Blackjack: Man Vs. Machine

Authors
Ron Hafzadi 316482637, Idan Shalom 208502047, Dan Fine 207882192
ron.hafzadi@mail.huji.ac.il idan.shalom@mail.huji.ac.il dan.fine@mail.huji.ac.il

Introduction

A game of Blackjack consists of a dealer and one or more players. Each player plays against the dealer. All players get two cards to begin and can ask for more until they bust (their total exceeds 21) or they stand (are happy with the total of their cards). Face cards count as 10, ace may be 1 or 11. The dealer is dealt one card face up and one face down and plays a fixed strategy: hit 16 or less (i.e., take a card), and stand on 17 or more. The player loses if they bust, wins if they do not bust and the dealer does. Otherwise, the player wins if his total is closer to 21 than the dealer's. In a tie scenario, the player does not win or lose money. If the player's first two cards total 21, this is called a blackjack, and he wins 1.5 times his bet (unless the dealer also has one and the no money changes hands) [1].

The game of Blackjack has been used as a model to explore decision-making, pattern recognition, and risk management, which are significant research areas in cognitive sciences and artificial intelligence. Unlike many games of chance, Blackjack is not purely based on luck; it requires players to make strategic decisions based on incomplete information, probabilistic outcomes, and evolving circumstances. In their article "Choices, Values and Frames" [2], Kahneman and Tversky have described the cognitive and the psycho-physical determinants of choice in risky and risk-less contexts. The process of mental accounting, in which people organize the outcomes of transactions, explains some anomalies in consumer behavior. In particular, they have shown the different value functions people exhibit in situations of gaining and losing money. Gambling games, and Blackjack in particular, have been studied numerous times in the past to study such phenomena further [3].

For these reasons, we aspire to solve the game using AI and simulate how humans may approach the game without prior knowledge but the game rules. More specifically, we will see how different types of risk attitudes affect the success rate in a game of Blackjack. Following Kahnmann and Tversky's Prospect theory, We will use 3 different risk attitudes: risk-lover, which will give low probabilities a higher value, risk-averse, which will give low probabilities a lower value, and risk-neutral, which will see the probabilities as they are. We will implement these attitudes using the ExpectiMax heuristic to simulate how different attitudes might affect actions in the game. Furthermore, to try and solve the game, we will have a Q-learning agent, which will hopefully converge to an optimal strategy ($basic\ strategy$, further discussion later). To benchmark the performance of the different agents, we will both compare the success rate to known deterministic strategies and compare the hit-stand tables of each combination. Finally, we will show how the success rate might affect the game's outcome using a "Last Man Standing" simulation.

Previous Work

Before any mathematical research of the game, several strategies were commonly used: "Never bust"- hit until 11, stand 12 and above, "Mimic the dealer"- hit until 16, stand 17 and above [4]. Edward Thorp, an American mathematics professor, described in his 1962 book "Beat the Dealer" a simple strategy that makes blackjack an almost even game [5]. Thorp's book revolutionized the game, regardless of one's mathematical background, every player could have a closed set of rules to have the highest likelihood to beat the casino. However, Thorp's book was written in 1966, and since then, the game has changed drastically: Thorp's theory was based on a single-deck game, which makes the player's actions much more dependent on previous cards, which contributed to the rising phenomenon of card counting. Card counting, although not illegal, is a way to beat the odds of the dealer and make the game uneven in favor of the players. The casinos in response, have changed the game during the years to have multiple decks, with random shuffling times, to not alter the casino-favoring probabilities of the game [6].

Nowadays, in response to the different game variants offered in casinos, many different closed optimal strategies (basic strategies) have been made, which were made simpler to calculate with numerical solutions via the rising power of modern computers [7]. Such strategies emerged using simulations of the different scenarios with different numbers of decks to give the best action in each scenario.

The rebirth of AI and machine learning gave another approach to research the game. Such researchers tried to use machine learning to converge into Thorp's basic strategy [8], and others have used reinforcement learning by simulating the game as a Markov chain and solving it with value iteration [9] and Q-learning [10] unsuccessfully, yielding to worse results than the basic strategy.

Previous works in the course have also tried to solve the game. Some have used Monte Carlo methods [11], while others tried to use Q-learning agents and card-counting agents [12] [13]. They both used a single deck and reached a result worse than Thorp's basic strategy, even when using card counting which we will not use, for the reasons mentioned.

Curiously, despite the existence of the relatively simple, well-known, and optimal strategy known as basic strategy, empirical studies have found that casino players play quite differently from that strategy [14]. For this reason, we aspire not only to solve the game but to give an in-depth view of the different personalities and strategies people can use. As the name of the course suggests, artificial intelligence's role is not only to solve human-related problems but to reflect on human cognition as a whole and serve as a model to study it.

Methodology

The Game

The Blackjack variant we implemented is a simpler version of the original game:

No Double

In most versions of the game, players have the option to double their bet before taking their first action. However, we have removed this option.

No Splitting

In most versions of the game, players have the option to split their hand on the first turn if they receive two identical cards, allowing them to place an additional bet on the split hand. However, we have removed this option.

No Card-Counting

To implement a more realistic game, we implemented the game such that each card has a probability of 1/52 to be drawn, independently of how many cards were drawn earlier. This way we eliminate the possibility of card-counting, for the reasons mentioned above.

Approaches

We implemented two distinct approaches to playing Blackjack: A Reinforcement Learning approach using Q-learning, and a probabilistic planning approach using the ExpectiMax algorithm.

Reinforcement Learning

The RL class implements a Q-learning algorithm for playing Blackjack.

Settings

State Space
Player sum: Values range from 1-32.
Dealer's up card: Values range from 1-11.
Usable ace: Boolean.
Action Space
{0:Stand, 1:Hit}
Q-Table
A 4-dimensional array of size 33x12x2x 2 representing the Q-values for each state-action pair.
Hyper-parameters
α: 0.01–Learning rate, determining how much new information overrides old information.
γ: 0.95–Discount factor, controlling the importance of future rewards.
ϵ: Initial value of 1–Exploration rate, controlling the balance between exploration and exploitation.
δ: 0.99995–Decay rate for ϵ, gradually reducing exploration over time.
Final ϵ: 0.1–The minimum exploration rate once decaying has converged.

Learning Process

The agent observes the current state. It selects an action using an ϵ-greedy policy, where it explores with probability ϵ and exploits (chooses the best-known action) with probability $1 - \epsilon$. After executing the action, the agent receives a reward and transitions to a new state. The Q-value for the chosen state-action pair is updated using the Q-learning update rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

As training progresses, ϵ decays according to the rule:
ϵ_t = max(ϵ_{min}, $\epsilon_t \cdot \delta$), where ϵ_{min} is set to 0.1 to ensure the agent continues to explore occasionally.

Reward Function

+1 for winning the hand.
-1 for losing the hand or busting.
0 for a tie.

Playing Method

After the training is completed, the learned policy for each player-dealer state pair is extracted and applied deterministically during gameplay, selecting the action with the highest Q-value.

ExpectiMax

The ExpectiMaxPlayer class implements an expectimax algorithm.

Settings

State Space
{Player sum ,Assessed Dealer sum}
Action Space
{0:Stand, 1:Hit}
Risk Preferences
The model incorporates three risk preferences, these preferences affect the evaluation function, allowing for different playing styles: $Risk - Averse$, $Risk - Loving$, and $Risk - Neutral$.

Decision Process

1.**Terminal State Check**
The algorithm first checks if the current state is terminal (dealer stands or player busts). If so, it evaluates the hand and returns a score.
2.**Player's Turn (Maximizing)**
For each card, it evaluates two scenarios:
a) Standing after this card.
b) Hitting again (if not busted).
Chooses the maximum expected value between standing and hitting. Weighs each outcome by the probability of drawing that card.
3.**Dealer's Turn (Chance Node)**
Simulates all possible dealer card draws. Calculates the expected value by summing probability-weighted outcomes for each card.
4.**Ace Handling**
Treats an Ace as 11 if it doesn't cause a bust, otherwise as 1.
5.**Recursive Exploration**
The function calls itself recursively, exploring all possible game states until reaching terminal states.

Evaluation Function

The `EvaluateHand` function evaluates a Blackjack hand by comparing the player's and dealer's card sums. The function takes into account whether the player wins or loses and adjusts the result based on the player's risk preference.

Base Value Calculation

Once the winner is determined, the function calculates a base value based on how close the player is to Blackjack(21):

$$\text{baseValue} = \begin{cases} \text{distance from 21} & \text{if the player wins,} \\ -\text{distance from 21} & \text{if the player loses.} \end{cases}$$

This base value reflects the difference between the player's hand and 21, and whether the player won or lost.

Risk Preferences

The `baseValue` is then modified based on the player's risk preference:

- **Risk-Averse**: The value is moderated by applying a logarithmic function, which grows slowly, so larger values have a smaller effect:

$$f(x) = \text{sgn}(x) \cdot \ln(|x| + 1)$$

- **Risk-Loving**: The value is exaggerated by applying an exponential function, which increases rapidly:

$$f(x) = \text{sgn}(x) \cdot \exp(|x|)$$

- **Risk-Neutral**: The base value is returned as-is without modification.

averageHitVal

In addition to the evaluation function, before deciding which action to take, we also calculate `averageHitVal`: This is the average expected value of hitting. However, the code also introduces variations based on risk preference:
For a risk-loving agent, the threshold is lowered (averageHitVal>StandVal-5), making them more likely to hit.
For a risk-averse agent, the threshold is raised slightly (averageHitVal>StandVal+0.5), making them less likely to hit.
For a risk-neutral player, the straight comparison is used.

Success Criteria

In order to evaluate the quality of our solutions, we will use multiple deterministic agents: Random, Never-Bust, Mimic-The-Dealer, and Basic-Strategy. We will compare these strategies to our agents using the following criteria:
1)Hit-Stand table comparison.
2)Win-rate over 10000 games (wins/wins+losses, ties omitted).
3)Simulation of number of games lasting starting with 1000$ with 10$ bet size.

Code

A Link to our repository can be found here [15].
The code is written in c# to be compatible with the Unity engine. The code has a class for a Player, and all of the agents are classes that inherit the Player class, overriding the method of "PlayTurn" and deciding how to act according to each agent's policy. The game is managed by a "GameManager" to fit for Unity.
We have used [16] To understand previous approaches for using RL to solve the game.
We have used [17] To understand better how to have the risk attitudes in ExpectiMax.
We have used [18] To help with the visualization of the game.
We have used [19] To help with the decaying epsilon.
We have used both Claude and Chat-GPT for both the backend and frontend of our game.

Results

Before showing our results, we will talk about the success of the control agents in our no-double no-split version of the game. As we mentioned, hands including ace are termed **soft**- as ace can be either 1 or 11. Hands not including ace are termed **hard**.

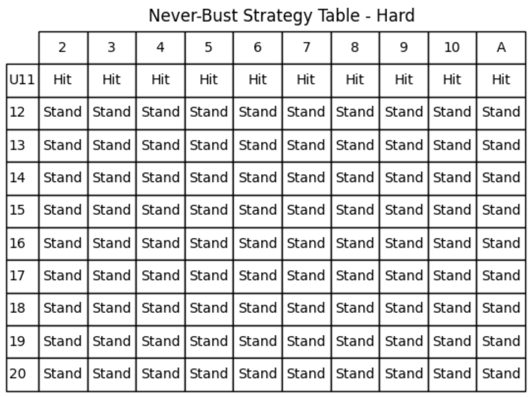
Control Agents

Random Agent

As mentioned before, The Random agent can either hit or stand with a 50-50% chance, regardless of the current state (table is redundant).
The Random agent got a **32.96%** winrate in a 10K rounds simulation.

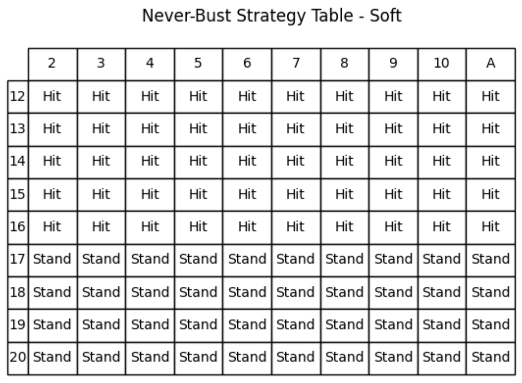
Never-Bust Agent

In our version of the game, The Never-Bust (NB) agent plays according to the following policy:


Never-Bust Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
13	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
14	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
15	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
16	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.


Never-Bust Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

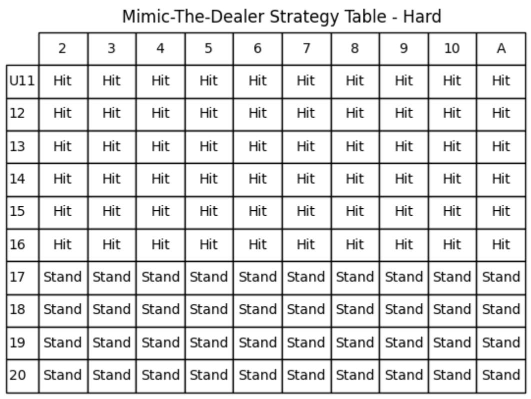
(b) Soft hand.

Figure 1: Never-Bust Policy.

The NB agent got a **45.36%** winrate in a 10K rounds simulation.

Mimic-The-Dealer Agent

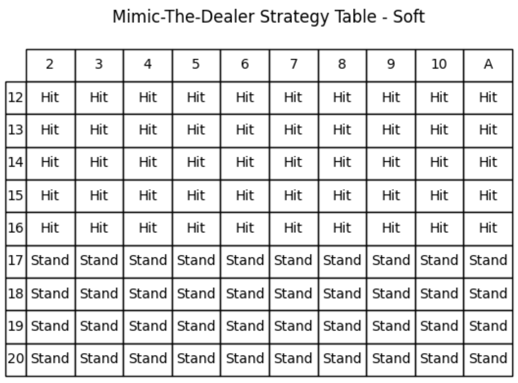
In our version of the game, The Mimic-The-Dealer (MTD) agent plays according to the following policy:



Mimic-The-Dealer Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.



Mimic-The-Dealer Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

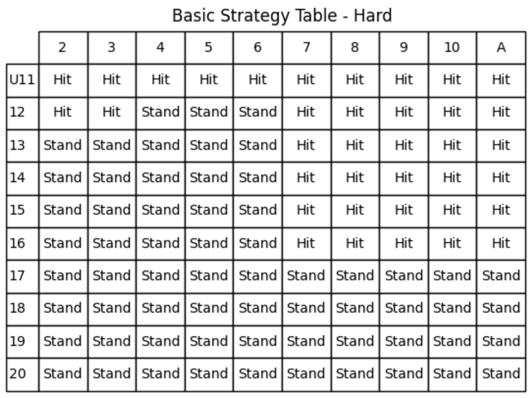
(b) Soft hand.

Figure 2: Mimic-The-Dealer Policy.

The MTD agent got a **45.37%** winrate in a 10K rounds simulation.

Basic-Strategy Agent

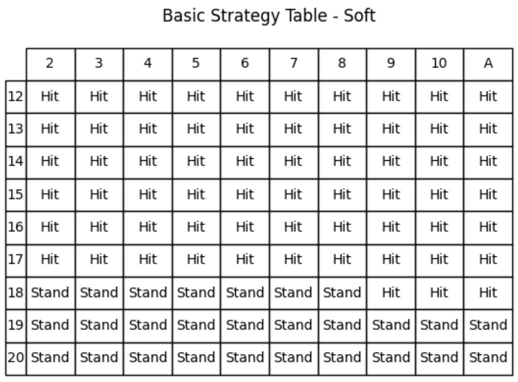
In our version of the game, The Basic-Strategy (BS) agent plays according to the following policy:



Basic Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.



Basic Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(b) Soft hand.

Figure 3: Basic-Strategy Policy.

The BS agent got a **46.98%** winrate in a 10K rounds simulation.

Control Agents Success

Before showing our agents, let’s compare the control agents’ success.

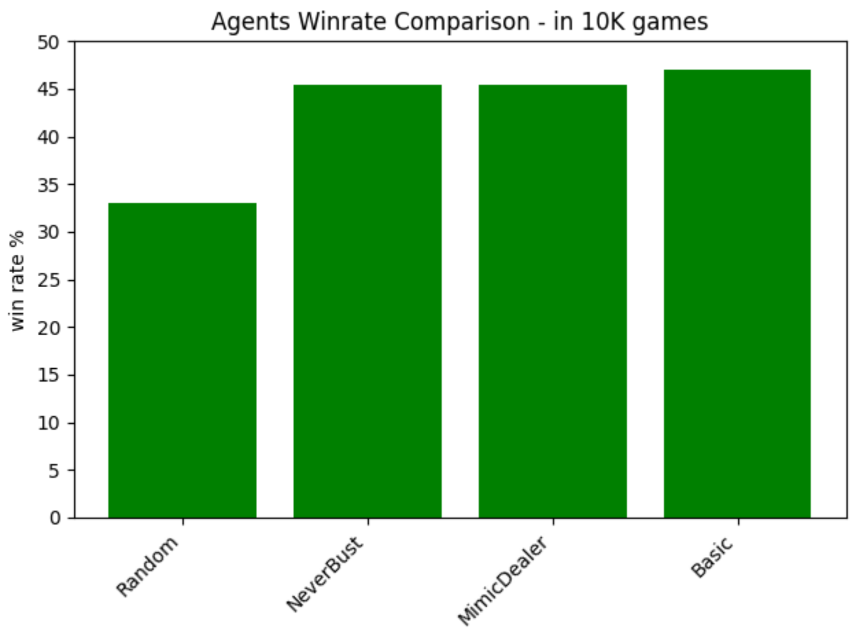


Figure 4: Success-Rate Of Control Agents

As already mentioned, our game version does not include doubling or splitting. Out of all the control agents, this hurts the BS agent the most, as he cannot double (on 11, for example), or split (with double aces, for example). This turns the odds against his favor, while the NB and MTD agents play with the same original policy.
Now let’s see how our agents competed against them.

Q-Learning Agent

After multiple tries with a constant epsilon and different parameters, we saw that our Q-learner converges to a non-optimal policy. We then decided to change our approach to using a decaying epsilon. This way our agent can re-explore states which could have led him to choose non-optimal actions.

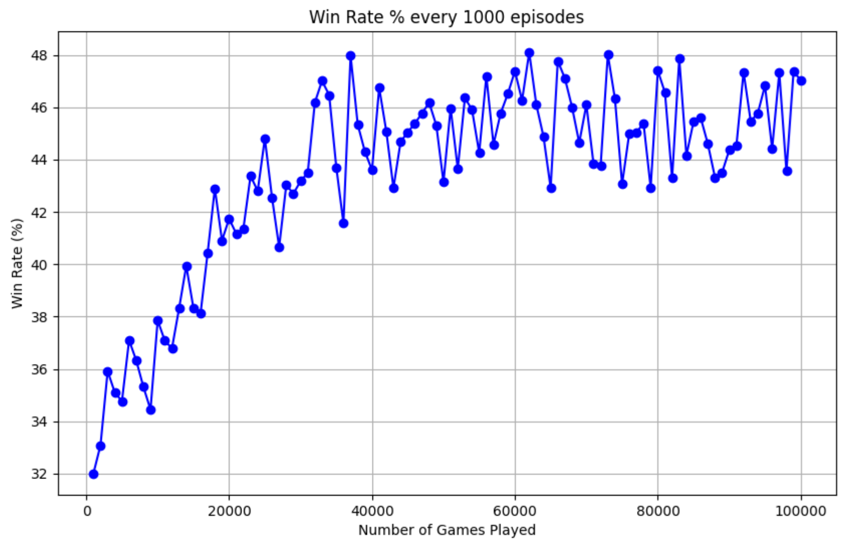
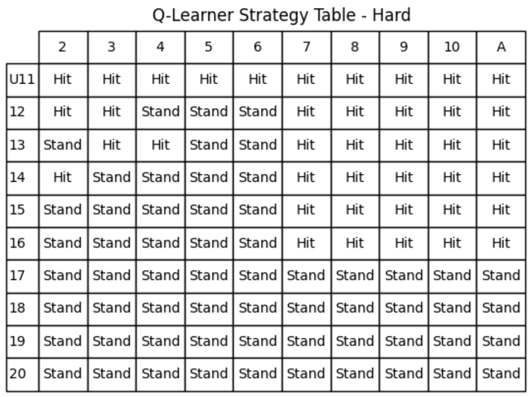


Figure 5: Q-learning training over 100K episodes

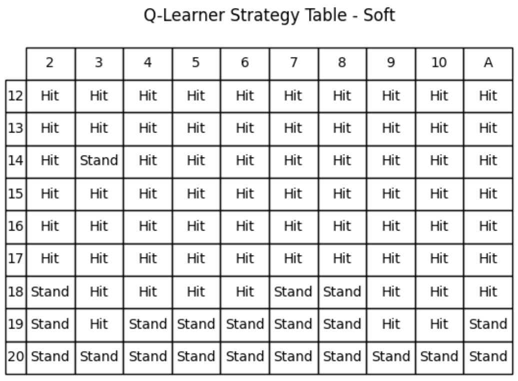
After 100K episodes, the Q-learner has converged to this policy:



Q-Learner Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Hit	Hit	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Hit	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.



Q-Learner Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Stand	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
17	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
18	Stand	Hit	Hit	Hit	Hit	Stand	Stand	Hit	Hit	Hit
19	Stand	Hit	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(b) Soft hand.

Figure 6: Q-Learning Converged Policy.

Comparing it to 3 there are a few anomalies not seen in the table; The Q-learner stands with 5 against 3 of the dealers, and with 4 against 4 of the dealers. Moreover, we can see that while the Hard table of the Q-learner is almost the same as that of the BS agent, The soft table is quite different. This can be explained by the fact that the Q-learner learns the game while playing it. The more card-combinations of the same kind he will see, the more likely he will converge to the correct, optimal, policy- for that specific combination. As soft hands are rarer, such as small hands like 3 or 4, it's likely that the Q-leaner did not have a chance to face enough of these combinations against each of the dealer's cards to converge to an optimal policy before the epsilon got too small. As this game is probabilistic, our agent could have been rewarded for a wrong decision (standing on 5 and the dealer busts) or punished for a correct decision.
All in all, using this strategy, our Q-learning agent got a **46.27%** winrate in a 10K rounds simulation, not far from the BS agent.

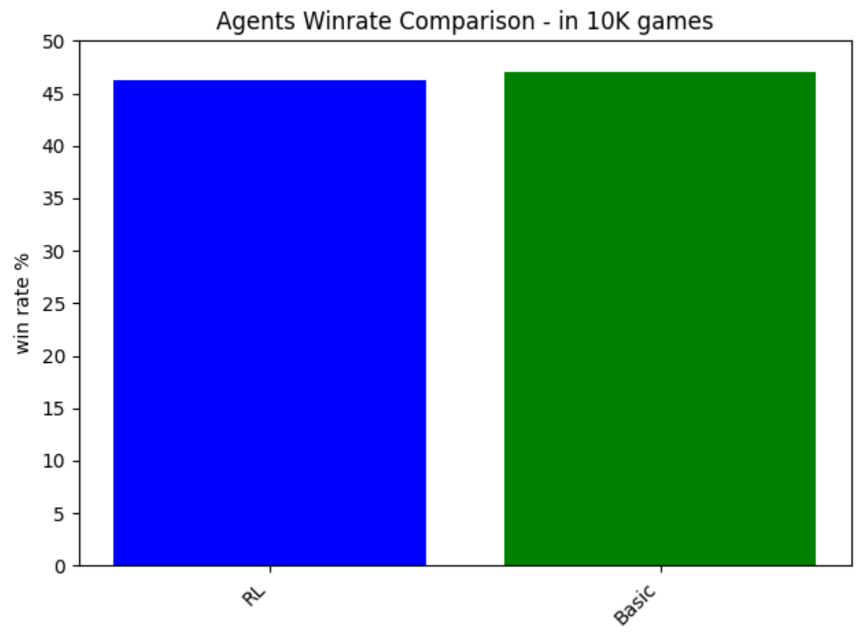


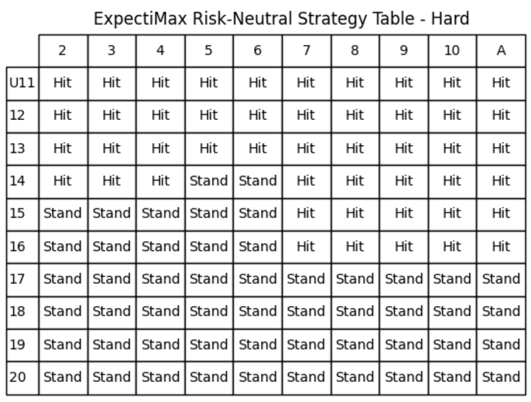
Figure 7: Success-Rate Of Q-Learning Agent

ExcpectiMax Agents

As we mentioned earlier, we made 3 agents. Risk-Neutral (ERN), Risk-Averse (ERA), and Risk-Loving (ERL).

Risk-Neutral

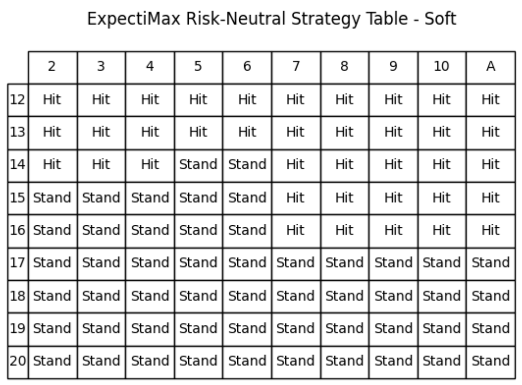
After multiple tries, we got a good result using the evaluation function mentioned in the methodology. The policy table is as follow:



ExpectiMax Risk-Neutral Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.



ExpectiMax Risk-Neutral Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

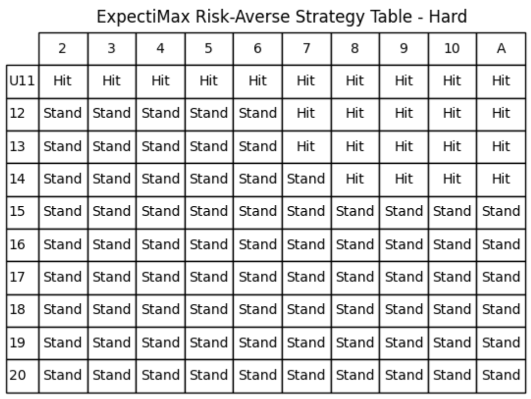
(b) Soft hand.

Figure 8: ExpectiMax Risk-Neutral Policy.

We can see how the ERN agent is somewhat similar to the BS agent. We can see how he hits very similarly in hard hands to BS, with a small exception: he stands with 14 against 5 and 6, and hits in 12 no matter what the dealer has. The reason might be that he thinks 5 and 6 are bad upcards for the dealer, and hopes he would bust.
Using this strategy, the ERN agent got a **46.04%** winrate in a 10K rounds simulation.

Risk-Averse

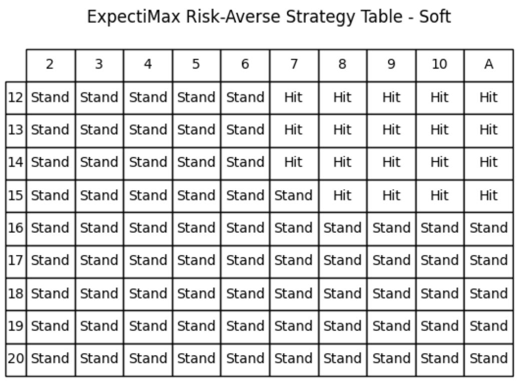
The ERA agent is based on the ERN agent, except he uses a log transformation in his hand evaluation. His policy table is as such:



ExpectiMax Risk-Averse Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Stand	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
16	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.



ExpectiMax Risk-Averse Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

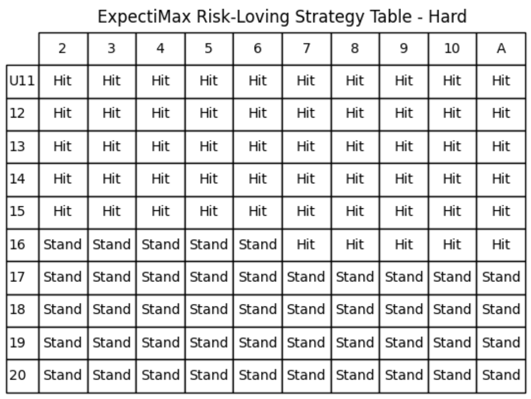
(b) Soft hand.

Figure 9: ExpectiMax Risk-Averse Policy.

We can see how the ERA agent is more conservative than the ERN agent. He stands on all hard 15s and only hits with 14 against 8, 9, 10, and ace. In the soft-hand table we can see that he uses the same strategy. This can be explained by the fact that the Risk-Averse evaluates his hand using the log-transform, making him more cautious to bust, which makes him stand much earlier than needed- taking fewer risks. We can also see that he resembles the NB agent. Using this strategy, the ERA agent got a **45.14%** winrate in a 10K rounds simulation.

Risk-Loving

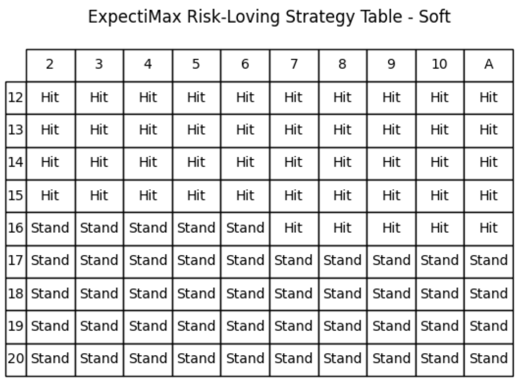
The ERL agent is based on the ERN agent, except he uses an exp transformation on his evaluation of a hand. His policy table is as such:



ExpectiMax Risk-Loving Strategy Table - Hard

	2	3	4	5	6	7	8	9	10	A
U11	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(a) Hard hand.



ExpectiMax Risk-Loving Strategy Table - Soft

	2	3	4	5	6	7	8	9	10	A
12	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
13	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
14	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
15	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

(b) Soft hand.

Figure 10: ExpectiMax Risk-Loving Policy.

We can see how the ERL agent is less conservative than the ERN agent. He hits on all hard 15s and only stands with 16 against 2-6. In the soft-hand table we can see that he uses the same strategy. This can be explained by the fact that the Risk-Loving evaluates his hand using the exp-transform, making him less cautious of busting, which makes him hit up to 15 included. We can also see that he resembles the MTD agent. Using this strategy, the ERL agent has got a **46.15%** winrate in a 10K rounds simulation.

ExpectiMax Comparison

All in all, we can see how our agents have gotten good results, similar to well-known strategies.

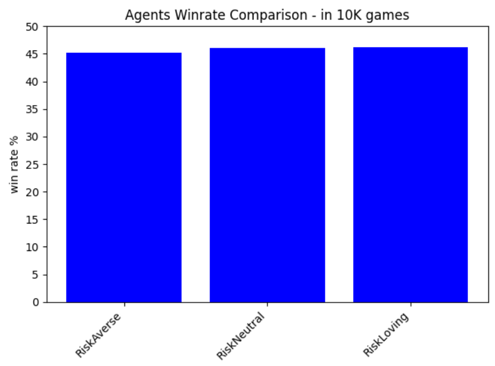


Figure 11: Success-Rate Of ExpectiMax Agents

Surprisingly, the ERL agent has succeeded the most. Although in hard hands ERN hits 6 times less than BS, while the ERL hits 18 times more than BS, the MTD strategy which resembles ERL proves that although not optimal, hitting above 12 and under 16 is not such a bad strategy. It yields less than 1% win-rate than any ExpectiMax agent, and less than 2% from BS. Moreover, the small differences in percentage can be attributed to luck alone as this is a probabilistic game after all.

Agents Comparison

Although not optimal, our results were very close to that of BS. The Q-learner had only .69 less win% than BS, and the ExpectiMax agents were not far from it.

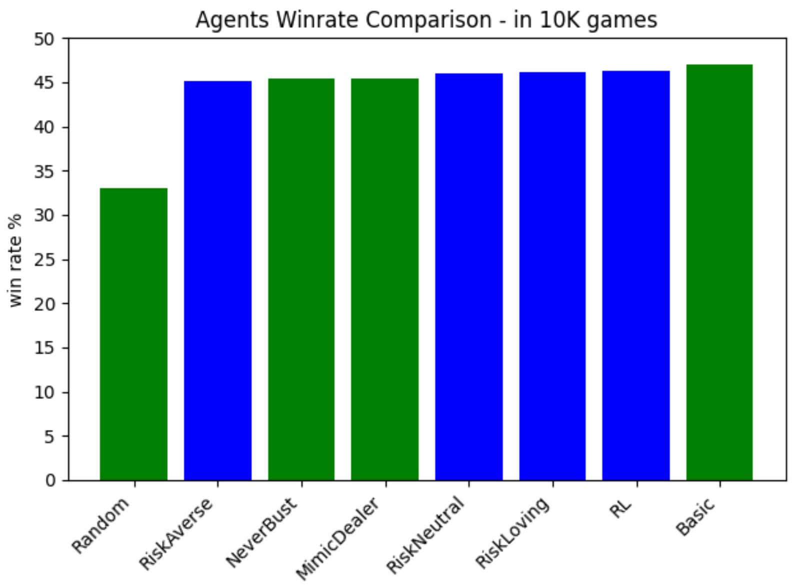
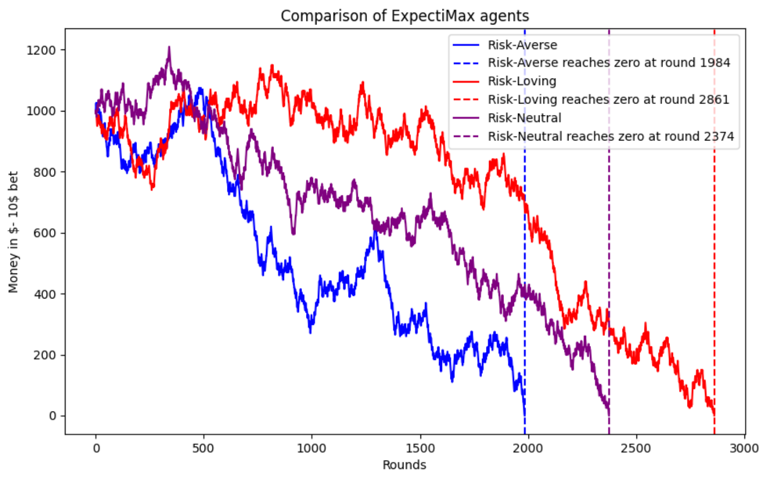


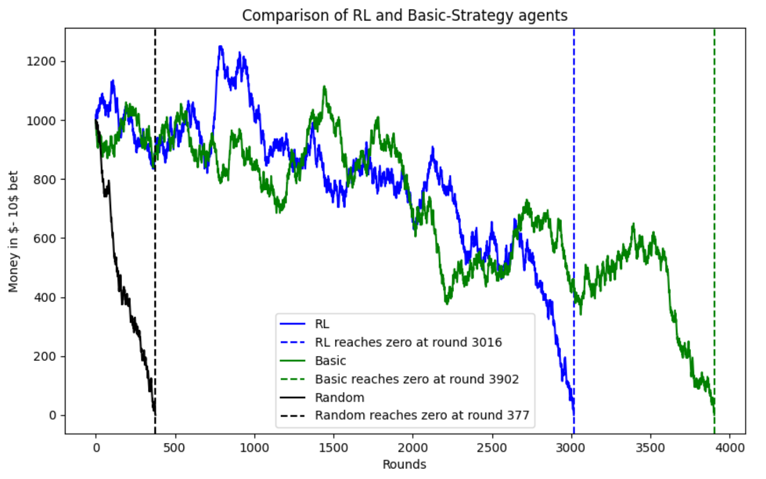
Figure 12: Success-Rate Of All Agents

Last Agent Standing

To have another evaluation of our solutions, we decided to do a more realistic simulation. Each agent starts with 1000$ in a 10$ bet blackjack, and we see who lasts the most number rounds.



(a) ExpectiMax agents.



(b) Random, RL and Basic agents.

Figure 13: Last Agent Standing.

In these graphs, we can see how small changes in win rate can alter the game's outcome in real-life scenarios. We can see how the random agent loses his money in under 500 rounds, while the ExpectiMax agents reach over 2K. Meanwhile, The Q-learning and BS agents have reached over 3K rounds. Another interesting thing to see here is that at some points in time, the Q-learning agent was far above the BS agent, but the correctness of the BS policy has proven to be better over time. Lastly, we can see how although optimal, not BS and no other agent has reached over 1400$, which shows the power casinos have over people, and how profitable these games are to them.

Summary

Our mission was to research different algorithms that could play Blackjack and find out if any of them could reach the Basic-Strategy's win percentage, which is the optimal solution. In some aspects, we have succeeded - our Q-Learning agent did very well compared to the Basic-Strategy. However, it still didn't reach it.

Q-learning can struggle in games like Blackjack or other probabilistic games for several reasons. These challenges arise due to the nature of the game dynamics, the exploration of the state space, and the handling of uncertainty. Blackjack has many possible game states, Q-learning works by learning an optimal action for each state-action pair, but with so many possible states in Blackjack, it can take a long time to explore and update the values for all relevant state-action combinations. Many states may only be visited occasionally, making it hard for Q-learning to accurately estimate the Q-values for those states. Moreover, Q-learning assumes a deterministic environment, where repeated actions in the same state should yield consistent outcomes over time. However, in probabilistic environments like Blackjack, the outcome of taking an action (like "hit") may vary depending on unseen cards, leading to punishing correct actions, and rewarding bad actions, causing inconsistent Q-value updates. We have tried to overcome this problem using decaying epsilon, but still, some state-action pairs might be much rarer than others, and even this solution couldn't overcome it.

Our second approach to the game was also not ideal. In our ExpectiMax implementation, we tried numerous evaluation functions until we got one that made sense. The biggest problem we faced was the punishment for busting, which in some cases might be worse than standing under 11. The busting method in Blackjack makes you lose your hand at that moment, no matter if the dealer busts later. Standing with a small sum can give you a chance to win if the dealer busts, which can happen quite often. The balance between punishing our agent for busting, but granting him reward for hitting Blackjack was not trivial, but we managed to do so.
Our Risk-Loving and Risk-Taking ExpectiMax agents also did not play as we hoped. In our hypothesis we wanted the Risk-averse to never hit above 12 making him much more similar to the Never-Bust agent, and the Risk-Lover agent to hit almost always in a try to get to 21 and get the 1.5 reward. We got more moderate solutions, which in the end resembled well-known strategies, and they all succeeded better than them.

All in all, we have seen how even when using ExpectiMax with different risk-attitudes, Q-learning, or even the famous Basic-Strategy, Nobody beats the casino.

References

[1] R. T. Duda, “Mathematics of the blackjack counting system,” 2004, accessed: 2024-08-25. [Online]. Available: https://services.math.duke.edu/~rtd/MEC/prob/blackjack.html

[2] D. Kahneman and A. Tversky, “Choices, values, and frames,” *American Psychologist*, vol. 39, no. 4, pp. 341–350, 1984, accessed: 2024-08-25. [Online]. Available: https://psycnet.apa.org/fulltext/1985-05780-001.pdf

[3] J. Hewig, R. Trippe, H. Hecht, M. G. Coles, C. B. Holroyd, and W. H. Miltner, “Decision-making in blackjack: An electrophysiological analysis,” *Cerebral Cortex*, vol. 17, no. 4, pp. 865–877, 2007. [Online]. Available: https://doi.org/10.1093/cercor/bhk040

[4] G. Keren and W. A. Wagenaar, “On the psychology of playing blackjack: Normative and descriptive considerations with implications for decision theory,” *Journal of Experimental Psychology: General*, vol. 114, no. 2, pp. 133–158, June 1985. [Online]. Available: https://psycnet.apa.org/fulltext/1986-08209-001.pdf

[5] E. O. Thorp, *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*, revised edition ed. New York: Vintage, 1966.

[6] O. Vancura, K. Fuchs, and J. Knott, *Knock-Out Blackjack: The Easiest Card Counting System Ever Devised*. New York: Square One Publishers, 1998, accessed: 2024-08-25. [Online]. Available: https://www.google.co.il/books/edition/Knock_out_Blackjack/K6H6Y3WtgoAC?hl=en&gbpv=1&dq=The+Easiest+Card+Counting+System+Ever+Devised&printsec=frontcover

[7] D. Fogel, “Evolving strategies in blackjack,” p. Published in: Proceedings of the 2004 Congress on Evolutionary Computation, June 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1331064

[8] S. Yakowitz and M. Kollier, “Machine learning for optimal blackjack counting strategies,” *Journal of Statistical Planning and Inference*, vol. 33, pp. 295–309, 1992, received 28 September 1990; revised manuscript received 23 September 1991. Recommended by S. Zacks.

[9] J. L. Popyack, “Blackjack-playing agents in an advanced ai course,” contact: (215) 895-1846, JPopyack@CS.Drexel.edu.

[10] C. De Granville, “Applying reinforcement learning to blackjack using q-learning,” contact: CHAZZ184@OU.EDU.

[11] Unknown, “Finding optimal strategy using monte carlo,” Project report, The Hebrew University of Jerusalem, 2021. [Online]. Available: https://www.cs.huji.ac.il/course/2021/ai/projects/old/Blackjack.pdf

[12] S. Swissa and D. Almog, “A.i. project – blackjack,” Project report, The Hebrew University of Jerusalem, 2014. [Online]. Available: https://www.cs.huji.ac.il/course/2021/ai/projects/2014/BlackJack/

[13] L. Zamir, S. Tubul, and E. Ami, “Blackjack: Final project,” Project report, The Hebrew University of Jerusalem, 2014. [Online]. Available: https://www.cs.huji.ac.il/course/2021/ai/projects/2014/BlackJack/

[14] M. Schiller and F. Gobet, “A comparison between cognitive and ai models of blackjack strategy learning,” September 2012. [Online]. Available: https://www.researchgate.net/publication/262366834_A_Comparison_between_Cognitive_and_AI_Models_of_Blackjack_Strategy_Learning

[15] I. Shalom, “Blackjack_ai,” 2024, accessed: 2024-09-15. [Online]. Available: https://github.com/Idan-Shalom/BlackJack_AI

[16] L. Rick, “Blackjack strategy table,” 2023, accessed: 2024-09-13. [Online]. Available: https://www.youtube.com/watch?v=9MArYMVWubA&t=315s

[17] EconPort, “Risk and uncertainty,” 2024, accessed: 2024-09-13. [Online]. Available: https://www.econport.org/content/handbook/decisions-uncertainty/basic/risk.html

[18] Mr.Kaiser, “How to make a game,” 2022, accessed: 2024-09-13. [Online]. Available: https://www.youtube.com/watch?v=pbMRqgGbkVE

[19] F. Foundation, “Blackjack tutorial,” https://gymnasium.farama.org/tutorials/training_agents/blackjack_tutorial/, 2023, accessed: 2024-09-14.