# Optimizing Blackjack games with Tabular and Deep Q-learning

**Hung Huu Hoang** hhoang26@uwo.ca
*Department of Computer Science*
*University of Western Ontario*
*London, Ontario*

**Editor:** Hung Huu Hoang

## Abstract

Blackjack is a game that is popular worldwide. It has also has a known optimal strategy often referred to as the Basic Strategy. In this project, I worked on training an RL agent using Q-learning (Tabular and DQN) to play the game optimally, i.e., as close to the optimal strategy as possible. My evaluation results also showed that, after up to 3 million training rounds, the RL agent could produce a strategy that is very close to the optimal one in the case where only Hit and Stand actions are considered. When Double-down and Split actions are also considered, the best results still have many differences compared with the optimal one. Another observation is that for Blackjack, Tabular Q-learning produced better playing strategies than DQN in most cases.

**Keywords:** Reinforcement learning, Blackjack game, Tabular Q-learning, DQN

## 1 Introduction

Blackjack is popular game that is played almost everywhere, in both casinos and for fun among friends and family members.

For Blackjack, the optimal playing strategy has been known since Baldwin et al. (1956); Apprenticeship (2024), making it a good game for learning and testing Reinforcement Learning (RL) algorithms.

In Blackjack, the observation and action spaces are relatively small, a few hundreds for observations and a few discrete actions. As such, this game seems amenable to Q-learning, a basic foundational algorithm in RL. The goal is to train the RL agent to produce a strategy as close to the optimal one as possible.

There are already many existing projects working on this problem Geiser and Hasseler (2024); De Granville; Zha et al. (2021), so the main contribution of this project is more about customizing the existing Gymnasium environment Foundation (2024) to support more actions: Double down and Split.

The results showed that a tabular Q-learning algorithm could deal well with this problem, achieving almost optimal solutions if we consider only the two basic actions of Hit and Stand. Introducing Double down and Split created significant problems for both tabular and Deep Q-learning, whose best achieved policy was still quite far from the optimal one.

## 2 Blackjack: Background and supported rules

### 2.1 Background on Blackjack

Although a globally popular game, there are many slight variations of Blackjack rules used in different parts of the world, with most of the differences lie in more advanced features or rules, such as insurance and the number of decks used. As these advanced actions are not yet supported in this project, most of the Blackjack rules described below are standard.

In Blackjack games, there is a dealer, who usually represents the casino, and one or many players. Winning or losing is between each player and the dealer, although players can also optionally bet among themselves, which is often known as “side bets”. In this project, side bets are not supported and our only concern is whether the RL agent, representing one player, wins or loses vis-a-vis the dealer.

In order to decide who wins the game, we need to sum up the values of their cards. In Blackjack, cards’ suits are not used. When summing, cards’ values are calculated as follows:

- For number cards (2 - 10): their value is the number.
- For face cards (Jack, Queen and King): their values are 10.
- For Ace (A): it can be counted as 1 or 11. Whenever it is possible to count Ace as 11 without busting the hand, i.e., making the sum exceed 21, then we will always count it as 11. This Ace counting approach may not be universal but this is the version supported in this project.
  Terminology: When a hand has an Ace that can be counted as 11, we will refer to that hand as “soft” or having a “usable Ace”, interchangeably. For example, a hand of Ace + 7 is counted as 18, and it is referred to as a “soft” 18 or 18 with a usable ace. A hand of Jack + 8 is also counted as 18 but because it does not have any Ace in it, it is called a “hard” 18.

A Blackjack, sometimes also called a “natural”, refers to the hand with exactly these 2 cards: Ace and a ten-valued card (either a 10 or a face card). Therefore, a Blackjack can only be obtained by the first 2 cards dealt by the dealer.

A standard flow of a Blackjack game takes place as follows:

1. The dealer deals 2 cards for each player, including themselves. Each player will only see their cards, while the first card of the dealer is shown to all players.
2. Players will then, after looking at their cards, and in turn, decide if they want to take additional cards or not. If they do, the dealer will give them more cards until they stop. This process repeats for each player.
   Terminology: When a player decides to take one more card: that action is known as “Hit”. And when they decide to stop taking cards, it is known as “Stand”.
3. The dealer then looks at their face-down card and decide to take more cards or not using the following rule: Dealers have to take more cards as long as the sum of their existing cards $< 17$. Whenever their sum is $\geq 17$, they have to stop.
   Note: there is another variation of this rule where the dealer has to hit on soft 17 (Ace + 6, for example). But in this project, the supported rule is that the dealer will stand on both hard and soft 17.

4. Once the dealer is done with taking additional cards (if any), they will show their cards. At this time, winning or losing between each player and the dealer can be decided based on the sum of their hands using the following procedure:
   If the player busted, i.e., sum of their cards $> 21$: the dealer wins, regardless of dealer's sum. Note that the dealer still wins in this case if their hand busted.
   If the player did not bust and the dealer busted: player wins.
   If both did not bust: then the person whose sum is larger wins. If the sums are equal, it is a draw.

The rules of Blackjack, as mentioned above, give dealer a significant advantage over players because when players go bust, dealer will always win, regardless of whether they will also eventually go bust or not. This rule is by far the most significant leverage that the dealer has.

### 2.2 Supported Blackjack rules

In addition to the basic Hit and Stand actions, Double down and Split are two additional actions that are also supported in this project. Their usage rules are as follows:

- Double down: this action means the player wants to double their existing bets. This action is only allowed on the first 2 cards, i.e., before taking any additional cards. And after double down, the player can only take 1 additional card and be done with their hand, even if their hand's sum then is still very small. Double down is allowed for any hand of any value.

- Split: this action means splitting their existing hand into 2 hands. Split is only allowed if their current hand has only 2 cards (i.e., before any hitting), and these 2 cards have the same value. Examples of hands that can be split are (2,2), (10, J) and (A, A). After splitting, each of their new hands will receive a new card so that they will have 2 hands each of 2 cards. The same standard rules are used to play each of these 2 hands.

Lastly, below are 2 additional notes regarding the Blackjack rules that are supported in this project's environment. Firstly, if a player gets Blackjack, dealer pays 3:2, i.e., players win 1.5x their betting amount. If the dealer has Blackjack, players still lose only their original bet amount. Secondly, in casino games, cards are taken from a shoe of 4 - 8 decks, each having 52 cards. In for-fun games at home, cards are taken from a single deck. In this project, for simplicity, an "infinite deck" is supported: a card once taken out will still be there available for the next draw, i.e., cards are taken with replacement, creating the same effect as an infinite deck.

## 3 Methodology

### 3.1 Environment

As described above on Blackjack rules, the environment for our RL agent can be modeled as follows for the 2 versions that are supported in this project.

**Version 1: Split not supported**

- Actions: 3 discrete values: Hit, Stand, Double down.
- Observations: a 3-tuple:
  - *Player's hand sum*: sum of the values of cards in the player's hand. Possible values: integers from 4 to 31.
  - *Dealer's up card*: integers from 2 to 26. Before the game (also referred to as episode) is completed, this value only shows the dealer's up card. This means that as long as the agent still needs to make a decision, it will only see the up-card's value. When the episode is completed, this value will show the dealer's hand sum for reference purpose. This explains why the range is from 2 to 26.
  - *Player has as a usable ace*: boolean, as defined in the Blackjack rules above.

**Version 2: Split is supported**

- Actions: 4 discrete values: Hit, Stand, Double, Split
- Observations: a 4-tuple: the first 3 as in version 1, then adding the 4th one: *Player hand can be split*, which is a boolean value.

All rewards before reaching the terminal state are 0. For terminal states (when the episode is completed), an agent get a reward of 1 for winning (or 1.5 for Blackjack and winning), 0 for a draw and -1 for losing. If Double down was selected, rewards will be doubled to become 2, 0 and -2, respectively.

### 3.2 Algorithms

In this project, I experimented with both (tabular) Q-learning and Deep Q-learning (DQN). For Blackjack's discrete actions and a small observation space, standard tabular Q-learning seems sufficient. However, for comparison purposes, I also experimented with DQN.

Below, I will give a short summary of Q-learning and DQN. For more comprehensive descriptions, readers are referred to Sutton and Barto (2018); Mnih et al. (2013).

Q-learning is a model-free value-based RL algorithm that updates the value of (s, a) after an action has been taken, i.e., a temporal-difference type of algorithm. Conceptually, it is a simple table where each row corresponds to a state (observation) and each column corresponds to an action. The value for a particular (state, action) pair is stored and updated in this table.

The update equation for tabular Q-learning is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s') + \gamma \max_{a'}(Qs',a') - Q(s,a)] \tag{1}$$

where:

- Q(s, a): current value of (s, a)
- R(s'): the reward obtained when in state s', which is the resulting state after taking action $a$ in state $s$.
- $\alpha$: learning rate. In this project, the optimal value found was 1e-3.

- $\gamma$: discount of future rewards. In this project, the optimal value found was 0.5 for Q-learning and 0.1 for DQN

There is also another important hyper-parameter, the epsilon value used in Q-learning's epsilon-greedy policy. For a probability ¡ epsilon, the agent will take a random action, and in the other scenario, the agent will take the action corresponding to the current highest Q (s, a) value. Epsilon usually starts at 1 at the beginning of the training and gradually reduced to a small value such as 0.1 at the end of last episode.

During inference, for each observed state s, we simply do a lookup in the Q table for s and choose the action that maximizes Q(s, a). The policy is now always a greedy one, and no longer an epsilon-greedy one as during training.

DQN is similar to Q-learning in that it also tries to find suitable values for Q(s, a) through training. But instead of using a Q table, DQN trains a neural network, such as a multi-layer feedforward network or a CNN, to estimate Q values. We can now use standard gradient descent to minimize the loss between the temporal difference and current Q values, calculated in the same fashion as in Q-learning above.

Besides using a neural network, there are 2 other new differences in DQN compared with Q-learning: the introduction of a replay memory and a new target Q-network, in addition to the usual Q-network. The replay memory is used to gather experiences in a buffer from which training observations are sampled. It has been shown that training using a batch sampled from replay memory helps stabilize the training process, compared with, say, just using new observations. The introduction of the target Q-network also helps increase stability, because the target value of the optimization is now fixed for a certain duration.

Once trained, given a state s, DQN(s) outputs a probability for each action. From this distribution, we can choose to sample an action from it, or to deterministically take the action with the highest probability as we would for this Blackjack game.

### 3.3 Implementations

I started off with the initial environment and Q-learning implementation from Gymnasium Foundation (2024) which was named ”Blackjack-v1”. From there, below are the major changes and extensions made:

- Fixing existing bugs in the current environment regarding rewards calculation.

- Extending this environment, which supports only Hit and Stand actions, to support Double Down and Split actions. Most of the changes were related to extending the observations modeling and adding logics to make sure only legal actions at a particular state are allowed. See Section Supported rules above for rules regarding these 2 actions. Supporting Split was more problematic because Split introduces 2 new hands. The following design choices were made:

    - When a hand is split, that episode is considered to have ended with a reward of 0. The 2 split hands are automatically dealt with 2 additional cards so that we have 2 hands each with 2 cards. These 2 hands are scheduled as the next 2 episodes for playing.

- For Q-learning, Q (s, a), where a is Split, is then updated 2 times using Q values of the next 2 hands (instead of being updated just 1 time as usual). This was so designed to reflect the fact that whether Split is a good action or not depends on what comes out of the subsequent 2 hands.

For DQN, I used the implementation of Stable-BaselineS3 (SB3) Raffin et al. (2021) where DQN is implemented as feedforward neural network with 2 hidden layer of 64 params at each layer. This implementation was used as-is and my experiments' focus for DQN was only on optimizing hyper-parameters. The customized environment used with Q-learning above also needed some modifications to work with SB3.

Lastly, the initial policy visualization in Foundation (2024) was also customized so that it looks similar in form to the optimal policy in Apprenticeship (2024), for easier comparison and reference.

### 3.4 Evaluation metrics

The most important evaluation metric used is how similar the produced policy is compared to the optimal one in Apprenticeship (2024). In all discussion below, if not explicitly indicated, this is the metric that was used to compare performance among models.

Another important metric is the average rewards earned when the agent is run through an evaluation that consists of 100,000 games (episodes). In practice, this is actually the metric that matters to players, but given that the optimal policy for Blackjack is already known, comparing the policies produces a more direct comparison.

The winning, drawing and losing rates (as percentage) are also measured for reference when needed.

## 4 Experiments and Results

### 4.1 Hyper-params search

As discussed above, one of the most important aspects of my experiments was about hyper-parameters search for learning rate (alpha), epsilon (in epsilon-greedy) and discount factor (gamma).

Learning rate and discount values were tested together to find the best combination. After that, epsilon values were tested using the best combination of discount and learning rate values found.

Table 1 below shows the values that were tested and the best selected values for Q-learning.

Near the end of the experiments, I also tried using a gradually reduced learning rate of 1/n where n is the number of times a (state, action) pair has been visited before, together with a discount of 0.5. But this method did not produce any better policies than using the best combination above.

For DQN, most of the hyper-params used the default values in SB3 Raffin et al. (2021). Table 2 below shows the 2 hyper-params that were different from SB3's default values.

| **Hyper-params** | **Tested values** | **Best values from experiments** |
|---|---|---|
| Learning rates + Discount values | • 1e-2, 1e-3, 5e-5 + 0.95<br>• 1e-2, 1e-3, 1e-4 + 0.5<br>• 1e-3 + 0.15 | 1e-3 + 0.5 |
| Epsilon (start → end) | • 1 → 0.1<br>• 1 → 0.5<br>• 1 → 1 (no decay) | 1 → 0.1 |

Table 1: Hyper-params search results for Q-learning

| **Hyper param** | **SB3's default** | **Used values** |
|---|---|---|
| Discount factor ($\gamma$) | 0.99 | 0.1 |
| Buffer size (replay buffer) | 1 million | 50K |

Table 2: Hyper-params that are different from SB3's default values

### 4.2 Evolution of Q-learning's best policy

Probably the most interesting aspect of training a toy agent like this is to see how the produced policy evolves over time.

The obtained policies for Q-learning agent after every 100K additional training episodes of the first 1 million episodes are shown in Appendix A.

### 4.3 Best obtained policies

Normally, there are only 3 policies of interest:

1. Policy for hard total (no ace that can be counted as 11 in the hand), which include decisions related to Hit, Stand, Double down
2. Policy for soft total (Hit, Stand, Double down)
3. Policy for split (Split or not)

However, Double down is an action that can be considered as a Hit when doubling down is not allowed. Reducing Double down to a Hit creates 2 additional scenarios where we are only concerned about Hit or Stand.

Table 3 below reports the best policies obtained for these 5 scenarios. For Tabular Q-learning, these are the best results after up to 3 million training rounds. For DQN, these are the best results obtained after training up to 2 million timesteps.

| **Scenario** | **Number of deviations of the best obtained policy from the optimal one** | **Models obtaining the best policy** |
|---|---|---|
| 1. Hard total (No double-down) | 1 | Q-learning with split supported |
| 2. Hard total with double-down | 4 | Q-learning with split supported |
| 3. Soft total (No double-down) | 0 | Q-learning with double-down supported but no split<br>DQN with double-down not supported |
| 4. Soft total with double-down | 14 | Q-learning with split supported |
| 5. Split or not (1 for split; 0 for no split) | 27 | Q-learning with split supported |

Table 3: Best obtained policies for different scenarios

The best obtained policy for 3 over 4 scenarios, with the exception of scenario #3 above concerning soft total when Double-down is not supported, were obtained with an environment supporting Split. This result was quite interesting because one would normally assume that an agent would learn more poorly in a more complex environment.

All of the best policies were obtained by Q-learning and DQN performed quite badly across all 5 scenarios, and could only co-obtain the optimal policy for Soft total when Double-down is not allowed.

I found this poor performance to be somewhat surprising, given that DQN is more powerful than Q-learning. But at the same time, there are also plausible explanations for it. The main cause, in my opinions, was that for DQN I only used the SB3 library as-is and did not modify the library to support some desirable customizations that had been done for Q-learning. The two most important customisations not available with DQN were:

1. Linking the rewards of the hand that was split to the 2 produced hands
2. Selecting only legal actions for a state, which was implemented by returning -100 as a big negative reward.

### 4.4 Best obtained rewards

For Blackjack games, the house (dealer) still has an edge, i.e., their winning rate is $> 50\%$. This means that over a long period, it is virtually guaranteed that their earning is positive and players will lose.

To quantify how much players lose, I also evaluated the trained agents over 100,000 episodes to see the average obtained rewards.

Table 4 below records rewards, winning percentage obtained by 3 agents:

1. Random agent: randomly select Hit or Stand (Double-down and Split not allowed)

2. Optimal agent: plays according to the optimal policy, with double-down and split supported.

3. Best trained RL agent, which is the agent that obtained the best policy for scenario 1 and 2 in Table 3. Double-down and split supported.

| **Agent** | **Average rewards and Win, Draw, Lose stats** |
|---|---|
| Random agent | • Avg reward: -0.38<br>• Win rate: 28.2%<br>• Draw rate: 4.0%<br>• Lose rate: 67.8% |
| Optimal-strategy agent | • Avg reward: -0.005<br>• Win rate: 43.4%<br>• Draw rate: 8.9%<br>• Lose rate: 47.7% |
| Best trained RL agent | • Avg reward: -0.006<br>• Win rate: 43.3%<br>• Draw rate: 9.0%<br>• Lose rate: 47.7% |

Table 4: Rewards and winning rates comparison

From Table 4, it can be seen that compared to a random agent, the best trained Q-learning model performs significantly better.

What is somewhat surprising is that although its produced policies still have many deviations from the optimal one, especially when double-down and split are concerned, the Q-learning agent performed virtually the same as the optimal one in 100K games with regard to the winning rates. The optimal agent out-performed in terms of rewards with a total loss of 485.5 units vs 606.5 units of the trained RL after 100K games.

In real Blackjack games, assume each bet is 1$, if a person can lose just about 500$ after 100K games just like this RL model, I think that is really a good result.

## 5 Conclusions and Future work

In this project, I trained an RL agent to play Blackjack where Double-down and Split actions are also supported.

Comparing the produced policies against the optimal one showed that were able to obtain almost the optimal policy when only Hit and Stand actions are considered. When also considering Double-down, the best obtained policy still left much to be desired with 3 and 14 deviations from the optimal policy for hard and soft count, respectively. For Split policy, the best obtained policy was the worst among trained policies: with 27 deviations from the optimal one, which is equivalent to a 27% error rate. This means that for Split action, further experimentations were needed, for both Q-learning and DQN.

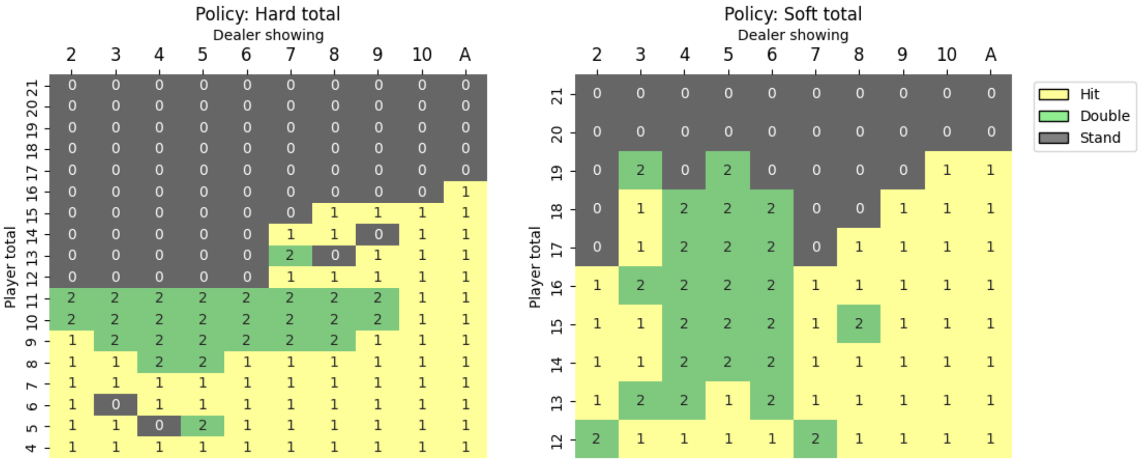
For all cases, the best policy was obtained by Q-learning while DQN could only obtain one best policy, jointly with Q-learning, for the case of a soft hand where only Hit and Stand actions were considered. For DQN, there are still a lot to be experimented with, especially with regard to the customisations that were done for Q-learning but not yet available for DQN. Another important hyper-param to explore for DQN was its network structure, whose default was a 2 hidden 64x64 feed-forward network.

In real-life Blackjack, the current environment also need to be extended or modified to support: a finite n-deck shoe composition (instead of the current infinite-deck setting), insurance and surrender options, as well as an option for dealers to stand on soft 17. When a finite deck composition is used (usually 6 or 8 decks), card counting then becomes possible. Dynamic betting strategy, i.e., using different amounts for different games, is also another good enhancement that goes together with cards counting.

It would be interesting to see how much further we can improve this agent with these optimizations and when the environment gets closer to a real-life one. These are left for future work.

## Appendix A. Evolution of policies by Q-learning

### Episode 100K



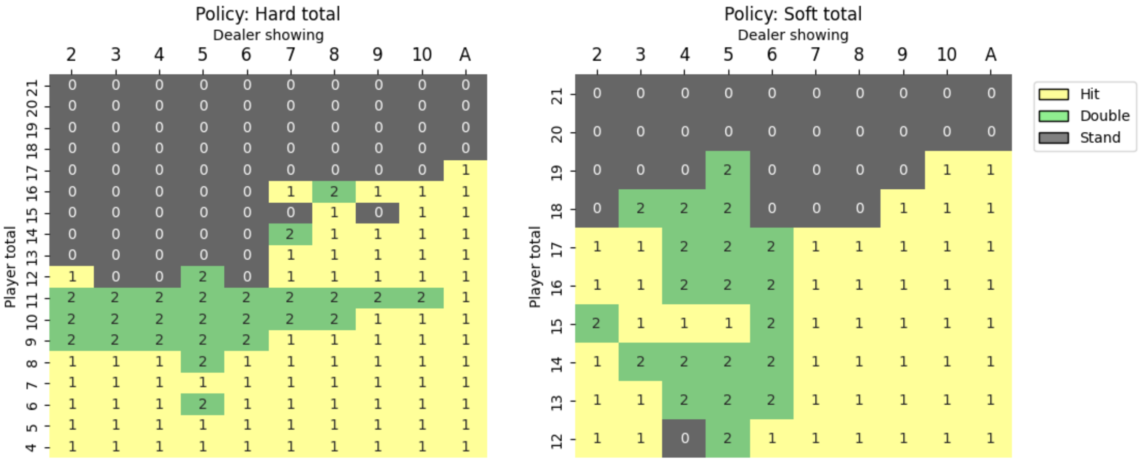
Policy: Hard total

| Player total | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 |
| 12 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 9 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 8 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| Player total | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 1 | 2 | 2 | 2 | 0 | 0 | 1 | 1 | 1 |
| 17 | 0 | 1 | 2 | 2 | 2 | 0 | 1 | 1 | 1 | 1 |
| 16 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 12 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 |

### Episode 200K



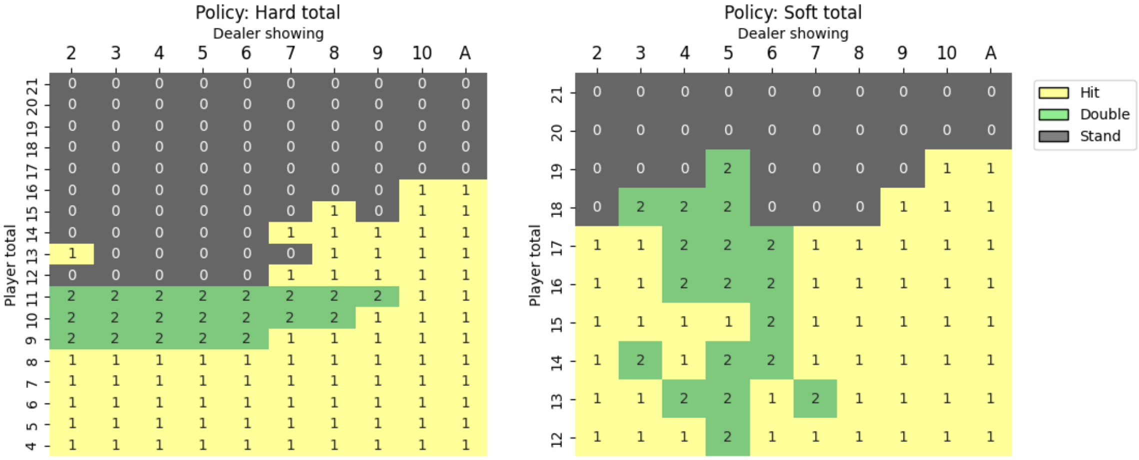
Policy: Hard total

| Player total | Dealer showing 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 16 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| Player total | Dealer showing 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

## Episode 300K



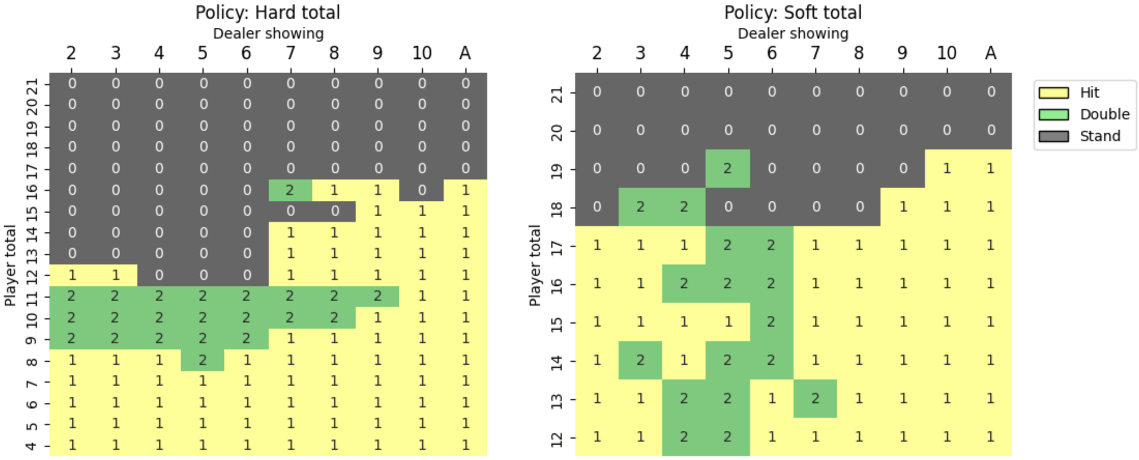
Policy: Hard total

| Player total | Dealer showing 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 12 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| Player total | Dealer showing 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

## Episode 400K



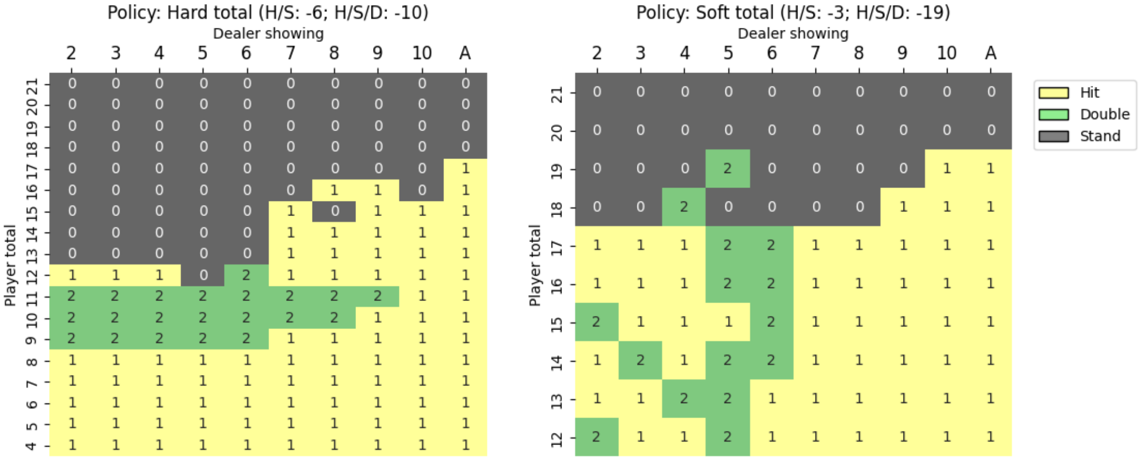
Policy: Hard total

| Player total | Dealer showing 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 0 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| Player total | Dealer showing 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

## Episode 500K



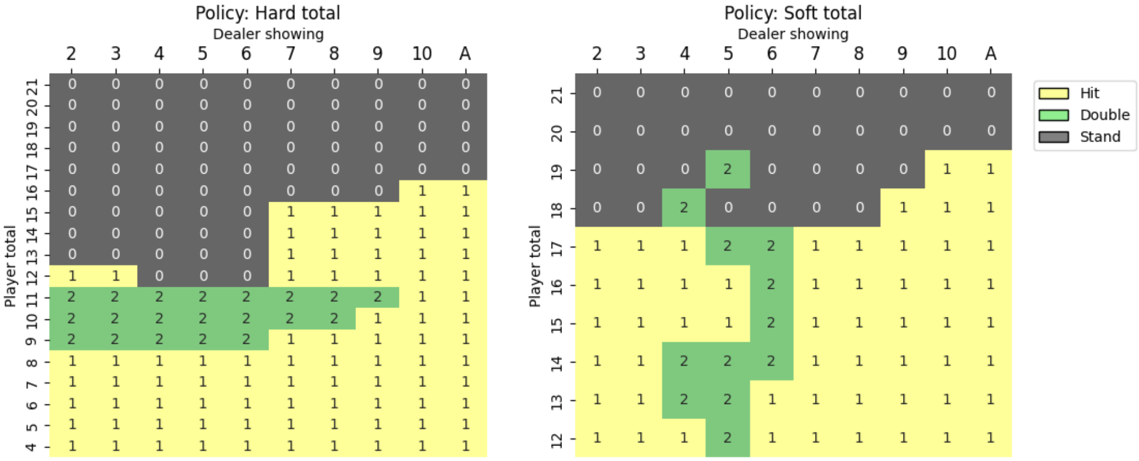
Policy: Hard total (H/S: -6; H/S/D: -10)

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 0 | 2 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total (H/S: -3; H/S/D: -19)

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

## Episode 600K



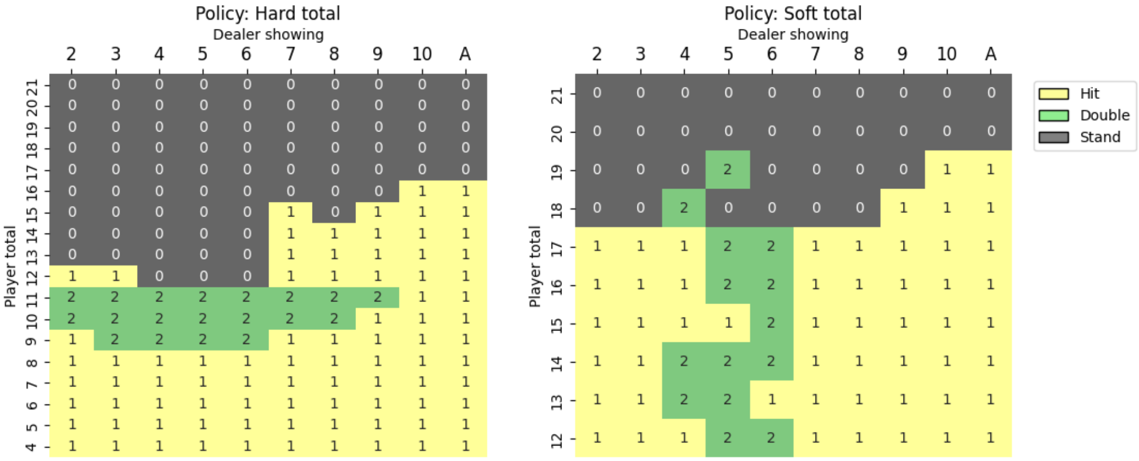
Policy: Hard total

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

## Episode 700K



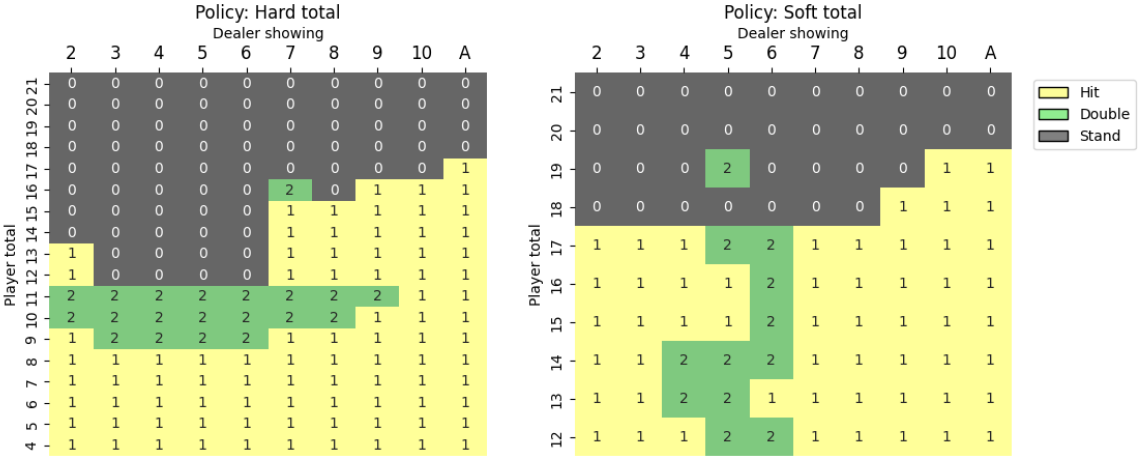
Policy: Hard total

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |

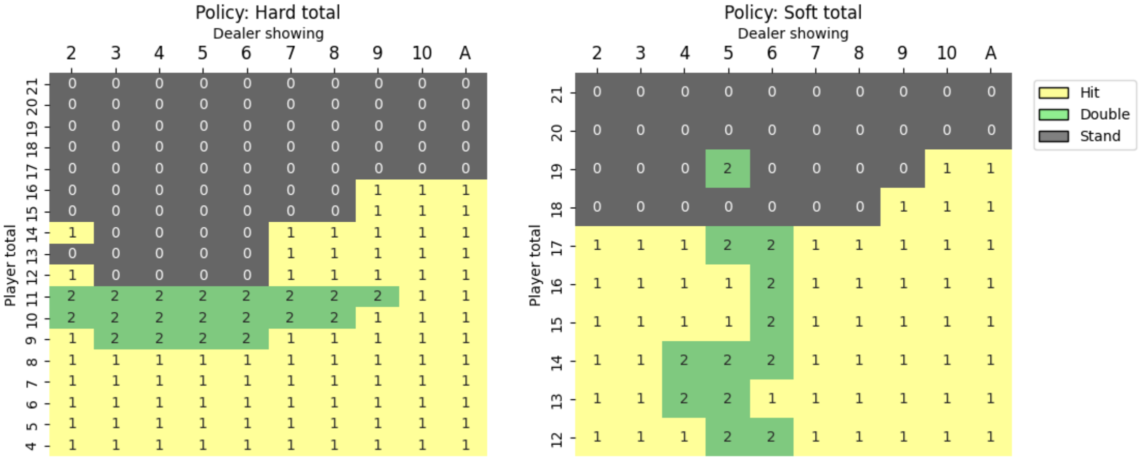
## Episode 800K



|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 16 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |

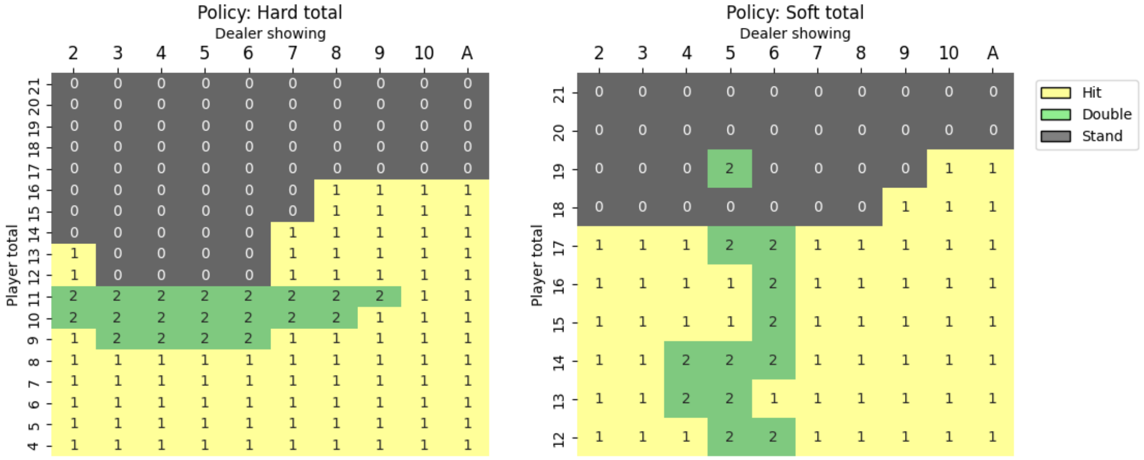
## Episode 900K



|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 14 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |

### Episode 1mil



Policy: Hard total

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 9 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Policy: Soft total

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 17 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |

## References

Blackjack Apprenticeship. Learn blackjack strategy, April 2024. URL `https://www.blackjackapprenticeship.com/blackjack-strategy-charts/`.

Roger R Baldwin, Wilbert E Cantey, Herbert Maisel, and James P McDermott. The optimum strategy in blackjack. *Journal of the American Statistical Association*, 51(275): 429–439, 1956.

Charles De Granville. Applying reinforcement learning to blackjack using q-learning. *University of Oklahoma*.

Farama Foundation. Blackjack — gymnasium, March 2024. URL `https://gymnasium.farama.org/environments/toy_text/blackjack/`.

Joshua Geiser and Tristan Hasseler. Beating blackjack - a reinforcement learning approach, February 2024. URL `https://web.stanford.edu/class/aa228/reports/2020/final117.pdf`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Daochen Zha, Kwei-Herng Lai, Songyi Huang, Yuanpu Cao, Keerthana Reddy, Juan Vargas, Alex Nguyen, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: a platform for reinforcement learning in card games. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 5264–5266, 2021.