How to get escorted out of the casino, with Reinforcement Learning

Kieran Rudd,[1, *] Michael Gamston,[1, †] and Scott Underdown[1, ‡]

[1] *School of Physics and Astronomy, University of Nottingham, Nottingham, NG7 2RD, UK*

(Dated: February 3, 2025)

Note: use plain text.
Can prob just use AI to assist in this once done.
Any acronyms ?

I. INTRODUCTION

Blackjack, also known as twenty-one, is the most widely played casino game in the world, largely due to its simple game structure in which a player attempts to get the highest score by drawing cards from a deck. With its long history, various strategies for increasing a players chance of winning, such as card counting, are commonplace. Whilst an optimum strategy for blackjack has been known by statisticians for decades, by drawing on the expected value of future cards [1], training an intelligent agent to learn the optimum strategy is an approach not as often taken. Using intelligent agents like this is known within machine learning communities as 'reinforcement learning'. The advantage of taking a reinforcement learning approach in this circumstance is rooted in the stochastic nature of blackjack, in which long term reward is prioritized. Additionally, its theorem proving nature offers extra insight, i.e., it can validate or indicate mathematical frameworks behind a system [2].

Reinforcement Learning is a subfield of machine learning concerned with teaching an 'agent' to find an optimal set of moves - optimal policy - in the context of a wider system. The foundational components compose a policy - mapping possible states to possible actions, a reward signal - defining the agent's goal, an environment - which the agent interacts with, and a value function - indicating the long-term desirability of a sequence of states. Probabilistic reinforcement learning incorporates uncertainty into the decision-making process. Markov Decision Processes (MDP) inherently involve probabilities by modelling state transitions and reward distributions. A reinforcement learning task is considered a MDP if it satisfies the Markov property, defined as the exclusive reliance of the likelihood of changing to a specific state on the present state and elapsed time, and not on previous states [3].

There are many variations to Blackjack, typically involving multiple players and a dealer, but for the purposes of this project a stylized version with a single player and a passive dealer was played. The game uses a standard deck of cards with numerical values equal to their number for cards 2-10, and equal to 10 for Jacks, Queens, and Kings. Aces are valued at 11 unless the sum of the cards in hand exceeds 21, in which case they are valued at 1. The sequence of play is as follows:

1. A card is dealt to the player with value C_1.
2. For n iterations, or until a total score of 21 is exceeded, the player can make one of two choices;
   (a) Stick, and end the game.
   (b) Hit, and receive another card with value C_{n+1}.
3. The final score is calculated using

$$S = \begin{cases} (\sum C_n)^2 & \text{if } \sum C_n \leq 21 \\ 0 & \text{if } \sum C_n > 21 \end{cases} \tag{1}$$

To approach this problem, two situations were considered. Infinite, in which the pile of cards being drawn from is infinite, meaning the probability of each card being drawn is equal, and finite, in which the pile of cards being drawn from is finite, meaning unequal probabilities. These two approaches were taken to provide a broad spectrum of results. This paper details the methodology taken for both problem situations, the results of each in context of an optimal result, and a conclusion on the efficacy of this approach.

II. METHODOLOGY

In training an agent to play Blackjack, an iterative Q-Learning approach was taken. Watkins' Q-Learning aims to learn the optimal 'q-value' for given state-action pairs in an environment, i.e., the respective value of making a certain move in a certain environmental state. This approach was chosen because it is a model-free, value-based, off-policy algorithm. Model free means that the agent does not build an understanding of its environment's functionality, value based means that the algorithm estimates the value of each state-action pair, and off-policy means that the optimal policy need not be followed. Q-learning solves problems modelled as MDP, meaning the probabilistic nature of Blackjack was realized by the model. The Bellman's equation for Q-

[*] efykr2@nottingham.ac.uk
[†] ppxmg5@nottingham.ac.uk
[‡] ppxsu1@nottingham.ac.uk

Learning is

$$Q_{new}(s,a) = Q_{old}(s,a) + \alpha(\underbrace{R(s,a) + \gamma MaxQ(s',a') - Q_{old}(s,a)}_{\text{Temporal Difference}}). \quad (2)$$

Where s, a are the current state and action, s', a' are the next state and action, α is the learning rate, γ is the discount factor, $R(s,a)$ is the reward received after taking action a in state s, and $Q(s,a)$ is the q-value for the state-action pair. The highlighted temporal difference is used to update an estimate based on other estimates, without waiting for a final outcome (known as bootstrapping) [3]. The learning rate α decayed throughout training with the aim of reducing probability induced learning oscillations. It took the equation,

$$\alpha = \alpha_{min} + (\alpha_{max} - \alpha_{min})e^{-\frac{1}{dt}} \quad (3)$$

For respective values for the minimum and maximum learning rate, decay rate d, and number of iterations over the training process t. The reward $R(s,a)$, passed to the agent to drive learning, was defined by

$$R(s,a) = S^2_{new} - (S^2_{old} - S^2_{new})\delta_{Ace} \quad (4)$$

Where $\delta_{Ace} = 1$ when an ace drops in value from 11 to 1, and $\delta_{Ace} = 0$ otherwise. This is used to proportionally penalize the agent for using an ace since it is less advantageous to hold an 1-valued ace than a 11-valued ace.

Q-tables encapsulate the state-action pairs, entirely defining all possible moves in the system with their respective Q-value. Each action taken recalculates respective state-action Q-values using equation 2. This converges towards a terminal value which, for probabilistic situations like this, is the expected sum of future rewards. From this, the optimal policy was determined by selecting the action with the highest q-value for each state.

Since an agent follows the optimal policy where possible, exploiting and not exploring new information, an Epsilon-greedy algorithm was employed. This intends to explore less taken state-action avenues $X\%$ of the time.

A. Infinite

In the 'infinite' situation, the probability of each card being drawn was equal, meaning that the expected value of the next drawn card remained constant, so retaining prior knowledge of cards drawn posed no advantage to the agent. Therefore, the environment was composed of two state variables - current score and usable ace - and one action variable - hit/stick.

B. Finite

In the 'finite' situation, cards were drawn from a pile of finite number, meaning that the probability of drawing respective cards changed as the game progressed. This introduced a challenge that would ideally have been addressed by providing the agent with all previously dealt cards, where the agent would have learned to predict the probabilities of new cards and adjust its play strategy accordingly. However, incorporating all previous cards into the agent's calculations comes at a significant computational cost.

To balance accuracy with computational efficiency, the probability of losing was integrated into the Q-table (a concept from reinforcement learning where Q-values represent the expected utility of taking an action). This approach was chosen over, say, card counting because it provided a more accurate representation of the environment while maintaining relatively low complexity. Since these probabilities could have been deduced from previously dealt cards, providing the agent with this essentially skipped the step of learning probabilistic expectations.

To implement this continuous probability into the Q-table of finite dimension size, the probabilities were discretized, i.e., continuous probabilities were separated into bins of size 2%. Therefore, the environment was composed of three variables - current score, usable ace, and probability of losing - and one action variable - hit/stick.

III. RESULTS & DISCUSSION

The training process for each model was evaluated using a cumulative average of the score, as illustrated in Fig. 1 and Fig. 2. The infinite situation converged after approximately " iterations, which is a considerable amount less than for the finite situation, with " iterations. This was a direct result of the difference in state spaces, where the finite situation required more iterations to converge its larger q-table. The two systems converged to similar cumulative average scores. This is notable because it indicates that the probability dimension in the finite q-table effectively managed inconsistent probabilities.

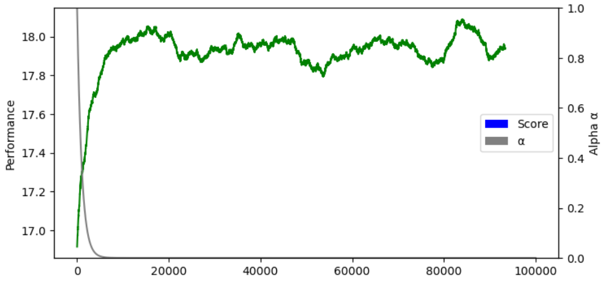

FIG. 1. Agent's training curve and learning rate in the infinite situation

The expected value of the next card in the infinite situation is 6.54, which is approximately the difference be-

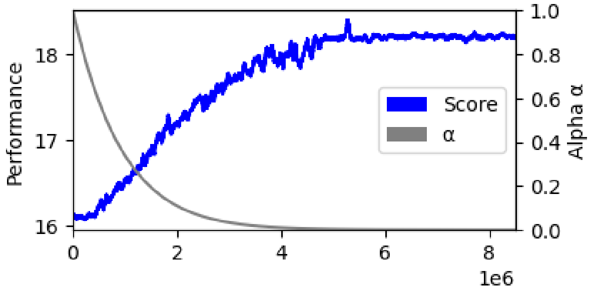

FIG. 2. Agent's training curve and learning rate in the finite situation.

tween 21 and its converged value. This validates the system, because the algorithm wants an even split of probability????

Stick values in each Q-table quickly converged to their value squared, as defined by equation 4, the reward equation. This was because the agent saw no future maximum value. In contrast, the Hit values in the Q-table took longer to converge. This was due to their probabilistic nature, in which, enough cards had to be dealt for

Optimal hyperparameters were selected for each situation via a grid-search approach, in which a selection of hyperparameters were tested and compared. Following this, Gamma γ was defined as 1 because this would ensure that the future maximum reward was as valuable as current reward. Similarly, an optimum Epsilon ϵ was defined as 0.1. Alpha decay rates, however, differed between the two systems, with infinite using an optimum of 0.01 and finite using 0.08.

The resulting optimal policies π^* are plotted in Fig. 3 for the infinite situation and Fig. 4 for the finite situation. The infinite policy hits until 14 with no unused ace, and until 17 with an unused ace, which differs from the statistical stance of sticking on 17 irrespective of the held cards. The differences between this policy and the statistical optimum is due to the squared reward defined in 1 and the absence of an active dealer. This drives the agent to be adverse to risk, preferring to receive a definitive reward.

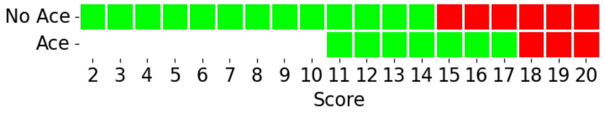

FIG. 3. Optimal policy for the infinite situation, with green indicating hit, and red indicating stick.

The finite policy, in contrast, did not fully converge due to some states being very rarely entered, such as a 30% chance of losing with a current score of 20. These values are missing in the respective Q-table. Despite this, commonly entered states developed a policy exhibiting logical patterns - sticking when loss is likely, and hitting when loss is unlikely. Additionally, the policy follows a diagonal split. This is because with a low card count, there is a higher future maximum value, meaning that even with a high chance of losing, the agent is willing to be

The rarity of entering regions in the top right and bottom left was due to The inequality between these areas was due to 4 × 10 valued cards.

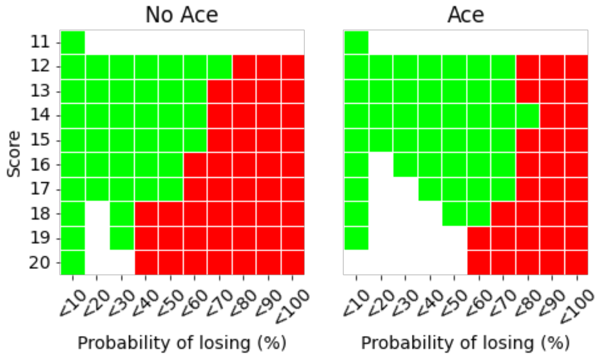

FIG. 4. Optimal policy for the finite situation, with green indicating hit, and red indicating stick. This model was trained over 400000 iterations and 2 decks. Probability bins of 10% increments are indicated accordingly. Missing policy points did not converge during training.

IV. CONCLUSIONS

This paper presents a reinforcement learning approach to solving Blackjack, with considerable attention paid to the games probabilistic nature. Watkin's Q-Learning was employed with a reward relative to the square of the card, for situations with constant and changing probabilities.

The training processes converged after

Optimal policies were created which, with no usable ace, stopped hitting at 15 and, with a usable ace, stopped hitting at 18. These values contrast a statisticians optimum.

[1] H. M. Roger R. Baldwin, Wilbert E. Cantey and J. P. McDermott, Journal of the American Statistical Association **51**, 429 (1956).

[2] K. A. Bidi, J.-M. Coron, A. Hayat, and N. Lichtlé, Reinforcement learning in control theory: A new approach to mathematical problem solving (2023), arXiv:2310.13072 [math.OC].

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An*

Introduction (A Bradford Book, Cambridge, MA, USA, 2018).