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 2, 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. A reinforcement learning task is considered a Markov Decision Processes (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. MDPs are used in sequential decision-making in probabilistic systems [3]. WRITE MORE ABOUT PROBABILISTIC REINFORCEMENT LEARNING.

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

Training of the models

The resulting optimal policies are plotted in III for the infinite situation and III for the finite situation.

Optimal hyperparameters were selected for each situation via a grid-search approach, in which a selection of hyperparameters were tested and compared. These hyperparameters are indicated in respective figures. epsilon =0.1 gamma = 1 alpha decay = varied.

Speak about convergence in each situation via graph

Speak about policies

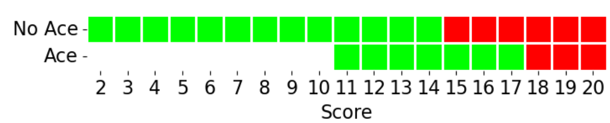

FIG. 1. Optimal policy for the infinite situation, with green indicating hit, and red indicating stick.

Ideas; * policy * ideal policy (needs mathematical modelling) * learning rates (and alpha rates?) : absolute and relative changes - derivatives * differences between held ace and no held ace * actual score (moving average?)

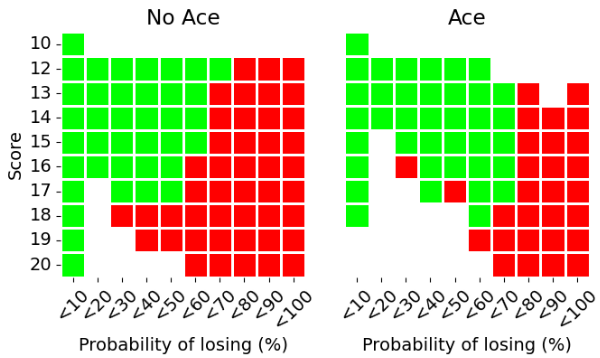

FIG. 2. Optimal policy for the finite situation, with green indicating hit, and red indicating stick. Probability bins of 10% increments are indicated accordingly.

IV. CONCLUSIONS

...4 pages, not including references. Allow space for abstract. Check: every bit of technical lingo is explained.

This paper successfully presents a Q-learning, reinforcement learning approach to blackjack's optimal policy - the set of best rewarding moves in game state. blah blah

[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).