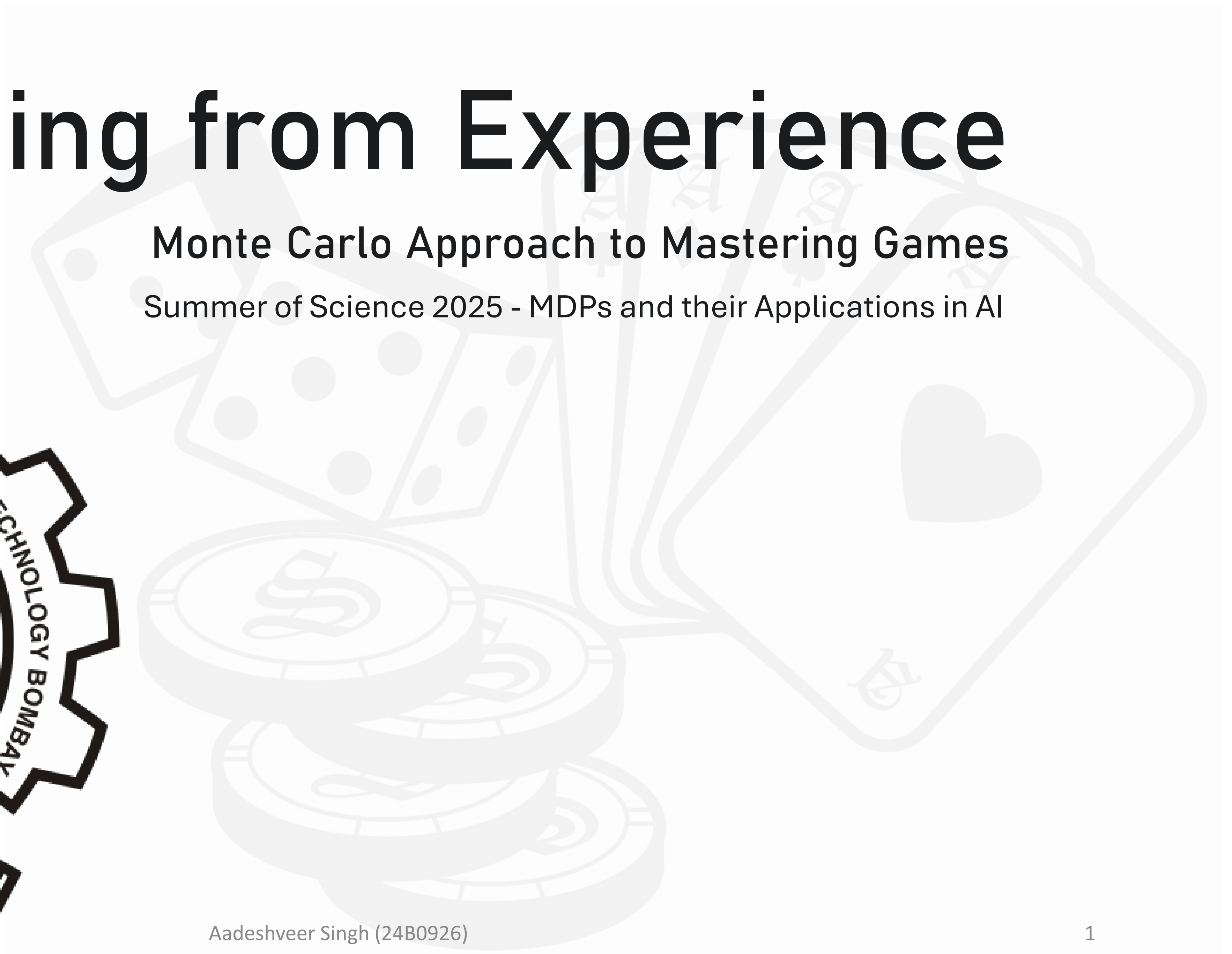
# Learning from Experience

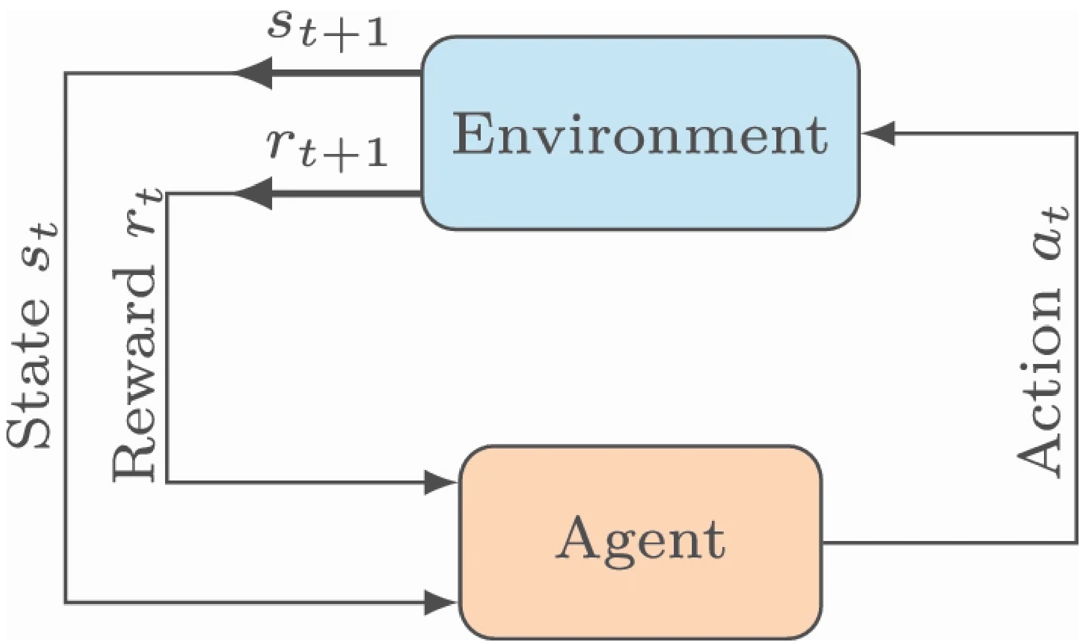
## Monte Carlo Approach to Mastering Games

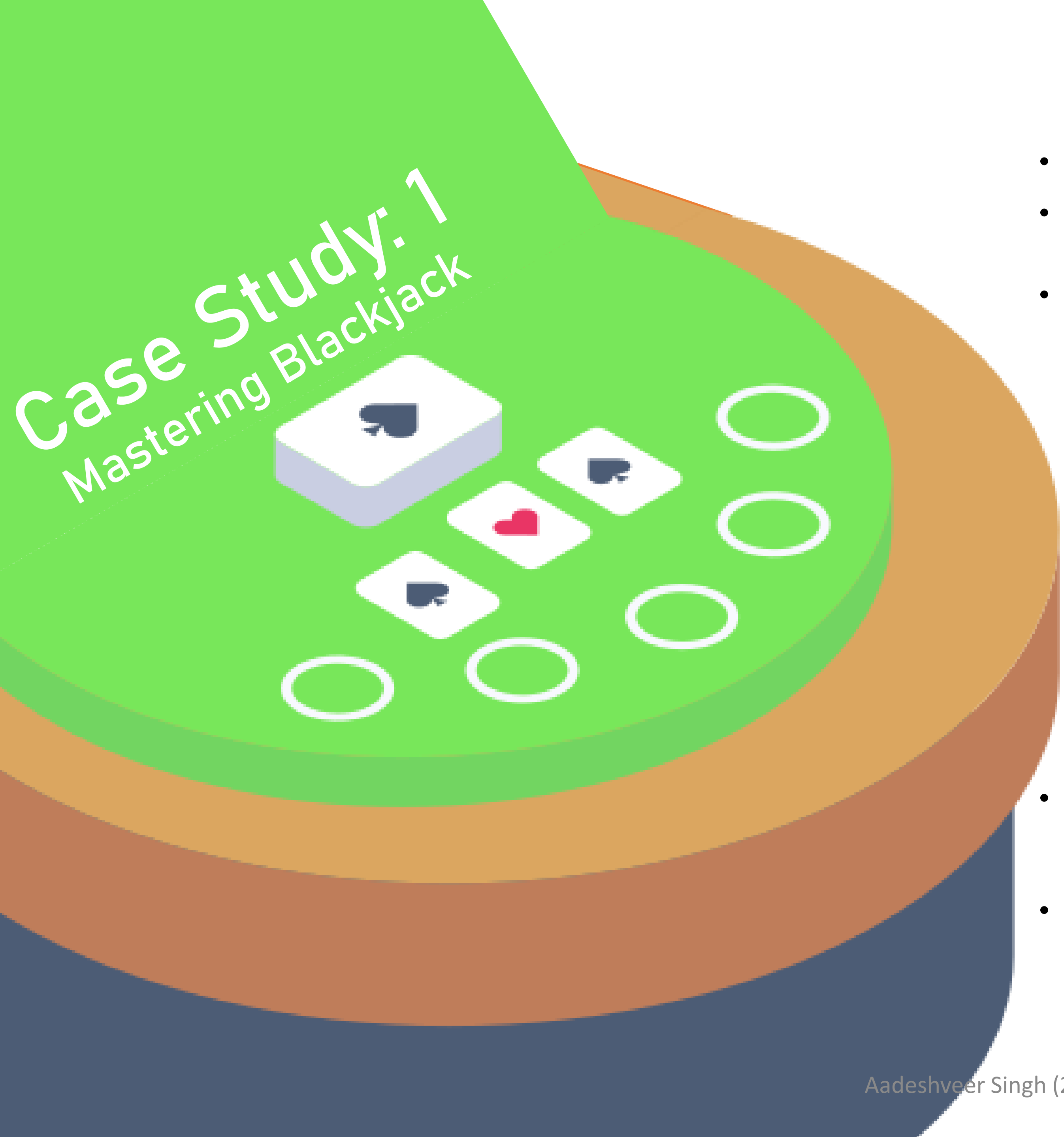
Summer of Science 2025 - MDPs and their Applications in AI



# What are Monte Carlo methods

- **Reinforcement learning** is a type of machine learning quite distinct from other two types(namely supervised and unsupervised). It tries to mock natural animal learning by letting the agent select actions and providing a reward or penalty feedback of how effective its action was. The agent then learns from the feedback and improves its policy.
- The environment is usually given by a probability function that decides the reward and new state based on present state and action chose($\boldsymbol{p(s', r|s, a)}$).Having access to this functions allows us to get the absolute optimal policy by theoretically proven methods. However **usually we don't have access to this function** and in such cases the absolute method fails.
- In case we don't have access to environment dynamics model free learning comes into play and Monte Carlo is one of these methods.
- **Monte Carlo** refers to the method of generating sample runs in the environment and training the agent based on the experience.

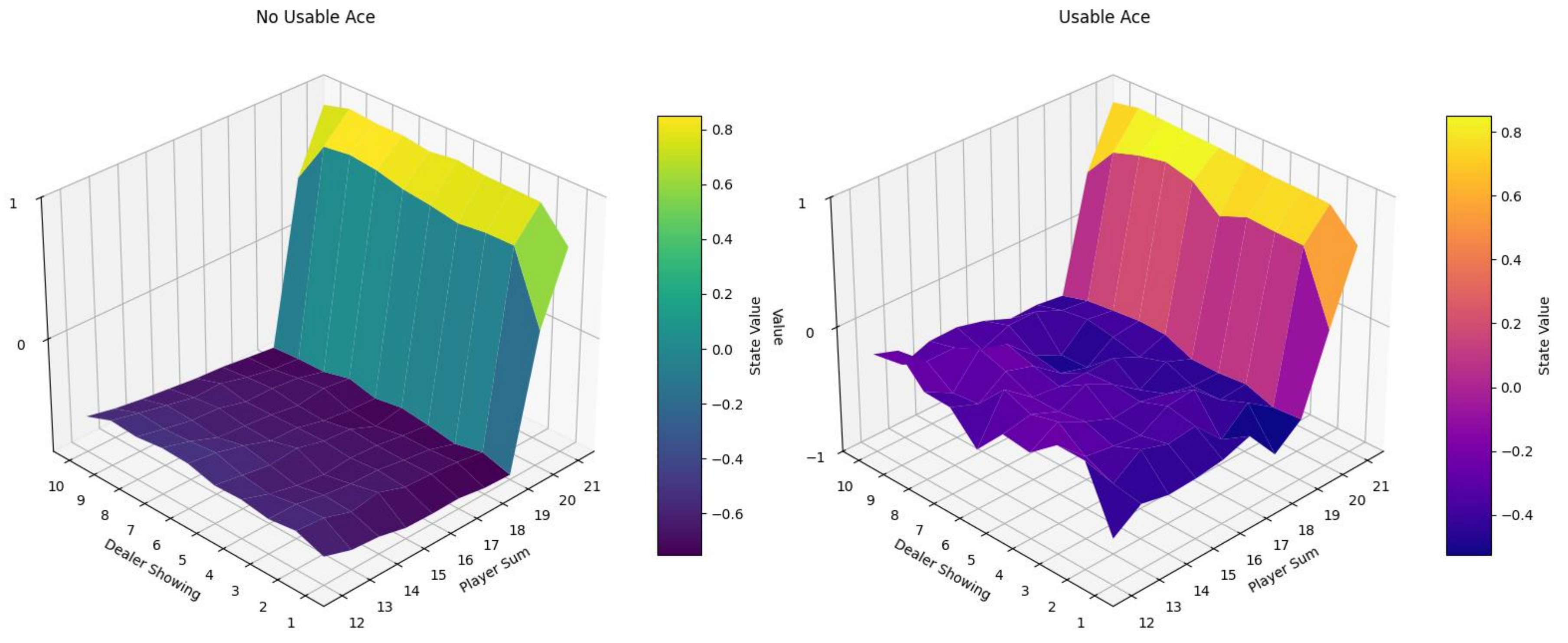




- Blackjack is a popular casino card game.
- Many variants of blackjack are played throughout the world. In this case study a very basic variant of game is chose.
- Basic rules:
  - At beginning the player and dealer are dealt 2 cards each.
  - The dealer has one card hidden and one card face up.
  - Aim of the player is to have sum of his cards to be as close as possible to 21 and not above it.
  - The value of every numeric card is its number itself, every face card is of value 10, and an Ace can be counted as 11 or 1 based on whichever is more profitable to the holder.
  - The player can choose to hit(draw another card) or stand.
  - I sum of player's hand exceeds 21 the player loses and is said to have busted.
  - Once the player stands, the dealer draws cards until the sum in his hand is less than 18.
  - I the dealer busts the player wins. Otherwise, the person with sum closest to 21 wins.
- Other variants also include insurance, doubling down, splitting and naturals. In order to keep the game simple for simulation these have been ignored.
- Again, A real blackjack game is played with a finite deck of cards giving advantage to a person who counts cards. Again, to keep it simple we will assume infinite decks i.e. all cards have equal probability of draw.

## MC Prediction

### Evaluating a Simple Strategy

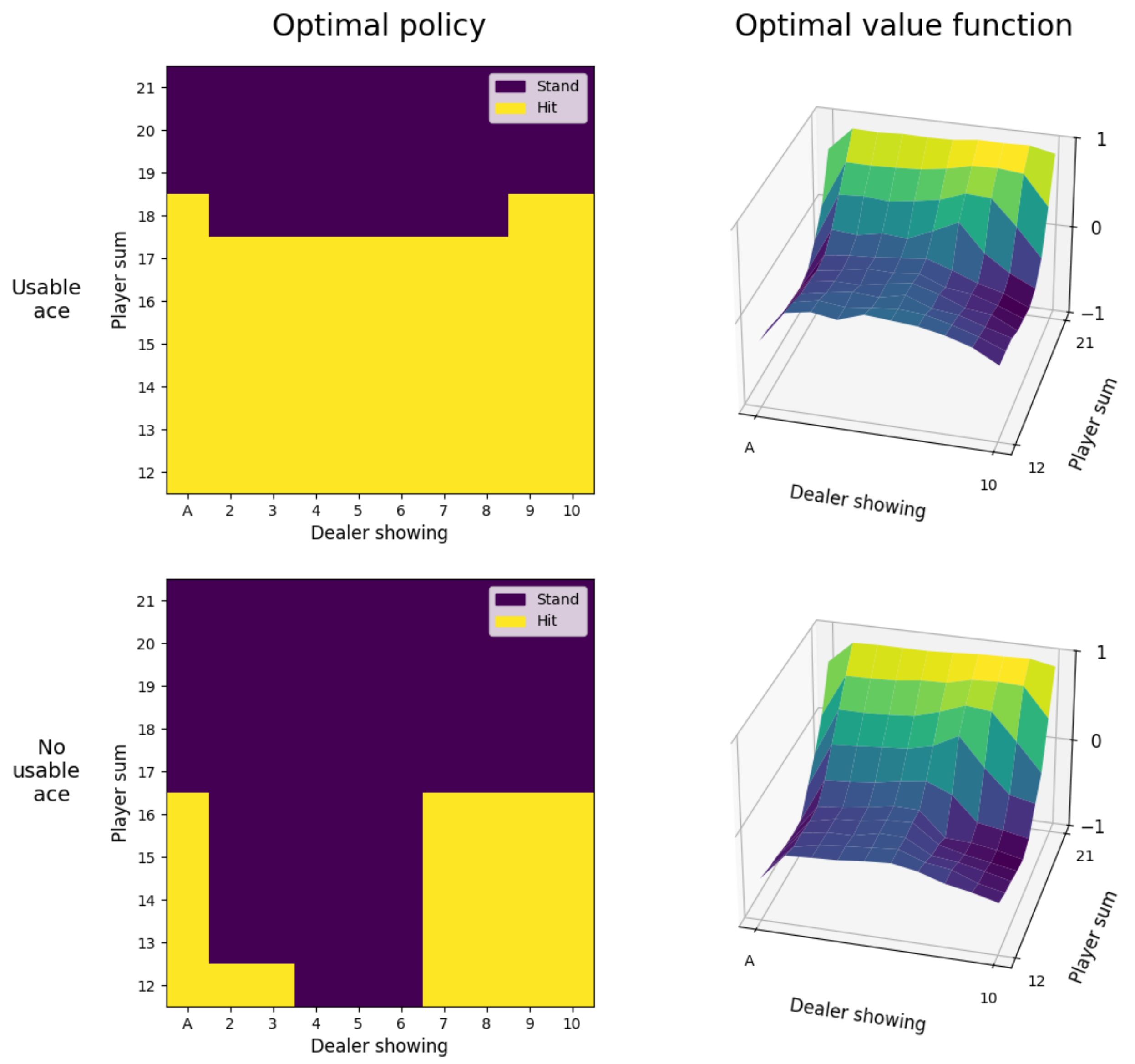


- The first step was not to jump directly to control and find winning strategy but to actually evaluate a given policy.
- The state of the Markov chain was designed to be a set of value of card dealer is showing, sum of cards in hand of player and if the player has a usable ace.
- The policy evaluated was to always hit unless sum is 20 or 21.
- It involved generating 100,000 games strictly following the given policy to gather experience.
- The evaluation converged to the one given in the figure, as the event of game reaching a state with usable ace is rare the value function is bumpier.
- Another observation is that the values are high at 20 and 21 owing to high probability of winning, however the values keep reducing as player sum increases as the player is force to hit at 19 where the probability of bust is large.
- Interestingly the values decrease substantially near the cases where dealer is showing a 10 or Ace as in such case dealer may win easily or may have a usable ace.

# MC Control

## Finding the Optimal Strategy

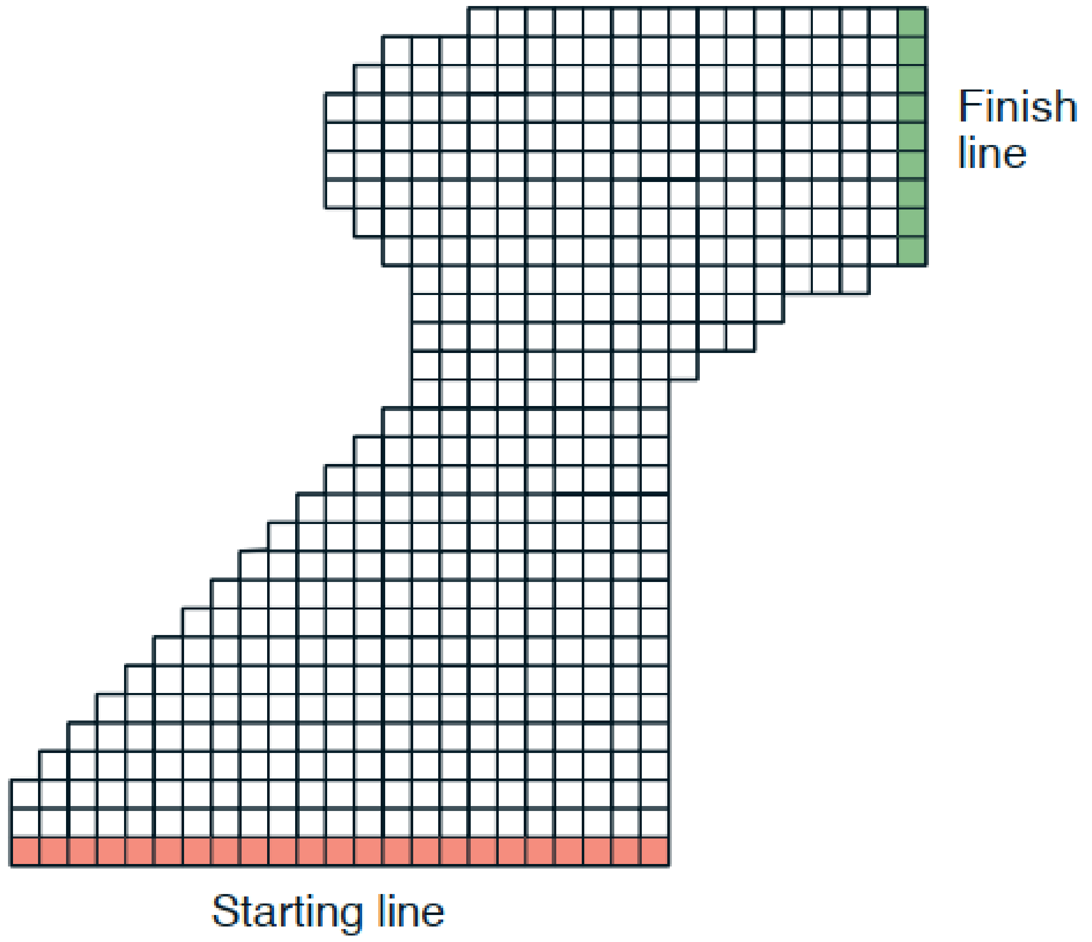
- Now that the prediction problem has been solved, we move on to the control problem, to find optimal strategy to play the game.
- The method used here is exploring starts. My initial idea was to draw random cards but it was not able to cover the entire state space as some states were very rare to reach.
- Thus, a new idea was applied to force start the game in every of 400 states at least 2000 times.
- Hence, after 800,000 episodes the agent learnt the optimal policy in this case as shown in the figure.
- It can be observed that the player learns to take chances if it has a usable Ace.
- Also, the agent learns that if the dealer shows a big card like Ace, 10 or 9, the best chances to win will be possible with a big sum and so it is worth taking the risk to hit at much higher sum.



# Case Study 2:

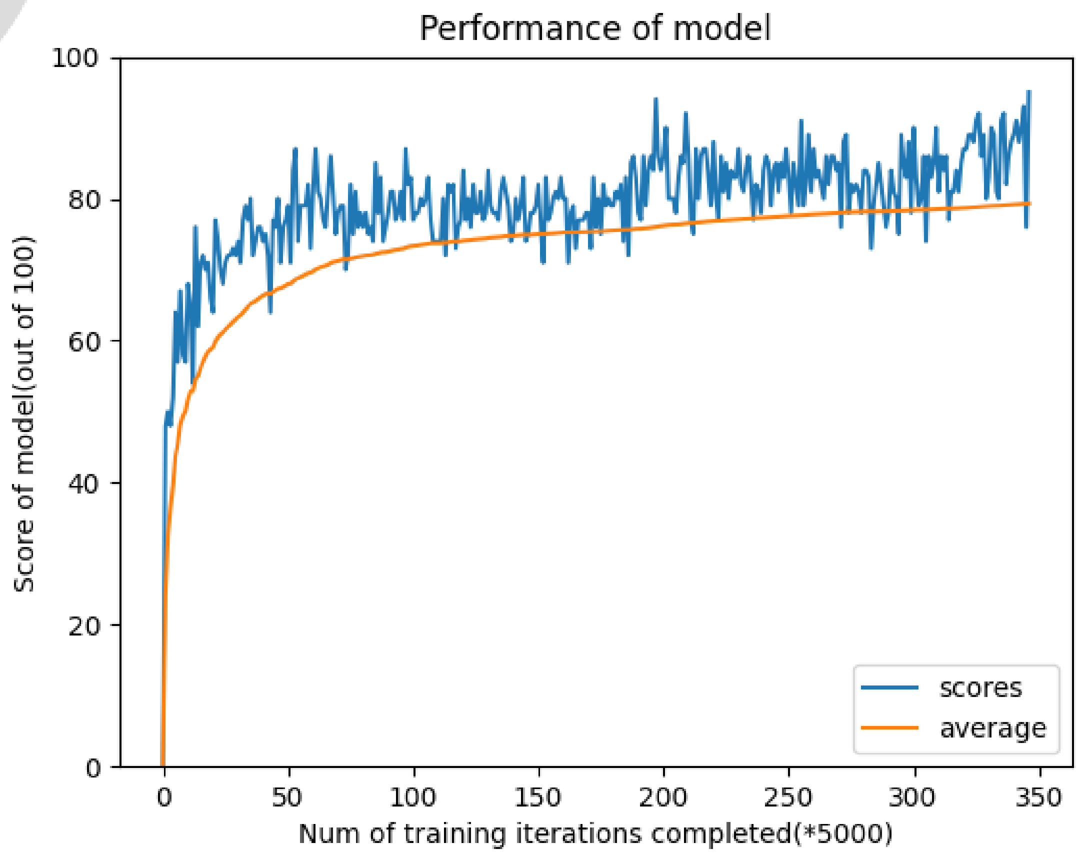
## The Racetrack Problem

- This problem is from Sutton and Barto, the agent must drive a car starting from the starting line and finish as fast as possible.
- The car this time however has momentum, i.e. the agent has control over the acceleration and not the velocity.
- The agent at a time step can accelerate in any of the four directions.
- This adds complexity to the problem as the agent must now understand momentum.
- The diagram shows a sample map as given in Sutton and Barto



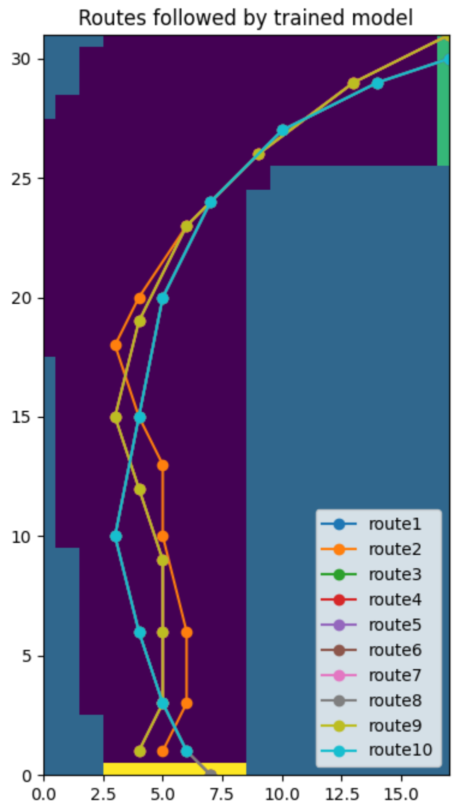
## Off-Policy MC Control for the Racetrack

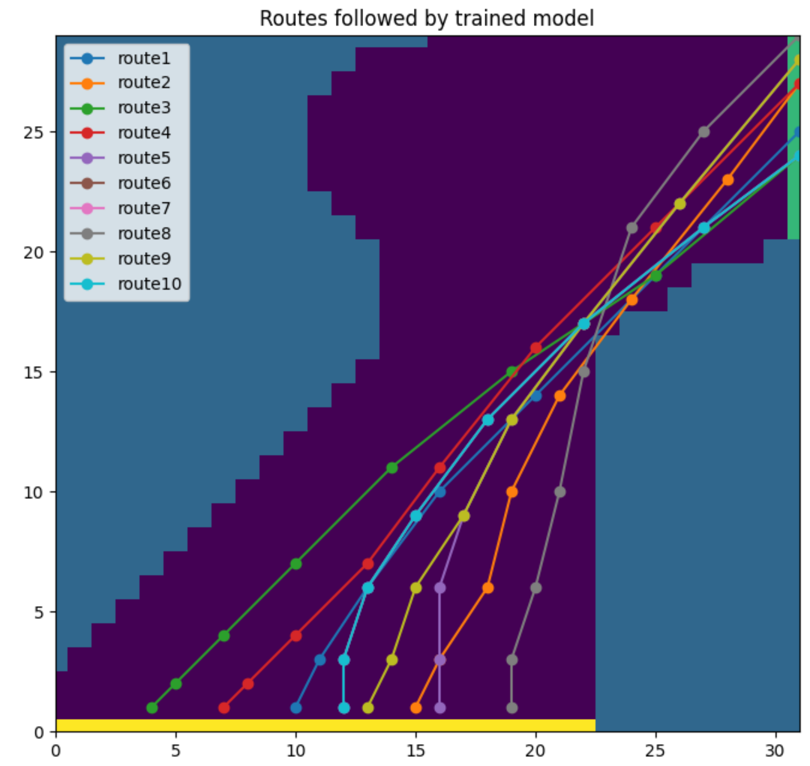
- Given the tricky situation the method used was simple, maintaining a state space of not just position but a set of position and velocity components in both directions.
- The agent was trained over millions of steps, Until It began scoring above a metric of 95. Meaning out of every 100 randomly generated test cases the model successfully reached the finished without hitting any wall.
- As you can see, its performance, measured by the success rate of completing a lap, steadily improves and converges at a high level, showing clear learning.
- The training was stopped when the agent scored >95 for first time and then some paths were generated to observe the training.



## The Result:
## The Learned "Racing Line"

- After training until the agent reached a 95% accuracy, the sample routes taken by the agent were generated for two different maps as shown in the diagram.
- It has learned a sophisticated policy. It accelerates on straightaways, decelerates just enough to navigate turns without crashing.
- In a sense it finds the optimal path around the track the 'racing line'. This is all learned purely from trial and error.



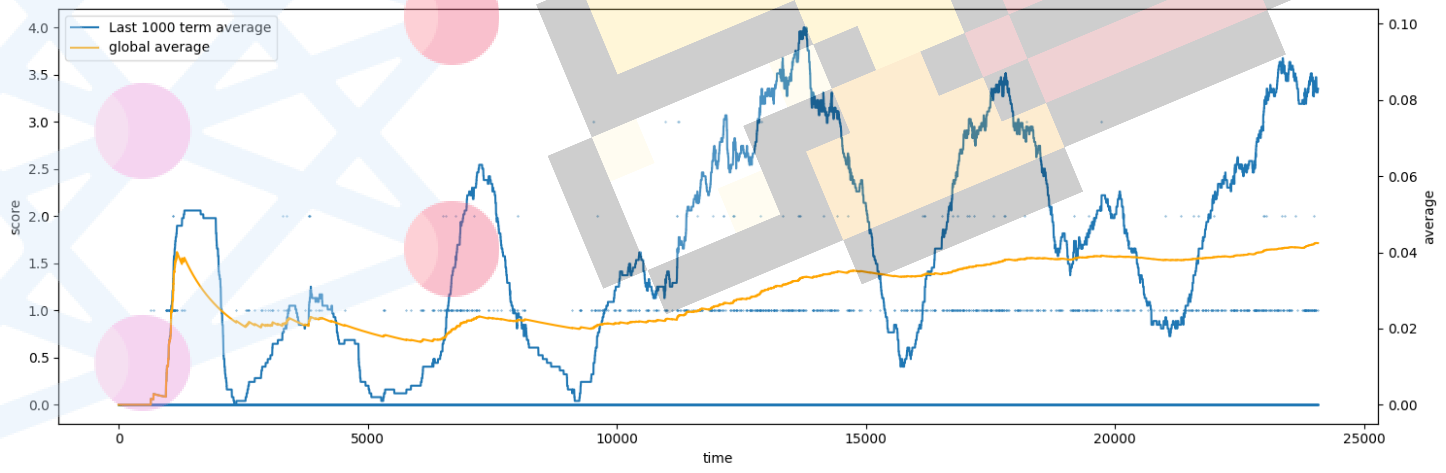


## Conclusion and What next?

Monte Carlo methods are a powerful way to solve problems purely from experience, finding many applications in real life. It allows us to find optimal strategies for complex tasks like Blackjack and Racetrack.

These methods are great for episodic tasks, but what if an episode is very long or never ends? This leads to other techniques like Temporal-Difference Learning and Deep Q-Networks, which I explored for my final Flappy Bird project.

Tackling the Flappy Bird problem highlighted the challenges of Deep RL, such as sample efficiency and catastrophic forgetting, providing valuable lessons for future work.



# Thank You