

Module 2

CSE3011 Reinforcement Learning

Credit Structure : 2-2-3



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Module 2 : Monte-Carlo (MC) methods

Topics : Monte Carlo methods, prediction and control tasks, Monte Carlo prediction: algorithm, types of MC prediction, examples, incremental mean updates, Monte Carlo Control: algorithm, on-policy MC control, MC with epsilon-greedy policy, off-policy MC control. Limitations of MC method.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- L01 : Understand MC method
- L02 : Understand 2 types of RL tasks : prediction and control
- L03 : Understand the advantages of MC over DP methods
- L04: Understand the different types of MC prediction
- L05: Apply MC prediction to train an agent to play the blackjack game
- L06: Understand the different types of MC control
- L07: Apply MC control to train an agent to play the blackjack game



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Introduction

- Model-free methods do not require the model dynamics of the environment to compute the value and Q functions in order to find the optimal policy.
- One such popular model-free method is the Monte Carlo (MC) method.
- The Monte Carlo method is a statistical technique used to find an approximate solution through sampling.

Introduction

- For instance, the Monte Carlo method approximates the expectation of a random variable by sampling, and when the sample size is greater, the approximation will be better.
- Let's suppose we have a random variable X and say we need to compute the expected value of X ; that is $E(X)$, then we can compute it by taking the sum of the values of X multiplied by their respective probabilities as follows:

$$E(X) = \sum_{i=1}^N x_i p(x_i)$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Introduction

- MC method, estimates the expected value of X by just sampling the values of X for some N times and compute the average value of X as the expected value of X as follows:

$$\mathbb{E}_{x \sim p(x)}[X] \approx \frac{1}{N} \sum_i x_i$$

- When N is larger the approximation will be better.
- Thus, with the Monte Carlo method, we can approximate the solution through sampling and our approximation will be better when the sample size is large.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Introduction

In reinforcement learning, we perform two important tasks, and they are:

- The prediction task
- The control task



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Prediction Task

- **The prediction tasks, take a policy π as an input and evaluate this policy, i.e to determine whether the policy is good or bad.**
- If the agent obtains a **good return** using the given policy then we can say that the policy is good.
- Thus, to evaluate the given policy, we need to understand what is the return the agent would obtain if it uses the given policy.
- To obtain the return, we **predict the value function or Q function using the given policy.**
- Hence, the policy is evaluated by predicting the value function or Q function using the given policy.

- The value function of a state gives the expected return starting from state s following the policy π .
- So, by predicting the value function following the policy π , we can understand what is the expected return the agent will obtain in each state if it uses the given policy π .
- If the return is good then the policy π is also good.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- The Q function of a state-action pair gives the expected return starting from state s and an action a following the policy π .
- So, by predicting the Q-function by following the policy π , we can understand what is the expected return the agent will obtain in each state-action pair if it uses the given policy π .
- If the return is good then the policy π is also good.
- **Prediction methods do not change the given policy.**



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Control Task

- Unlike the prediction task, in the control task, we will not be given any policy as an input. **In the control task, our goal is to find the optimal policy.**
- So, we will start off by initializing a random policy and we try to find the optimal policy iteratively.
- That is, we try to find an optimal policy that gives the maximum return.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo Prediction

- First, let's recap the definition of the value function of a state,
- The value function or the value of the state s is the expected return the agent would obtain starting from the state s and following the policy π .
- It can be expressed as:

$$V^{\pi}(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo Prediction

- But, in the Monte Carlo prediction method, the value of a state is calculated by sampling the episodes (trajectories) following the input policy π for N times.
- take the average return of a state across the sampled N episodes instead of taking the expected return.

$$V(s) \approx \frac{1}{N} \sum_{i=1}^N R_i(s)$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Grid world environment example

Our goal is to reach the state I from the state A without visiting the shaded states, and the agent receives +1 reward when it visits the unshaded states and -1 reward when it visits the shaded states:

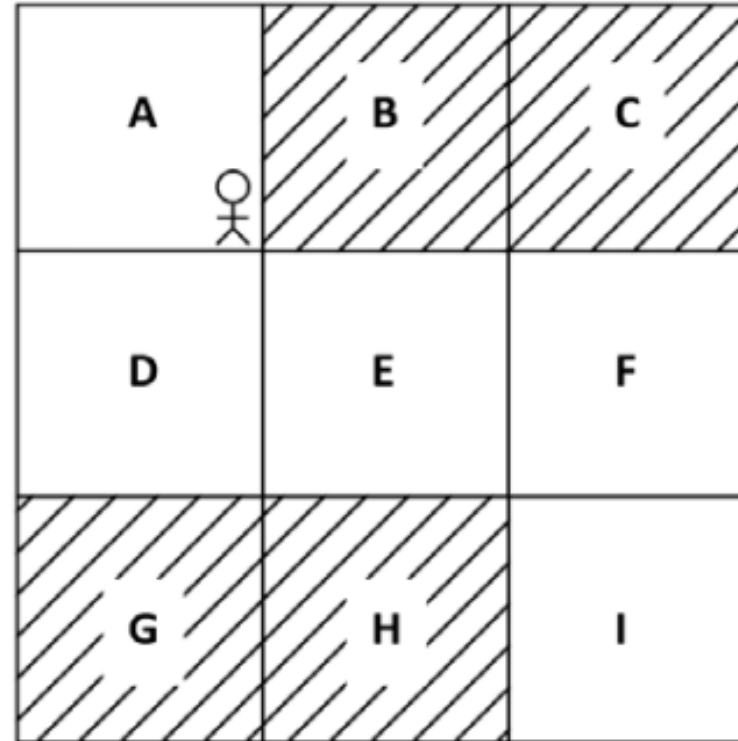


Figure 4.1: Grid world environment



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Grid world environment example

- Let's say we have a stochastic policy π . Let's suppose, in state A, our stochastic policy π selects action down 80% of time and action right 20% of the time, and it selects action right in states D and E and action down in states B and F 100% of the time.
- First, we generate an episode τ_1 using our given stochastic policy π as Figure shows:



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Grid world environment example

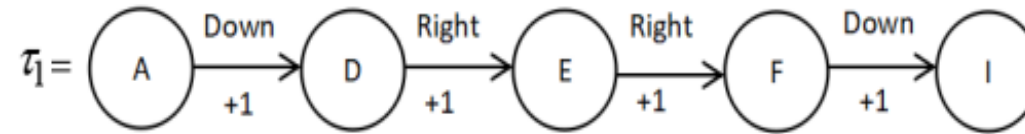
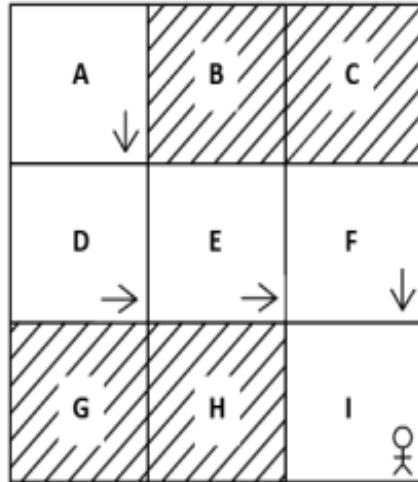


Figure 4.2: Episode τ_1



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Grid world environment example

For a better understanding, let's focus only on state **A**. Let's now compute the return of state **A**. The return of a state is the sum of the rewards of the trajectory starting from that state. Thus, the return of state **A** is computed as $R_1(A) = 1+1+1+1 = 4$ where the subscript 1 in R_1 indicates the return from episode 1.

Say we generate another episode τ_2 using the same given stochastic policy π as Figure 4.3 shows:

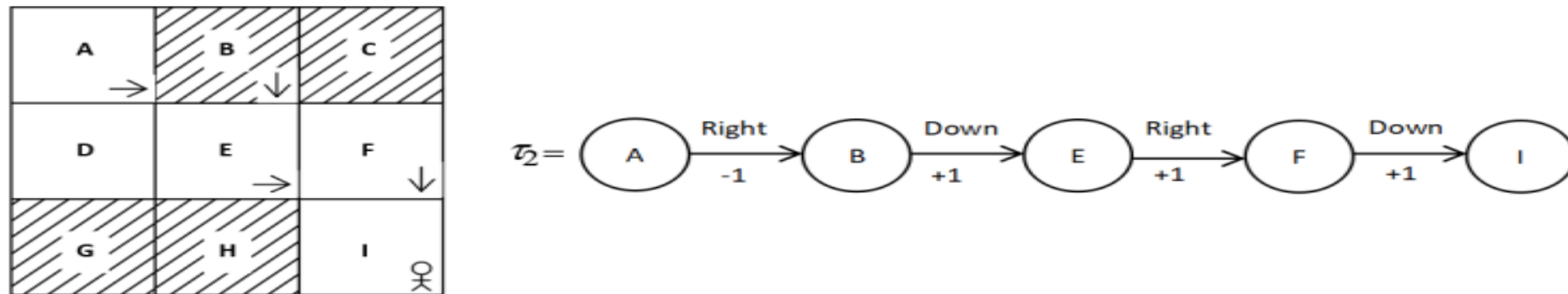


Figure 4.3: Episode τ_2

Let's now compute the return of state **A**. The return of state **A** is $R_2(A) = -1+1+1+1 = 2$.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Grid world environment example

Say we generate another episode τ_3 using the same given stochastic policy π as Figure 4.4 shows:

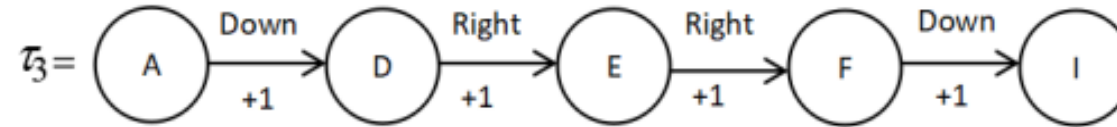
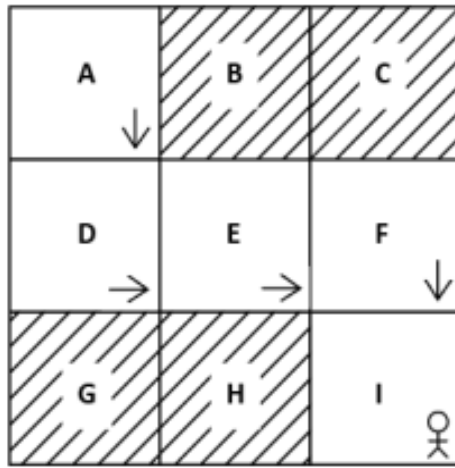


Figure 4.4: Episode τ_3

Let's now compute the return of state **A**. The return of state **A** is $R_3(A) = 1+1+1+1 = 4$.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Grid world environment example

The value of a state can be approximated by computing the average return of the state across some N episodes (trajectories):

$$V(s) \approx \frac{1}{N} \sum_{i=1}^N R_i(s)$$

We need to compute the value of state A , so we can compute it by just taking the average return of the state A across the N episodes as:

$$V(A) \approx \frac{1}{N} \sum_{i=1}^N R_i(A)$$

We generated three episodes, thus:

$$V(A) \approx \frac{1}{3} \sum_{i=1}^3 R_i(A)$$

$$V(A) = \frac{1}{3} (R_1(A) + R_2(A) + R_3(A))$$

$$V(A) = \frac{4 + 2 + 4}{3} = 3.3$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo Prediction

- In the Monte Carlo prediction method, to predict the value of a state (value function) using the given input policy π ,
 - we generate some N episodes using the given policy and
 - then we compute the value of a state as the average return of the state across these N episodes.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

The Monte Carlo prediction algorithm is given as follows:

1. Let $\text{total_return}(s)$ be the sum of return of a state across several episodes and $N(s)$ be the counter, that is, the number of times a state is visited across several episodes. Initialize $\text{total_return}(s)$ and $N(s)$ as zero for all the states. The policy π is given as input.
2. For M number of iterations:
 1. Generate an episode using the policy π
 2. Store all the rewards obtained in the episode in the list called rewards
 3. For each step t in the episode:
 1. Compute the return of state s_t as $R(s_t) = \text{sum}(\text{rewards}[t:])$
 2. Update the total return of state s_t as $\text{total_returns}(s_t) = \text{total_return}(s_t) + R(s_t)$
 3. Update the counter as $N(s_t) = N(s_t) + 1$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

3. Compute the value of a state by just taking the average, that is:

$$V(s) = \frac{\text{total_return}(s)}{N(s)}$$

The preceding algorithm implies that the value of the state is just the average return of the state across several episodes.

To get a better understanding of how exactly the preceding algorithm works, let's take a simple example and compute the value of each state manually. Say we need to compute the value of three states s_0 , s_1 , and s_2 . We know that we obtain a reward when we transition from one state to another. Thus, the reward for the final state will be 0 as we don't make any transitions from the final state. Hence, the value of the final state s_2 will be zero. Now, we need to find the value of two states s_0 and s_1 .



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

Step 1:

Initialize the $\text{total_return}(s)$ and $N(s)$ for all the states to zero as *Table 4.1* shows:

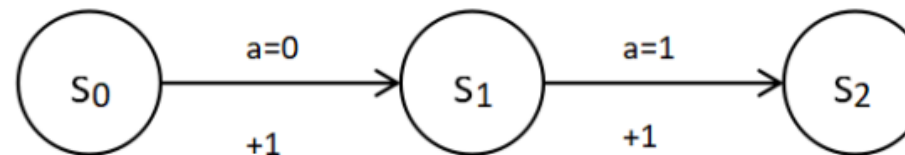
State	$\text{total_return}(s)$	$N(s)$
s_0	0	0
s_1	0	0

Table 4.1: Initial values

Say we are given a stochastic policy π ; in state s_0 our stochastic policy selects the action 0 for 50% of the time and action 1 for 50% of the time, and it selects action 1 in state s_1 for 100% of the time.

Step 2: Iteration 1:

Generate an episode using the given input policy π , as *Figure 4.5* shows:



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

Store all rewards obtained in the episode in the list called rewards. Thus,
rewards = [1, 1].

First, we compute the return of the state s_0 (sum of rewards from s_0):

$$\begin{aligned} R(s_0) &= \text{sum}(\text{rewards}[0:]) \\ &= \text{sum}([1, 1]) \\ &= 2 \end{aligned}$$

Update the total return of the state s_0 in our table as:

$$\begin{aligned} \text{total_returns}(s_0) &= \text{total_returns}(s_0) + R(s_0) \\ &= 0 + 2 = 2 \end{aligned}$$

Update the number of times the state s_0 is visited in our table as:

$$\begin{aligned} N(s_0) &= N(s_0) + 1 \\ &= 0 + 1 = 1 \end{aligned}$$

Now, let's compute the return of the state s_1 (sum of rewards from s_1):

$$\begin{aligned} R(s_1) &= \text{sum}(\text{rewards}[1:]) \\ &= \text{sum}([1]) \\ &= 1 \end{aligned}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

Update the total return of the state s_1 in our table as:

$$\begin{aligned}\text{total_returns}(s_1) &= \text{total_returns}(s_1) + R(s_1) \\ &= 0 + 1 = 1\end{aligned}$$

Update the number of times the state s_1 is visited in our table as:

$$\begin{aligned}N(s_1) &= N(s_1) + 1 \\ &= 0 + 1 = 1\end{aligned}$$

Our updated table, after iteration 1, is as follows:

State	total_return(s)	N(s)
s_0	2	1
s_1	1	1

Table 4.2: Updated table after the first iteration



MC prediction algorithm

Iteration 2:

Say we generate another episode using the same given policy π as Figure 4.6 shows:

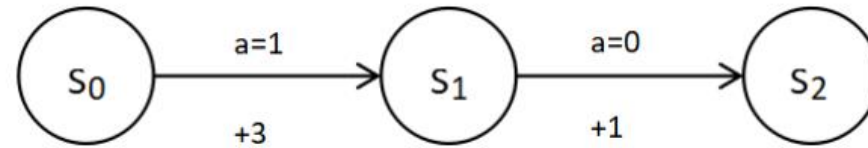


Figure 4.6: Generating an episode using the given policy π

Store all rewards obtained in the episode in the list called rewards. Thus, rewards = [3, 1].

First, we compute the return of the state s_0 (sum of rewards from s_0):

$$\begin{aligned} R(s_0) &= \text{sum}(\text{rewards}[0:]) \\ &= \text{sum}([3, 1]) \\ &= 4 \end{aligned}$$

Update the total return of the state s_0 in our table as:

$$\begin{aligned} \text{total_returns}(s_0) &= \text{total_returns}(s_0) + R(s_0) \\ &= 2 + 4 = 6 \end{aligned}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

Update the number of times the state s_0 is visited in our table as:

$$\begin{aligned} N(s_0) &= N(s_0) + 1 \\ &= 1 + 1 = 2 \end{aligned}$$

Now, let's compute the return of the state s_1 (sum of rewards from s_1):

$$\begin{aligned} R(s_1) &= \text{sum}(\text{rewards}[1:]) \\ &= \text{sum}([1]) \\ &= 1 \end{aligned}$$

Update the return of the state s_1 in our table as:

$$\begin{aligned} \text{total_returns}(s_1) &= \text{total_returns}(s_1) + R(s_1) \\ &= 1 + 1 = 2 \end{aligned}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

Update the number of times the state is visited:

$$\begin{aligned} N(s_1) &= N(s_1) + 1 \\ &= 1 + 1 = 2 \end{aligned}$$

Our updated table after the second iteration is as follows:

State	total_return(s)	N(s)
s ₀	6	2
s ₁	2	2

Table 4.3: Updated table after the second iteration

Since we are computing manually, for simplicity, let's stop at two iterations; that is, we just generate only two episodes.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction algorithm

Step 3:

Now, we can compute the value of the state as:

$$V(s) = \frac{\text{total_returns}(s)}{N(s)}$$

Thus:

$$V(s_0) = \frac{\text{total_return}(s_0)}{N(s_0)} = \frac{6}{2} = 3$$

$$V(s_1) = \frac{\text{total_return}(s_1)}{N(s_1)} = \frac{2}{2} = 1$$

Thus, we computed the value of the state by just taking the average return across multiple episodes. Note that in the preceding example, for our manual calculation, we just generated two episodes, but for a better estimation of the value of the state, we generate several episodes and then we compute the average return across those episodes (not just 2).



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Types of MC prediction

- First-visit Monte Carlo
- Every-visit Monte Carlo



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



First-visit Monte Carlo

- In the **first-visit Monte Carlo method**, if the same state is visited again in the same episode, we don't compute the return for that state again.
- For example, consider a case where an agent is playing snake and ladder. If the agent lands on a snake, then there is a good chance that the agent will return to a state that it had visited earlier.
- So, when the agent revisits the same state, we don't compute the return for that state for the second time.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



First-visit Monte Carlo Algorithm

The following shows the algorithm of first-visit MC; as the point in bold says, we compute the return for the state s_t only if it is occurring for the first time in the episode:

1. Let $\text{total_return}(s)$ be the sum of return of a state across several episodes and $N(s)$ be the counter, that is, the number of times a state is visited across several episodes. Initialize $\text{total_return}(s)$ and $N(s)$ as zero for all the states. The policy π is given as input
2. For M number of iterations:
 1. Generate an episode using the policy π
 2. Store all the rewards obtained in the episode in the list called rewards
 3. For each step t in the episode:



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



First-visit Monte Carlo Algorithm

If the state s_t is occurring for the first time in the episode:

1. Compute the return of the state s_t as $R(s_t) = \text{sum}(\text{rewards}[t:])$
 2. Update the total return of the state s_t as $\text{total_return}(s_t) = \text{total_return}(s_t) + R(s_t)$
 3. Update the counter as $N(s_t) = N(s_t) + 1$
3. Compute the value of a state by just taking the average, that is:

$$V(s) = \frac{\text{total_return}(s)}{N(s)}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Every-visit Monte Carlo Algorithm

Here, **we compute the return every time a state is visited in the episode.**

The algorithm of every-visit Monte Carlo is the same as the one we saw earlier at the beginning of this section and it is as follows:

1. Let $\text{total_return}(s)$ be the sum of the return of a state across several episodes and $N(s)$ be the counter, that is, the number of times a state is visited across several episodes. Initialize $\text{total_return}(s)$ and $N(s)$ as zero for all the states. The policy π is given as input



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Every-visit Monte Carlo Algorithm

2. For M number of iterations:
 1. Generate an episode using the policy π
 2. Store all the rewards obtained in the episode in the list called rewards
 3. For each step t in the episode:
 1. Compute the return of the state s_t as $R(s_t) = \text{sum}(\text{rewards}[t:])$
 2. Update the total return of the state s_t as $\text{total_return}(s_t) = \text{total_return}(s_t) + R(s_t)$
 3. Update the counter as $N(s_t) = N(s_t) + 1$
3. Compute the value of a state by just taking the average, that is:

$$V(s) = \frac{\text{total_return}(s)}{N(s)}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Remember that the only difference between the first-visit MC and every-visit MC methods is that in the first-visit MC method, we compute the return for a state only for its first time of occurrence in the episode but in the every-visit MC method, the return of the state is computed every time the state is visited in an episode.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Blackjack game with MC prediction

- Blackjack also known as 21 is a popular card game.
- It has a **player and a dealer**.
- Goal of the player is
 - to have a collection of cards that sum upto 21
or
 - Have a larger value than the sum of the dealer's cards but not exceeding 21.
- If one of the above two criteria is true, then the player 'wins' or the dealer 'wins'.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- Value of cards J, K, and Q is 10.
- Value of Ace can be 1 or 11, depending on the player's choice during the game.
- Value of rest of the cards is their face value, ex, value of card 2 is 2
- There can be many players at a time but only one dealer.
- All players compete with only the dealer, not within them.
- Consider a case with one player(which is ourself) and one dealer.
- Player performs one of the actions :
 - **hit** --- get one more card.
 - **Stand** ---- don't need any more cards, but request the dealer to show their cards.



- Initially both player and dealer are given 2 cards
- Both cards of the player are face up (visible to the dealer)
- Only one of the dealer's card is face up (visible to the player) and the other is face down.
- Let's see the different cases of the game :
 - The player wins
 - The player loses
 - The player goes bust
 - Usable ace
 - Non-usable ace



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Different cases of the game

- Case 1: when the player wins the game



Figure 4.7: The player has 20, and the dealer has 2 with one card face down

- The player's card value is already large 20,
- Hence the player 'stands'

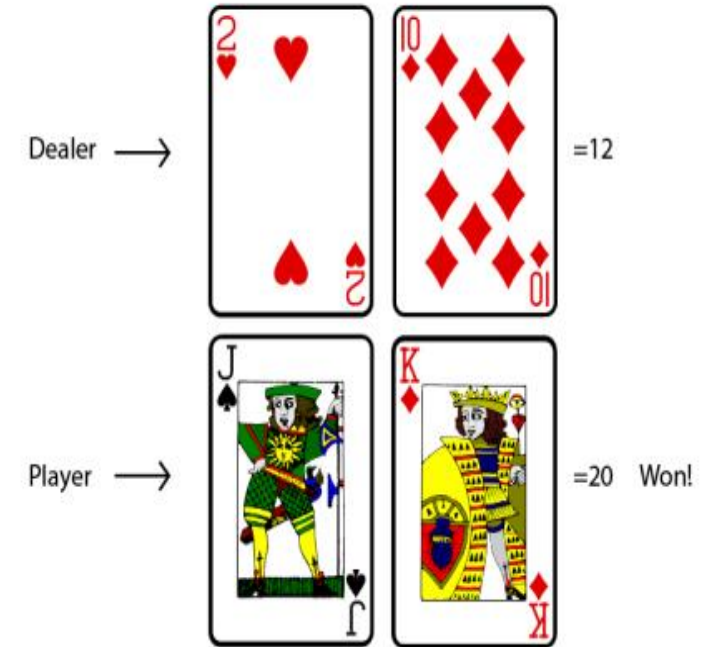


Figure 4.8: The player wins!

- The player's sum is greater than the dealer's and also doesn't exceed 21, hence the player wins.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- Case 2: when the player loses the game

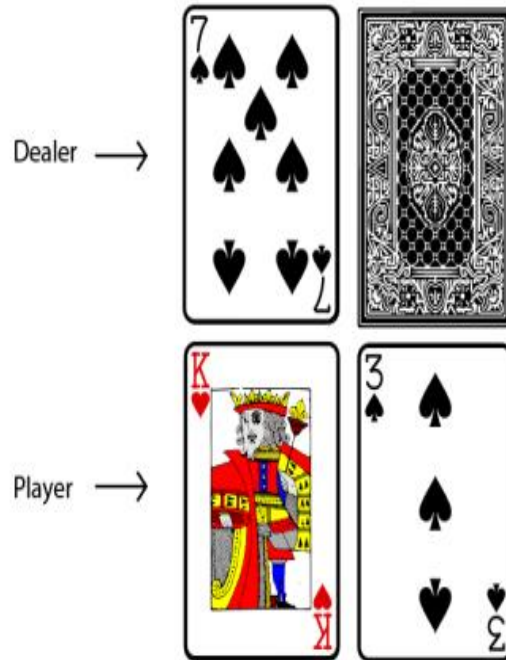


Figure 4.9: The player has 13, and the dealer has 7 with one card face down

- Assume the player is optimistic that the dealer's value cannot exceed 13,
- Hence the player 'stands'

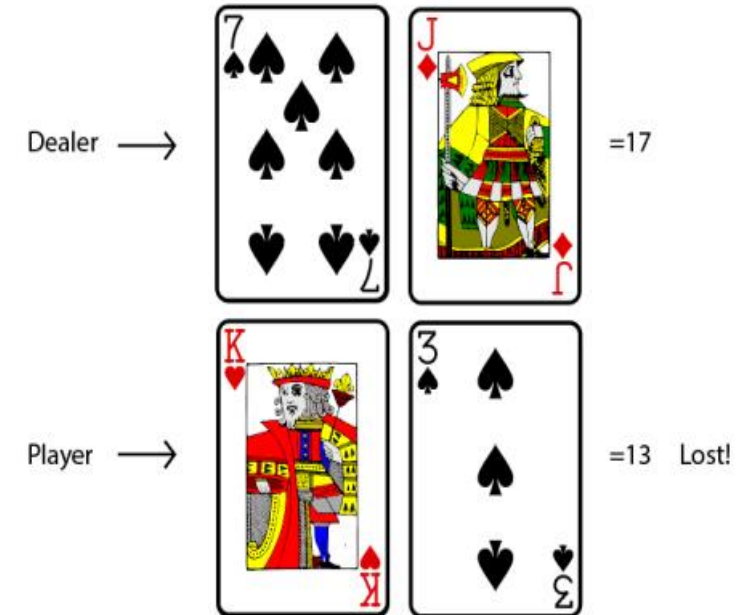


Figure 4.10: The dealer wins!

- The dealer's value is greater than the player's value but doesn't exceed 21.
- Hence the player loses



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



• Case 3: when the player goes bust



Figure 4.11: The player has 8, and the dealer has 10 with one card face down

- Players total value is only 8.
- Hence the player performs 'hit' to get a new card.

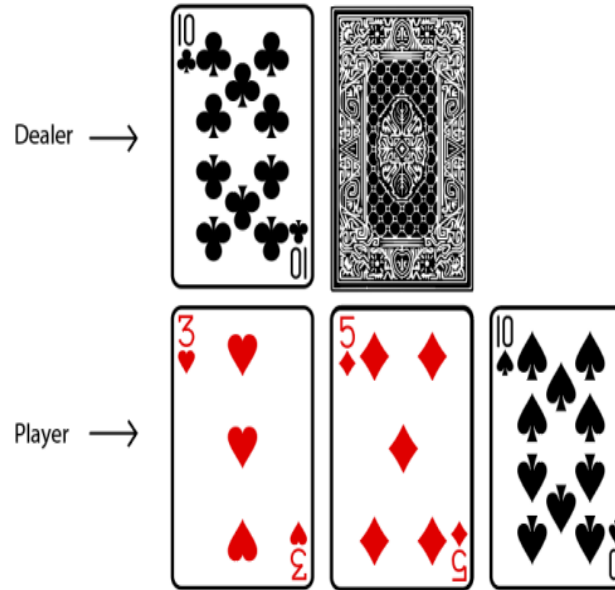


Figure 4.12: The player has 18, and the dealer has 10 with one card face down

- Assume the new card is 10
- Now the player's value is 18
- If the player again performs 'hit' greedily to get a new card,
- Assume the next card is a Q



Figure 4.13: The player goes bust!

- The player's sum is now 28, which exceeds 21.
- **This is a bust and the player losses.**



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- Case 4: Usable Ace:



Figure 4.14: The player has 10, and the dealer has 5 with one card face down

- Player performs 'hit'
- Gets an Ace
- Decides the value of the Ace as 11

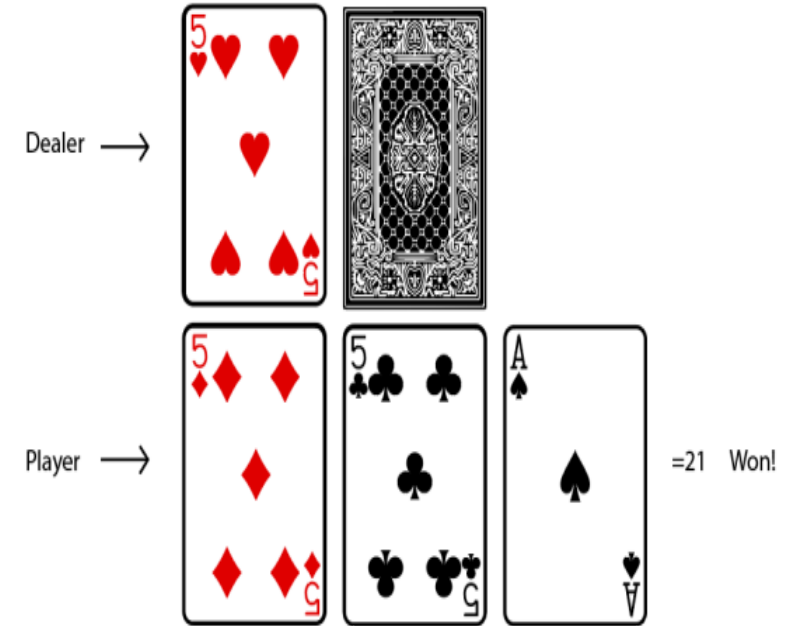


Figure 4.15: The player uses the Ace as 11 and wins the game



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



• Case 5: Unusable Ace

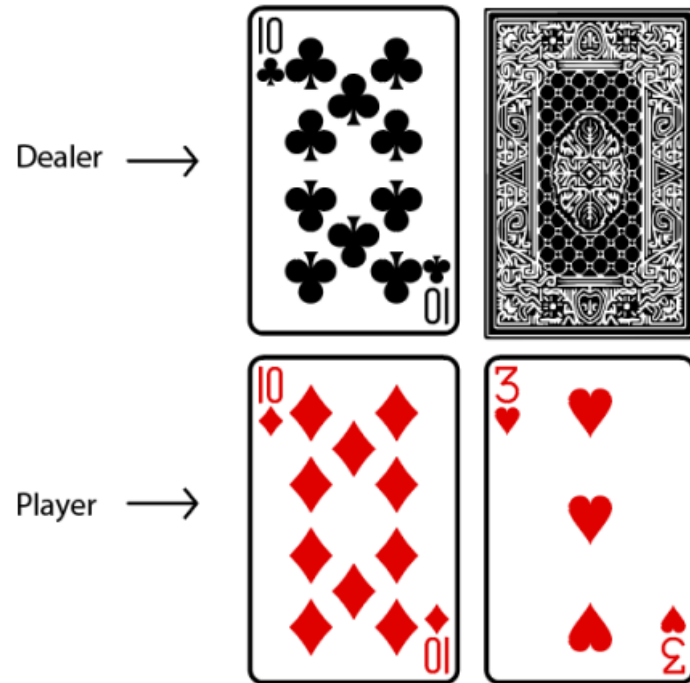


Figure 4.16: The player has 13, and the dealer has 10 with one card face down

- Player performs 'hit'
- Gets an Ace
- Player has to use the value of the Ace as 1

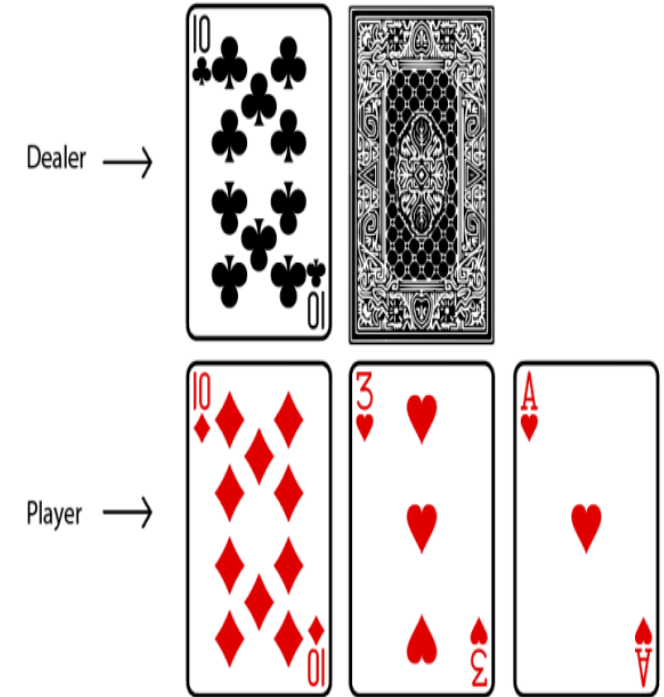


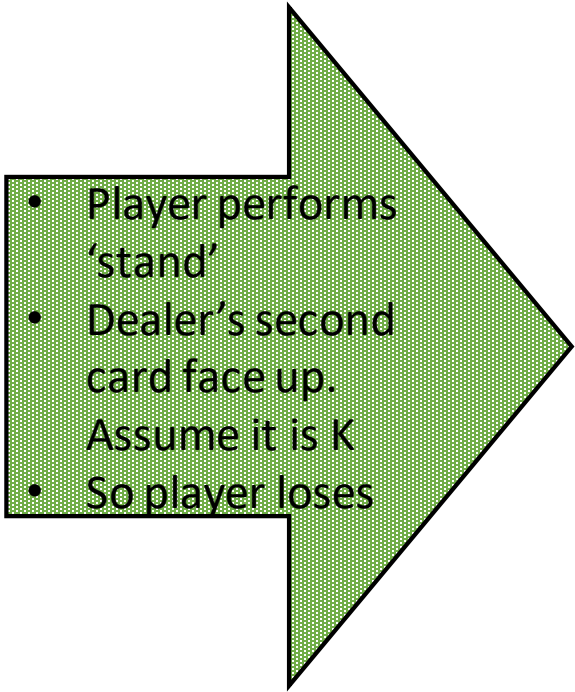
Figure 4.17: The player has to use the **Ace** as a 1 else they go bust



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- 
- Player performs 'stand'
 - Dealer's second card face up.
Assume it is K
 - So player loses

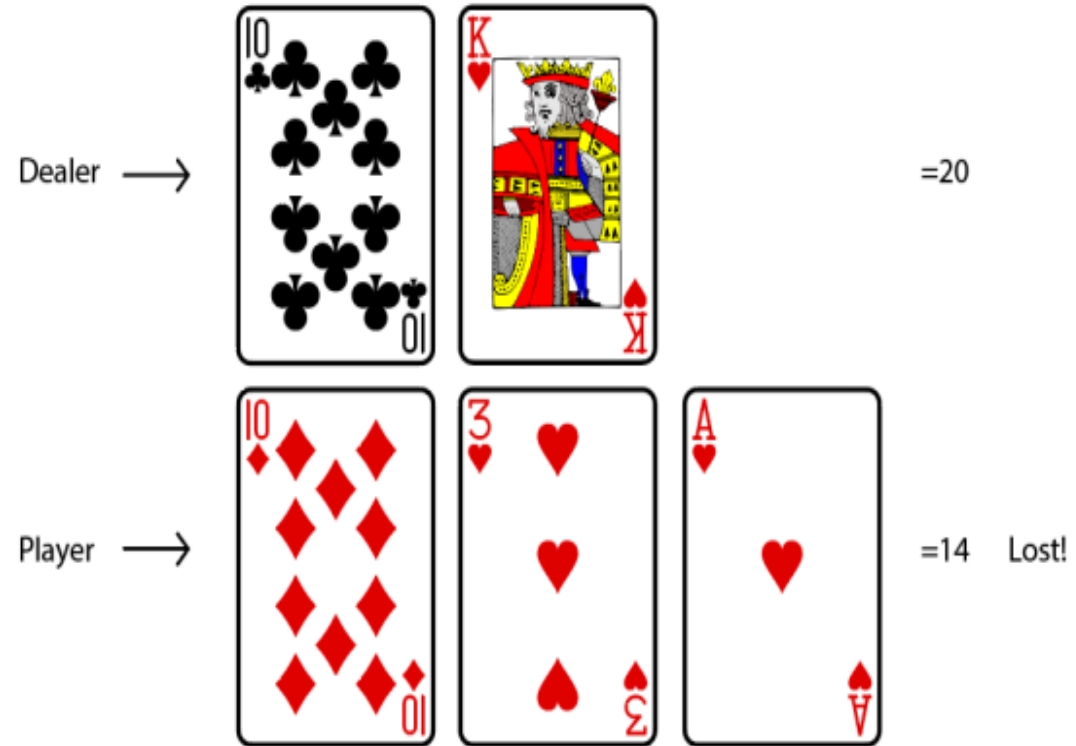


Figure 4.18: The player has 14, and the dealer has 20 and wins



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



- **Case 6: game is a draw**

If both the player and the dealer's sum of cards value is the same, say 20, then the game is called a draw.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Incremental mean updates

- In both first-visit MC and every-visit MC, we estimate the value of a state as an average (arithmetic mean) return of the state across several episodes as shown as follows:

$$V(s) = \frac{\text{total_return}(s)}{N(s)}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Incremental mean updates

- Instead of using the arithmetic mean to approximate the value of the state, we can also use the incremental mean, and it is expressed as:

$$N(s_t) = N(s_t) + 1$$

$$V(s_t) = V(s_t) + \frac{1}{N(s_t)} (R_t - V(s_t))$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Incremental mean updates

- Consider our environment as non-stationary. In that case, we don't have to take the return of the state from all the episodes and compute the average.
- As the environment is non-stationary we can ignore returns from earlier episodes and use only the returns from the latest episodes for computing the average. Thus, we can compute the value of the state using the incremental mean as shown as follows:



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Incremental mean updates

$$V(s_t) = V(s_t) + \alpha(R_t - V(s_t))$$

Where $\alpha = 1/N(s_t)$ and R_t is the return of the state s_t .



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction (Q function)

- We generate several episodes using the given policy π ,
- Then, we calculate the $\text{total_return}(s, a)$, the sum of the return of the state-action pair across several episodes.
- We calculate $N(s, a)$, the number of times the state-action pair is visited across several episodes.
- Then we compute the Q function or Q value as the average return of the state-action pair as shown as follows:

$$Q(s, a) = \frac{\text{total_return}(s, a)}{N(s, a)}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction (Q function)

Say we have two states s_0 and s_1 and we have two possible actions 0 and 1.

Now, we compute $\text{total_return}(s, a)$ and $N(s, a)$.

Let's say our table after computation looks like Table 4.4:

State	Action	$\text{total_return}(s,a)$	$N(s,a)$
s_0	0	4	2
s_0	1	2	2
s_1	0	2	2
s_1	1	2	1

Table 4.4: The result of two actions in two states



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction (Q function)

- Thus, we can compute the Q value for all state-action pairs as:

$$Q(s_0, 0) = \text{total_return}(s_0, 0) / N(s_0, 0) = 4/2 = 2$$

$$Q(s_0, 1) = \text{total_return}(s_0, 1) / N(s_0, 1) = 2/2 = 1$$

$$Q(s_1, 0) = \text{total_return}(s_1, 0) / N(s_1, 0) = 2/2 = 1$$

$$Q(s_1, 1) = \text{total_return}(s_1, 1) / N(s_1, 1) = 2/1 = 2$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction (Q function) algorithm

The algorithm for predicting the Q function using the Monte Carlo method is as follows.

1. Let $\text{total_return}(s, a)$ be the sum of the return of a state-action pair across several episodes and $N(s, a)$ be the number of times a state-action pair is visited across several episodes.

Initialize $\text{total_return}(s, a)$ and $N(s, a)$ for all state-action pairs to zero.

The policy π is given as input

MC prediction (Q function) algorithm

2. For M number of iterations:
 1. Generate an episode using policy π
 2. Store all rewards obtained in the episode in the list called rewards
 3. For each step t in the episode:
 1. Compute return for the state-action pair, $R(s_t, a_t) = \text{sum}(\text{rewards}[t:])$
 2. Update the total return of the state-action pair, $\text{total_return}(s_t, a_t) = \text{total_return}(s_t, a_t) + R(s_t, a_t)$
 3. Update the counter as $N(s_t, a_t) = N(s_t, a_t) + 1$
3. Compute the Q function (Q value) by just taking the average, that is:

$$Q(s, a) = \frac{\text{total_return}(s, a)}{N(s, a)}$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC prediction of the Q function

- We have two types of MC— first-visit MC and every-visit MC.
- **In first-visit MC**, we compute the return of the state-action pair only for the first time the state-action pair is visited in the episode
- **In every-visit MC** we compute the return of the state-action pair every time the state-action pair is visited in the episode.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Incremental mean

We can compute the Q value using the incremental mean as shown as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R_t - Q(s_t, a_t))$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo control

- In the control task, our goal is to find the optimal policy.
- Unlike the prediction task, here, we will not be given any policy as an input.
- we will begin by initializing a random policy, and then we try to find the optimal policy iteratively.
- An **optimal policy is that which gives the maximum return.**
- if we have a Q function, then we can extract policy by selecting an action in each state that has the maximum Q value as the following shows:
$$\pi = \arg \max_a Q(s, a)$$
- From the new Q function, we extract a new policy. We repeat these steps iteratively until we find the optimal policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo control

- **Iteration 1**: Let π_0 be the random policy
 - Step 1.1 : Use π_0 to generate an episode
 - Step 1.2 : Find Q^{π_0} by taking the average return of the state-action pair.
 - Step 1.3: From Q^{π_0} extract a new policy π_1 . This will not be an optimal policy. Hence need one more iteration.
- **Iteration 2**: Use the new policy π_1 , got from the previous iteration.
 - Step 1.1 : Use π_1 to generate an episode
 - Step 1.2 : Find Q^{π_1} by taking the average return of the state-action pair.
 - Step 1.3: From Q^{π_1} extract a new policy π_2 . If this policy is optimal, then stop, else, generate one more iteration.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo control

Iteration 3— Now, we use the new policy π_2 derived from the previous iteration to generate an episode and compute the new Q function Q^{π_2} . Then, from this Q function Q^{π_2} , we extract a new policy π_3 . If π_3 is optimal we stop, else we go to the next iteration.

We repeat this process for several iterations until we find the optimal policy π^* as shown in *Figure 4.21*:

$$\pi_0 \rightarrow Q^{\pi_0} \rightarrow \pi_1 \rightarrow Q^{\pi_1} \rightarrow \pi_2 \rightarrow Q^{\pi_2} \rightarrow \pi_3 \rightarrow Q^{\pi_3} \rightarrow \dots \rightarrow \pi^* \rightarrow Q^{\pi^*}$$

Figure 4.21: The path to finding the optimal policy



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithm

- Once we have the Q function, we extract a new policy by selecting an action in each state that has the maximum Q value.
- In the next iteration, we use the extracted new policy to generate an episode and compute the new Q function (Q value) as the average return of the state-action pair.
- We repeat these steps for many iterations to find the optimal policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithm

- One more thing, we need to observe that just as we learned in the first-visit MC prediction method, here,
- we compute the return of the state-action pair only for the first time a state-action pair is visited in the episode



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithmic steps

1. Let $\text{total_return}(s, a)$ be the sum of the return of a state-action pair across several episodes and $N(s, a)$ be the number of times a state-action pair is visited across several episodes. Initialize $\text{total_return}(s, a)$ and $N(s, a)$ for all state-action pairs to zero and initialize a random policy π
2. For M number of iterations:
 1. Generate an episode using policy π
 2. Store all rewards obtained in the episode in the list called rewards



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithmic steps

3. For each step t in the episode:

If (s_t, a_t) is occurring for the first time in the episode:

1. Compute the return of a state-action pair, $R(s_t, a_t) = \text{sum}(\text{rewards}[t:])$
2. Update the total return of the state-action pair as, $\text{total_return}(s_t, a_t) = \text{total_return}(s_t, a_t) + R(s_t, a_t)$
3. Update the counter as $N(s_t, a_t) = N(s_t, a_t) + 1$
4. Compute the Q value by just taking the average, that is,

$$Q(s_t, a_t) = \frac{\text{total_return}(s_t, a_t)}{N(s_t, a_t)}$$

4. Compute the new updated policy π using the Q function:

$$\pi = \arg \max_a Q(s, a)$$

5. If this new policy is optimal, then stop. Else repeat step 2,3 and 4 using this new policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC Control methods

- **On-policy control**—In the on-policy control method, the agent **behaves** using one policy and also tries to improve that same policy.
- That is, **we generate episodes** using one policy and also improve the same policy iteratively to find the optimal policy.
- For instance, the MC control method, which we just learned above, can be called on-policy MC control as we are generating episodes using a policy π , and we also try to improve the same policy π on every iteration to compute the optimal policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC Control methods

- **Off-policy control**—In the off-policy control method, the agent behaves using one policy and tries to improve a different policy π .
- That is, in the off-policy method, we generate episodes using one policy and we try to improve the different policy iteratively to find the optimal policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



on-policy Monte Carlo control

There are two types of on-policy Monte Carlo control methods:

- Monte Carlo exploring starts
- Monte Carlo with the epsilon-greedy policy



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo exploring starts algorithm

1. Let $\text{total_return}(s, a)$ be the sum of the return of a state-action pair across several episodes and $N(s, a)$ be the number of times a state-action pair is visited across several episodes. Initialize $\text{total_return}(s, a)$ and $N(s, a)$ for all state-action pairs to zero and initialize a random policy π
2. For M number of iterations:
 1. **Select the initial state s_0 and initial action a_0 randomly such that all state-action pairs have a probability greater than 0**
 2. Generate an episode from the selected initial state s_0 and action a_0 using policy π
 3. Store all the rewards obtained in the episode in the list called rewards



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo exploring starts algorithm

4. For each step t in the episode:

If (s_t, a_t) is occurring for the first time in the episode:

1. Compute the return of a state-action pair, $R(s_t, a_t) = \text{sum}(\text{rewards}[t:])$
2. Update the total return of the state-action pair as $\text{total_return}(s_t, a_t) = \text{total_return}(s_t, a_t) + R(s_t, a_t)$
3. Update the counter as $N(s_t, a_t) = N(s_t, a_t) + 1$
4. Compute the Q value by just taking the average, that is,

$$Q(s_t, a_t) = \frac{\text{total_return}(s_t, a_t)}{N(s_t, a_t)}$$

5. Compute the updated policy π using the Q function:

$$\pi = \arg \max_a Q(s, a)$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo exploring starts algorithm

- One of the major drawbacks of the exploring starts method is that it is not applicable to every environment.
- That is, we can't just randomly choose any state-action pair as an initial state-action pair because in some environments there can be only one state-action pair that can act as an initial state-action pair.
- So we can't randomly select the state-action pair as the initial state-action pair.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo exploring starts algorithm

- For example, suppose we are training an agent to play a car racing game; we can't start the episode in a random position as the initial state and a random action as the initial action because we have a fixed single starting state and action as the initial state and action.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo with the epsilon-greedy policy

- So, now the question is whether the agent should explore all the other actions in the state and select the best action as the one that has the maximum Q value

or

- exploit the best action out of already-explored actions. This is called an **exploration-exploitation dilemma**.
- The agent must decide whether to exploit the current best-known policy or explore new policies to improve its performance

Exploration-Exploitation dilemma

- The exploration-exploitation dilemma is a problem that can be encountered in most of the data driven decision making process when there exists some kind of **feedback loop between data gathering and decisions making**
- They are two possible behaviors when facing a decision making problem that both have pros and cons.
- **exploitation** consists of taking the decision assumed to be optimal with respect to the data observed so far.
- This « safe » approach tries to avoid bad decisions as much as possible but also prevents from discovering potential better decisions.

- **exploration** consists of not taking the decision that seems to be optimal, based on the fact that observed data are not sufficient to truly identify the best option.
- This is a more « risky » approach and can sometimes lead to poor decisions
- but also makes it possible to discover better ones, if there exists any.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Examples

- **restaurant selection**: going to your favourite restaurant (exploitation) or trying an unknown restaurant (exploration)
- **movies recommendation**: recommending the user's best rated movie type (exploitation) or trying another movie type (exploration)
- **oil drilling**: drilling at the best known location (exploitation) or trying a new location (exploration)
- **clinical trials**: using the best known treatment (exploitation) or trying a new experimental one (exploration)



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo with the epsilon-greedy policy

- To avoid this dilemma, we introduce a new policy called the **epsilon-greedy policy**.
- Here, all actions are tried with a non-zero probability (epsilon). With a probability epsilon, we explore different actions randomly and with a probability $1 - \epsilon$, we choose an action that has the maximum Q value.
- That is, with a probability epsilon, we select a random action (exploration) and with a probability $1 - \epsilon$ we select the best action (exploitation).



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Action at time(t) $\left\{ \begin{array}{ll} \max Q_t(a) & \text{with probability } 1-\epsilon \\ \text{any action (a)} & \text{with probability } \epsilon \end{array} \right.$

```
p = random()  
  
if p < ε:  
    pull random action  
else:  
    pull current-best action
```



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo with the epsilon-greedy policy

- In the epsilon-greedy policy, if we set the value of epsilon to 0, then it becomes a greedy policy (only exploitation).
- when we set the value of epsilon to 1, then we will always end up doing only the exploration.
- So, the value of epsilon has to be chosen optimally between 0 and 1.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo with the epsilon-greedy policy

- Say we set $\epsilon = 0.5$; then we will generate a random number from the uniform distribution and if the random number is less than ϵ (0.5), then we select a random action (exploration).
- if the random number is greater than or equal to ϵ then we select the best action, that is, the action that has the maximum Q value (exploitation).



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Monte Carlo with the epsilon-greedy policy

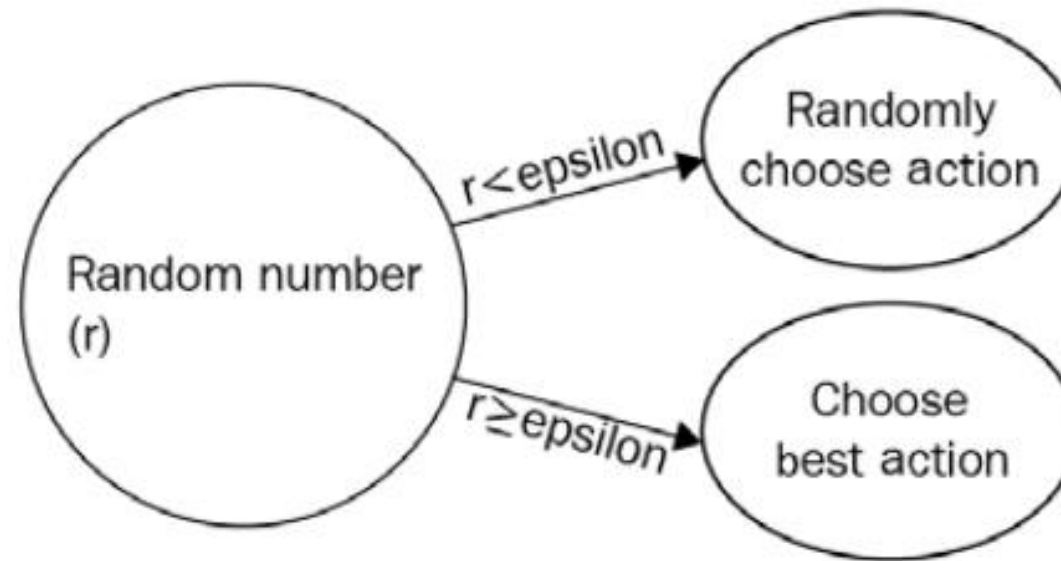


Figure 4.22: Epsilon-greedy policy



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithm with the epsilon-greedy policy

The MC control algorithm with the epsilon-greedy policy

The algorithm of Monte Carlo control with the epsilon-greedy policy is essentially the same as the MC control algorithm we learned earlier except that here we select actions based on the epsilon-greedy policy to avoid the exploration-exploitation dilemma. The following steps show the algorithm of Monte Carlo with the epsilon-greedy policy:

1. Let $\text{total_return}(s, a)$ be the sum of the return of a state-action pair across several episodes and $N(s, a)$ be the number of times a state-action pair is visited across several episodes. Initialize $\text{total_return}(s, a)$ and $N(s, a)$ for all state-action pairs to zero and initialize a random policy π



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithm with the epsilon-greedy policy

2. For M number of iterations:

1. Generate an episode using policy π
2. Store all rewards obtained in the episode in the list called rewards
3. For each step t in the episode:

If (s_t, a_t) is occurring for the first time in the episode:

1. Compute the return of a state-action pair, $R(s_t, a_t) = \text{sum}(\text{rewards}[t:])$
2. Update the total return of the state-action pair as $\text{total_return}(s_t, a_t) = \text{total_return}(s_t, a_t) + R(s_t, a_t)$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC control algorithm with the epsilon-greedy policy

3. Update the counter as $N(s_t, a_t) = N(s_t, a_t) + 1$
4. Compute the Q value by just taking the average, that is,

$$Q(s_t, a_t) = \frac{\text{total_return}(s_t, a_t)}{N(s_t, a_t)}$$

4. Compute the updated policy π using the Q function. Let $a^* = \arg \max_a Q(s, a)$. The policy π selects the best action a^* with probability $1 - \epsilon$ and random action with probability ϵ

As we can observe, in every iteration, we generate the episode using the policy π and also we try to improve the same policy π in every iteration to compute the optimal policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Off-policy Monte Carlo control

- Off-policy Monte Carlo is another interesting Monte Carlo control method. In the off-policy method, we use two policies called the behavior policy and the target policy. As the name suggests, we behave (generate episodes) using the behavior policy and we try to improve the other policy called the target policy.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Off-policy Monte Carlo control

In the on-policy method, we generate an episode using the policy π and we improve the same policy π iteratively to find the optimal policy. But in the off-policy method, we generate an episode using a policy called the behavior policy b and we try to iteratively improve a different policy called the target policy π .



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Off-policy Monte Carlo control algorithm

The algorithm is given as follows:

1. Initialize the Q function $Q(s, a)$ with random values, set the behavior policy b to be epsilon-greedy, and also set the target policy π to be greedy policy.
2. For M number of episodes:
 1. Generate an episode using the behavior policy b
 2. Initialize return R to 0



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Off-policy Monte Carlo control algorithm

3. For each step t in the episode, $t = T-1, T-2, \dots, 0$:
 1. Compute the return as $R = R + r_{t+1}$
 2. Compute the Q value as $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R_t - Q(s_t, a_t))$
 3. Compute the target policy $\pi(s_t) = \arg \max_a Q(s_t, a)$
3. Return the target policy π



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Need of importance sampling

- We are finding the target policy π from the Q function, which is computed based on the episodes generated by a different policy called the behavior policy, our target policy will be inaccurate. This is because the distribution of the behavior policy and the target policy will be different.
- So, to correct this, we introduce a new technique called importance sampling. This is a technique for estimating the values of one distribution when given samples from another.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Importance sampling types:

- Ordinary importance sampling

Here, the importance sampling ratio will be the ratio of the target policy to the behavior policy $\frac{\pi(a|s)}{b(a|s)}$

- Weighted importance sampling

Here, the importance sampling ratio will be the weighted ratio of the target policy to the behavior policy $w \frac{\pi(a|s)}{b(a|s)}$.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Off-policy Monte Carlo control algorithm

The complete algorithm of the off-policy Monte Carlo method is explained in the following steps:

1. Initialize the Q function $Q(s, a)$ with random values, set the behavior policy b to be epsilon-greedy, and target policy π to be greedy policy and initialize the cumulative weights as $C(s, a) = 0$
2. For M number of episodes:
 1. Generate an episode using the behavior policy b
 2. Initialize return R to 0 and weight W to 1
 3. For each step t in the episode, $t = T-1, T-2, \dots, 0$:
 1. Compute the return as $R = R + r_{t+1}$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Off-policy Monte Carlo control algorithm

2. Update the cumulative weights $C(s_t, a_t) = C(s_t, a_t) + W$
3. Update the Q value as
$$Q(s_t, a_t) = Q(s_t, a_t) + \frac{W}{C(s_t, a_t)} (R_t - Q(s_t, a_t))$$
4. Compute the target policy $\pi(s_t) = \arg \max_a Q(s_t, a)$
5. If $a_t \neq \pi(s_t)$ then break
6. Update the weight as $W = W \frac{1}{b(a_t | s_t)}$

3. Return the target policy π



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



MC limitation

- Issue with the Monte Carlo method is that it is applicable only to episodic tasks. We learned that in the Monte Carlo method, we compute the value of the state by taking the average return of the state and the return is the sum of rewards of the episode.
- But when there is no episode, that is, if our task is a continuous task (non-episodic task), then we cannot apply the Monte Carlo method.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013

