UNIT 5 MONTE CARLO METHODS FOR MODEL FREE PREDICTION

Ami Munshi

Syllabus

5.	**Monte Carlo Methods for Model Free Prediction and Control** Monte Carlo Prediction, Monte Carlo Estimation of Action Values, Monte Carlo Control, Off-policy Prediction via Importance Sampling, Off-Policy Monte Carlo Control, Importance Sampling on Truncated Return	**05**

Course Outcomes

After completion of the course, students will be able to –

1. Apply the basics of Reinforcement Learning (RL) to compare with traditional control design
2. Correlate how RL relates and fits into the broader umbrella of machine learning, deep learning
3. Recommend value functions and appropriate algorithms for optimal decision-making
4. Design a dynamic programming approach to an industrial control problem

References

Text Books

1. Laura Graesser and Wah Loon Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, 1st Edition, Pearson India/Padmavati Publisher, 2022.
2. Richard Sutton and Andrew Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, MIT Press, 2018.
3. Abhishek Nandy and Manisha Biswas, *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*, 1st Edition, Apress Publisher, 2017.

Reference Books

1. Nimish Sanghi, *Deep Reinforcement Learning with Python: With PyTorch, TensorFlow and OpenAI*, 2nd edition, Apress Publisher, 2021.
2. Alexander Zai and Brandon Brown, *Deep Reinforcement Learning in Action*, 1st Edition, Manning Publisher, 2020.
3. Csaba Szepesvari, *Algorithms for Reinforcement Learning*, 3rd Edition, Morgan & Claypool Publisher, 2019.

Dr Saeed Saeedvand
https://youtu.be/oGB3dkI-Lt0?feature=shared

https://www.youtube.com/watch?v=xGMjVX59aVE
https://www.youtube.com/watch?v=oGB3dkI-Lt0&list=PLZ_sI4f41TGvthD8dA7daahlbLV0yDW0w&index=4

Monte Carlo- An Introduction

- Here we do not assume complete knowledge of the environment
- Monte Carlo methods require only experience
  - sample sequences of states, actions, and rewards from actual or simulated interaction with an environment
- Learning from actual experience is striking because it requires no prior knowledge of the environment's dynamics, yet can still attain optimal behavior
- Learning from simulated experience is also powerful

Monte Carlo- An Introduction

- Although a model is required, the model need only generate sample transitions, not the complete probability distributions of all possible transitions that is required for dynamic programming (DP)
- Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns
- We will consider Monte Carlo methods for episodic tasks
- We assume that
  - the experience is divided into episodes and
  - No matter what action is taken, the episode always terminates
- On completion of one episode, the value estimates and policy are changed
- Monte Carlo is incremental episode-by-episode but not step-by-step (online) sense

Monte Carlo- An Introduction

- The term “Monte Carlo” is often used more broadly for any estimation method whose operation involves a significant random component
- Here we use it specifically for methods based on averaging complete returns
- Monte Carlo methods sample and average returns for each state–action pair much like the bandit methods
- The main difference is that now there are multiple states, each acting like a different bandit problem (like an associative-search or contextual bandit) and the different bandit problems are interrelated
- Return after taking an action in one state depends on the actions taken in later states in the same episode
- Because all the action selections are undergoing learning, the problem becomes nonstationary from the point of view of the earlier state

Monte Carlo Prediction

- We will use Monte Carlo methods for learning the state-value function for a given policy
- Recall
  - Value of a state is the expected return
  - Which is expected cumulative future discounted reward
  - That is starting from that state
- An obvious way to estimate it from experience, then, is simply to average the returns observed after visits to that state
- As more returns are observed, the average should converge to the expected value
- This idea underlies all Monte Carlo methods

Monte Carlo Prediction

- Suppose we wish to estimate the value of a state s under policy π, given a set of episodes $v_\pi(s)$ is obtained by following π and passing through s
- Each occurrence of state s in an episode is called a visit to s
- s may be visited multiple times in the same episode
- Two methods
  - First Visit
  - Every Visit
- Let us called first time it is visited in an episode as the first visit to s
- First-visit Monte Carlo method estimates $v_\pi(s)$, as the average of the returns following first visits to s
- Every-Visit Monte Carlo method averages the returns following all visits to s

Monte Carlo Policy Evaluation

- Its goal is to learn state value function v_π from episodes of experience under policy π: $S_1, A_1, R_2, \dots, S_k$
- Return is the total discounted reward for an episode ending at time $T > t$

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$$

- Value function is the expected return

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

- MC policy evaluation use sample average return instead of expected return

First Visit Monte Carlo Example

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow$ average$(Returns(S_t))$

First-visit MC on policy evaluation

Initialize N(s) =0, G(s) =0, for all s ϵS

Loop (for each episode)

Sample episode i: $S_{i,1}, A_{i,1}, R_{i,2}, S_{i,2}, A_{i,2}, R_{i,3,\ldots}, A_{i,T_i-1} R_{i,T_i}, S_{i,T_i}$

Define: $G_{i,t} = R_{i,t+1} + \gamma R_{i,t+2} + \gamma^2 R_{i,t+3} + \cdots + \gamma^{T_i-t-1} R_{i,T_i}$ as the return from time step t to the end of episode i

For each state s visited in episode i

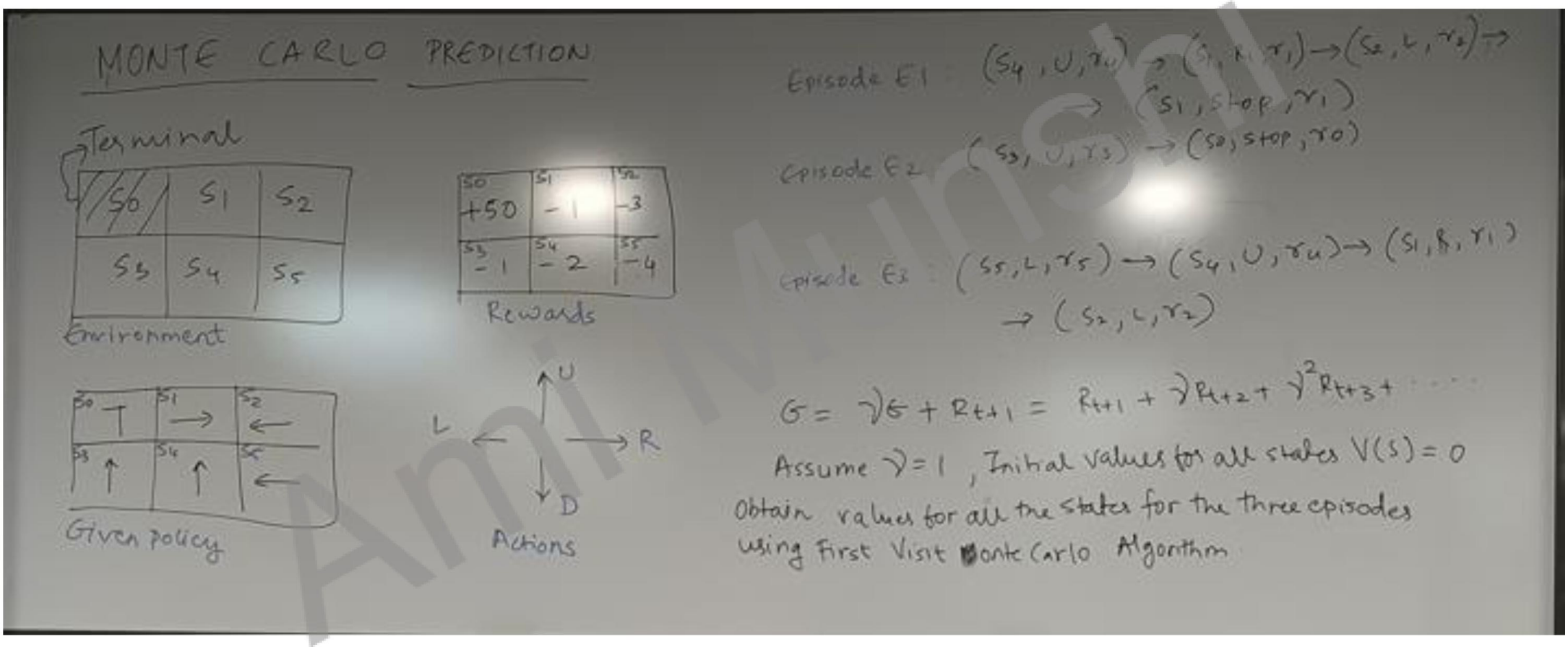
If s is visited at first time in episode i

Increment counter of total first visits: N(s)=N(s)+1

Increment total return G(s)=G(s)+$G_{i,t}$

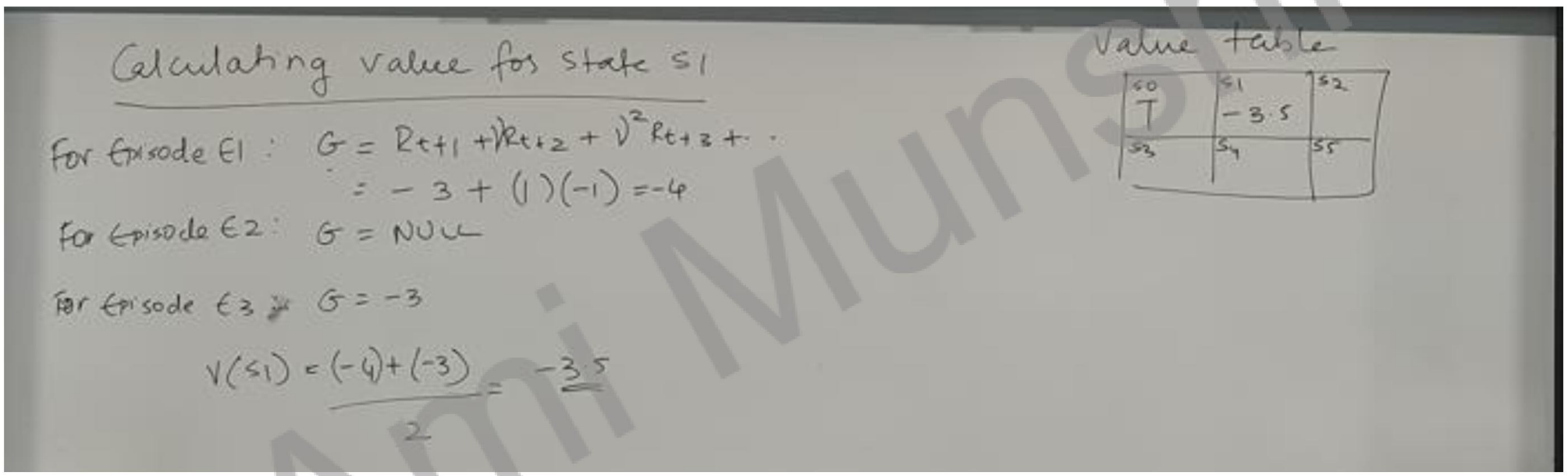
Update estimate V^{π}(s)=G(s)/N(s)

Example on First Visit Monte Carlo



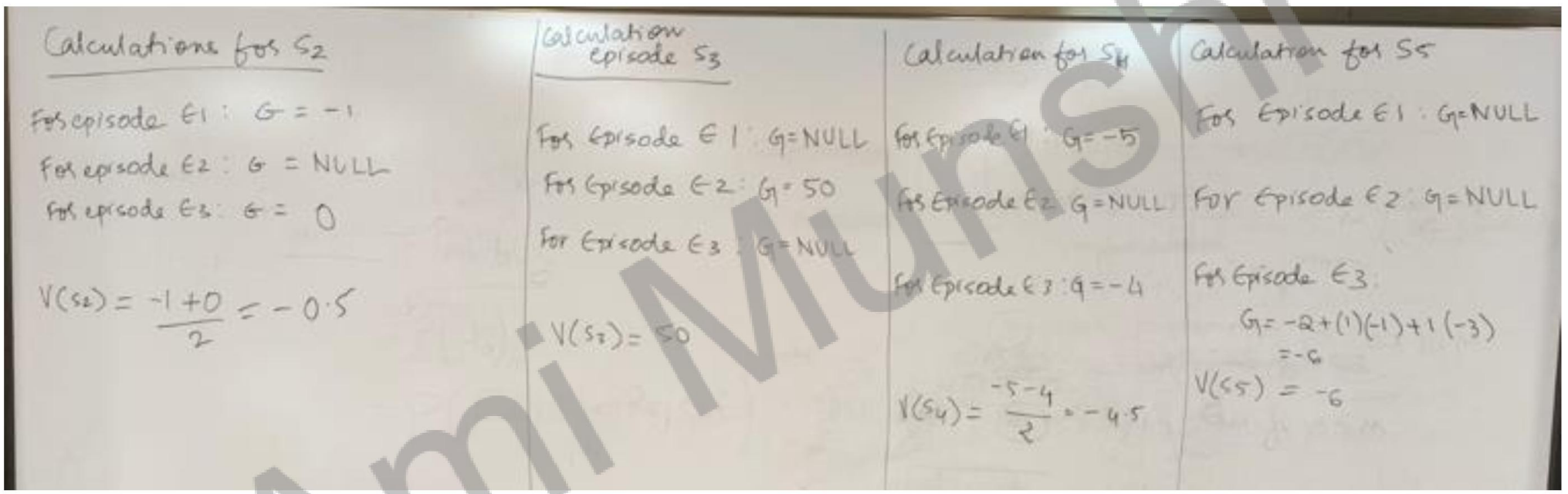
Ref: Dr Saeed Saeedvand: https://youtu.be/oGB3dkI-Lt0?feature=shared

Example on First Visit Monte Carlo



Ref: Dr Saeed Saeedvand: https://youtu.be/oGB3dkI-Lt0?feature=shared

Example on First Visit Monte Carlo



Value table

T	-3.5	-0.5
50	-4.5	-6

Every Visit Monte Carlo Example

Every-visit MC on policy evaluation

Initialize N(s) =0, G(s) =0, for all s ϵS

Loop (for each episode)

Sample episode i: $S_{i,1}, A_{i,1}, R_{i,2}, S_{i,2}, A_{i,2}, R_{i,3,\ldots}, A_{i,T_i-1} R_{i,T_i}, S_{i,T_i}$

Define: $G_{i,t} = R_{i,t+1} + \gamma R_{i,t+2} + \gamma^2 R_{i,t+3} + \cdots + \gamma^{T_i-t-1} R_{i,T_i}$ as the return from time step t to the end of episode i

For each state s visited in episode i

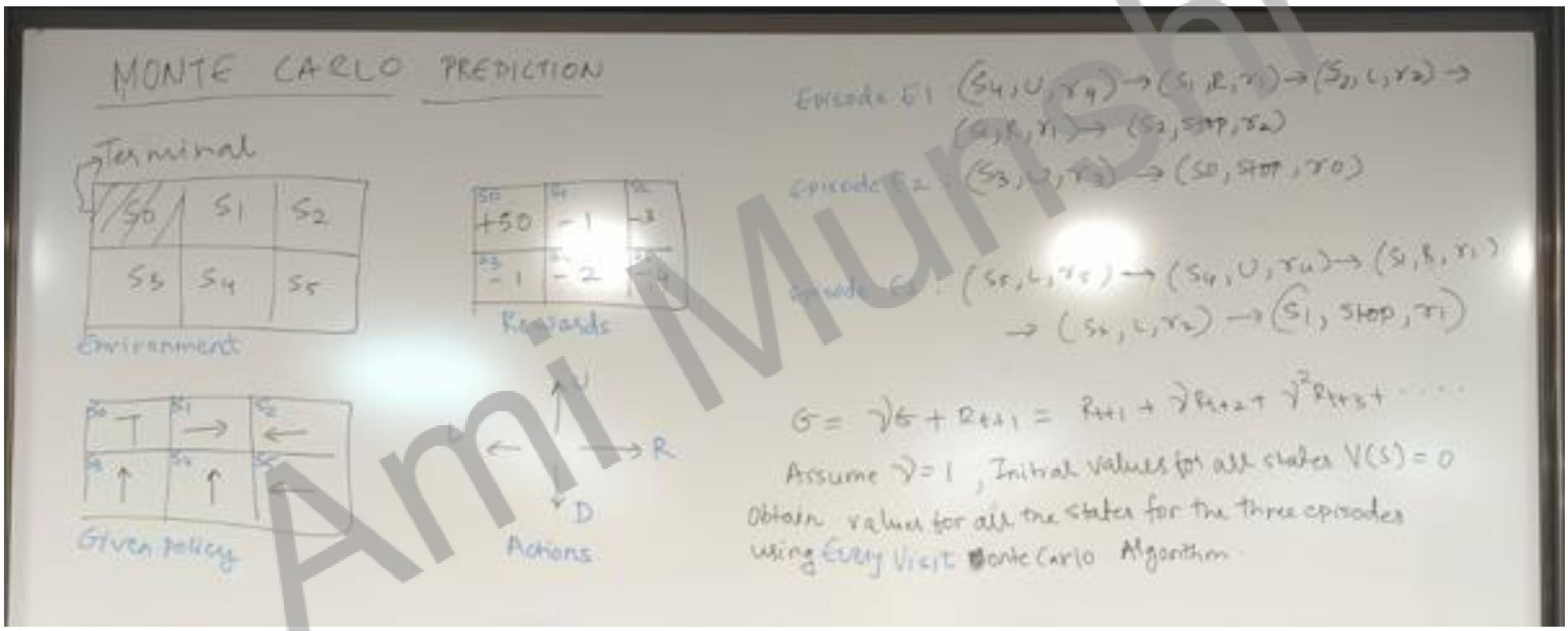
For **every time** t that state s is visited in episode i

Increment counter of total visits : N(s)=N(s)+1

Increment total return G(s)=G(s)+$G_{i,t}$

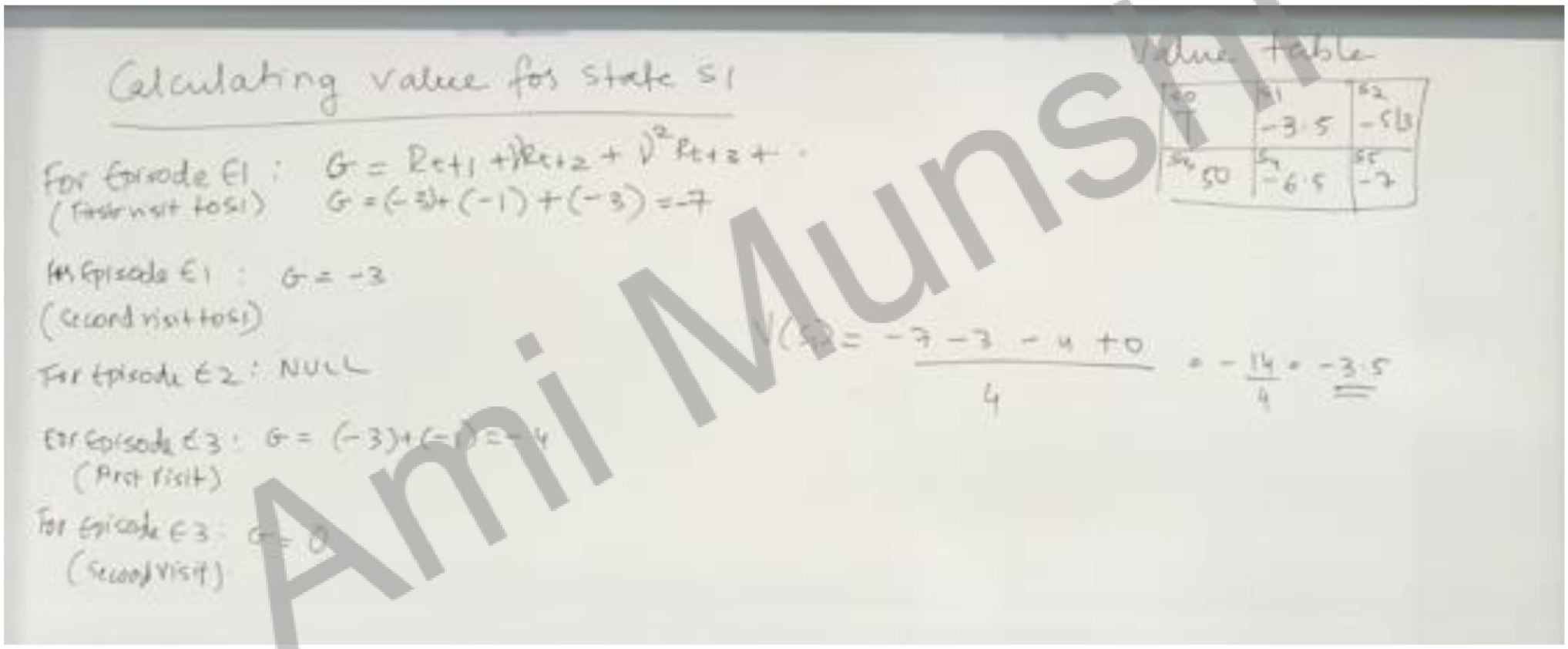
Update estimate V^{π}(s)=G(s)/N(s)

Example on Every Visit Monte Carlo



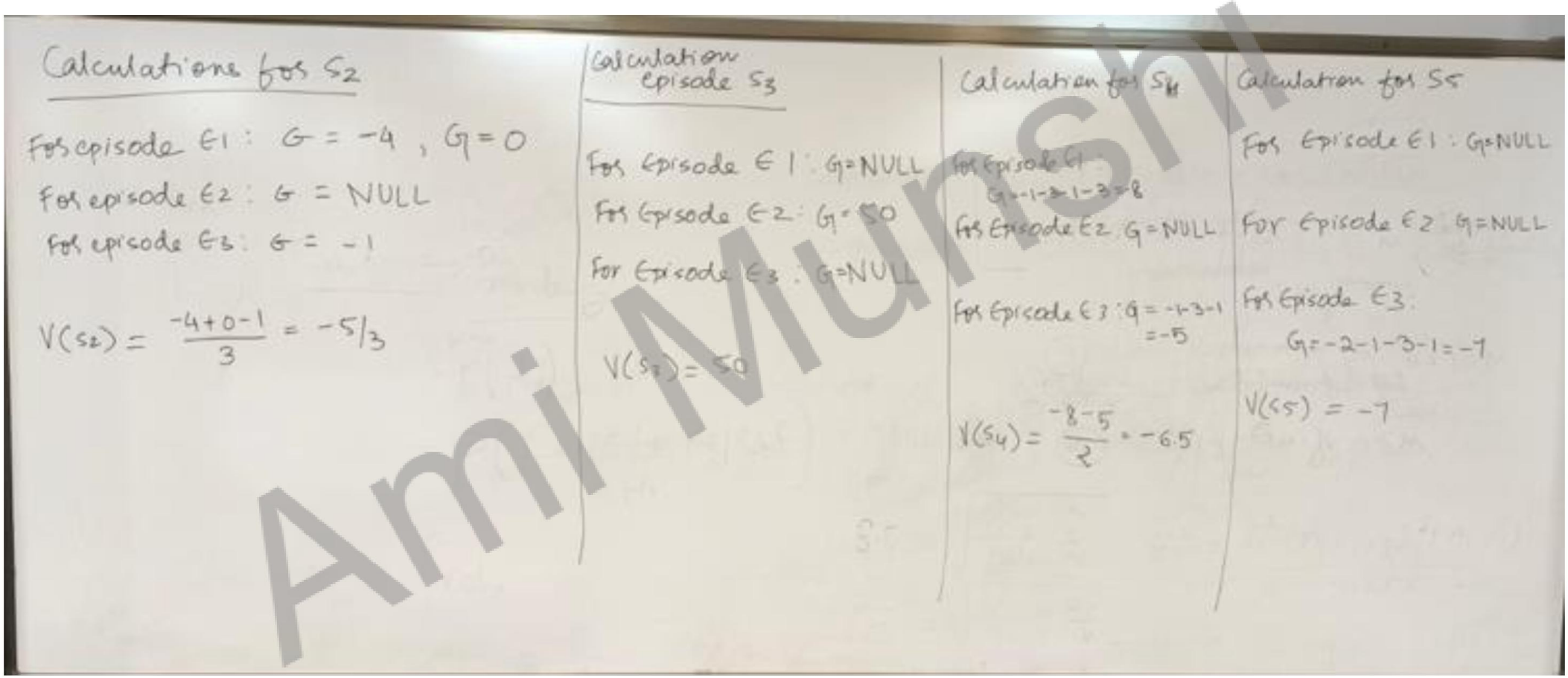
Ref: Dr Saeed Saeedvand: https://youtu.be/oGB3dkI-Lt0?feature=shared

Example on Every Visit Monte Carlo



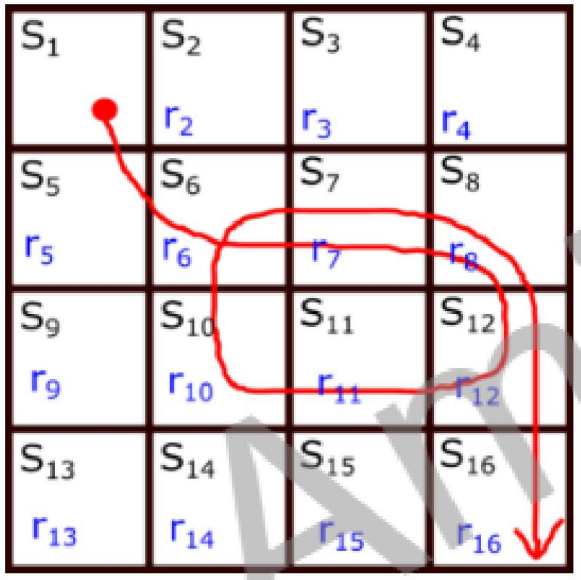
Ref: Dr Saeed Saeedvand: https://youtu.be/oGB3dkI-Lt0?feature=shared

Example on Every Visit Monte Carlo



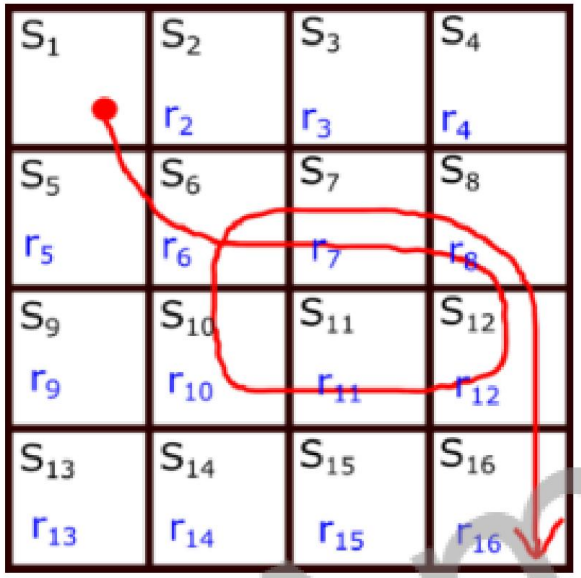
Ref: Dr Saeed Saeedvand: https://youtu.be/oGB3dkI-Lt0?feature=shared

Example



- Consider the following environment
- Trajectory followed during one episode is shown
- Actions are {L,R,U,D}
- Rewards in each state are mentioned
- Write the episode under policy π in the following format $\{S_t, A_t, R_{t+1}\}$

https://aleksandarhaber.com/monte-carlo-method-for-learning-state-value-functions-first-visit-method-reinforcement-learning-tutorial/


S1
S2
r2
S3
r3
S4
r4
S5
r5
S6
r6
S7
r7
S8
r8
S9
r9
S10
r10
S11
r11
S12
r12
S13
r13
S14
r14
S15
r15
S16
r16

Example

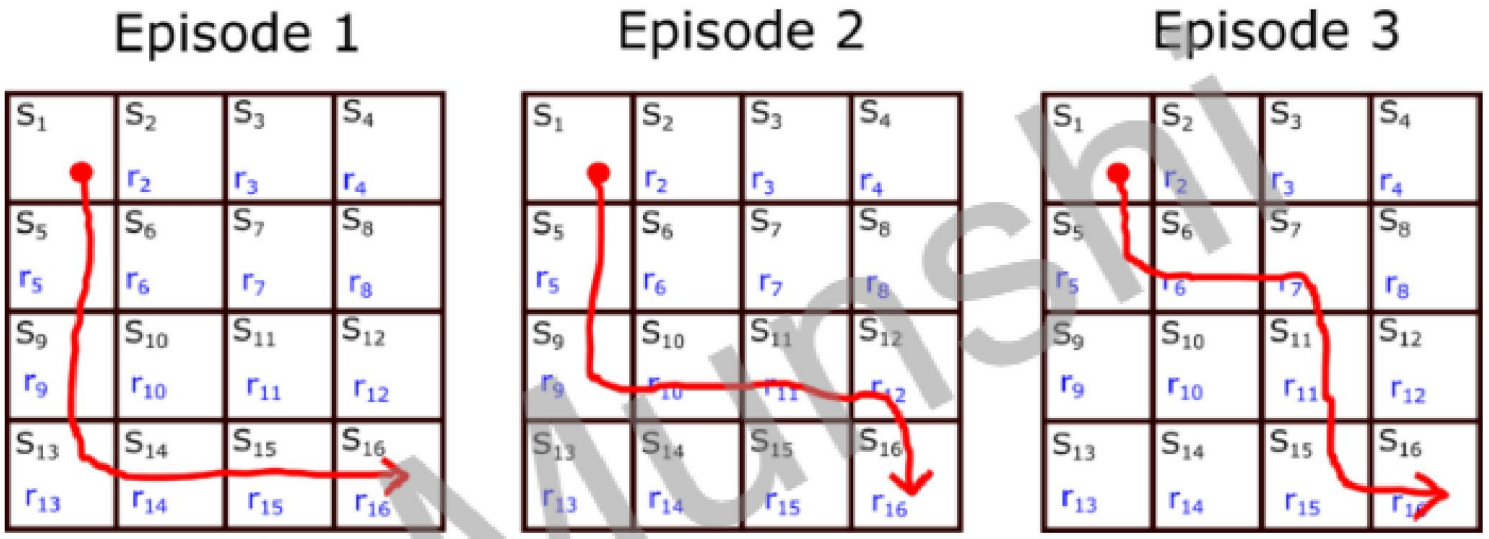


Figure 2: three episodes.

- For the given three episodes,
  - Evaluate the value of state s_1 using First Visit Monte Carlo and Every Visit Monte Carlo algorithms

https://aleksandarhaber.com/monte-carlo-method-for-learning-state-value-functions-first-visit-method-reinforcement-learning-tutorial/

Example

Assume you have data in the form of just the following 5 complete episodes for an MRP.

- *Episode 1:* A 2 A 6 B 1 B 0 T
- *Episode 2:* A 3 B 2 A 4 B 2 B 0 T
- *Episode 3:* B 3 B 6 A 1 B 0 T
- *Episode 4:* A 0 B 2 A 4 B 4 B 2 B 0 T
- *Episode 5:* B 8 B 0 T

In the episodic data presented above, the non-terminal states are labeled A and B, the numbers denote *rewards*, and all states end in a terminal state T. Assume discount factor $\gamma = 1$. Answer the following:

(a) Given only this data and experience replay (repeatedly and endlessly drawing an episode at random from this pool of 5 episodes), calculate the value function estimates, i.e. $V(\mathtt{A})$ and $V(\mathtt{B})$, that (a) First-Visit Monte-Carlo and (b) Every-Visit Monte-Carlo converge to.

Ref: IIT Kharagpur, Midsem paper 2022

Example

First-Visit Monte Carlo averages the returns starting from the first occurrence of each of the states across all episodes. Therefore, the First-Visit Monte Carlo Value Function estimate (with experience replay) would converge to:

$$V(\mathrm{A}) = \frac{(2+6+1+0)+(3+2+4+2+0)+(1+0)+(0+2+4+4+2+0)}{4} = \frac{33}{4} = 8.25$$

$$V(\mathrm{B}) = \frac{(1+0)+(2+4+2+0)+(3+6+1+0)+(2+4+4+2+0)+(8+0)}{5} = \frac{39}{5} = 7.8$$

Ref: IIT Kharagpur, Midsem paper 2022

Example

Every-Visit Monte Carlo averages the returns starting from each occurrence of each of the states across all episodes. Therefore, the Every-Visit Monte Carlo Value Function estimate (with experience replay) would converge to:

$$V(\mathrm{A}) = \frac{[(2+6+1+0)+(6+1+0)]+[(3+2+4+2+0)+(4+2+0)]+[(1+0)]+[(0+2+4+4+2+0)+(4+4+2+0)]}{2+2+1+2}$$

$$= \frac{(9+7)+(11+6)+(1)+(12+10)}{7} = \frac{56}{7} = 8$$

$$V(\mathrm{B}) = \frac{[(1+0)+(0)]+[(2+4+2+0)+(2+0)+(0)]+[(3+6+1+0)+(6+1+0)+(0)]+[(2+4+4+2+0)+(4+2+0)+(2+0)+(0)]+[(8+0)+(0)]}{2+3+3+4+2}$$

$$= \frac{(1+0)+(8+2+0)+(10+7+0)+(12+6+2+0)+(8+0)}{14} = \frac{56}{14} = 4$$

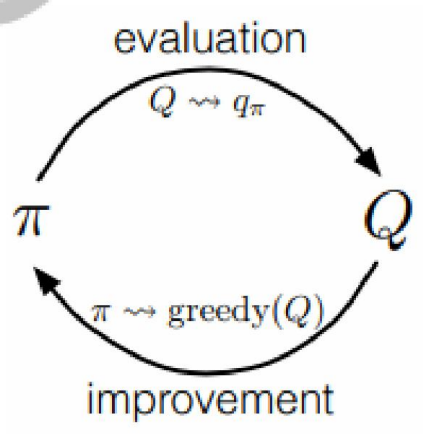
Ref: IIT Kharagpur, Midsem paper 2022

Monte Carlo Control

- Monte Carlo estimation can be used in control, that is, to approximate optimal policies
- The overall idea is to proceed according to the same pattern as in the DP, according to the idea of generalized policy iteration
- In GPI one maintains both an approximate policy and an approximate value function

Monte Carlo Control

- Value function is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function, as suggested by the diagram



Monte Carlo version of classical policy iteration

- In this method, we perform alternating complete steps of policy evaluation and policy improvement, beginning with an arbitrary policy ↑0 and ending with the optimal policy and optimal action-value function

$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_{*}$$

where E is policy evaluation and I is policy improvement

Monte Carlo Control-Exploring Starts

- For Monte Carlo policy iteration it is natural to alternate between evaluation and improvement on an episode-by-episode basis
- After each episode, the observed returns are used for policy evaluation, and then the policy is improved at all the states visited in the episode
- A complete simple algorithm along these lines, which we call Monte Carlo ES, for Monte Carlo with Exploring Starts

Monte Carlo Exploring Start (ES) Algorithm for Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \text{argmax}_a \, Q(S_t, a)$

Assign arbitrary value for Q-values and Policy first

Repeat Until Converge (no policy change) – optimal policy π.

a) Select random state s and action a pair in the environment

b) Generate an episode by policy π (T steps)

- Start from selected state s and run selected action a

c) For each state-action pair compute the discounted returns and create list

- Return is calculation of only all next states of each state in episode

$$V_\pi(s) = E_\pi[G(s)], G = \gamma g + R_{t+1}$$

d) Average the value of list and update the Q-value for state-action pairs

$$Q(s_i, a_j) = AVG(List(s_i, a_j))$$

f) Update the Policy

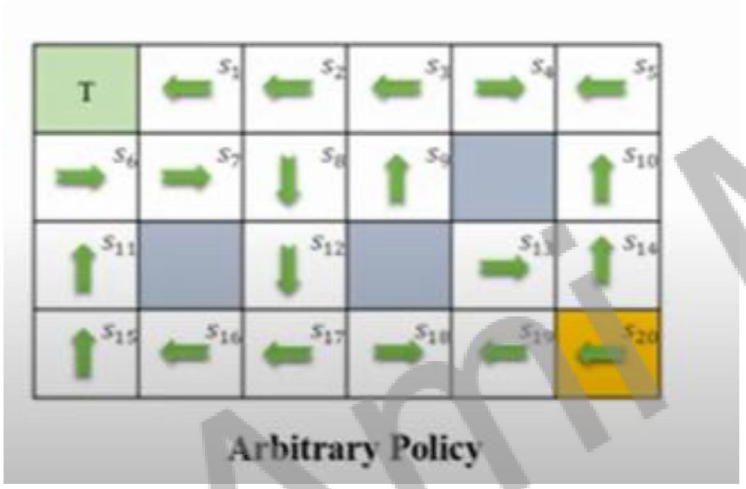
$$\pi(s) = argmax_a Q(s, a)$$

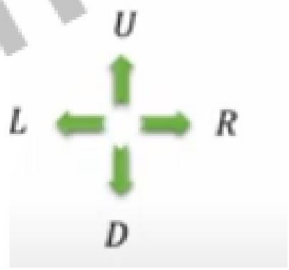
Ref: Dr Saeed Saeedvand: https://youtu.be/oGB3dkI-Lt0?feature=shared

Example on Monte Carlo ES

- Robot has to go from start to target
- Reward
  - +100 for target
  - -1 for each other step
- Q value initially zero for simplicity

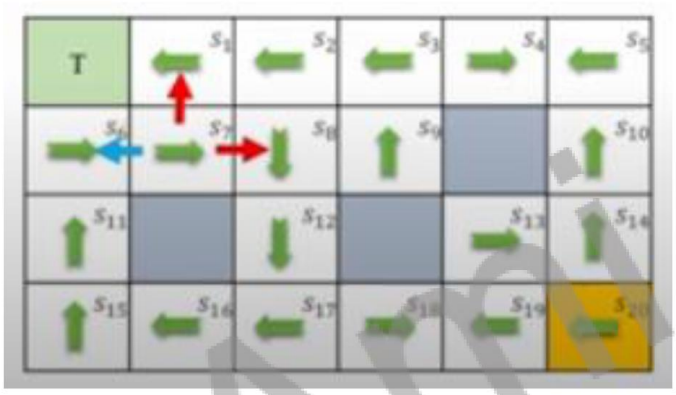
Assign Arbitrary Policy and zero Q values

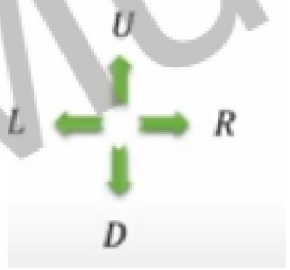




Step a: Select random state s and random action. Consider only possible action from each state

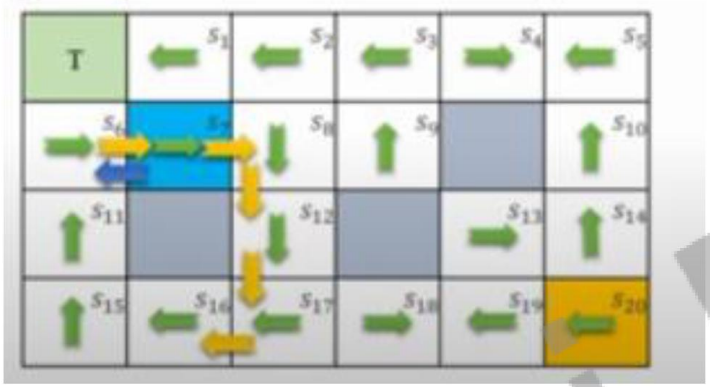
Episode 1
Random state (S_7, L), T = 6



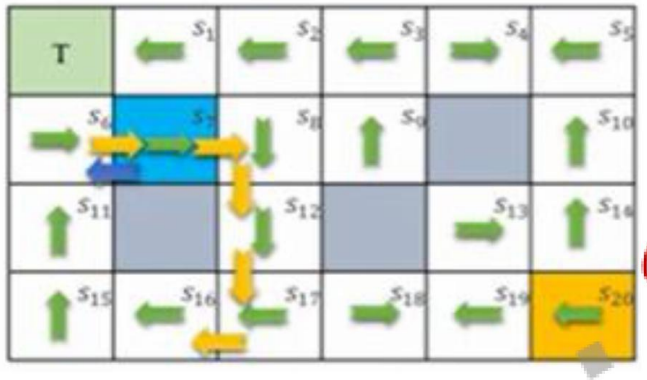


$(S_7, L) \rightarrow (S_6, R) \rightarrow (S_7, R) \rightarrow (S_8, D) \rightarrow (S_{12}, D) \rightarrow (S_{13}, L) \rightarrow (S_{12}, L)$

Step b: Generate Episode by policy π. Start from random state s and select the action according to the policy



Step c: For each state action pair, compute the discounted return and create list



Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]

Discounted Return = $G = \gamma G + R_{t+1}$, $\gamma = 0.9$

$(s_7, L) \rightarrow (s_6, R) \rightarrow (s_7, R) \rightarrow (s_8, D) \rightarrow (s_{12}, D) \rightarrow (s_{17}, L) \rightarrow (s_{16}, L)$

$G(s_{17}, L) = 0.9 \times 0 - 1 = -1$

$(s_{12}, D) = 0.9 \times (-1) - 1 = -1.9$

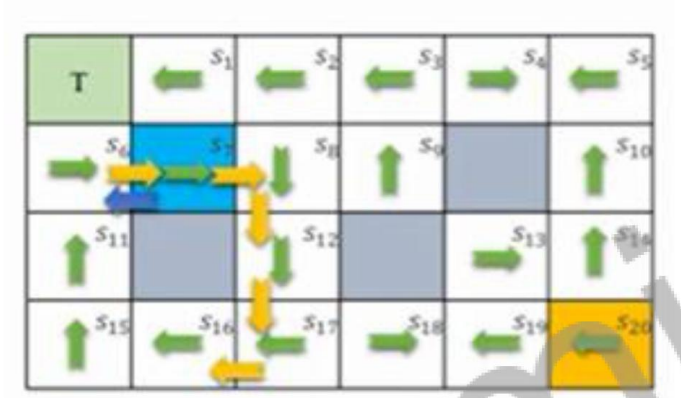
$(s_8, D) = 0.9 \times (-1.9) - 1 = -2.71$

$(s_7, R) = 0.9 \times (-2.71) - 1 = -3.439$

$(s_6, R) = 0.9(-3.439) - 1 = -4.095$

$(s_7, L) = 0.9(-4.095) - 1 = -4.685$

Step d: Average value of list and update Q value

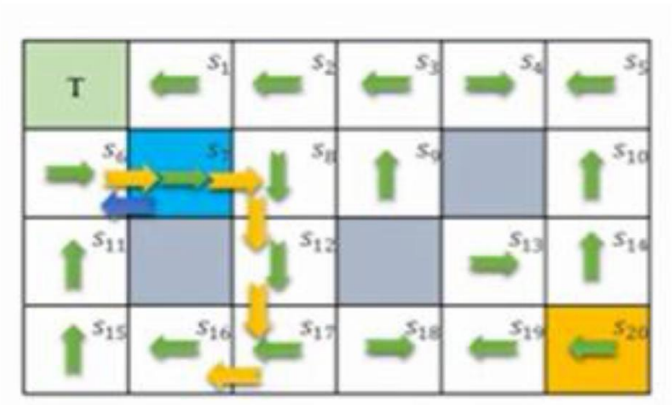


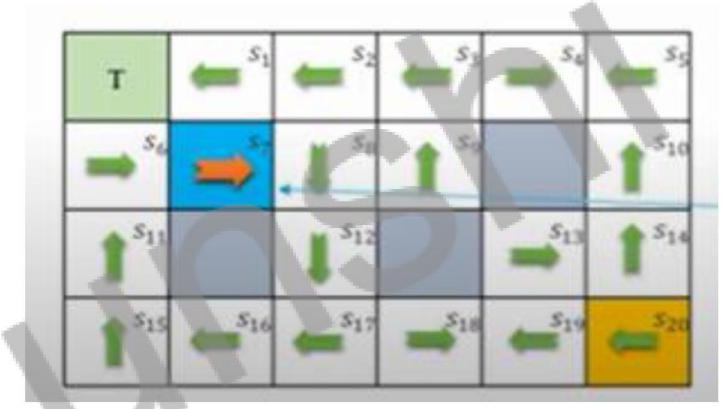
Average list

Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]

Average list will remain the same as there is only one item in the list

Step e: Update the policy

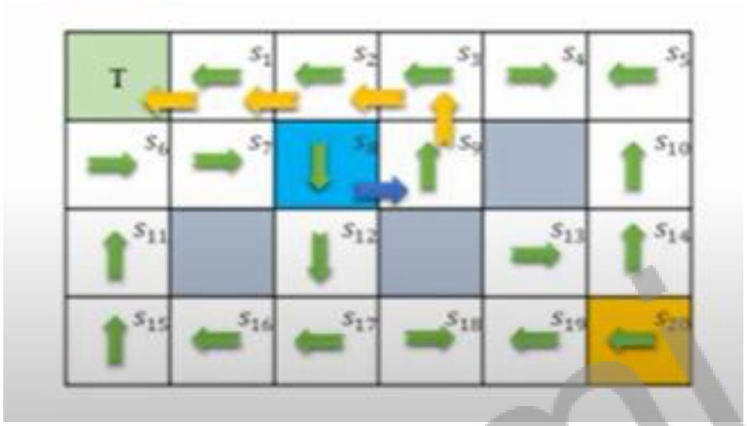




Generate Episode E2

Random start at state S8

T=6 Random action for S_8 is R



$(S_8, R) \rightarrow (S_9, U) \rightarrow (S_3, L) \rightarrow (S_2,$

$\rightarrow (S_1, L) \rightarrow (S_0, \text{stop})$

$G = \gamma G + R_{t+1}$, $\gamma = 0.9$

$(S_1, L) = 0.9 \times 0 + 100 = 100$

$(S_2, L) = 0.9 \times 100 - 1 = 89$

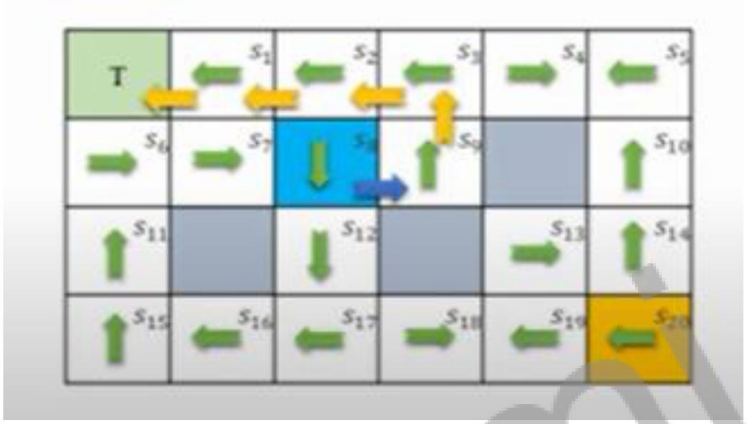
$(S_3, L) = 0.9 \times 89 - 1 = 79.1$

$(S_9, U) = 0.9 \times 79.1 - 1 = 70.19$

$(S_8, R) = 0.9 \times 70.19 - 1 = 62.17$

Episode E2

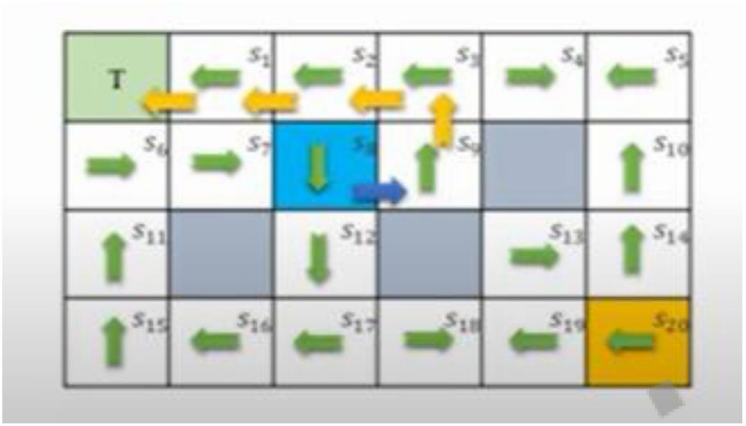
$(S_8, R) \rightarrow (S_9, U) \rightarrow (S_3, L), (S_2, L) \rightarrow (S_1, L) \rightarrow (S_0, \text{stop})$

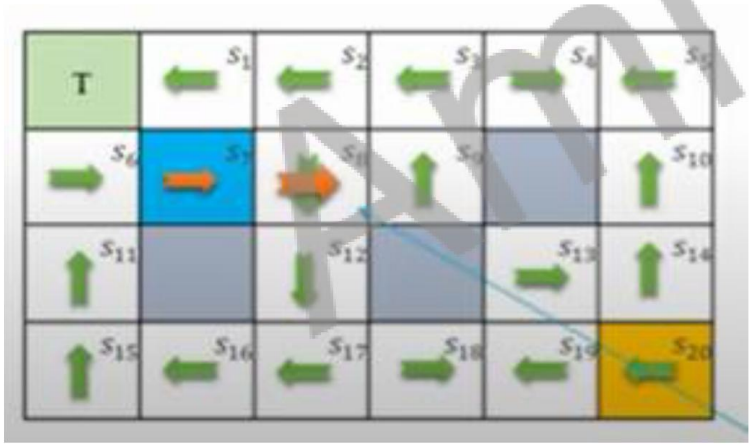


Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]

Returns_list(s_1, L)=[100]
Returns_list(s_2, L)=[89]
Returns_list(s_3, L)=[79.1]
Returns_list(s_9, U)=[70.19]
Returns_list(s_8, R)=[62.171]

Take Average of the list and Update Policy after Episode 2





Update the Policy

$Q(s_{17}, L)$=[−1]
$Q(s_{12}, D)$=[−1.9]
$Q(s_8, D)$=[−2.71]
$Q(s_7, R)$=[−3.439]
$Q(s_6, R)$=[−4.095]
$Q(s_7, L)$=[−4.685]
$Q(s_1, L)$=[100]
$Q(s_2, L)$=[89]
$Q(s_3, L)$=[79.1]
$Q(s_9, U)$=[70.19]
$Q(s_8, R)$=[62.171]

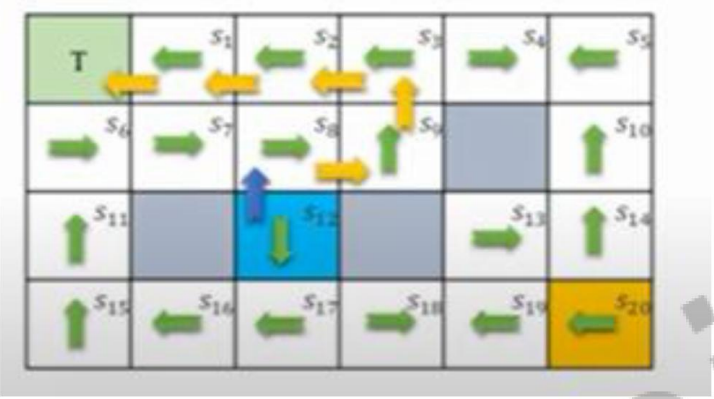
Average of the list:

Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]
Returns_list(s_1, L)=[100]
Returns_list(s_2, L)=[89]
Returns_list(s_3, L)=[79.1]
Returns_list(s_9, U)=[70.19]
Returns_list(s_8, R)=[62.171]

Generate Episode E3

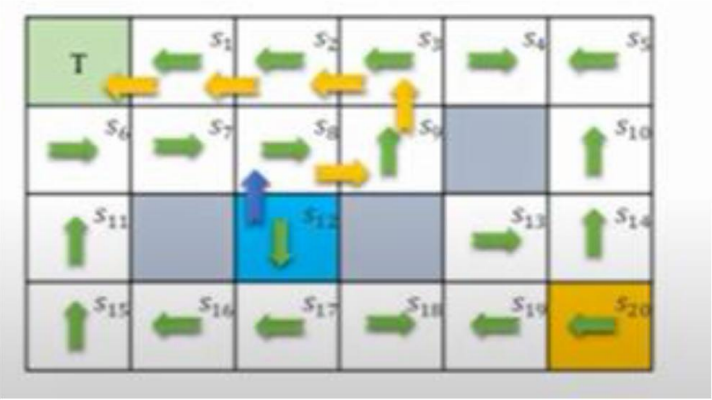
Random start state is S12 and random action is U

T=6



$$(s_{12}, U) \rightarrow (s_8, R) \rightarrow (s_9, U) \rightarrow (s_3, L) \rightarrow (s_2, L) \rightarrow (s_1, L) \rightarrow (s_0, stop)$$

Compute returns for Episode E3



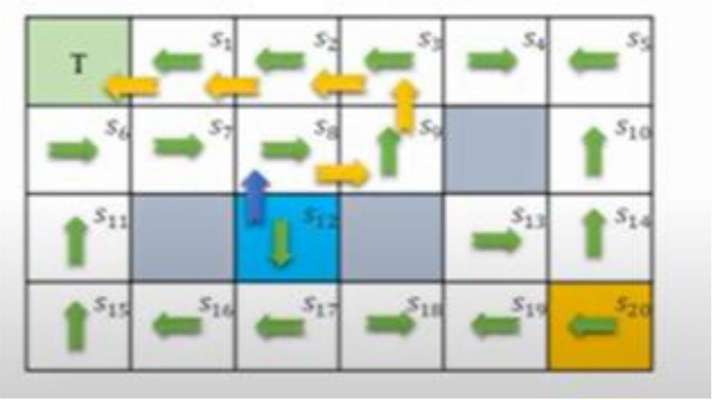
$$(s_{12}, U) \rightarrow (s_8, R) \rightarrow (s_9, U) \rightarrow (s_3, L) \rightarrow (s_2, L) \rightarrow (s_1, L) \rightarrow (s_0, stop)$$

Discounted returns: $G = \gamma G + R_{t+1}$ $\gamma = 0.9$

$(s_1, L) = (0.9 \times 0) + 100 = 100$
$(s_2, L) = (0.9 \times 100) - 1 = 89$
$(s_3, L) = (0.9 \times 89) - 1 = 79.1$
$(s_9, U) = (0.9 \times 79.1) - 1 = 70.19$
$(s_8, R) = (0.9 \times 70.19) - 1 = 62.171$
$(s_{12}, U) = (0.9 \times 62.171) - 1 = 54.953$

$(s_{12}, U) \rightarrow (s_8, R) \rightarrow (s_9, U) \rightarrow (s_3, L) \rightarrow (s_2, L) \rightarrow (s_1, L) \rightarrow (s_0, stop)$

Average the return list after episode E3



Returns obtained

$(s_1, L) = (0.9 \times 0) + 100 = 100$
$(s_2, L) = (0.9 \times 100) - 1 = 89$
$(s_3, L) = (0.9 \times 89) - 1 = 79.1$
$(s_9, U) = (0.9 \times 79.1) - 1 = 70.19$
$(s_8, R) = (0.9 \times 70.19) - 1 = 62.171$
$(s_{12}, U) = (0.9 \times 62.171) - 1 = 54.953$

Returns list after episode E2

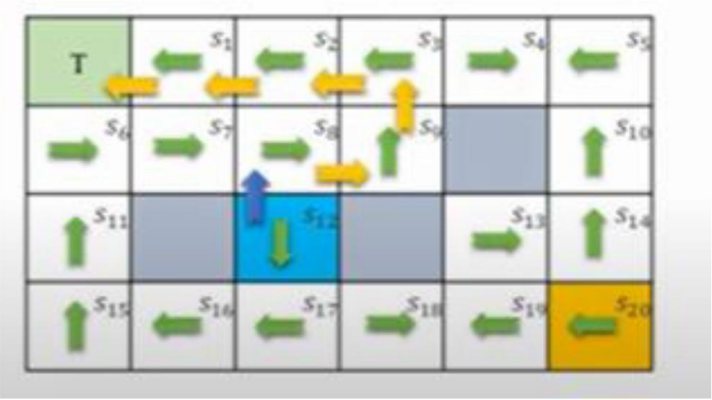
Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]
Returns_list(s_1, L)=[100]
Returns_list(s_2, L)=[89]
Returns_list(s_3, L)=[79.1]
Returns_list(s_9, U)=[70.19]
Returns_list(s_8, R)=[62.171]

Returns list after episode E3

Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]
Returns_list(s_1, L)=[100, **100**]
Returns_list(s_2, L)=[89, **89**]
Returns_list(s_3, L)=[79.1, **79.1**]
Returns_list(s_9, U)=[70.19, **70.19**]
Returns_list(s_8, R)=[62.171, **62.171**]
Returns_list(s_{12}, U)=[54.953]

Average the return list after episode E3

$$(s_{12}, U) \rightarrow (s_8, R) \rightarrow (s_9, U) \rightarrow (s_3, L) \rightarrow (s_2, L) \rightarrow (s_1, L) \rightarrow (s_0, stop)$$

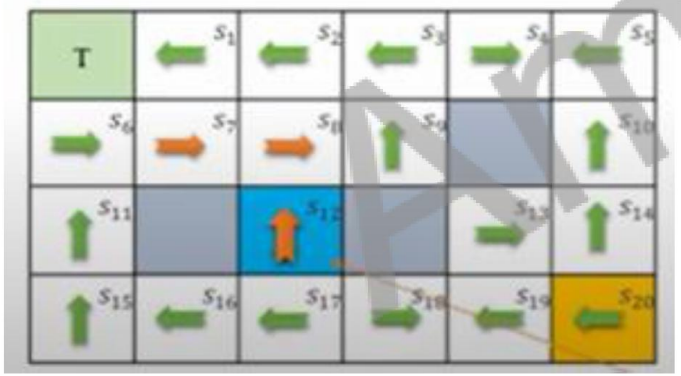


Returns list after episode E3

Returns_list(s_{17}, L)=[−1]
Returns_list(s_{12}, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]
Returns_list(s_1, L)=[100, **100**]
Returns_list(s_2, L)=[89, **89**]
Returns_list(s_3, L)=[79.1, **79.1**]
Returns_list(s_9, U)=[70.19, **70.19**]
Returns_list(s_8, R)=[62.171, **62.171**]
Returns_list(s_{12}, U)=[54.953]

Average return list and update the Q values

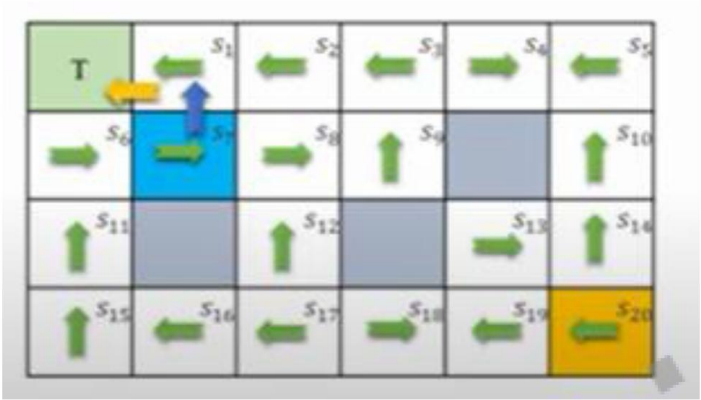
Q(s_17, L)=[−1]
Q(s_12, D)=[−1.9]
Q(s_8, D)=[−2.71]
Q(s_7, R)=[−3.439]
Q(s_6, R)=[−4.095]
Q(s_7, L)=[−4.685]
Q(s_1, L)=[100]
Q(s_2, L)=[89]
Q(s_3, L)=[79.1]
Q(s_9, U)=[70.19]
Q(s_8, R)=[62.171]
Q(s_{12}, U)=[54.953]



Generate Another Episode E4

Random start state is S7 with random action U

T=6



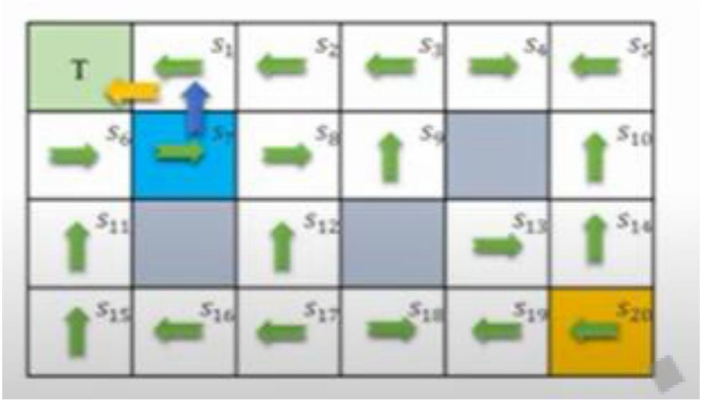
$$(s_7, U) \rightarrow (s_1, L) \rightarrow (s_0, stop)$$

Discounted returns: $G = \gamma G + R_{t+1}$ $\gamma = 0.9$

Compute returns

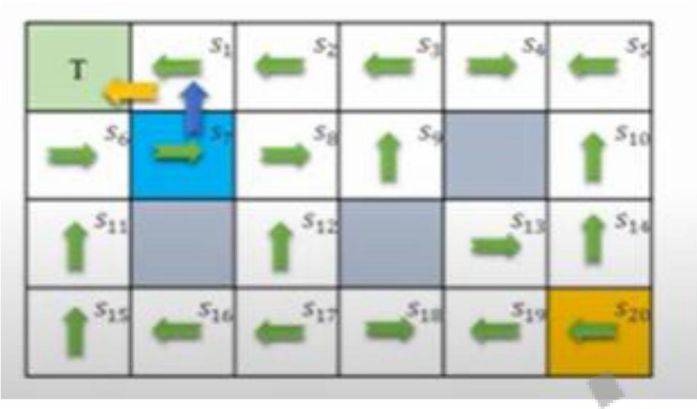
$$(s_1, L) = (0.9 \times 0) + 100 = 100$$
$$(s_7, U) = (0.9 \times 100) - 1 = 89$$

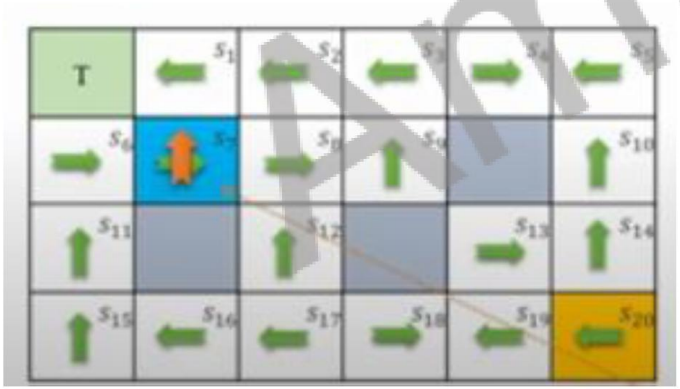
Update the return list after E4



Returns_list(s_17, L)=[−1]
Returns_list(s_12, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]
Returns_list(s_1, L)=[100, 100, **100**]
Returns_list(s_2, L)=[89, 89]
Returns_list(s_3, L)=[79.1, 79.1]
Returns_list(s_9, U)=[70.19, 70.19]
Returns_list(s_8, R)=[62.171, 62.171]
Returns_list(s_{12}, U)=[54.953]
Returns_list(s_7, U)=[89]

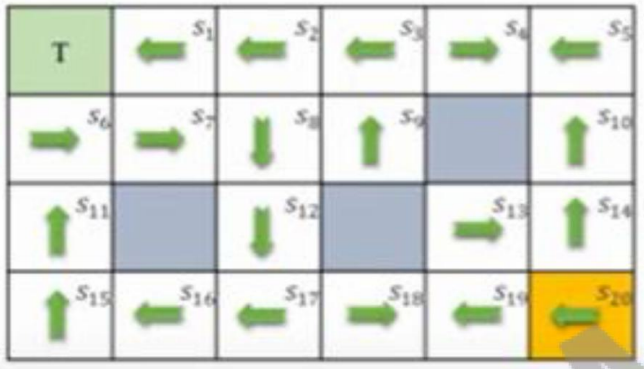
Average return list and update the policy



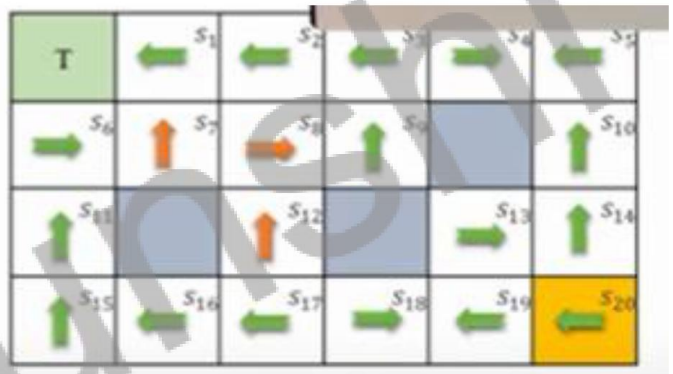


Returns_list(s_17, L)=[−1]
Returns_list(s_12, D)=[−1.9]
Returns_list(s_8, D)=[−2.71]
Returns_list(s_7, R)=[−3.439]
Returns_list(s_6, R)=[−4.095]
Returns_list(s_7, L)=[−4.685]
Returns_list(s_1, L)=[100, 100, **100**]
Returns_list(s_2, L)=[89, 89]
Returns_list(s_3, L)=[79.1, 79.1]
Returns_list(s_9, U)=[70.19, 70.19]
Returns_list(s_8, R)=[62.171, 62.171]
Returns_list(s_{12}, U)=[54.953]
Returns_list(s_7, U)=[89]

$Q(s_{17}, L)$=[−1]
$Q(s_{12}, D)$=[−1.9]
$Q(s_8, D)$=[−2.71]
$Q(s_7, R)$=[−3.439]
$Q(s_6, R)$=[−4.095]
$Q(s_7, L)$=[−4.685]
$Q(s_1, L)$=[100]
$Q(s_2, L)$=[89]
$Q(s_3, L)$=[79.1]
$Q(s_9, U)$=[70.19]
$Q(s_8, R)$=[62.171]
$Q(s_{12}, U)$=[54.953]
$Q(s_7, U)$=[89]



Arbitrary Policy



Updated Policy after 4 episodes

Monte Carlo Epsilon Greedy Policy (Without EC)

- ✓ In some problems we can not calculate all edge cases.
  - ➢ for example in an application that resetting the environment always goes back to one state not random one!
- ✓ In such cases MC Exploring start becomes infeasible!

Solution

- ✓ We **eliminate randomly selection** for all **starting points** from MC Exploring Starts.
- ✓ **Apply random policy** sometimes by **Epsilon-Greedy** technique

Monte Carlo Epsilon Greedy Policy (Without EC)

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \text{arg}\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Off Policy Prediction

- Dilemma face by control methods
  - They seek to learn action values conditional on subsequent optimal behavior, but they need to behave non-optimally in order to explore all actions (to find the optimal actions)
  - How can they learn about the optimal policy while behaving according to an exploratory policy?

Off Policy Prediction

- On-policy approach is actually a compromise
  - It learns action values not for the optimal policy, but for a near-optimal policy that still explores
- A more straightforward approach is to use two policies,
  - one that is learned about and that becomes the optimal policy
  - and one that is more exploratory and is used to generate behavior
- Policy being learned about is called the target policy,
- Policy used to generate behavior is called the behavior policy
- **In this case we say that learning is from data "off" the target policy, and the overall process is termed off-policy learning**

On Policy & Off Policy comparison

- On-policy methods are generally simpler and are considered first
- Off-policy methods require additional concepts and notation,
- The data is due to a different policy and hence off-policy methods are often of greater variance and are slower to converge
- On the other hand, off-policy methods are more powerful and general
- Off policy methods include include on-policy methods as the special case in which the target and behavior policies are the same

Off Policy Prediction

- Suppose we wish to estimate v_π or q_π, but all we have are episodes following another policy b, where $b \neq \pi$
- In this case, π is the target policy, b is the behavior policy, and both policies are considered fixed and given

Off Policy Prediction

- In order to use episodes from b to estimate values for π, we require that every action taken under π is also taken, at least occasionally, under b
- That is, we require that $\pi(a|s) > 0$ implies $b(a|s) > 0$
- This is called the assumption of $coverage$
- It follows from coverage that b must be stochastic in states where it is not identical to π
- The target policy π, on the other hand, may be deterministic, and, in fact, this is a case of particular interest in control applications
- In control, the **target** policy is typically the deterministic greedy policy with respect to the current estimate of the action-value function.
- This policy becomes a deterministic optimal policy while the **behavior** policy remains stochastic and more exploratory, for example, an ϵ-greedy policy
- In prediction problem, π is unchanging and given

Importance Sampling

- **Importance sampling** is a Monte Carlo method for evaluating properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest
  - Ref: Wikipedia

Off policy methods using importance sampling

- Importance sampling is a general technique for estimating expected values under one distribution given samples from another
- We apply importance sampling to off-policy learning by weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies, called the importance-sampling ratio

Off policy methods using importance sampling

- Relative probability of the trajectory under the target and behavior policies (the importance sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k,A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k,A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}.$$

Although the trajectory probabilities depend on the MDP's transition probabilities, which are generally unknown, they appear identically in both the numerator and denominator, and thus cancel.

The importance sampling ratio ends up depending only on the two policies and the sequence, not on the MDP