UNIT-4

MONTE CARLO METHODS

- Monte Carlo methods allow RL agent to directly learn the value functions from experience.
- The experience is o two types:
  - Actual Experience – agent directly interact with the real world
  - Simulated Experience – agent interacts with the simulated environment
- MC methods does not require the probability distribution of the state and reward signal.
- MC methods just need what is the current state and what is the next state and action.
- MC methods solve RL problem based on averaging of sample returns.
- In MC methods episodic tasks on considered but no continuing tasks.

Monte Carlo Prediction

- It involves learning the state values function for a given policy ($V\pi(S)$).
- The value of a state is defined as the expected return starting from that state.
- In MC prediction, there are 2 methods for computing the state value
  1. First visit MC method
  2. Every visit MC method
- Suppose we wish to estimate Vπ(S) – the value of ‘g’ state ‘S’ under policy π give a set of episodes obtained by following π and passing through ‘S’.
- Each occurrence of state S in an episode is called a visit to S.

$$E_1: S_4 \rightarrow S_1 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_5$$

- FIRST VISIT MC - computes Vπ(S) as the average of the returns following first visit to S.
- EVERY VISIT MC – computes Vπ(S) as the average of the returns following all visits to S.

Algorithm First Visit MC method

Initialize:

$\pi \leftarrow$ Policy to be evaluated
$V \leftarrow$ An arbitrary state value function
Return(S) $\leftarrow$ An empty list, for all s ϵS.

Repeat forever:

Generate an episode using π

For each state S appearing in the episode

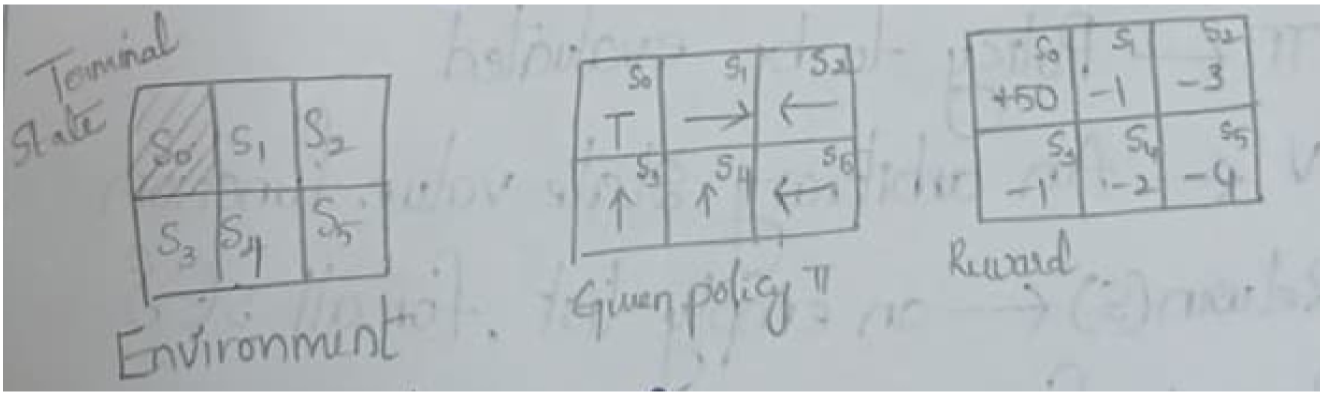
G $\leftarrow$ return following the first occurrence of S.

Append G to Returns (S)

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

End

Example:



Generate episodes (T=3)

E_1:$S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_1$

E_2:$S_3 \rightarrow S_0$

E_3:$S_5 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2$

Compute the value of S_1

E_1: $S_1 \rightarrow S_2 \rightarrow S_1$ $\therefore G = -3+(-1) = -4$

E_2: NULL

E_3: $S_1 \rightarrow S_2$ $\therefore G = -3$

$V(S_1)$ = Avg return = $(-4+(-3)) / 2 = -3.5$

Compute the value of S_2

E_1: $S_2 \rightarrow S_1$ $\therefore G = -1$

E_2: NULL

E_3: S_2 is terminal state $\therefore G = 0$

$V(S_2)$ = Avg return = $(-1+0) / 2 = -0.5$

Algorithm Every Visit MC method

Initialize:

$\pi \leftarrow$ Policy to be evaluated
$V \leftarrow$ An arbitrary state value function
Return(S) $\leftarrow$ An empty list, for all s ϵ S.

Repeat forever:

Generate an episode using π

For each state S appearing in the episode

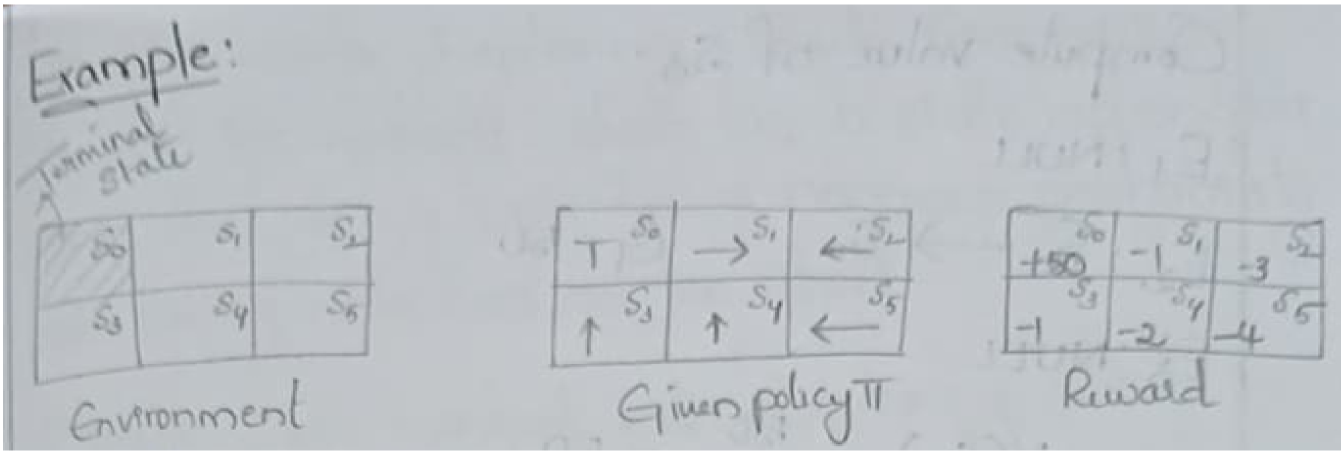
G $\leftarrow$ return following the first occurrence of S.

Append G to Returns (S)

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

End

Example:



Generate episodes (T=3)

E_1:$S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_1$

E_2:$S_3 \rightarrow S_0$

E_3:$S_5 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2$

Compute the value of S_1

E_1: $S_1 \rightarrow S_2 \rightarrow S_1$ $\quad \therefore$ G = -3+(-1) = -4

E_1: $S_2 \rightarrow S_1$ $\quad \therefore$G = 0

E_2: NULL

E_3: $S_1 \rightarrow S_2$ $\quad \therefore$ G = -3

V(S_1) = Avg return = (-4+0+(-3)) / 3 = -2.33

Compute the value of S_2

E_1: $S_2 \rightarrow S_1$ $\quad \therefore$G = -1

E_2: NULL

E_3: S_2 is terminal state $\quad \therefore$ G = 0

E_3: $S_1 \rightarrow S_2$

V(S_2) = Avg return = (-1+0) / 2 = -0.5

Compute the value of S_3

E_1: NULL

E_2: $S_3 \rightarrow S_0$ $\quad\quad\therefore$ G = 50

E_3: NULL

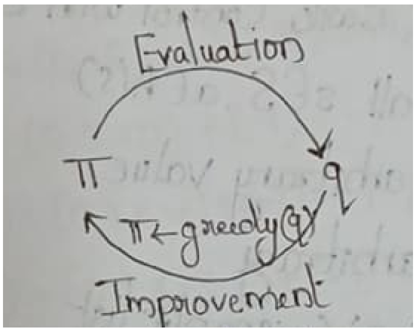
$V(S_3)$ = Avg return = 50/1 = 50

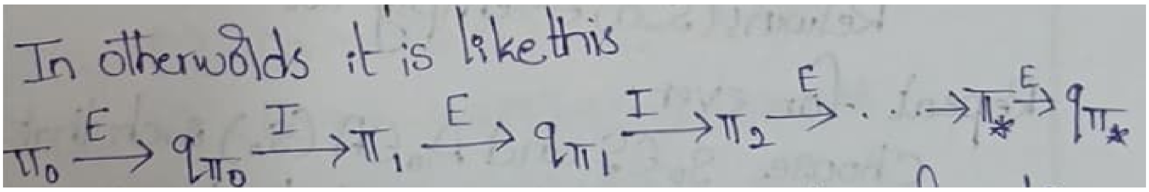
Monto Carlo Estimation of Actions Values:

- In some of the problems model of the environment is not available. Ex: - Autonomous Car
- For such problems it is useful to estimate action value rather than state value.
- To prove this first consider the policy evaluation problem for action values.
- The policy evaluation problem for action value is to estimate $q\pi(S,a)$.
- $q\pi(S,a)$ is the expected return when starting in state S, the taking action 'a' and thereafter following policy π.
- One technique used to compute $q\pi(S,a)$is **Monte Carlo Exploring Starts (MCES)**

Monto Carlo Exploring Starts (MC – ES)

- Uses Monte Carlo Control.
- Given a policy, evaluate this policy by computing the q values and then improve the policy using greedy q value. This process will be repeated until we get an optimal policy.





The greedy policy for any action value function q is given as

$$\Pi(S)\ \underset{a}{\operatorname{argmax}}\ q(S,a)$$

Policy improvement can be done by constructing π_{k+1} as the greedy policy w.r.t $q\pi_k$

This can be written as

$$q\pi_k(S, \pi_{k+1}(s)) = q\pi_k(s, \operatorname{argmax} q\pi_k(S,a))$$

$$= \max_{a} q\pi_k(S,a)$$

$$\geq q\pi_k(s, \pi_k(S))$$

$= V\pi_k(S)$

Algorithm Monte Carlo Control with Exploring Starts

Initialize, for all s ϵS, a ϵA(S)

Q(S,a) $\leftarrow$ arbitrary value

π(S) $\leftarrow$ arbitrary

Returns(S,a) $\leftarrow$ empty list

Repeat for ever:

Choose S_0 ϵS and A_0 ϵA(S_0) such that all pairs have probability > 0

Generate an episode starting from S_0, A_0 following π

For each pair S,a appearing in the episode :

G $\leftarrow$ return following the first occurrence of S,a

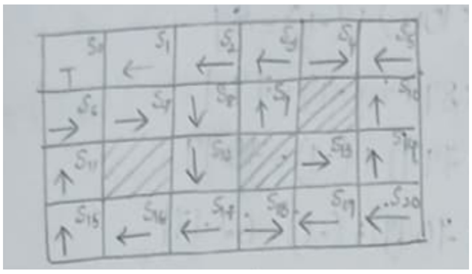
Append G to Returns (S,a)

Q(S,a) $\leftarrow$ avg (returns(S,a))

For each S in the episode

$$\Pi(S)\ \underset{a}{\text{argmax}}\ Q(S,a)$$

Example:



Reward of S_0 = 100

Reward of all other states =-1, γ = 0.9

G = γG + R_{t+1}

1) (S_7,L)

(S_7,L) = (S_6,R) $\rightarrow$ (S_7,R) $\rightarrow$ (S_8,D) $\rightarrow$ (S_{12},D) $\rightarrow$ (S_{17},L) $\rightarrow$ (S_{16},L)

Q(S_7 ,L) = (0.9 * 0)+(-1) =-1

$Q(S_{12}, D) = (0.9 * -1)+(-1) = -1.9$

$Q(S_8, D) =(0.9 * -1.9)+(-1) = -2.71$

$Q(S_7 ,R) =(0.9 * -2.71)+(-1) = -3.439$

$Q(S_6, R) =(0.9 * -3.439)+(-1) = -4.095$

$Q(S_7,L) =(0.9 * -4.095)+(-1) = -4.685$

$\Pi(S_7) = \max\{Q(S_7,R),Q(S_7,L)\}$

$= \max\{-3.439,-4.685\}$

$=-3.439$

2) (S_8,L)

$(S_8,R) \rightarrow (S_9,U) \rightarrow (S_3,L) \rightarrow (S_2,L) \rightarrow (S_1,L) \rightarrow (S_0,Stop)$

$Q(S_1 ,L) = (0.9 * 0)+(100) =100$

$Q(S_2, D) = (0.9 * 100)+(-1) = 89$

$Q(S_3, D) =(0.9 * 89)+(-1) = 79.1$

$Q(S_9 ,R) =(0.9 * 79.1)+(-1) = 70.19$

$Q(S_8, R) =(0.9 * 70.19)+(-1) = 62.171$

$\Pi(S_7) = \max\{Q(S_8,D),Q(S_8,R)\}$

$= \max\{-2.71,62.171\} = 62.171$ [Take Right in S_8]

3) (S_{12},U)

$(S_{12},U) \rightarrow (S_8,R) \rightarrow (S_9,U) \rightarrow (S_3,L) \rightarrow (S_2,L) \rightarrow (S_1,L) \rightarrow (S_0,Stop)$

$Q(S_1 ,L) = (0.9 * 0)+(100) =100$

$Q(S_2, L) = (0.9 * 100)+(-1) = 89$

$Q(S_3, L) =(0.9 * 89)+(-1) = 79.1$

$Q(S_9 ,U) =(0.9 * 79.1)+(-1) = 70.19$

$Q(S_8, R) =(0.9 * 70.19)+(-1) = 62.171$

$Q(S_{12}, U) =(0.9 * 62.171)+(-1) = 54.953$

$\Pi(S_{12}) = \max\{Q(S_{12},D),Q(S_{12},U)\}$

$= \max\{-1.9,54.953\} = 54.953$ [Take Up in S_{12}]

4) (S_7,U)

$(S_7,U) \rightarrow (S_1,L) \rightarrow (S_0,Stop)$

$Q(S_1 ,L) = (0.9 * 0)+(100) =100$

$Q(S_7, U) = (0.9 * 100)+(-1) = 89$

$\Pi(S_7) = \max\{Q(S_7,D),Q(S_7,U)\}$

$= \max\{-3.439, 89\} = 89$ [Take Right in S_7]

Monte Carlo Control Without Exploring Starts

- The assumption of exploring starts may not work always.
- For example, if we want to learn directly by interacting with the env then starting conditions are not very useful.
- A more common way is to allow the agent to start at any position and to select all actions infinitely aften.
- There are 2 approaches for ensuring this
    1. On-policy methods – where we try to evaluate or improve the policy that we have.

- Only one policy exists

Ex:-

- Monte Carlo Control with exploring starts.
- Monte Carlo Control without exploring starts.
- First – visit MC method
- Every – visit MC method

2) Off policy methods – where we have 2 methods, 1 is used for evaluation and other is used for improvement.

Ex:- Monte Carlo Control with important Sampling

- Monte Carlo Control with out exploring starts is a non-policy method which generally uses $\sum$ –Soft policy.
- $\sum$ –Soft policy – It is a policy for which $\pi\left(\frac{a}{s}\right) \geq \frac{\varepsilon}{|A(S)|}$, $\forall s \epsilon S_1$, $a \epsilon A(S)$ and $\epsilon > 0$.
- Among $\sum$ –Soft policy , ε - Greedy policy is more popular . Hence MC control without Exploring starts uses ε - Greedy policy.
- The ε - Greedy policy is defined as follows
    - All non – greedy actions are selected with a probability of $\frac{\varepsilon}{|A(S)|}$
    - The greedy action is selected with a probability of 1 – $\varepsilon + \frac{\varepsilon}{|A(S)|}$

Algorithm for first visit MC control without Exploring starts

Initialize for all s ϵS, a ϵA(S)

Q(S,a) $\leftarrow$ arbitrary

Returns(S,a) $\leftarrow$ arbitrary

$\pi\left(\frac{a}{s}\right) \leftarrow$ an arbritrary $\sum$ –Soft policy

Repeat for ever

a) Generate an episode using π

b) For each pair S,a appearing in the episode

   G $\leftarrow$ return following first occurrence of S,a

   Append G to Return (S,a)

   Q(S,a) $\leftarrow$ average(Returns(S,a))

c) For each S in the episode:
   a* $\leftarrow$ Argmax Q(S,a)
   For all a ϵ A(S)

$$\pi\left(\frac{a}{s}\right)=\left\{1-\varepsilon+\frac{\varepsilon}{|A(S)|}\, if\ a=a^{¿} \wedge ¿ \frac{\varepsilon}{|A(S)|}\, if\ a\neq a^{¿}\right.$$

End

Off – Policy Prediction via Importance Sampling

Two types of MC Learning

MC – Learning

On – Policy Learning

- Uses Same policy for evaluating and improving
- Ex:

  First visit MC

  Every visit MC

  MC control with ES

  MC control without ES

- Drawback:

  It is only a compromise i.e it learns a near optimal policy but not an exact optimal policy.

Off – Policy Learning

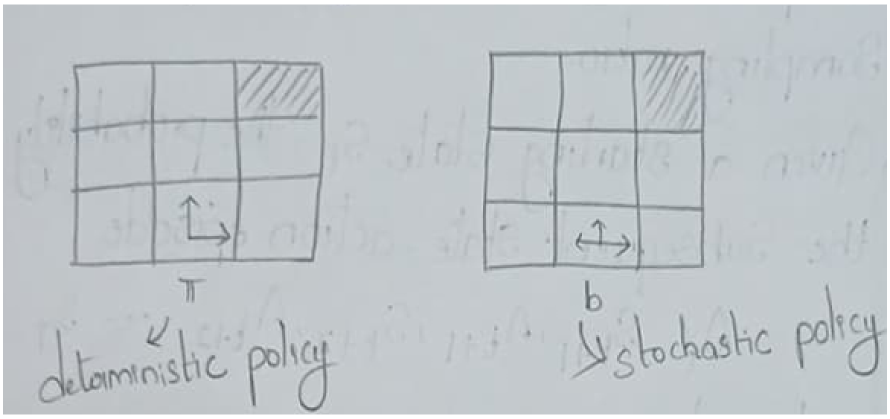
- Uses one policy for evaluating and for generating the data
- There are two policies

  Ex: Off policy prediction

- But off – Policy Learning always learns optimal policy

- Off policy methods uses two Separate policies

1. Behavior policy (b): Used to generate data.
2. Target policy (π) : is evaluated and improved to become an optiml policy based on the data generated.



- The objective is to estimate Vπ or qπ
- One requirement for doing this is coverage criteria.

Coverage Criteria:

- Every action taken under policy π must also be taken under policy b. Hence, it follows that

$$\pi(S) > 0 \text{ ¿>¿ } b(S) > 0 \ \forall s \epsilon S$$

- One way to ensure this requirement is using importance Sampling.

Importance Sampling

- Importance Sampling takes the returns of the episodes and weights them relatively based on the probabilities of occurring of them in both the policies. It is called the importance Sampling ratio.
- Given a starting state S_t, the probability of the Subsequent state – action episode A_t, S_{t+1}, A_{t+1}, S_{t+2}, A_{t+2}, ..., S_T under policy π is given as

$$P_r\left\{A_t, A_{t+1}, \ldots\ldots, \frac{S_T}{S_t}, A_{t:T-1} \quad \pi\right\} = \pi\left(\frac{A_t}{S_t}\right), P\left(\frac{S_{t+1}}{A_{t+1}}\right), \pi\left(\frac{A_{t+1}}{S_{t+1}}\right), P\left(\frac{A_{t+2}}{S_{t+2}}\right), \ldots\ldots, P\left(\frac{S_T}{S_{T-1}}, A_{T-1}\right)$$

$$¿\prod_{K=t}^{T-1} \pi\left(\frac{A_K}{S_K}\right), P\left(\frac{S_{K+1}}{S_K}, A_K\right)$$

- Now consider the importance Sampling ratio,
- It is given as

$$\frac{\int ¿\left(\prod_{K=t}^{T-1} \pi\left(\frac{A_K}{S_K}\right), P\left(\frac{S_{K+1}}{S_K}, A_K\right)\right)}{\prod_{K=t}^{T-1} b\left(\frac{A_K}{S_K}\right), P\left(\frac{S_{K+1}}{S_K}, A_K\right)} = \prod_{K=t}^{T-1} \pi\left(\frac{A_K}{S_K}\right), b\left(\frac{A_K}{S_K}\right)$$

- Once, we compute $\int$ then we can estimate V(S) as

$$V(S) = \sum_{t \in \tau(S)} \frac{\left(\int_{t:T-1} G_t\right)}{¿\tau(S) \vee ¿} ¿$$

- Here, $\tau(S)$ is the set of all the time stamps when S is visited.

V(S) = $(\sum_{t \in \tau(S)} \int_{t:T-1} G_t) / ¿ (\sum_{t \in \tau(S)} \int_{t:T-1}) ¿$

Incremental Implementation

- Suppose we have a sequence of returns G_1, G_2.....G_{n-1}, all starting in the same state and the corresponding weights w_1, w_2.....w_{n-1} then V_n can be computed as

$$V_n = \frac{\sum_{k=1}^{n-1} w_k G_k}{\sum_{k=1}^{n-1} w_k}, n \geq 2$$

- Instead, computing V_n using above equation. We can efficiently compute V_n incrementally as

$$V_{n+1} = V_n + \frac{w_n}{C_n}[G_n - V_n]$$

And C_{n+1} = $C_n + w_{n+1}$

And C_0 = 0