श्री गणेशाय : नमः

Reinforcement Learning -

Problem Formulation and Terminology :-

Agent : (Robot)

Environment : Setup of a problem.

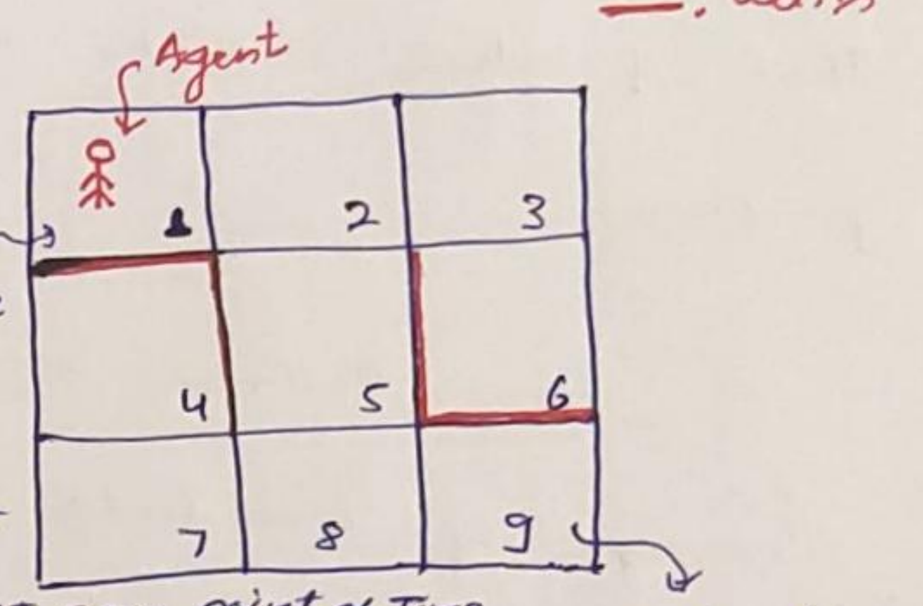
States : L1 - L9 (Nine Locations) state

Tells you what is the current state of an agent in the environment.

Actions : → ← ↓ ↑ All actions that an agent can take at any point of Time.

Reward : Reward is a function on which you perform an action R(S, a) on state S and gets the instantaneous Reward.

eg- R(L, →) ⟶ R(2,).



Bellman Equations :

value-function : V(S) :- value function is a function on a given state, you return a numerical value called as value in the state S.

$S \xrightarrow{a} S'$ (current state, next state)

eg- 1 → 2

2 ↓ 5

$V(S) = \max_{a \in} \left(R(S, a) + \gamma \cdot V(S') \right)$

immediate Rewards; value of next state (value of future states)

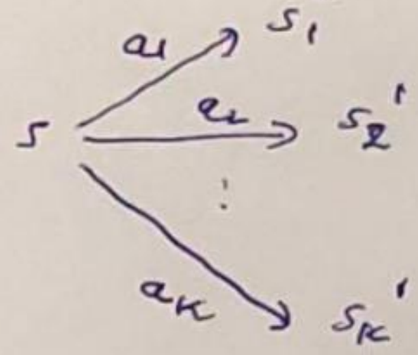
a ∈ a, a₂ ... aₙ

0 ≤ discounting Factor ≤ 1

Bellman equations.

Bellman equations says that the value in any state S. is the maximum of the immediate rewards that you will get by performing all actions on state S.

↳ Bellman equations can be represented as an recursive equation.



↳ That problem can be represented As an optimization problem.

↳ So a Bellman equation represent the weighted (sum) average of immediate returns that we are getting and not giving much importance to future rewards by using a discount factor. ($0 \leq \gamma \leq 1$)

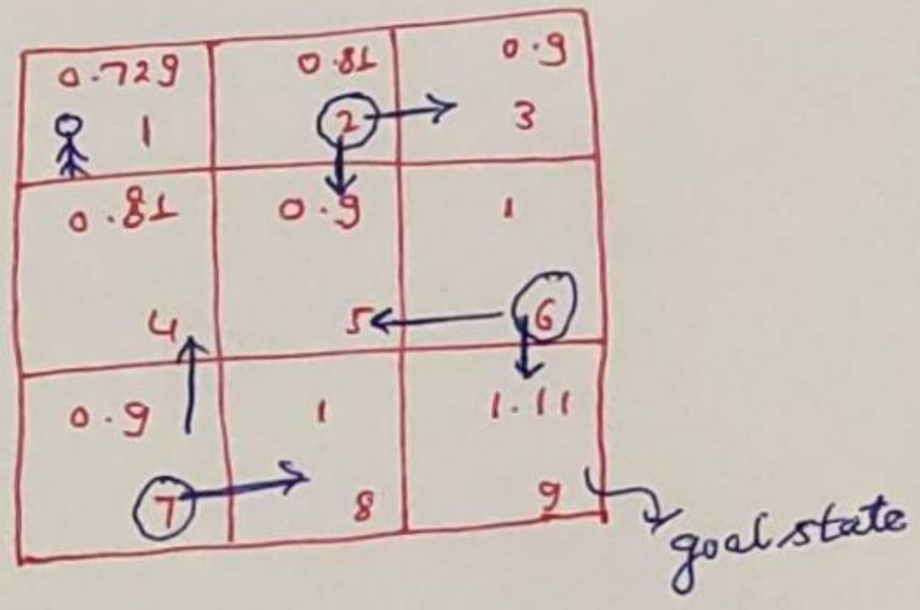
(Constant Rewards) future value

eg- $V(s) = \max_a (R(s,a) + \gamma V(s'))$

↳ Discount factor

Let $\gamma = 0.9$

Assume $R(s,a) = 0 \quad \forall \; s \notin L9$



$V(L9) = 1.11$

$V(L6) = ?$

$V(s) = \max_a (\gamma V(s'))$

$V(L9) = 1.11$

$V(L6) = 1 \quad (0.9 \times 1.11)$

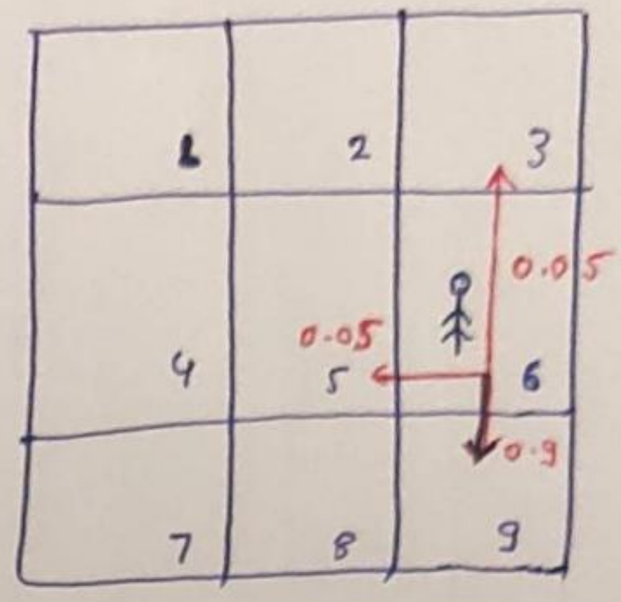
$V(L8) = 1 \quad (0.9 \times 1.11)$

$V(L7) = 0.9$

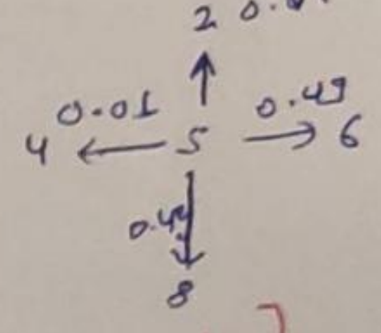
$V(L2) = 0.81$

↳ Always go to higher state than the current state if we are able to obtain the value state matrix.

Markov Decision Process:- (stocastic Process):-



means at each state instead of a one deterministic action, in MDP (stocastic Process), probabilities are associated with each action then such type of a system is called as MDP.



$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s,a,s') \cdot V(s') \right]$$

mathematical representation of this eqn.

Expected value = $E[V(s')]$

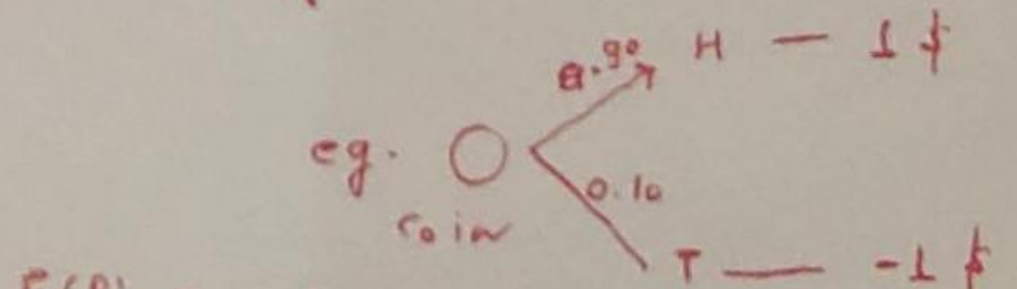
$0.9 * V(L9) +$

$0.05 * V(L3) +$

$0.05 * V(L5)$

$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s') \right]$$

Expected value of $V(s')$



$E(P) = 0.90 \times 1 + (0.10 \times -1) = 0.8$

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Introduction to Reinforcement Learning

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

श्री गणेशाय : नमः

POORNIMA

POORNIMA
COLLEGE OF ENGINEERING
DETAILED LECTURE NOTES

PAGE NO. .....

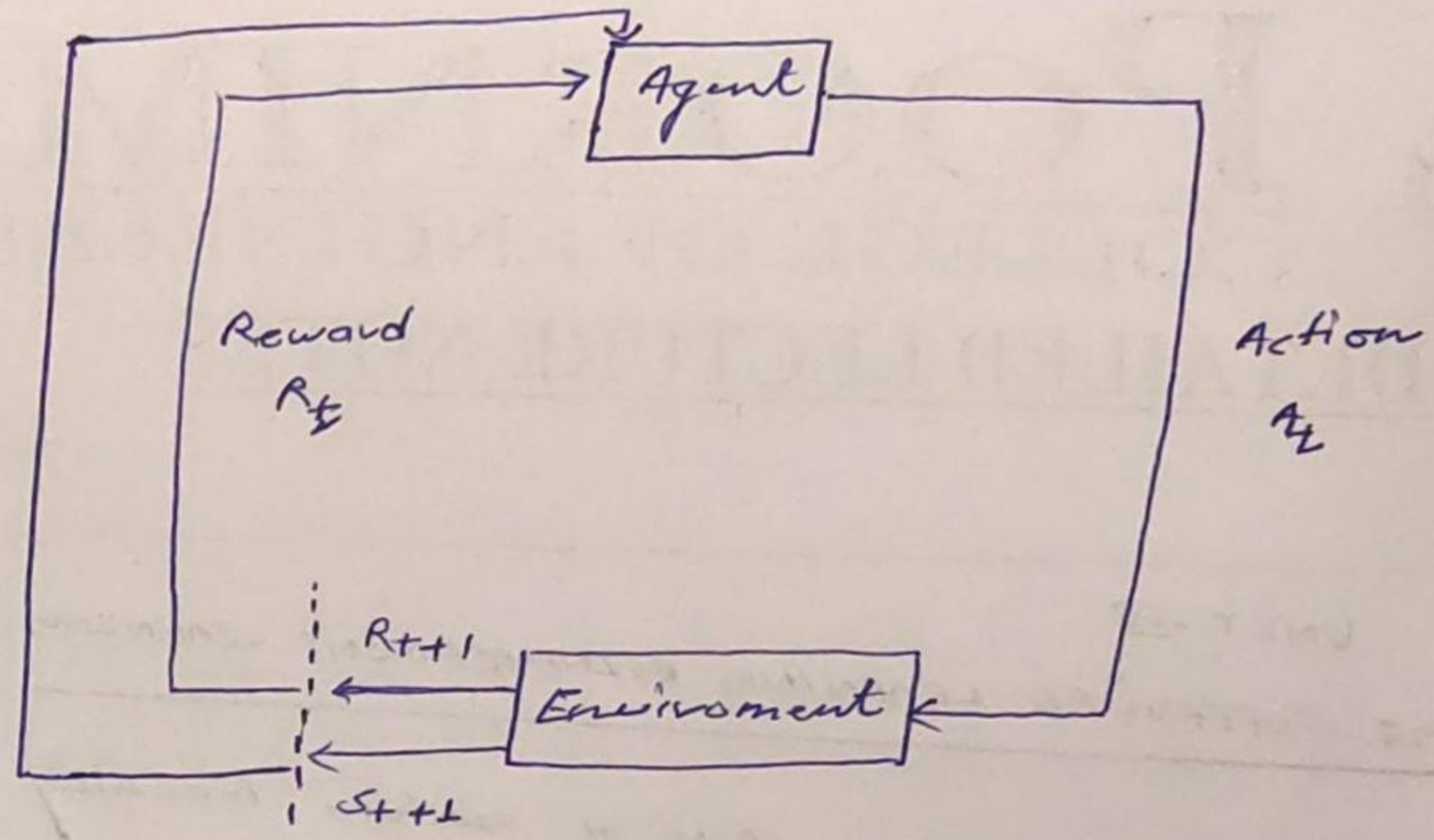
UNIT - IV

SEMI SUPERVISED LEARNING, REINFORCEMENT LEARNING

Reinforcement Learning:- is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error by using feedback from its own actions and experience.

↳ Reinforcement learning uses rewards and punishments as signals for positive and negative behaviour.

↳ Objective of reinforcement learning is to find a suitable action model that would maximize the total cumulative reward of the agent



Key- Elements of RL :-

Agent :- An agent is a s/w program that learn to make intelligent decisions.

eg- chess player
Mario

Environment :- The environment is the world of the agent. The agent stays within environment.

eg- chessboard. & Environment chessplayer & Agent
Super Mario Bros Mario

State and Action :- A state is a position or a moment in the environment that the agent can be in.

→ state is usually denoted by S.

The Agent interacts with the environment and moves from one state to another by performing in an action.

→ The Action is usually denoted by a.

Rewards :- The agent interacts with the environment by performing an action and moves from one state to another.

A reward is nothing but a numerical value, say +1, for a good action and -1 for a bad action.

↦ Reward is denoted by r.

In RL setting, we will not teach the Agent what to do or how to do it. instead we will give a reward to the Agent for every action it

it does. We will give +ve reward to the agent when it perform a good action and negative award to the agent when it performs a bad action.

→ So RL can be viewed as a trial and error learning process where the agent tries out different actions and learns the good actions, which gives a positive reward.

The RL ALGORITHM:-

step1: Firstly, the agent interacts with the enviroment by performing an action.

step2: By performing an action, the agent moves from one state to another.

step3: The Agent will receive an 'rl' award based on action it performed.

step4: Based on reward, the agent will understand whcather the action is good or bad.

step5:- if the action was good, i.e if the agent received a +ve reward, then the agent will prefer performing that action, else the

the Agent will try perform other actions in

search of a +ve reward.

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Terms Related to Reinforcement Learning

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

PAGE NO. .....

policy :

→ Deterministic policy - tells the Agent to perform one particular action in a state.

$a_t = \mu(s_t)$

a_t ← Action At Time T. (Own Action).

μ ↓ policy

s_t → state s at Time T

for eg $\mu(A) = \text{Down}$

ii) stochastic policy :- Maps the state to a probability distribution over an action space.

Means for a given state s, the stochastic policy will return a prob. dist over an action space, Here the agent will perform different actions each time based on prob. distribution returned by stochastic policy.

$a_t \sim \pi(s_t)$

Episode :- Agent - environment interaction starting from the initial state until final state is reached is called an episode.

↳ Also known as Trajectory. denoted by τ.

objective.
↳ To learn optimal policy.

episode $(s_0, a_0 r_0, s_1, a_1, r_1 \ldots\ldots s_T)$
s_0 ↑ Initial state
s_T ↑ Final state

Finite Horizon - If the agent - environment interaction stops at a particular time stemp, then horizon is called a finite horizon.

Infinite Horizon - If the " " " never stops then it is called Infinite horizon.

Return and discount Factor :-
↳ A return can be defined as the sum of rewards obtained by agent in an episode.

$$R(\tau) = r_0 + r_1 + r_2 + \ldots + r_T$$

$$R(\tau) = \sum_{t=0}^{T} r_t$$

↳ Goal of our agent is to maximize the return.

Discrete Tasks -
continuous Tasks

↳ $R(\tau) = r_0 + r_1 + r_2 + \ldots + r_\infty$

$R(\tau) = \gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots + \gamma^\infty r_\infty$

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

Discount-factor value
- small → more importance to immediate rewards than future rewards
- large → more importance to future rewards compared to immediate rewards
- = 0 → nothing learns, since it concentrate on immediate reward.
- = 1 → learning continuous every time

The value Function - Also known as state-value function, denotes the value of a state

$$v^{\pi}(s) = [R(\tau) \mid s_0 = s]$$

↳ value function can be defined as expected return that the agent would obtain starting from state s by following policy π.

$$V^{\pi}(s) = E\left[R(\tau) \mid s_0 = s\right]$$

$\tau \sim \pi$

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: IV

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Markov Decision Process

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO. .....

Markov Decision Process:- provides a mathematical framework for solving the RL problem.

The Markov property and Markov chain:-

The Markov property states that the future depends only on present and not on the past. The Markov chain, also known as Markov process, consisting a sequence of states that strictly obey the Markov property.

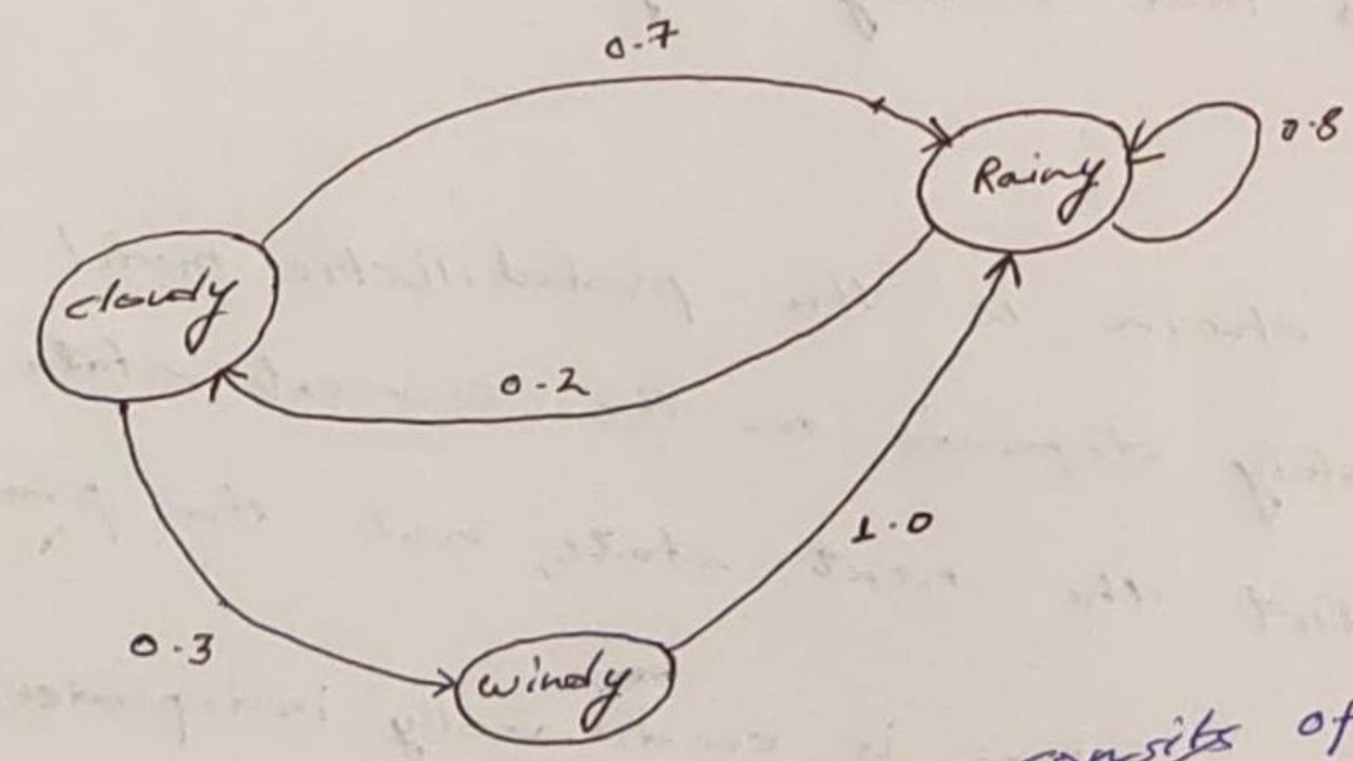
↳ Markov chain is the probabilistic model that solely depends on the current state to predict the next state, not the previous states, i.e. future is conditionally independent of the past.

Transition Probability:- Moving from one state to another is called a transition and its probability is called as Transition Probability.

Is denoted by $P(S'|S)$.

eg-

Current state	Next state	Transition Probability
cloudy	Rainy	0.7
cloudy	windy	0.3
Rainy	Rainy	0.8
Rainy	cloudy	0.2
Rainy	Rainy	1.0
windy	Rainy	



Markov chain or Markov process consits of a set of states along with their transition probabilities.

Markov Reward Process :- (MRP):

↳ MRP is an extension of Markov chain along with one additional function i.e reward function.

↱ so MRP consists of states, a transition probability and also a reward function.

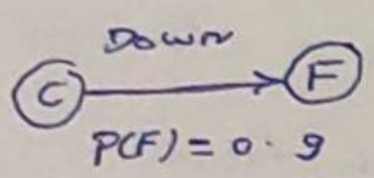
↱ Reward function denotes rewards that we receive in each state. Reward function is denoted by R(s).

↱ MRP consist of states S, a transition prob. P(s'|s), and a reward function R(s).

The Markov Decision Process :- MDP is an extension of MRP with actions.

so MDP consist of set of states S, a Transition probability $P(S'|S,a)$, a reward function $R(S)$ $R(S,a,S')$

and also action.

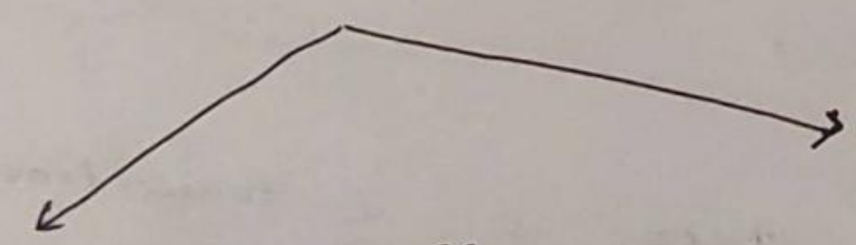


$R(A, right, B) = -1$

↳ $E(X) = \sum_{i=1}^{N} x_i P(x_i)$

Action space :- The set of all possible actions in the enviroment is called as Action space.

eg - Grid World - Action space - { Left, Right, UP, Down }



Policy :- A policy defines the agent behaviour in an enviroment.

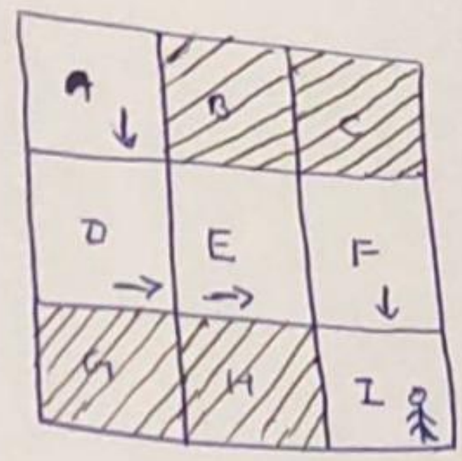
policy tells the agent what action to perform in each state.

↱ Firstly we choose random policy.

PAGE NO. .....

optimum policy

State	Action
A	Down
D	Right
E	Right
F	Down



↳ optimum policy tells the agent to perform the correct action in each state so that the agent can receive good reward.

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: IV

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

BELLMAN EQUATION

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

श्री गणेशाय नमः

POORNIMA
COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO. .....

BELLMAN EQUATION:- The bellman equation, named after Richard Bellman, helps us to solve Markov decision process. (MDP).

↳ BELLMAN Equation widely used for finding the optimal value and Q function recursively. that can be used to find the optimal policy.

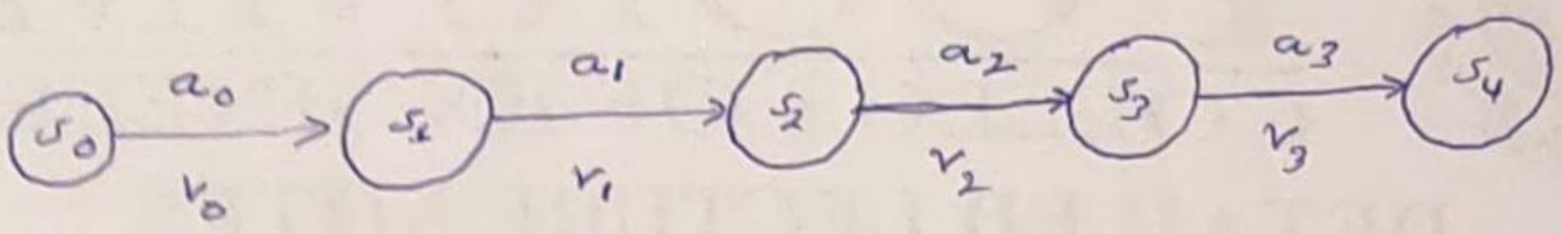
BELLMAN EQUATION FOR VALUE FUNCTION:- BELLMAN EQUATION states that the value of a state can be obtained as a sum of immediate reward and discounted value of the next state.

- BELLMAN Equation of the value function can be expressed as-

$$V(s) = R(s, a, s') + \gamma V(s')$$

Here $R(s, a, s')$ - implies the immediate reward obtained while performing an action a in state s and moving to N.S s'.

For eg- Say we generate a trajectory I using some policy π:



Let suppose we need to compute the value of state S2.

So According to BELLMAN Equation-

$$V(s_2) = \underbrace{R(s_2, a_2, s_3)}_{r_2} + \underbrace{\gamma V(s_3)}_{\text{Discounted value of next state.}}$$

$$V(s_2) = r_2 + \gamma V(s_3)$$

In More Generalised Form-

$$\boxed{V^{\pi}(s) = R(s, a, s') + \gamma V^{\pi}(s')}$$

where π implies that we are using policy π. RHS Term $R(s, a, s') + \gamma V^{\pi}(s')$ is often called as - BELLMAN backup.

Note- Previous equation will work only when we have deterministic environment.

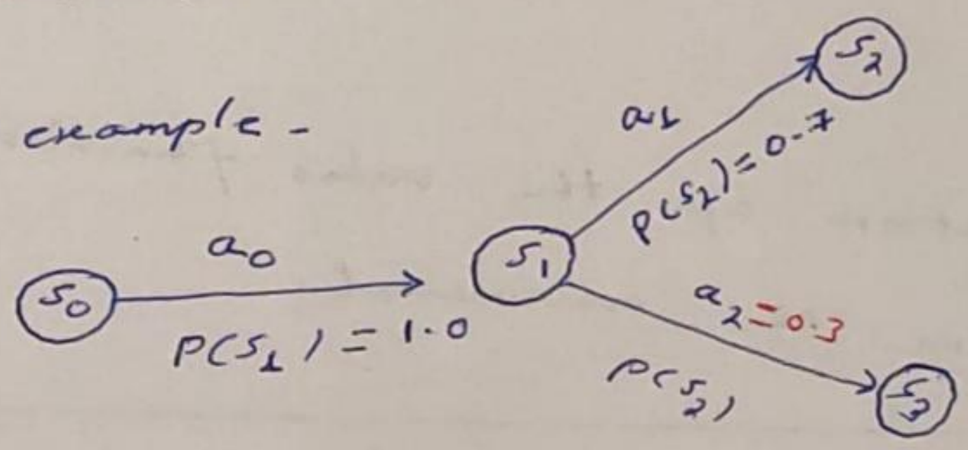
Q- Function :- Q-function is a

POORNIMA
COLLEGE OF ENGINEERING
DETAILED LECTURE NOTES

PAGE NO.

For stochastic Environment-

For example -



Here our Bellman equation with the expectation (i.e weighted Average), i.e sum of Bellman Backup multiplied by the corresponding Transition probability.

$$V^{\pi}(s) = \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma V^{\pi}(s') \right]$$

Here $P(s' \mid s, a)$ - denotes the transition probability of reaching s' by performing an action a in state s.

$[R(s, a, s') + \gamma V^{\pi}(s')]$ - Bellman backup.

Thus we can write.

$$V(s_1) = P(s_2 \mid s_1, a_1)\left[R(s_1, a_1, s_2) + V(s_2)\right]$$

$$+ P(s_3 \mid s_1, a_1)\left[R(s_1, a_1, s_3) + V(s_3)\right]$$

$$V(s_1) = 0.70\left[R(s_1, a_1, s_2) + V(s_2)\right] + 0.30\left[R(s_1, a_1, s_3)\right.$$

$$\left. + V(s_3)\right]$$

Thus the bellman equation of the value function including stochasticity present in the environment.

$$V^{\pi}(s) = \sum_{s'} P(s' \mid s, a)\left[R(s, a, s') + \gamma V^{\pi}(s')\right]$$

→ decide the quality of an action.

BELLMAN EQUATION of the Q Function :- states that the Q value of a state-action pair can be obtained as a sum of the immediate reward and the discounted Q value of a next state action pair.

$$q(s, a) = R(s, a, s') + \gamma q(s', a')$$

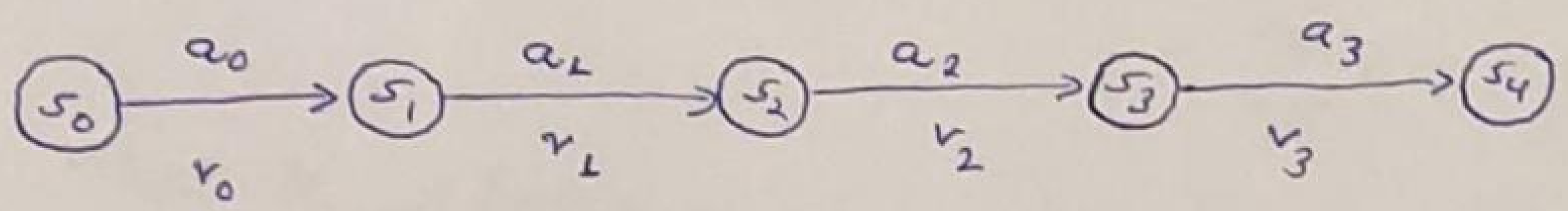
Here $R(s, a, s')$ - implies the immediate reward obtained while performing an action a in state s and moving to the next state s'.

γ - is the discount factor.

$V(s)$ = max [$R(s,a)$ + γ $\sum$ $P(s$ ... of state s
... take ... a given state s which is the best
inorder to get the highest reward.

$Q(s', a')$ - is the Q value of next-state-action pair.

For eg-



Suppose we need to compute the Q-value of a state-action pair (s_2, a_2).

$$Q(s_2, a_2) = \underbrace{R(s_2, a_2, s_3)}_{\text{immediate Reward}} + \underbrace{\gamma Q(s_3, a_3)}_{\text{discounted Q value of next-state-action pair.}}$$

$$\boxed{Q(s_2, a_2) = r_2 + \gamma Q(s_3, a_3)}$$

In More Generalized - representation -

$$\boxed{Q^{\pi}(s, a) = R(s, a, s') + \gamma Q^{\pi}(s', a')}$$

Here π - implies that we are using the policy π and RHS is Bellman Backup.

Scanned by TapScanner

Th...

For stochastic Environment -

$$q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)\left[R(s,a,s') + \gamma q^{\pi}(s'a')\right]$$

Q- Function :- Q-function is a simple function of state s and action a.

Q- function describe at a given state s which is the best action should I take inorder to get the highest reward.

$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s,a,s') . V(s') \right]$$

↓

$Q(s,a)$:- Q- Function

↳ Goal of Q function is to maximize the total reward

so Deep - Q - Learning is all about computing the Q function

$$V(s) = \max_a Q(s,a)$$

. so $Q(s,a) = R(s,a) + \gamma . \sum_{s'} P(s,a,s') . V(s')$

$E[V(s')]$

can be expressed as

$\max_{a'} Q(s'a')$

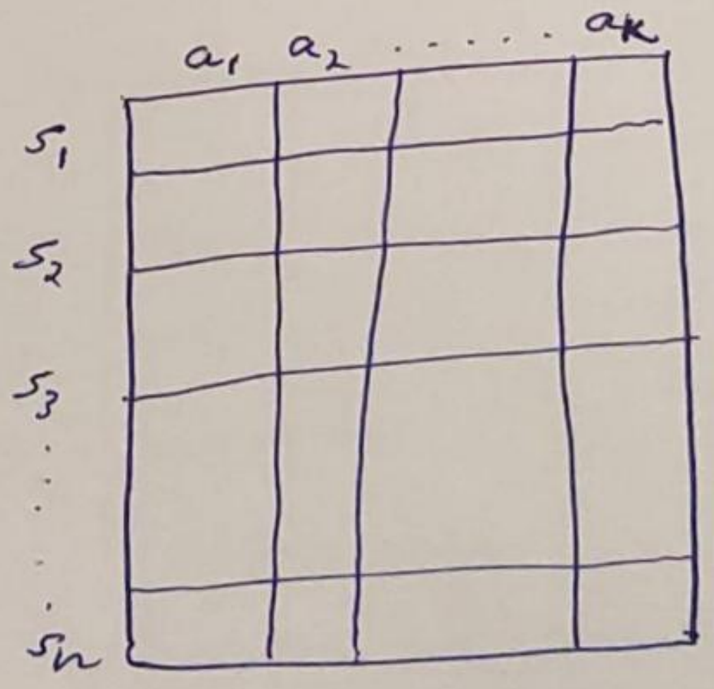
$$Q(s,a) = R(s,a) + \gamma \sum_{s'} \left[P(s,a,s') . \max_{a'} Q(s',a') \right]$$

Here $Q(s,a)$ is also a recurrence (recursive equation)

Let $P(s,a,s') = 1$

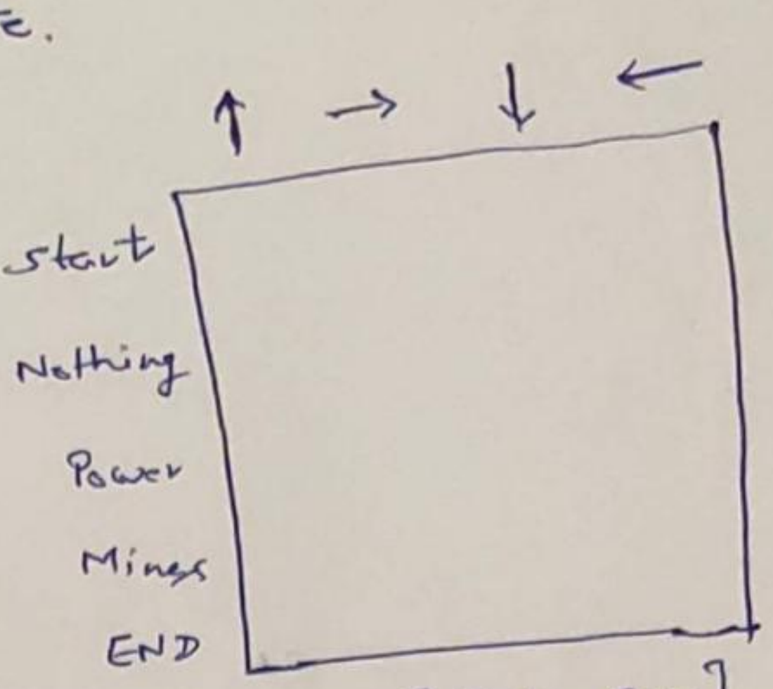
$Q(s,a) = R(s,a) + \gamma . \max_{a'} Q(s',a')$

$s \xrightarrow{a} s'$



So a Q-Table is just a simple lookup table where we calculate the maximum expected future rewards for actions at each state.

↳ Q-Table, columns are the actions start and rows are states.



$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots][s_t, a_t]$$

given state and action

Q-Learning algo^{m} process-

Initialize Q-table

choose an action

Perform an action

Measure reward

update Q-table

} After a lot of iteration a good Q-table is ready.

Q- learning algorithm -

For each s, a initialize the table entry $\hat{Q}(s,a)$ to zero.

observe the current state s.

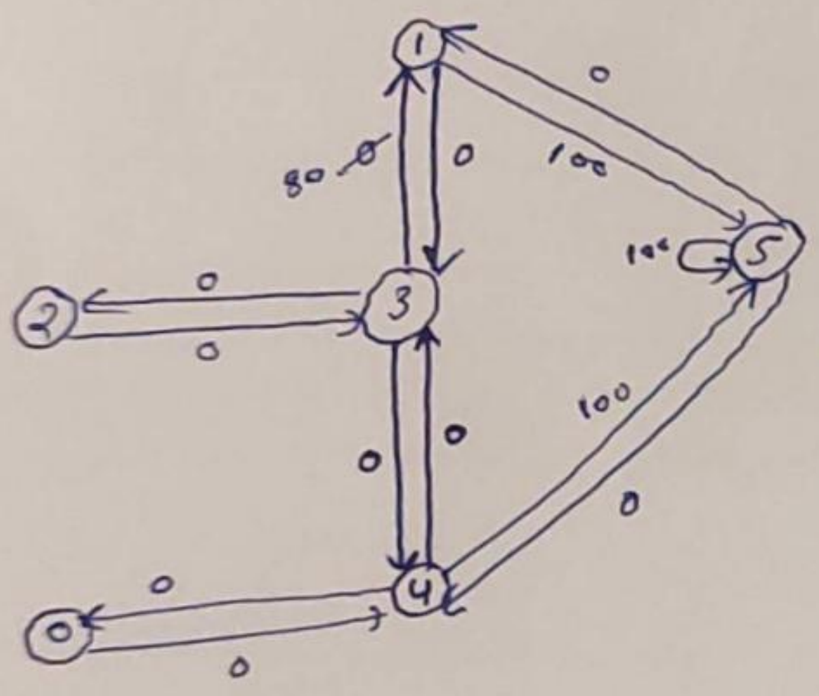
Do forever:

- Select an action a and execute it.
- Receive immediate reward r.
- observe the new state s'
- update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} Q'(s',a')$$

- set $s \leftarrow s'$

eg-



$\gamma = 0.8$

$s = 3$ $\quad a = 2, (1), 4$

$r = 0$ $\quad s' = 1$

$$\hat{Q}(3,1) = 0 + 0.8 \max \hat{Q}(0,100)$$

$$= 0 + 0.8 * 100 = 80$$

$s = s' = 1$

$s = 1$ $\quad a = 3, (5)$

$r = 100, \quad s' = 5$

$$\hat{Q}(1,5) = 100 + 0.8 \max(0,0)$$

$$= 100$$

Same process is repeated until all zero replaced with converged values.

Q- Learning:-

Q- function - denoted as $Q(s, a)$ ← quality of a action.

- basically describe at a given state s, which is the best action should I take inorder to get highest reward.

- Quality of Reward.

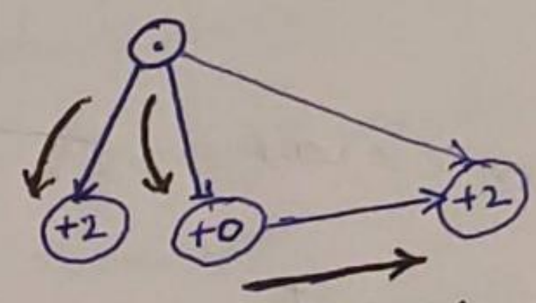
- Goal - Maximize the total reward.

ideal case $R = R_t + R_{t+1} + R_{t+2} + \ldots + R_T$

eg - ④ → [100] in next 2 days.

④ → 100 in next 200 days

Here $\gamma = +2$ (near related to future)



Total reward = 0 + 2 (there is reward is delayed) future

Rewards should come early as possible. For later rewards we panelize the reward if the come late.. by a discount factor γ.

$$R_t = r_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-1} R_T$$

γ ↑ discount factor

$0 < \gamma \leq 1$

uncertainty going to increase as you go in future.

$$Q(s, a) = r + \gamma \max Q(s', a')$$

r ↑ Immediate reward

Q function

BELLMAN EQ

Goal of Q-learning - Learn an approximate value of Q function

	Left	Right	Jump
s_0	0.3C	0.4L	0.57 ← Q-values ← Best action for a state
s_1			
s_2			

$Q(s, a) \longrightarrow \hat{Q}(s, a)$

Target Approximation.

$$\text{MSE LOSS} = (Q(s, a) - \hat{Q}(s, a))^2$$

PAGE NO. .....

```
import numpy as np
import matplotlib.pyplot as plt
import os
from collections import deque
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
import random
%matplotlib inline
class Agent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.memory = deque(maxlen=2000)
        self.gamma = 0.95  # discount factor
# Exploration vs Exploatation Tradeoff.
# Exploration: Good in the begining → helps you to
                                    explored various random
                                    things.
# Exploitation: Sample good experiences from the
                                    past (Memory).
```

self. epsilon = 1.0 # 100% Random Exploration in the beginning.

self. epsilon_decay = 0.995

self. learning_rate = 0.001

self. epsilon_min = 0.01 (1% probability that you can opt the random explorations)

1 ——— 0
(Random) (Based visual experience learning from past)

श्री गणेशाय : नमः

Policy Evaluation using Monte-Carlo Methods:-

Model Free Methods in RL:- In MDP, we are given all components to solve a problem, but what happen if we are not given some of the components. (for eg- transition probabilities and rewards i.e known as dynamics of the environment.)

can we solve that problem?
(for example- for a given position in chess what will be the next state/next position)
learning c programming from net)
(eg-2.

↳ This type of learning is called model-free learning.

↳ In Model-free, we just focus on figureout the value functions directly from the interaction with the environment.

↳ so barically a Monte-Carlo Method is a model free method which is used to solve that kind of problems as discussed previously.

Monte-Carlo Approach:- Monte-Carlo method is a statistical technique used to find an approximate solution through sampling. It approximates the expectation of a random variable by sampling and when the sample is greater, the approximation will be better.

↳ Expectation of a random variable x is given by.

$$E(x) = \sum_{i=1}^{N} x_i P(x_i)$$

$$E_{x \sim p(x)}[X] \approx \frac{1}{N} \sum_{i} x_i$$

we can estimate the expected value of x for some N Times and compute the average value of x as Expected value of [x]

↳ Monte Carlo is a model free method, meaning it doesn't require the model dynamics to compute the value function.

so a value function in R.L is denoted as

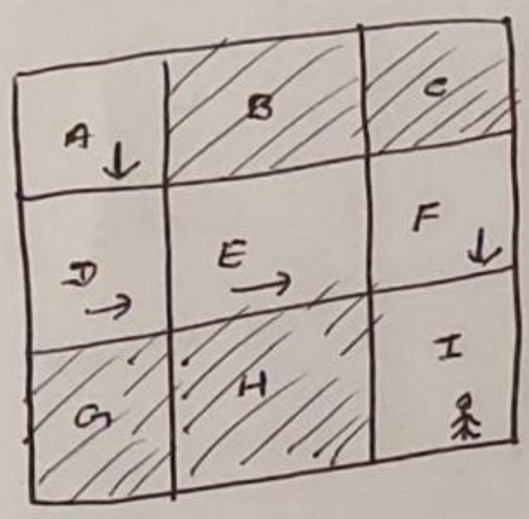
Ps.-

$$v^{\pi}(s) = E_{\tau \sim \pi}[R(\tau) \mid s_0 = s]$$

↑ expected return of a state by following a policy π

In monte carlo prediction method, we approximate the value of a state by taking the average return of a state across N episodes instead of taking the expected return.

For eg- (1).



τ_1 = A —down, +1→ D —Right, +1→ E —Right, +1→ F —Down, +1→ I

$R_1(A) = 4$

episode 1

τ_2 = A —Right, -1→ B —Down, +1→ E —Right, +1→ F —Down, +1→ I

episode 2

$R_2(A) = 2$

$\tau_3 = \tau_1$

$R_3(A) = 4$

So According to Monte-Carlo Method, the value of a state can be approximated by computing the average return across N episode i.e. (trajectories)

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

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

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

$$V(A) = 3.3$$

PAGE NO.

Algorithm :- MC Prediction :

[total_return(S) = sum of return of visited state across several episode
N(S) = counter / number of time we visited state
in which it occurs several episode]
[policy π is given as input]

Step 1 - For M number of iteration :
1. Generate an episode using the policy π.
2. Store all rewards in the episode in the list called rewards.
3. For each step t in episode :
1. Compute the return of state S_t as
$R(S_t)$ = sum(rewards[t:])
2. Update total return of state S_t
total_return(S_t) = total_return(S_t) + $R(S_t)$
3. Update the counter $N(S_t)$ = $N(S_t)$ + 1

Step 2: Compute the value of a state by just

taking the average i.e.

$$V(S) = \text{total\_return}(S) / N(S)$$

श्री गणेशाय : नमः :

Model-based Reinforcement Learning:- In model based learning, an agent will have a complete description of the environment.

↳ Here the Agent knows the complete dynamics of its environment like transition probability and rewards that the agent will going to earn, then type of learning is called model-based learning.

↳ Here the agent will knows the model dynamics to find the optimal policy.

PAGE NO. .....

Policy iteration and value iteration:-

Reinforcement Learning

Agent - policy: $P(a/s)$

(policy is a function that maps states to an action)

or prob. distribution for a action in a given state)

$a \rightarrow$ action

$s \rightarrow$ state

$r \rightarrow$ reward

Environment - Transition dynamics - describe How a states in environment changes.

$p(s', r \mid s, a)$

↑ next state ↑ current state

and how the agent gets rewards. so its a joint distribution of next state and reward based on the current state and action applied by the the agent.

③ state -

④ Action -

⑤ Reward -

State Transition prob-

$$p(s'|s,a) = \sum_{r \in R} p(s', r|s, a)$$

Total Discounted return-

(weight sum of rewards).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{N-1} R_{t+N}$$

$\gamma \in [0, 1]$

↑

Discounted Factor.

value of a state -

(with respect to a policy π)

$$v_\pi(s) = E_\pi[G_t \mid s_t = s]$$

$$v_\pi(s) = E_\pi[G_t \mid s_t = s]$$

$$= E_\pi[R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^N R_{t+N} \mid s_t = s]$$

$$= E_\pi[R_{t+1} + \gamma G_{t+1} \mid s_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \left[r + \gamma E_\pi[G_{t+1} \mid s_{t+1} = s] \right]$$

dynamics — next state value

Policy

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s' r} p(s', r|s, r) \left[r + \gamma v_\pi(s') \right]$$

↑

current state

∴ Bellman - equation

BELLMAN optimality Eq^{n} -

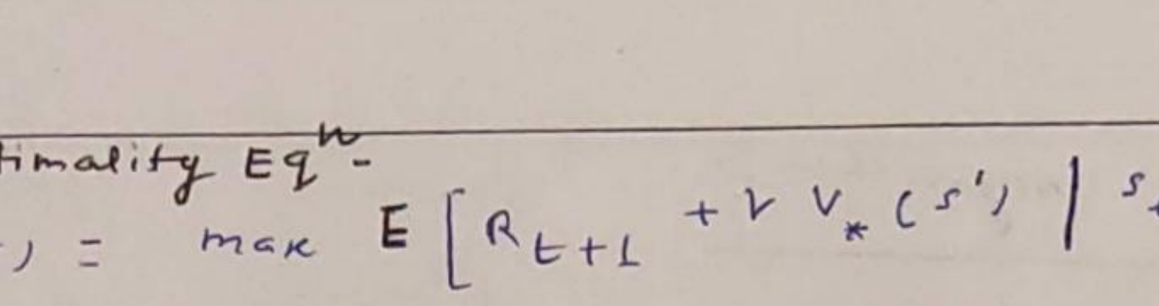
$$V_*(s) = \max_a \mathbf{E}\left[R_{t+1} + \gamma V_*(s') \mid s_t = s,\ a_t = a\right]$$

$$V_*(s) = \max_a \sum_{s'r} p(s', r \mid s, a)\left[r + \gamma V_*(s')\right]$$

For deterministic environment

$\gamma = 1.0$

eg.



1.5 ← Max

1.1

$$V_*(s) = R_2 + \gamma V_*(s_2) = 1.5$$

Finding optimal policy

policy - means what action to take in a certain state to maximize the reward.

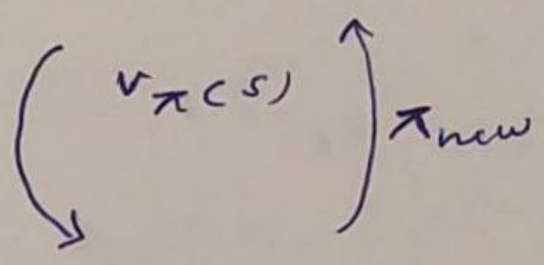
① policy iteration Algorithm - } To find the optimal policy.

② value " - "

① Policy Iteration Algorithm :-

it has two parts.

1) policy Evaluation - Compute values for the states for all state in environment.



2) policy improvement - improve the policy

$\pi_1 \xrightarrow{\text{Evaluate}} v_{\pi_2} \xrightarrow{\text{improve}} \pi_2 \xrightarrow{\text{Evaluate}} v_{\pi_2} \xrightarrow{\text{improve}} \cdots \pi_*$

Set the random policy.

iterative policy evaluation (π) ← random policy to be evaluated at initial.

{

1. Set the arbitrary values for the states.

2. Using π compute new values for the states.

$$v_{new}(s) \leftarrow \sum_{s', r} p(s', r \mid s, \pi(s)) [r + v_{old}(s')]$$

for All states

3. Repeat from (2) until convergence of the state values.

}

eg:-

$$\pi(s) = \begin{cases} a_1 & \text{if } s = s_1 \\ a_2 & \text{if } s = s_2 \\ a_3 & \text{if } s = s_1 \\ a_4 & \text{if } s = s_4 \end{cases}$$

policy improvement steps :-

For all state $s \in S$. update $\pi(s)$ as

$$\pi(s) \leftarrow \underset{a}{\text{argmax}} \sum_{s', r} p(s', r | s, a)[r + \gamma v(s')]$$

Repeat

Repeat

for all $s \in S$

$$v(s) \leftarrow \sum_{s', r} p(s', r | s, \pi(s)) [r + \gamma v_{old}(s)]$$

} Policy ~~improvement~~ Evaluation

until convergence

for all $s \in S$

$$\pi(s) \leftarrow \underset{a}{\text{argmax}} \sum_{s', r} p(s', r | s, a)[r + \gamma v(s')]$$

} policy improvement

until Convergence

Return $\pi(s)$

Value - iteration Algo^m -

Repeat

For all $s \in S$

$$V(s) \leftarrow \max_a \sum_{s',r} p(s', r) | (s, a) [r + \gamma V_{old}(s)]$$

until convergence

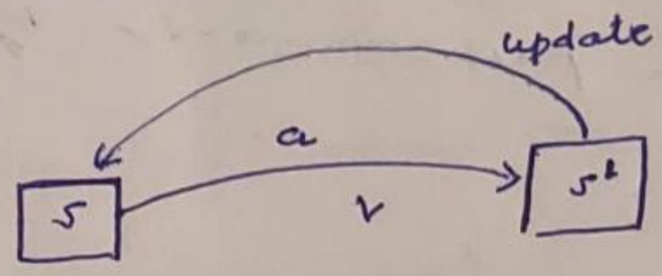
$V_*(s) = V(s)$ for all $s \in S$

Return $\pi(s) = \arg\max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V_*(s)]$

श्री गणेशाय: नम:

Temporal Difference Learning:-

↳ Also known as TD(0).



Here we can simply apply an action on state s and move to next state s' and then we can obtain the update on previous state by observing the value of r as well as our prev guess of the value of next state s'.

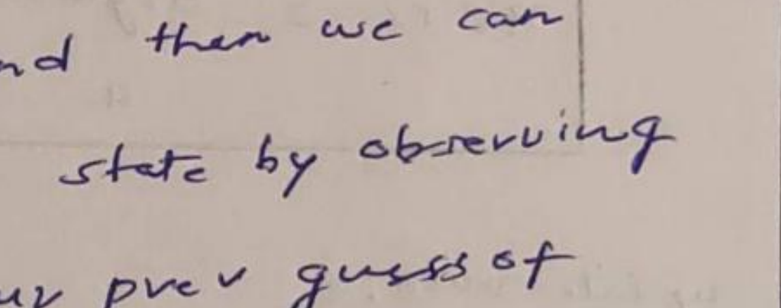
$V_\pi(s)$

(value Function)

value of a state is the expected return that we obtain. if we apply policy π from state s.

$q_\pi(s, a)$ — q-value state action pair

↳ q value is something that is associated with the state as well as a particular action.



$q_\pi(s,a) = \frac{1}{N} \sum G_i$

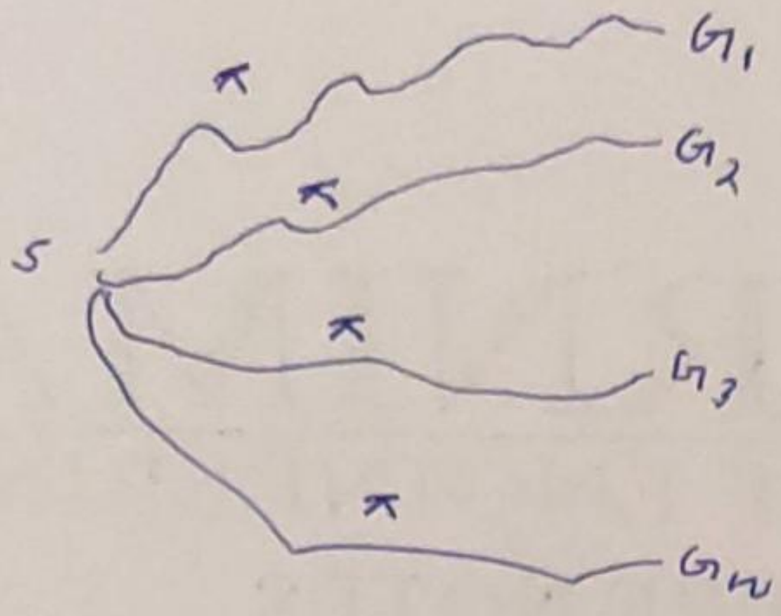
$v^*(s)$ - optimal value of value function

(maximum return that we can get)

(stochastic Environment)

$v^*(s) = v_{\pi^*}(s)$

that depends upon the policy π that can give maximum rewards.



$$V_\pi(s) = \frac{1}{N} \sum G_i$$

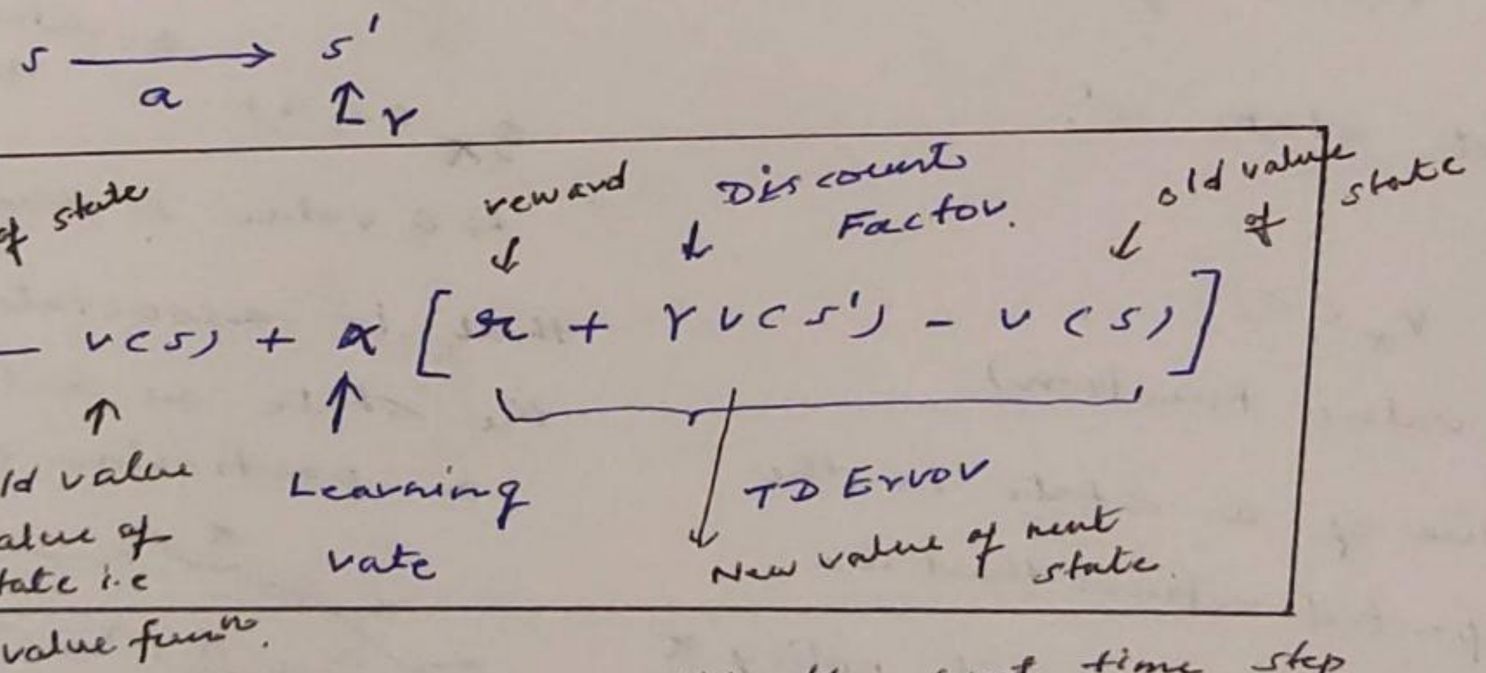
↳ optimal q-value for state-action pair.

$$q^*(s, a) = q_{\pi^*}(s, a)$$

that is the maximum q-value that you can get from that state and action pair onwards by following policy π.

$$\pi(s) = \underset{a}{\operatorname{argmax}}\ q^*(s, a)$$

TD(0) update rule:-



TD(0) only have to wait until the next time step to update the old value

SARSA Algorithm:- (SARSA is an acronym for State-Action Reward-State-Action).

↳ SARSA is an On-policy Temporal Difference Learning. (Here the agent behaves using one policy and tries to improve the same policy).

↳ Its only talk about the Q-value not the value of state that is value function.

$$\pi(s) = \begin{cases} \underset{a}{\operatorname{argmax}}\, Q(s,a), & \text{with } p = 1-\varepsilon \\ \text{or} \\ \text{random } a, \text{ with } p = \varepsilon \end{cases}$$

$s \xrightarrow{a} s' \xrightarrow{a'} \cdots$

$s, a \uparrow$; $s' \downarrow r$; $a' \uparrow$ E-greedy

$s, a \uparrow$ E greedy

(epsilon greedy policy) - may not be 100% randomizations at start.

(Here we select a random action with prob. epsilon and select action with prob 1-epsilon.

$\varepsilon = 0.02$

Discount Factor

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$$

Learning rate

SARSA Update Rule

SARSA Algorithm

initialize the Q-table

say $Q(s, a) = 0$ for all (s, a) pairs.

Repeat for N episodes

$a =$ E-greedy at s_0

Repeat for k-steps.

At any state s.

① Execute a, get reward $r \longrightarrow$ state s'

② Next action $a' =$ E-greedy at s'

③ Get $Q(s', a')$ from the table

④ update

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$

⑤ Set

$a \leftarrow a'$, $s \leftarrow s'$

END OF FOR

END OF FOR

PAGE NO. .....

eg-

	1	2	3	4
1	S	F	F	F
2	F	H	F	H
3	F	F	F	H
4	H	F	F	G

Q- Table with random values.

State	Action	Value
(1,1)	Up	0.5
⋮	⋮	⋮
(4,2)	Up	0.3
(4,2)	Down	0.5
(4,2)	Left	0.1
(4,2)	Right	0.8
⋮	⋮	⋮
(4,4)	Right	0.5

highest (s, a) Epsilon - value

Right (4,3) Up 0.6
(4,3) Down 0.3
(4,3) Left 1.0
(4,3) Right 0.9

Let reward $r = 0$ learning rate = 0.1 discount factor $\gamma = 1$

Now According to SARSA update rule

$$Q(s,a) = Q(s,a) + \alpha(R + \gamma Q(s', a') - Q(s,a))$$

$$Q((4,2), right) = Q((4,2), right) + \alpha(R + \gamma Q((4,3), a') - Q(4,2), right)$$

$$Q((4,2), right) = Q((4,2), right) + 0.1(0 + 1 * Q((4,3), a') - Q(4,2), right)$$

$$Q((4,2), right) = 0.8 + 0.1(0 + 1 + Q((4,3), a') - 0.8)$$

Now again select $(Q(4,3), a')$ right) Here 0.9 randomly

$$Q((4,2), right) = 0.8 + 0.1(0 + 1 + (0.9) - 0.8)$$

$$Q(4,2, right = 0.81$$