Machine Learning

Unit-4

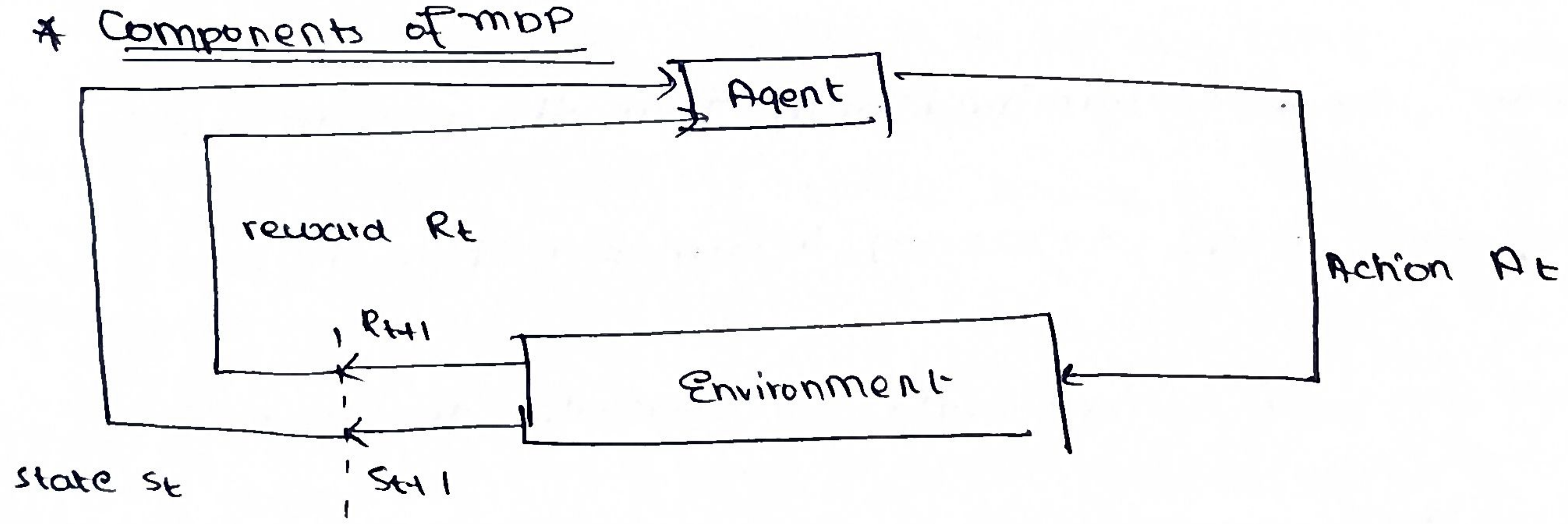
* Markov Decision Processes

→ MDPs model sequential-decision making scenarios with probabilistic dynamics.

→ They are used to design intelligent machines or agents that need to function longer in an environment where actions can yield uncertain results.

→ MDP models are used in probabilistic planning & reinforcement learning. (RL)

* Components of MDP



→ Reinforcement learning is based on the concept of MDP. A MDP is defined as a tuple (S, A, T, R, γ)

States S - possible situations the decision maker can be in

Actions A - the choices the decision maker can make at each state

Transitions T - The probability of moving from one state to another after taking an action

Reward R : The immediate rewards received after taking an action.

Discount Factor : γ - a value $[0,1]$, and takes care of the rewards the agent achieved in the past, present and future.

Policy (π) ~~is the ag~~ helps the agent determine the optimal action given the current state so that it gains the maximum reward

$\pi : S \to A$

<u>MDP Property</u> → usage of the Markov property, which states that future can be determined only from the present state that encapsulates all the necessary information from the past.

i.e MDP uses only current state to evaluate the next actions without depending on the previous states or actions.

$$S_{t+1} = f(s_t, a_t)$$

<u>Bellman Equation</u> - The Bellman equation represents the max reward an agent can receive if they make the optimal decision now and for all future decisions

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

* SARSA and Q-Learning?

Reinforcement Learning

↙ ↘

Model-based → MCTS

Model-free

↙ ↘

on-policy → SARSA

off policy → Q-Learning

Q-Learning? → a model-free reinforcement learning algorithm for learning a policy, which tells an agent what action to take under what circumstances

→ The use of the max function over the available actions makes the Q-learning algorithm an off-policy approach

Algorithm

$\boxed{\text{Initialization}}$ set Q(s,a) to small random values for all s & a.

$\boxed{\text{Repeat}}$:

initialize s

repeat :

- selection action a using ε-greedy or another policy
- take action a and receive reward r

- sample new state s'

- update $Q(s,a) \leftarrow Q(s,a) + \mu(r + \gamma \max Q(s', a') - Q(s,a))$

- set $s \leftarrow s'$

- For each step of the current episode

• until there are no more episodes

SARSA

→ SARSA is a model-free reinforcement learning algorithm.

→ SARSA = state action reward state action

→ In SARSA, the time difference value is calculated using the current state-action combo & the next state-action combo.

Algorithm

Initialization

- set $Q(s,a)$ to small random values for all s & a

Repeat:

- initialize s
- choose action a using current policy

repeat:

take action a and receive reward r

sample new state s'

choose action a' using the current policy

update $Q(s,a) \leftarrow Q(s,a) + \mu(r + \gamma Q(s',a') - Q(s,a))$

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

· For each step of the current episode

Until there are no more episodes

* K-Means Clustering

Obtain 2 clusters $k = 2$

Let the cluster centers be Instance 1 and Instance 3

Instance	X	Y
1	1.0	1.5
2	1.0	4.5
3	2.0	1.5
4	2.0	3.5
5	3.0	2.5
6	5.0	6.0

The points are
(0, 1) → C1
(3, 3.16) → C1
(1, 0) → C2
(2.24, 2) C2
(2.24, 1.41) → C2
(6.02, 5.41) → C2

new $C_1 = ((1+1)/2, (1.5+4.5)/2)$

new $C_2 = ((2+2+3+5)/4, (1.5+3.5+2.5+6.0)/4)$

→ contd

Ans Compute Euclidean distance from C_1 (1, 1.5) | from C_2 (2, 1.5)

Instance 1 $\sqrt{0} = 0$ | $\sqrt{(-1)^2 + 0^2} = 1$

Instance 2 $\sqrt{0 + (3)^2} = 3$ | $\sqrt{(-1)^2 + (3)^2} = 3.16$

Instance 3 $\sqrt{(1)^2} = 1$ | $= 0$

Instance 4 $\sqrt{1 + 4} = 2.23$ | $\sqrt{0 + (2)^2} = 2$

Instance 5 $\sqrt{4 + 1} = 2.24$ | $\sqrt{2} = 1.41$

Instance 6 $\sqrt{} = 6.02$ | $= 5.41$

Unit - 4

Principles of Machine Learning

* K- Means Clustering

Q1. SAT Compute 2 clusters using K- Means clustering where the initial clusters are (1, 1) and (5, 7). Perform one iteration

A	B
1	1
1.5	2
3	4
5	7
3.5	5
4.5	5
3.5	4.5

	distance from C_1 (1,1)	distance from C_2 (5,7)	Cluster
(i) (1,1)	$D = 0$	$D = \sqrt{(4)^2 + (-6)^2}$	C_1
(ii) (1.5,2)	$D = \sqrt{(0.5)^2 + (1)^2} = 1.11$	$D = \sqrt{(3.5)^2 + (5)^2}$	C_1
(iii) (3,4)	$D = \sqrt{(2)^2 + (3)^2} = \underline{4}$	$\sqrt{(2)^2 + (3)^2} = \underline{4}$	C_1 or C_2
(iv) (5,7)	$D = \sqrt{(4)^2 + (6)^2}$	$D = 0$	C_2
(v) (3.5,5)	$D = \sqrt{(2.5)^2 + (4)^2} = 4.716$	$= \sqrt{(1.5)^2 + (2)^2}$	C_2
(vi) (4.5,5)	$D = \sqrt{(3.5)^2 + (4)^2}$	$D = \sqrt{(0.5)^2 + (2)^2}$	C_2
(vii) (3.5,4.5)	$D = \sqrt{(2.5)^2 + (3.5)^2}$	$D = \sqrt{(1.5)^2 + (2.5)^2}$	C_2

*Action Selection, Policy and Discounting in RL

Action Selection → process by which an agent decides what action to take at each time step.

Some strategies are:

(i) ϵ- Greedy strategy

1. with probability ϵ, the agent selects a random action (exploration)

2. with probability $1-\epsilon$, the agent selects the action that maximizes the estimated value function. (exploitation)

(ii) Softmax Action Selection

→ determines the probability of selecting an action a using the softmax distribution

(iii) Deterministic Action selection : rule-based

$a = \pi(s)$

policy

(iv) Stochastic Action Selection : probability based

$a = \pi(s|a)$

(v) Greedy : always choose the current best value (exploitation)

Policy

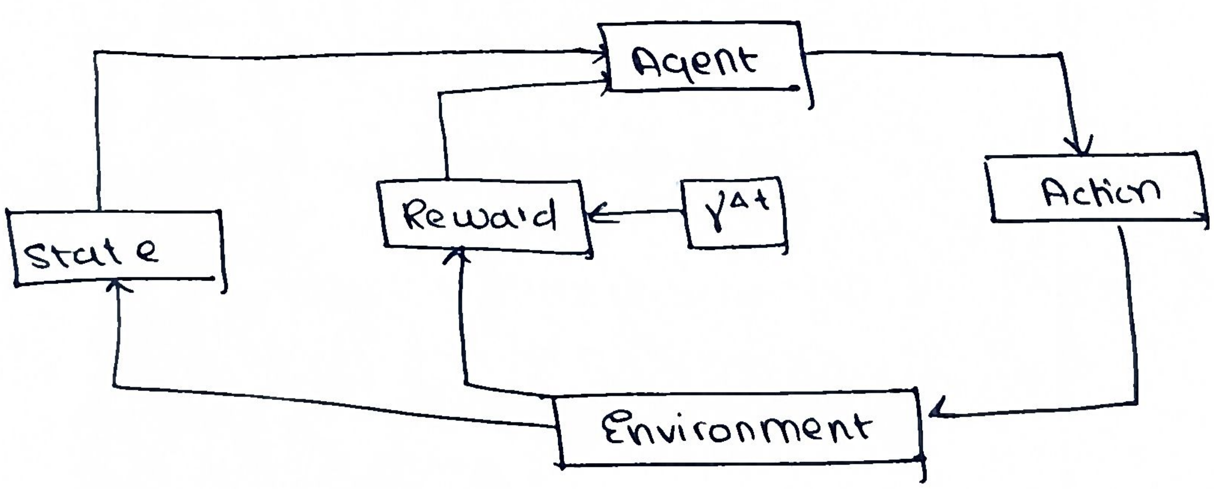
- In each state, what action should the agent take
- For any state - s, policy $\pi(s)$ gives the action
- Optimal policy gives the optimal action in every state. Leads to the highest reward.

Discount Factor

- determines how much the RL agent cares about rewards in the distant future relative to those in the immediate future.

If $\gamma = 0 \Rightarrow$ agent only learns about actions that produce an immediate reward

$\gamma = 1 \Rightarrow$ agent evaluates each of its actions based on the sum total of all its future rewards



$\gamma^{\Delta t}$ = discount factor