Reinforcement Learning

pointments

RL is basically a learning model that learns from its environment. Unlike supervised ML where a dataset for training is already given RL models basically learns from a trial & error method.

Elements of RL

Agent – the thing performing all the tasks

Environment – the surrounding ~~eg. the~~ any game. ~~on~~ of the agent i.e. the place where agent will perform task (like battlefield of any game)

Policy – agent's way of behaving at a given time

Reward signal – defines goal of a RL prob. At each step, the agent receives certain reward which ~~is~~ we have to maximize.

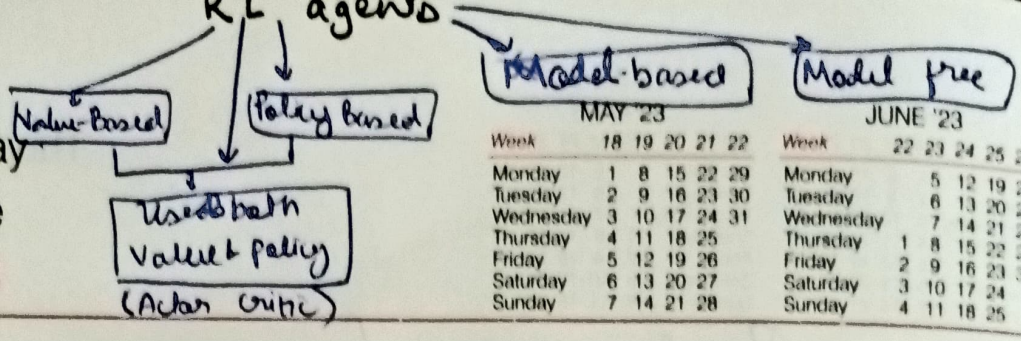
Value function – total amt. of reward the agent expects to accumulate over future.

Model (not needed in model-free learning ~~ones~~) – basically used to train the model on a dummy environment before running it on the actual Environment.

State – information used to determine what happens next. (In markove state, state is sufficient for predicting future, history is not needed)

✍

Rewards are short-term indicator but value$_{fn}$ is the cumulative sum of rewards hence, long-term indication



02 | Friday, June 2023

153rd Day | 22nd Week

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

Appointments

Markov Decision Process (MDP)

9.00 ~~Markov~~

- Finite MDP (finite states)
- Infinite MDP (infinite states)

10.00

⇒ In this, agent directly observes the environment state

11.00

12.00

$$O_t = S^a = S_t^e$$

S^a: agent state; S_t^e: environment state

1.00

2.00 A state is markov if and only if state is sufficient statistic of future

3.00 i.e. $P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, \ldots, S_t]$

4.00 State transition matrix – defines the probability of going into another state.

5.00

$P_{ss'}$ → Prob. of transition probabilities from

6.00 all states s to all successor states s'

7.00

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

9.00

Markov Chain ⇒ need just 2 parameters

S → set of states

P → state trans prob matrix

10.00

Markov Reward Process → 4 params

S, P, R, γ

R → reward fn ; γ → discount factor

11.00

12.00

Return ⇒ $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

1.00

⇒ represents future rewards.

if $\gamma \to 0$: Myopic {can simplify that means we don't care all future rewards}

if $\gamma \to 1$: far-sighted {we care consider future rewards}

2.00

3.00

4.00

Value fn.

$$V(s) = E[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]$$

5.00

$$= R_s + \gamma \sum_{s' \in S} P_{ss'} \cdot V(s')$$

6.00

R_s : Reward at current state

$\sum_{s' \in S} P_{ss'} \cdot V(s')$: discount value of the value fn of all future states with their probability of state transition

7.00

Sunday 04

$$\begin{bmatrix} V(1) \\ \vdots \\ V(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ & & \\ P_{11} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} V(1) \\ \vdots \\ V(n) \end{bmatrix}$$

05 Monday June 2023

Assignments

8:00

i.e. $v = R + \gamma P v$

$\Rightarrow v = (1 - \gamma P)^{-1} R$

10:00

BUT Complexity comes is $O(n^3)$!

11:00

So we need other methods for large MRPs

12:00

MDP $\Rightarrow$ needs S, A, P, R, γ

1:00

$A \rightarrow$ set of actions

2:00

Policy $\pi \rightarrow$ ~~set~~ function of actions over given states.

3:00

$$\pi(a|s) = P[A_t = a \mid S_t = s]$$

4:00

So, now, $$P^{\pi}_{s,s'} = \sum_{a \in A} \pi(a|s) P^{a}_{ss'}$$

5:00

$$R^{\pi}_{s} = \sum_{a \in A} \pi(a|s) R^{a}_{s}$$

6:00

Value of π ($v_\pi(s)$)

7:00

$\Rightarrow v_\pi = E_\pi [R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$

Action-value of π ($q_\pi(s,a)$):

$\Rightarrow q_\pi(s,a) = E_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$

Appointments

9.00

10.00

11.00

12.00

1.00

2.00

3.00

4.00

5.00

6.00

7.00

Optimal value f^n

→ optimal state-value f^n : $V_*(s) = \max_\pi V_\pi(s)$

→ Optimal action-value f^n : $q_*(s,a) = \max_\pi q_\pi(s,a)$

basically acting greedily to get best performance

Optimal policy is the policy at which we achieve $q_*(s,a)$ & $V_*(s)$.

$V_*(s) = \max_a q_*(s,a)$ ⇒ at particular action 'a' the value f^n is the max of q action value f^n.

{ Value of being in a particular state under optimal policy is determined by selecting the action that maximizes expected return }

So, $q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \underbrace{V_*(s')}_{\max_a q_*(s,a)}$

$V_*(s) = \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s')$

For optimal policy, we maximize π for a set of actions

$\pi^* = \arg\max_a [q^*(s,a)]$

07 | Wednesday June 2023

158th Day | 23rd Week

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

Appointments

9.00

10.00

11.00

12.00

1.00

2.00

3.00

4.00

5.00

6.00

7.00

Dynamic Programming ⇒ Optimising a program used for Dynamic because the problem breaks & solution can be cached and reused

⇒ Hence DP is req for MDP

MDP ⇒ requires two kinds of sol"

Prediction
↳ predicts value f" v_π

control
↳ gives optimal policy. π_*

Iterative Policy Evaluation

What is Policy Evaluation?

⇒ process of determining state-value f" or action-value f" for a given policy in a MDP in order to estimate how good that policy is.

How to do it?

iteratively updates the value function until it converges to the true value f" for the policy. In case of iterative policy evaluation we update using Bellman Eq".

Appointments

8.00 We can backup the value f^n using synchronous backup methods or asynchronous backups.

10.00

Synchronous backup

11.00

$$V_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s') \right)$$

12.00

here s' is the successor state of s

1.00

or

$$V^{k+1} = R^{\pi} + \gamma P^{\pi} V^k$$

2.00

3.00

So basically if we determine our policy by acting greedily wrt V_k after a number of iterations we will converge to some policy π which is the optimum policy. This is called policy improvement. Policy evaluation & Policy Improvement go together in RL

4.00

5.00

6.00

7.00 So, for a given policy π

- Evaluate the policy
  $V_\pi(s) = E[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$
- Improve the policy by acting greedily.

In this process of policy evaluation we always get π^*.

09 Friday June 2023

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

160th Day | 23rd Week

Appointments

Value Iteration

⇒ Goal is to find the optimal value function which represents the maximum expected return in a MDP.

So we start with an initial estimate of the value function then improve it by iterative updates.

It can be understood as a special case of policy iteration, where we stop after one sweep and update our policy.

Can be written as,

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_k(s')\right]$$

The value iteration terminates when it converges to a value V^*.

So, it effectively combines in each step one sweep of policy evaluation & one sweep of

JULY '23

Week	26	27	28	29	30
Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

AUGUST '23

Week	31	32	33	34	35
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Saturday
June
2023

10

23rd Week | 161st Day

Appointments

9.00 policy improvement.

10.00

Synchronous Dynamic Programming

⇒ Complexity per iteration ⇒ $O(mn^2)$ for $V_\pi(s)$
↳ matrix
n states

⇒ Complexity $O(m^2n^2)$ per iteration for $q_\pi(s,a)$

1.00

2.00 Asynchronous Dynamic Programming

3 ways ⇒ ① In-place dynamic programming
3.00 ② Prioritised sweeping
③ Real-time dynamic programming

4.00

① In-place : stores only 1 copy of value fn

5.00

$$V(s) \leftarrow \max_{a \in A} \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V(s') \right)$$

6.00

7.00 ② Prioritised sweeping : Backup the state with largest bellman error i.e. $\left| \max_{a \in A} \left(R_s^a + \gamma \sum_{s \in S} P_{ss'} V(s') \right) - V(s) \right|$

Sunday 11

③ Real-time Dynamic Programming :

→ use agent's experience to guide the selection of states
→ After each time step, backup the state S_t

$$V(S_t) \leftarrow \max_{a \in A} \left(R_{s_t}^a + \gamma \sum_{s \in R} P_{s_t s'}^a V(s') \right)$$

12 | Monday
June
2023

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

163rd Day | 24th Week

Appointments

9.00 # Model-free Prediction

10.00 ↳ Two ways: ① Monte-Carlo Learning
② Temporal Difference learning

11.00

12.00 ## Monte-Carlo Learning

1.00 → learn directly from episodes of experience.

2.00 Q. What does episode of experience mean?

3.00 Ans → So we basically run our agent in the environment to gain experience. ~~That~~
4.00 Each run will be an episode. From the episodes of the run experience we
5.00 will have a set of states with their values, reward etc. We use that in
6.00 MC learning.

7.00 Here the value is the mean return.
Why?
In each to get proper estimate of return it is better to take average of the return at that state in each episode.

JULY '23

Week	26	27	28	29	30
Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

AUGUST '23

Week	31	32	33	34	35
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Tuesday
June
2023

13

24th Week | 164th Day

Appointments

9:00

10:00

11:00

12:00

1:00

2:00

3:00

4:00

5:00

6:00

7:00

There are two types of MC-Policy Evaluation.

① First-visit (commonly used)

② Every visit

In an episode we may visit a particular state more than once, in that case if we consider only the state visited first it is first-visit other wise every-visit.

First-Visit Monte-Carlo Policy Evaluation

For a policy π we have:

$S_0, A_0, R_1, S_1, A_1, R_2, \ldots S_{T-1}, A_{T-1}, R_T$

(S_0, A_0, R_1 — one episode)

For each episode:

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

② ~~Initialize~~ $G(s) \leftarrow 0$

② For each step of episode, $t = T-1, T-2, \ldots 0$

$G(s) \leftarrow G(s) + R_{t+1}$ {Total Return}

③ increment counter $N(s) \leftarrow N(s) + 1$

④ Value: $V(s) = G(s) / N(s)$

As $N(s) \to \infty$; $V(s) \to V_\pi(s)$

14 | Wednesday June 2023

165th Day | 24th Week

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

Appointments

9.00 **Incremental Monte-Carlo Learning**

10.00 if we update after each ~~episode~~ step in an episode

11.00 $S_1, A_1, R_2 - - - , S_T$

12.00 $N(S_t) \leftarrow N(S_t) + 1$

$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} \left(G_t - V(S_t) \right)$

1.00 we know the future ~~reward~~ return

2.00 So we care only about the current episode & not the old episodes

3.00

Hence better to name MC learning.

4.00

~~In every visit MC learn~~

5.00 ~~$V(S_t) \leftarrow V(S_t) + \alpha$~~

6.00

7.00

Incremental Mean

$$\mu_k = \frac{1}{k} \sum_{j=1}^{k} x_j = \underbrace{\mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})}_{\text{increment mean}}$$

Temporal-Difference Learning

⇒ Learning from incomplete episodes by bootstrapping.

Simplest TD-learn algorithm: TD(0)

$$V(S_t) \leftarrow V(S_t) + \underbrace{\alpha}_{=1/N(s)} \left(\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{estimated return instead of actual return.}} - V(S_t) \right)$$

δ_t (TD error) $= R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

Major advantage of TD ⇒ can work in non terminating environment

TD has low variance & some bias
(MC → high vars, no bias)
TD is more sensitive to initial value.

Bias/Variance Trade-off

$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$ ⇒ unbiased estn of $v(s)$

~~True TD target $R_{t+1} + \gamma V(S_{t+1})$ ⇒ unbiased estimate~~

TD target ⇒ $R_{t+1} + \gamma V(S_{t+1})$ ⇒ biased

G_t ⇒ depends on many actions, TD target ⇒ depends on one action

n-step return

⇒ calculating the return for 'n' steps

$$G_t^{(n)} = R_{t+1} + \sum_{k=1}^{n-1} \gamma^k R_{t+k+1} + \gamma^n V(S_{t+n})$$

16 | Friday June 2023

167th Day | 24th Week

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

Appointments

9.00 ~~TD(1)~~

Two diff time steps can be combined by averaging for combing n-steps, $G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$

10.00 TD(λ)

⇒ we used λ-return.

11.00

form of TD(λ)

12.00 $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{\lambda} - V(S_t) \right)$

1.00 ~~⇒ basically gives less weight to~~

2.00 ~~⇒ can be computed only for complete episodes~~

3.00 Value of λ determines how much weight we want to give to our past value

4.00

It is a midway between TD(0) & MC learning

5.00

if $\lambda \to 1$ ⇒ more weight to past states & more like MC (more variance)

6.00 if $\lambda \to 0$ ⇒ more bias to recent state visits & more like TD(0)

7.00

✍ Bootstrapping - using estimates of future values.

Sampling → collecting data or experience from the env.

⇒ Sum of offline updates is identical for forward-view & backward view TD(λ)

JULY '23

Week	26	27	28	29	30
Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

AUGUST '23

Week	31	32	33	34	35
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Saturday
June
2023

17

24th Week | 168th Day

Appointments

9.00 Forward View TD(λ)

10.00 ⇒ Uses complete episode to update value from n-step

11.00 ⇒ ~~offline update~~

12.00

1.00 Backward view TD(λ)

2.00 ⇒ Updates every step for incomplete sequences.

3.00 ⇒ ~~Online update~~

⇒ Keep an eligibility trace for every state s

4.00 ⇒ $V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$

↑ eligibility trace

5.00

6.00 when λ=0, only current state is updated

so $E_t(s) = 1$

7.00 ∴ TD(λ) becomes equal to TD(0) update

Sunday 18

⇒ Eligibility trace:. maintain a balance b/w recency & frequency-based heuristics.

in simple words, What cause some action?

the most recent action or the most frequent one.

Eligibility traces ~~balances~~ needs both.

19 Monday June 2023

170th Day | 25th Week

Appointments

9:00

10:00

11:00

12:00

1:00

2:00

3:00

4:00

5:00

6:00

7:00

MAY '23

Week	18	19	20	21	22
Monday	1	8	15	22	29
Tuesday	2	9	16	23	30
Wednesday	3	10	17	24	31
Thursday	4	11	18	25	
Friday	5	12	19	26	
Saturday	6	13	20	27	
Sunday	7	14	21	28	

JUNE '23

Week	22	23	24	25	26
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	
Sunday	4	11	18	25	

Q-learning

(or off-policy TD control)

~~Q~~ Q-values

$q^*(s, a)$ = expected return while starting in s, taking action a & acting optimally

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Here, learned action value f^n Q directly approximates q_* independent of the policy being followed.

So, $Q(s, a) \rightarrow$ expected cumulative reward we iteratively update until it converges to q^*.

~~& we chase actions randomly~~

we chase random actions greedily but not absolutely greedily we chase random actions with prob. ε

JULY '23

Week	26	27	28	29	30
Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

AUGUST '23

Week	31	32	33	34	35
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Tuesday
June
2023

20

25th Week | 171st Day

Appointments

9.00 Advan of Q learn ⇒ converges to optimal policy even if we act sub optimally

10.00

11.00 Learning rate should be such as

$$\sum_{t=0}^{\infty} \alpha_t(s,a) = \infty \quad \text{but} \quad \sum_{t=0}^{\infty} \alpha_t^2(s,a) < \infty$$

12.00

1.00

2.00 ⇒ Not possible to visit all state (in case of continuous envs.)
So we use Approximate Q-learning

3.00

so instead of having a set of Q values

4.00

we can have

5.00 ① Parameterize Q-function
which is

6.00 ① can be linear f^n of features
$Q_\theta(s,a) = \theta_0 f_0(s,a) + \theta_1 f_1(s,a) + \dots + \theta_n f_n(s,a)$

7.00 ② neural net

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \left[\frac{1}{2} \left(Q_\theta(s,a) - R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right)^2 \right]$$

21 | Wednesday
June
2023

172nd Day | 25th Week

Appointments

9:00

So what happens in Approximate Q learning

10:00

Instead of storing all the state-action pairs we use a parameterized fn like nn to approximate the Q-values.

11:00

we updat θ until we converge to θ* such that there is minimum discrepancy between approximate Q-values & true Q-values

12:00

We generally use nn to approx functions

1:00

So we need to approximate θ* & nn is a good way to do that

2:00

So what exactly happen in DQN?

3:00

⇒ We have an experience replay buffer to store agent's experience

4:00

⇒ We use CNN to estimate Q-values
NN takes states as input & give Q-values as output for each action

5:00

⇒ We acts greedily (or ε-greedily) to select action

⇒ action to perform, next state is observed & reward is received for that action.

6:00

7:00

JULY '23

Week	26	27	28	29	30
Monday	31	3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday	2	9	16	23	30

AUGUST '23

Week	31	32	33	34	35
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

Thursday
June
2023

22

25th Week | 173rd Day

Appointments

9.00

10.00

11.00

12.00

1.00

2.00

3.00

4.00

5.00

6.00

7.00

⇒ DQN will sample a batch of expens from replay buffer

⇒ for each experen target Q-value is calculate using separate target netwo to address moving target issue

⇒ the params of main network are updated using gradien - descent to minimize the estimate Q-valu & target Q-valu

Q. What is target Q value?

Desired Q-values that the network wants to approximate. They are estimated using a target network (copy of the main network) It generates target Q values by taking the maximum Q-value ove all possible actions and the next state acc- to target network.

Since these targets are computed frm Q-values they constanly changing which cause moving target problem.

By decoupling target value from rapidly chang Q-value we can resolve moving target issue.