Reinforcement Learning

DATE: PAGE:

Example where Reinforcement learning is used :

used in manufacturing optimization problems in which a sequence of actions must be chosen, and the reward to be maximized is the value of the goods produced minus the costs involved. It includes sequential scheduling problems such as choosing which taxis to send for passengers in a large city, where the reward to be maximized is a fn of the wait time of the passengers & the total fuel costs of the taxi fleet.

Each time the agent performs an action in its environment, a trainer may provide reward or penalty to indicate the desirability of the resulting state.

For ex, when training an agent to play a game the trainer might provide a positive reward when game is won, -ve reward when game is lost, & zero reward in all other states. The task of the agent is to learn from this indirect, delayed reward, to choose sequence of actions that produces the greatest cumulative reward. Thus sequence of actions that achieve its goals is referred as control-policy. & this is what is to be learnt by the agent. The task of agent is to perform all possible sequence of actions (for state-action transitions), observe their consequences & learn a control-policy. The control-policy we

for moving from state $s1$, action $a1$ is performed & reward $r1$ is given. It goes to state $s2$.

to where? to the Agent or

Described by a set of possible states S.

Environment interacts by performing action

Agent can be a robot

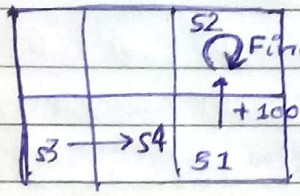
can be from a set of possible actions A.

Actually, a +ve reward is assigned to the state-action transition that immediately result in winning the game & not to the action.

DATE: PAGE:

desire is the one that, from any initial state, chooses actions that maximize the reward accumulated over time by the agent. The agent's task is to learn a control policy, $\pi : S \rightarrow A$, that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay.

i) (Delayed) reward is associated NOT with the action but with the state - action transition.



is given to the state-action transition which immediately leads you to the final state.

OR

From state S1, when agent goes up (action) transition happens to adjacent state S2.

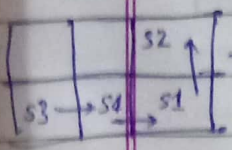
If state S2 happens to be a final state then +ve reward is awarded to this state - action transition.

∴ Reward is awarded after the transition happened depending on the state S2

∴ It is referred to as Delayed Reward.

Know that one transition is b/w 2 (adjacent) states ONLY

S1 transition S2 where S1 & S2 are adjacent states.

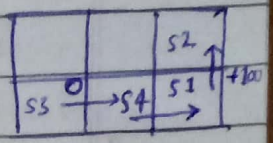


Transition : S3 → S4

Action : Turn Right

End State : S4

Is S4 a final state ? NO



∴ In state - action transition

S3 → S4 where

action performed is going right,

(delayed) reward given = 0

Consider transition b/w states S4 & S1

S4 → S1

Action : Turn Right

Is S1 a final state ? NO

∴ In state - action transition

S4 → S1

(delayed) reward given = 0

(ii) Note that,

first time when transition happened from S3 to S4 (by action : Turn Right) we did not have any reward associated with that state - action transition.

i.e., agent did not have prior knowledge about the effect of its action on the environment.

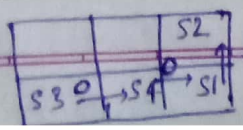
But next time,

it would already know (will have prior knowledge) the effect of such a state - action transition

It knows that if it takes

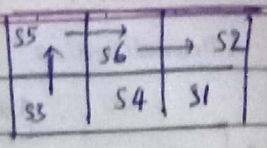
S3 → S4 it'll get reward of 0

Assuming the agent has learnt



DATE: PAGE:

but is unaware of



S3 → S5 (action: go UP)
S5 → S6 (action: go RIGHT)
S6 → S2 (action: Go RIGHT)

∵ it never performed these actions

Should one try exploration of unknown states & actions (to gather new information), or favor exploitation of states & actions that it has already learned & will yield high reward (to maximize its cumulative reward).

iii) Life-long learning:

Agents while learning for one tasks within an environment, often will learn several related tasks in the same environment, using the same sensors.

For ex: a mobile robot may need to learn or its main goal could be to dock on its battery charger when the battery is low, but in order to be able to do this it may also learn how to navigate through corridors.

After having learnt how to navigate & reach the charger, one can leverage the same knowledge / experience to use it for task such as picking up output from the printer.

* In this chapter, read objective fn = Policy.

DATE: PAGE:

iv) Partially observable states :

Sometimes state - action transition made may not be independent of the previous state - action transition.

Sometimes you'll require to consider previous observation along with the current & then choose action for next state - action transition.

For ex : a robot with a forward - pointing camera cannot see what is behind it.

In such cases, it may be necessary for the agent to consider its previous observations together with its current sensor data when choosing actions, & the best policy may be one that chooses actions specifically to improve the observability of the environment.

Generally,

where $\delta(s_t, a_t)$ which is a fn that gives the succeeding state s_{t+1} and $r(s_t, a_t)$ which is the reward that environment responds to the agent

$s_t \in$ the finite set of states S

& $a_t \in$ " " " " actions that can be performed A.

∵ Here 2 depend only on the current state & action, it is called as Markov Decision Process

The learning Task :

Considering Markov decision Process, the task of the agent is to learn a policy,

$$\pi : S \rightarrow A$$

for selecting its next action a_t, based on the current observed state s_t; that is

$$\pi(s_t) = a_t$$

But this policy that we learn should be such that it produces the greatest possible cumulative reward for the robot over time.

Let $V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$ [Discounted Cummulative Reward]

where the sequence of rewards r_{t+i} is generated by beginning at state s_t & repeatedly using the policy to select actions as described above (i.e., $a_t = \pi(s_t)$, $a_{t+1} = \pi(s_{t+1})$ etc)

Here, $0 \leq \gamma < 1$ is a constant that determines the relative value of delayed versus immediate rewards.

Few other defines of other reward that can be used are :

1. Discounted cummulative Reward,
$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$
2. Finite Horizon Reward,
$$V^{\pi}(s_t) \equiv r_t + r_{t+1} + r_{t+2} + \cdots$$
3. Average Reward,
$$V^{\pi}(s_t) = r_t + r_{t+1} + \cdots r_{t+h}$$

h

DATE: PAGE:

∴ The learning task is to learn a policy π that maximizes $V^{\pi}(s)$ for all states s

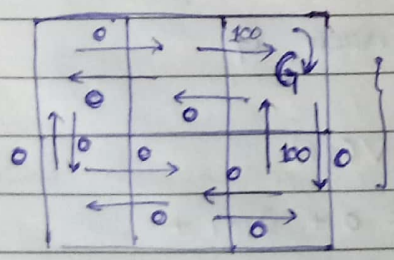
we will call such a policy an optimal policy & denote it by π^*.

$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\ V^{\pi}(s),\ (\forall s)$$

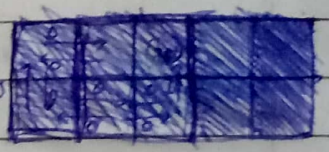
Notations used :

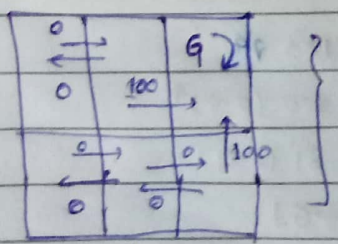
$V^*(s)$: discounted cummulative reward that agent can obtain starting from state s

$V^*(s)$: max. discounted cummulative reward that agent can obtain starting from state s ; that is, the discounted cumulative reward obtained by following the optimal policy π^* beginning at state s



Wrong. There is no coming back transition from final state.





r(s, a) or immediate reward values.

G stands for Goal state. correct. G is an absorbing state.

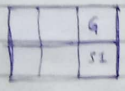
Step 1 : State - action transitions are initialized with reward values. These reward values are immediate reward values.

r(s, a)

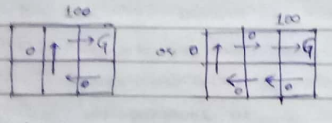
DATE: PAGE:

Rest = immediate reward values are what the agent receives if he performs action a being in state s. A makes transition to one of the adjacent states depending on the type of action performed.

Let $\gamma = 0.9$



If we are in state S1 & want to achieve goal G, we can do it in many ways



1 way to do so

another sequence of state action transitions starting from S1 to reach G.

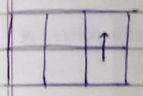
$V(S1)$
$= 0 + \gamma(0) + \gamma^2(100)$
$= \gamma^2 100$
$= 0.81(100)$
$= 81$

$V(S1)$
$= 0 + \gamma(0) + \gamma^2(0) + \gamma^3(0) + \gamma^4(100)$
$= 100\,\gamma^4$
$= 100\,(0.9)^4$
$= 81\,(0.81)$
$= 65.61$

or

$V(S1)$
$= 0 + \gamma(0) + \gamma^2(100)$
$= 100\,(0.81)$
$= 81$

or

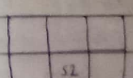


or many more

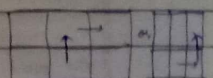
$V(S1) = 100$

∴ This sequence of state-action transition gives maximum discounted cumulative reward starting from state S1, ∴ this maximum discounted cumulative reward for state S1 is denoted as $V^*(S1) = 100$ obtained by following optimal policy π^* starting at state S1.

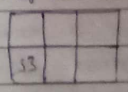
Similarly,



for state S2, optimal policy

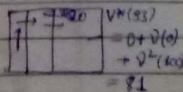


[optimal sequence of state-action transition that maximizes discounted cumm. reward is :



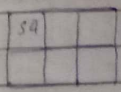
for S3

or



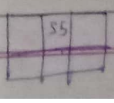
$V^*(S3) = 0 + 0.9(100) = 90$

$V^*(S3) = 0 + \gamma(0) + \gamma^2(100) = 81$

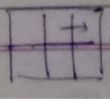


for S4

$V^*(S4) = 0 + 0.9(100) = 90$



for S5

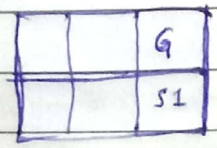


$V^*(S5) = 100$

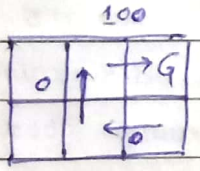
DATE: PAGE:

These immediate reward values are what the agent receives if he performs action a leaving in state s. & makes transition to one of the adjacent states depending on the type of action performed.

Let $\gamma = 0.9$

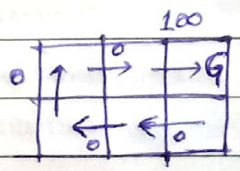


If we are in state S1 & want to achieve goal G, we can do it in many ways



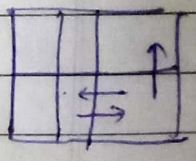
1 way to do so

or



another sequence of state action transitions starting from S1 to reach G.

or



$V(S1) = 0 + \gamma(0) + \gamma^2(100)$

$= \gamma^2 100$

$= 0.81 (100)$

$= 81$

$V(S1) = 0 + \gamma(0) + \gamma^2(0) + \gamma^3(0) + \gamma^4(100)$

$= 100\,\gamma^4$

$= 100 (0.9)^4$

$= 81 (0.81)$

$= 65.61$

$V(S1) = 0 + \gamma(0) + \gamma^2(100)$

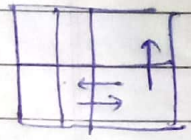
$= 100 (0.81)$

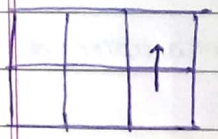
$= 81$

PAGE:

DATE: PAGE:

∵ This sequence of state-action transition gives maximum discounted cummulative reward starting from state s1, ∴ this maximum discounted cummulative reward for state s1 is denoted as $V^*(s1) = 100$ obtained by following optimal policy π^* starting at state s1.



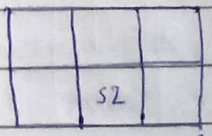


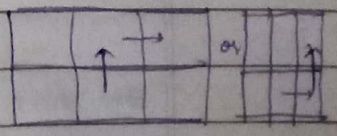
or many more.

$V(s1) = 100$

$V(s1)$
$= 0 +$
$\gamma(0)$
$+$
$\gamma^2(100)$
$= 100\,(0.81)$
$= 81$

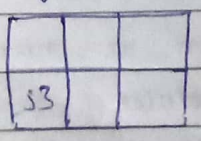
Similarly, for state s2, optimal policy



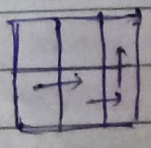


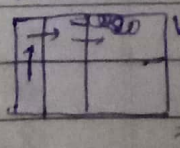
$V^*(s2) = 0 + (0.9)(100) = 90$

[optimal sequence of state-action transition that maximizes discounted cumm. reward is :

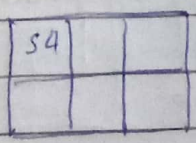


for s3

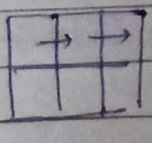




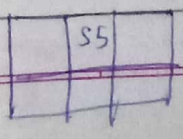
$V^*(s3) = 0 + \gamma(0) + \gamma^2(100) = 81$



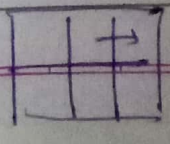
for s4



$V^*(s4) = 0 + 0.9(100) = 90$



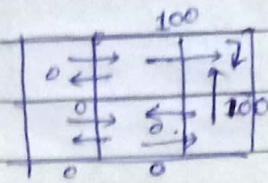
for s5



$V^*(s5) = 100$

DATE: PAGE:

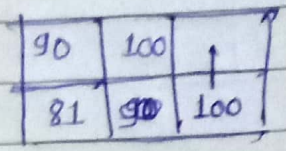
So we had



$r(s, a)$

or Immediate reward values

Now we have



$V^*(s)$

or Maximum cummulative discounted value s obtained for each s

Q Learning :

Generally,

in order to learn a mapping like

$$\pi^* : S \rightarrow A$$

we are supposed to have training examples of the form $< s, a >$.

But we do not have that.

Instead, the training information available to the learner is the sequence of immediate reward values

$$r(s_i, a_i)$$

for $i = 0, 1, 2, \ldots$

In such cases in order to learn the optimal policy for π^*,

it is easier to learn a numerical evaluation fn defined over states first & then implement the optimal policy in terms of this evaluation fn.

state s state s immediate successor state of s

DATE: PAGE:

$$\therefore \pi^*(s) = \underset{a}{\operatorname{argmax}} [r(s,a) + \gamma V^*(\delta(s,a))]$$

(recall that $\delta(s,a)$ denotes the state resulting from applying action a to state s
i.e., immediate successor state)

So learning V^* is a useful way to learn the optimal policy when the agent has perfect knowledge of δ & r.
But in case δ & r is unknown, learning V^* is unfortunately of no use for selecting optimal actions since the above action can't be used.

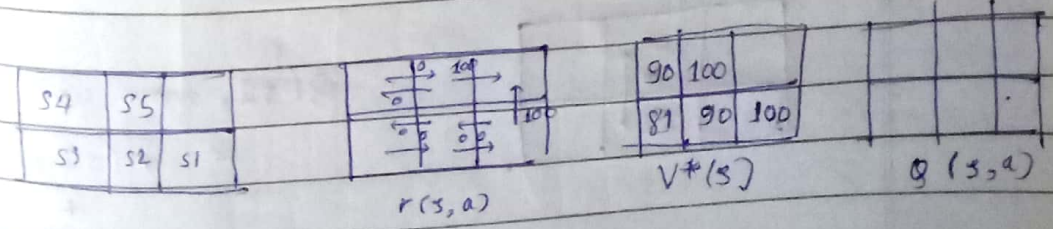
Q Function:

Let the evaluation fn be,

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

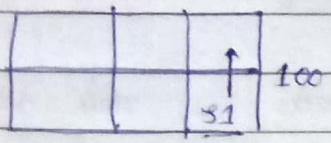
where
current state is s
reward obtained by applying action a to the current state s is $r(s,a)$
& the transition happens to a new state $\delta(s,a)$ on applying action a.

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



Calculating Q(s, a) :

DATE: PAGE:



Q(s1, move up)

= r(s1, move up)

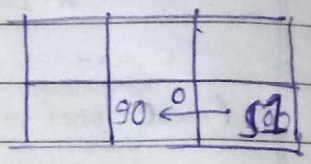
+ γ V*(G)

= 100

+

0.9(0)

= 100



Q(s1, turn left)

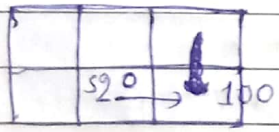
= r(s1, turn left)

+

γ V*(s2)

= 0 + 90(0.9)

= 81



Q(s2, turn right)

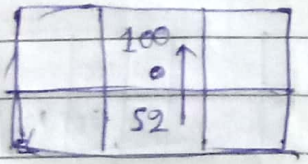
= r(s2, turn right)

+

γ V*(s1)

= 0 + 0.9(100)

= 90



Q(s2, move up)

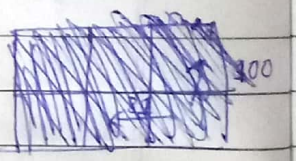
= r(s2, move up)

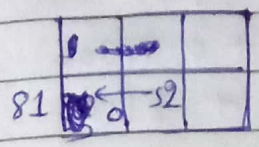
+

γ V*(s5)

= 0 + 0.9(100)

= 0 + 90 = 90





$Q(s_2, \text{turn left})$

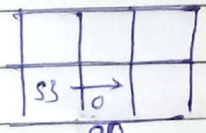
$= r(s_2, \text{turn left})$

$+ \gamma$ maximal discounted cummulative score starting from s_3 to reach the goal G

$= 0 + 0.9(81)$

$= [72.9]$

$= 72$



$Q(s_3, \text{turn right})$

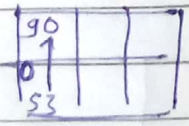
$= r(s_3, \text{turn right})$

$+$

γ maximal discounted cummulative score starting from s_4 to reach the goal G

$= 0 + 0.9(90)$

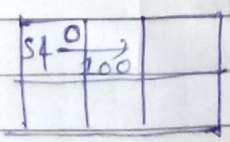
$= 81.0$



$Q(s_3, \text{move up})$

$= 0 + 0.9(90)$

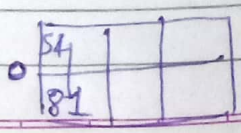
$= 81.0$



$Q(s_4, \text{turn right})$

$= 0 + 0.9(100)$

$= 90$

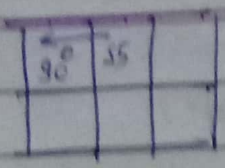


$Q(s_4, \text{move down})$

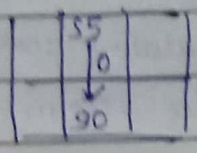
$= 0 + 0.9(81)$

$= [72.9]$

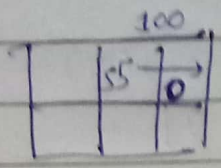
$= 72$



$Q(55, \text{turn left}) = 0 + 0.9(90)$
$= 0 + 81.0$
$= 81$

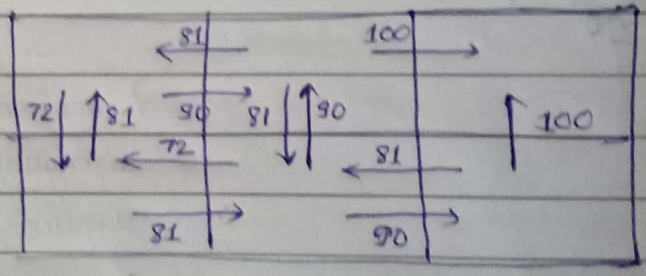


$Q(55, \text{move down}) = 0 + 0.9(90)$
$= 81$



$Q(55, \text{turn right}) = 100 + 0.9(0)$
$= 100$

∴ we get



$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$$

90	100	0
81	90	100

$V^*(s)$

DATE: PAGE:

Note that :

max { Q(s1, move up), Q(s1, turn left) }

→ max (100, 81)

= 100 = $V^*(s1)$

Surprise! Surprise!

max { Q(s2, move up),
Q(s2, turn left),
Q(s2, turn right) }

= max { 90, 72, 90 }

= 90 = $V^*(s2)$

and in general

$V^*(s) = \max_{\forall \text{ all valid actions } a' \text{ that can be performed in state } s} Q(s, a')$

DATE: PAGE:

Using this relation,

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$$

can be re-written as

$$Q(s, a) = r(s, a) + \gamma \max_{a'} \{Q(\delta(s, a), a')\}$$

for all valid actions a' that can be performed on immediate successor state s.