

Learning

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 within the cases 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 cost 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 moving from
state s_1 , action a_1

is performed &
reward r_1 is given.

to whom? It goes to state s_2 .

To the
Agent

or

For ex,

When training an agent
to play a game the

Described by
a set of possible

Environment

states S ,

interacts

by performing
action.



Agent

Trainer might provide

a positive reward when

game is won,

a set of possible

can be a

-ve reward when

actions A ,

robot

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 produce the greatest cumulative

reward. This sequence of actions that achieve its goals is

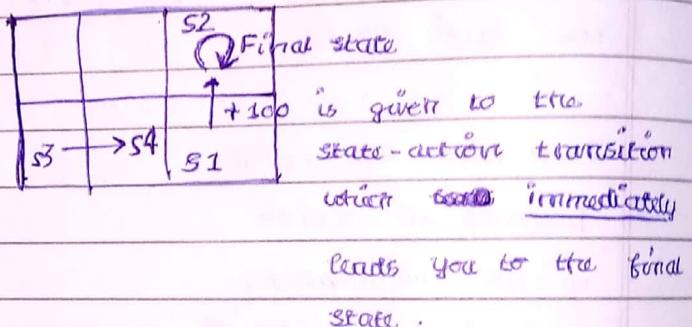
referred as control policy & this is what is to be

learned by the agent. The task is to perform all

possible sequences of actions (or state-action transitions), observe their consequences & learn a control-policy. The control-policy we

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 the rewards, with future rewards discounted exponentially by their delay.

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



From state s_1 ,

when agent goes up (action)

know that one

transition is b/w 2 adjacent states ONLY

transition

happens to adjacent

s_1 transition s_2

state s_2 .

where s_1 & s_2 are adjacent states

If state s_2 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 s_2

∴ It is referred to as delayed reward.

Learning

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 moving from
state s_1 , action a_1

is performed a
reward r_1 is given.

to whom? It goes to state s_2 .

to the
Agent

For ex,

when training an agent

described by
a set of possible

to play a game the

states S .

Environment

trainers might provide

π

interacts
by performing
action

Adversely,

a +ve reward when

Agent

is assigned

game is won,

can be from

to the

-ve reward when

can be a

state-action

transition

robot

result

game is lost,

immediately

& zero reward in all other states. The task of the agent

is to learn

from this indirect, delayed reward, to choose

the game &

sequence of actions that produce the greatest cumulative

not to

rewards. These sequence of actions that achieve its goals is

the action

referred as control-policy. & this is what is to be

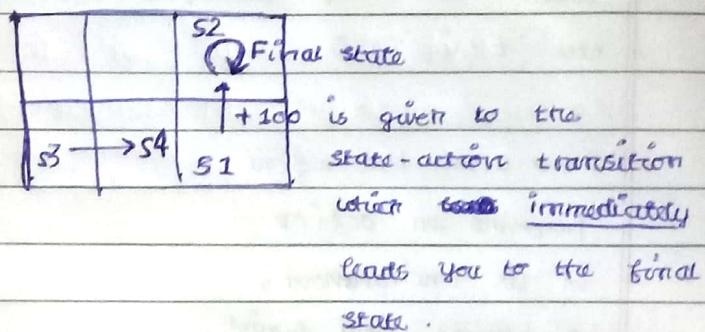
learnt by the agent. The task is to perform all

possible sequences of actions (or state-action transitions), observe

their consequences & learn a control-policy. The control-policy we

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.



[on]

From state s_1 ,

when agent goes
up (action)

transition

happens to adjacent

state s_2 .

State s_2 happens

to be a final state

then the reward is

awarded to this

state-action transition.

∴ Reward is awarded

after the transition happened

depending on the state s_2

∴ It is referred to as Delayed Reward.

Know that one

transition is b/w 2 (adjacent)

states ONLY

s_1 transition s_2

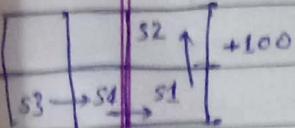
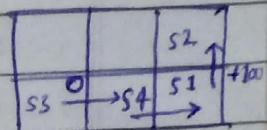
where s_1 & s_2 are

adjacent states.

Transition = $s_3 \rightarrow s_4$

Action: Turn Right

End

State = s_4 Is s_4 a final state? NO

So to state-action transition

 $s_3 \rightarrow s_4$ whereaction performed is going right,
(delayed) reward given = 0Consider transition b/w states s_4 & s_1 $s_4 \rightarrow s_1$

Action: Turn Right

Is s_1 a final state? NO

So to state-action transition

 $s_4 \rightarrow s_1$

(delayed) reward given = 0

ii) Note that,

first time when transition

happened from s_3 to s_4 (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,

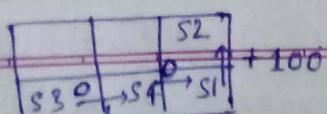
it would already know (will have prior knowledge)
the effect of such a state-action transition.

ie knows that if it takes

 $s_3 \rightarrow s_4$ it'll get reward of 0

simply

Assuming the agent has learnt



but is unaware of



$S3 \rightarrow S5$ (action: go UP)

$S5 \rightarrow S6$ (action: go RIGHT)

$S6 \rightarrow S2$ (action: go RIGHT)

∴ it never performed these actions

should ~~try~~ one try

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

iii) Life-long learning :

A agent while learning for one tasks within an environment, often will learn several related tasks \Rightarrow 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 chargers 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 chargers,

one can leverage the same knowledge / experience

to use it for task such as

picking up output from the printer.

iv. Partially observable states :

~~Assumption~~

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
sensory data when choosing actions, &
the best policy may be one that
chooses actions specifically to improve
the observability of the environment.

Generally,

where 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

& $a_t \in \{ \dots \} \subset A$

actions that can be performed

A.

"
fn
that
gives
the
succeeding
state
 s_{t+1}
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.

$$\text{Let } V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \\ = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \quad [\text{Discounted Cumulative Reward}]$$

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., $a_t = \pi(s_t)$),

$$a_{t+1} = \pi(s_{t+1}) \text{ etc.}$$

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

Four other types of other reward

that can be used are

1. Discounted cumulative Reward,

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

2. Finite Horizon Reward,

$$V^\pi(s_t) = r_t + r_{t+1} + r_{t+2} + \dots$$

3. Average Reward,

$$V^\pi(s_t) = \frac{r_t + r_{t+1} + \dots + r_{t+h}}{h}$$

∴ 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^* .

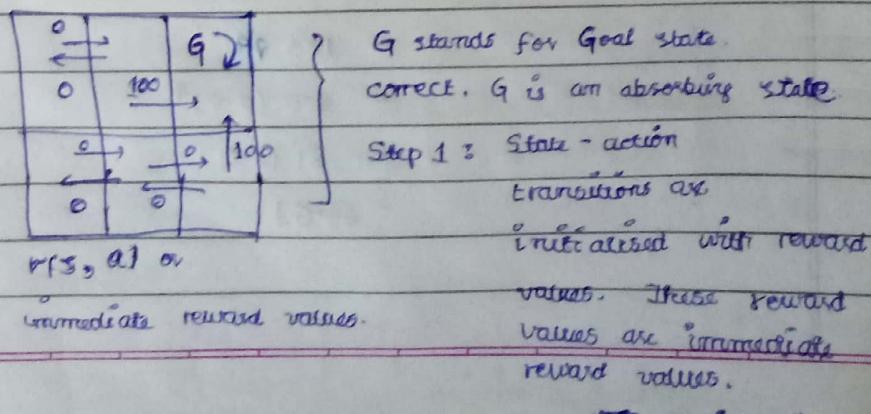
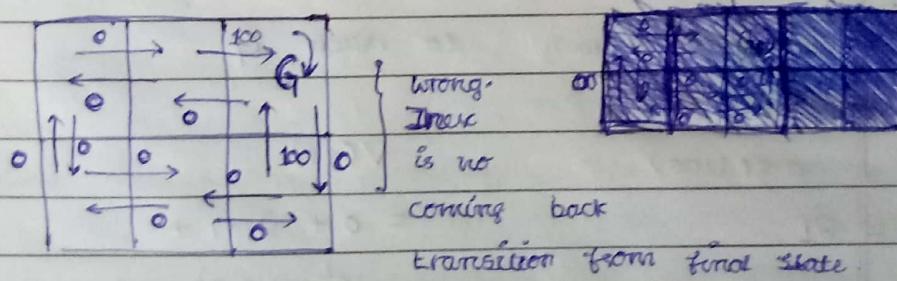
$$\pi^* = \operatorname{argmax}_{\pi} V^\pi(s), \forall s$$

Notations used :

$V(s)$: discounted cumulative reward

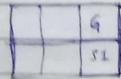
that agent can obtain starting from state s .

$V^*(s)$: max. discounted cumulative 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

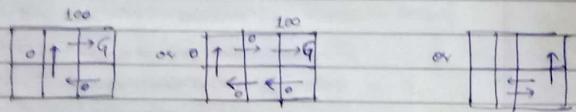


R_t = immediate reward received
are what the agent receives
if it performs action a_t
being in state s_t & makes
transition to one of the
adjacent states depending on
the type of action performed.

Let $\gamma = 0.9$



If we are in state s_1 &
want to achieve goal G ,
we can do it in many ways



1 way to another sequence
of state-action transitions

$$V(s_1)$$

$$= 0 + \gamma^0$$

scattered from s_1 to reach G .

$$+ \gamma^L(100)$$

$$= 0 \cdot 100$$

$$V(s_1)$$

$$= 100(0 \cdot \gamma)$$

$$= 0 + \gamma^0 +$$

$$\gamma^L(0) + \gamma^L(0) +$$

$$\gamma^L(100)$$

$$= 100 \cdot \gamma^4$$

$$= 100(0 \cdot \gamma)^4$$

$$= 81(0 \cdot \gamma)$$

$$= 65.61$$

* This sequence of state-action translation

gives maximum discounted cumulative reward starting

from state s_1 , i.e. this maximum discounted cumulative reward for

state s_1 is denoted
as $V^*(s_1) = 100$
obtained by following

$$V(s_1) = 100$$

optimal policy π^* starting
at state s_1

Similarly,

for State s_2 ,

optimal sequence of state-action translation that maximizes

discounted cumulative reward is

$$V^*(s_2) = 0 + (0 \cdot \gamma)/100 = 90$$

$$V^*(s_2) = 0 + \gamma^0 + \gamma^L(90)$$

$$= 81$$

$$V^*(s_3) = 0 + 0 \cdot \gamma/100 = 90$$

$$V^*(s_3) = 0 + \gamma^0 + \gamma^L(90)$$

$$= 81$$

$$V^*(s_4) = 0 + 0 \cdot \gamma/100 = 90$$

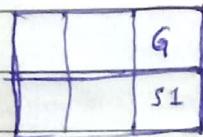
$$V^*(s_4) = 0 + \gamma^0 + \gamma^L(90)$$

$$= 81$$

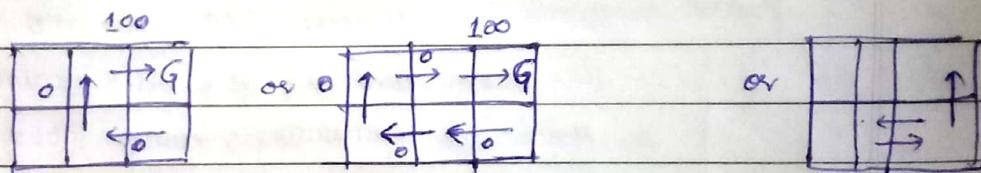
$$V^*(s_5) = 100$$

These are immediate reward values
are what the agent receives
if it performs action a
being in state s . It 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

$$V(s_1) = 0 +$$

$V(s_1)$

starting from s_1

$$\gamma(0)$$

$$= 0 + \gamma(0) + \gamma^2(100) \quad \text{to reach } G.$$

$$+$$

$$\gamma^2(100)$$

$$\gamma^2(100)$$

$$= 0.81(100)$$

$V(s_1)$

$$= 100(0.81)$$

$$= 81$$

$$= 0 + \gamma(0) +$$

$$= 81$$

$$\gamma^2(0) + \gamma^3(0) +$$

$$\gamma^4(100)$$

$$= 100\gamma^4$$

$$= 100(0.9)^4$$

$$= 81(0.81)$$

$$= 65.61$$

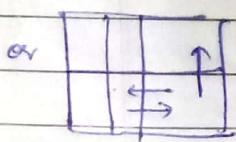
" The sequence of state-action transition

gives maximum discounted cumulative reward starting from state s_1 , i.e. thus maximum discounted cumulative reward for

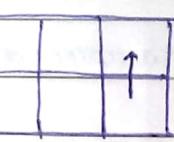
\uparrow

or many more

State s_1 is denoted as $V^*(s_1) = 100$ obtained by following



or



or many more

$$V(s_1)$$

$$= 0 +$$

$$\gamma(0)$$

$$+$$

$$\gamma^2(100)$$

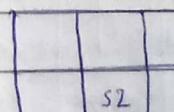
$$= 100(0.8)$$

$$= 81$$

$$V(s_1) = 100$$

optimal policy π^* starting at state s_1 .

Similarly,



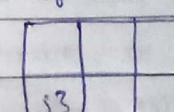
For state s_2 ,

optimal policy

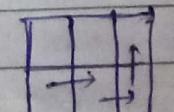
$$V^*(s_2) = 0 + (0.9)(100) = 90$$

optimal sequence of state-action transition that maximizes discounted cumm.

reward is:



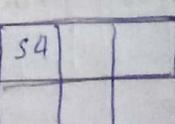
for s_3



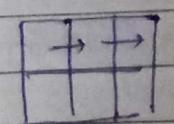
$$V^*(s_3) = 0 + 0.9(100) = 90$$

$$+ \gamma^2(100)$$

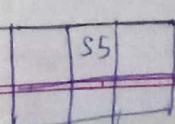
$$= 81$$



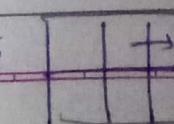
for s_4



$$V^*(s_4) = 0 + 0.9(100) = 90$$

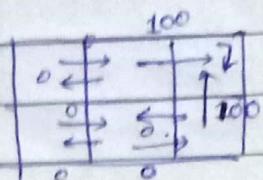


for s_5



$$V^*(s_5) = 100$$

So we had



$$r(s, a)$$

or intermediate
reward values

90	100	
81	90	100

$$V^*(s)$$

or maximum cumulative
discounted value
is obtained for
each s

Q Learning :

Generally,

in order to learn a mapping π^*

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

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 $i = 0, 1, 2, \dots$

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

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

state s state s'

DATE:

immediate successors

PAGE: state of

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

(recall that $s'(s, a)$ denotes the state resulting from applying action a to state s
i.e., ~~successor state~~ 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 abv action can't be used.

Q Function:

Let the evaluation fcn be,

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

where

current state is s

reward obtained by applying

action a to the current state s is $r(s, a)$

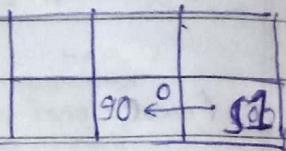
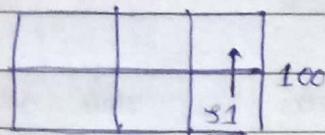
& the transition happens to a

new state $s'(s, a)$ on applying

action a .

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

s_4	s_5		$\xrightarrow{0} 100$	$90 \quad 100$	
s_3	s_2	s_1	$\xrightarrow{0} 0 \quad \xrightarrow{0} 100$	$80 \quad 90 \quad 100$	$Q(s, a)$
			$r(s, a)$	$V^*(s)$	

 $Q(s_1, \text{move up})$

$$= r(s_1, \text{move up})$$

$$\downarrow V^*(s_1)$$

$$= 100$$

+

$$0.9(0)$$

$$= 100$$

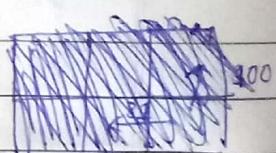
 $Q(s_1, \text{turn left})$

$$= r(s_1, \text{turn left})$$

$$\downarrow V^*(s_2)$$

$$= 0 + 90(0.9)$$

$$= 81$$

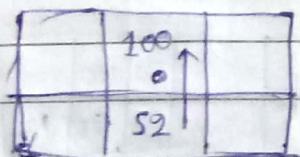
 $Q(s_2, \text{turn right})$

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

$$\downarrow V^*(s_1)$$

$$= 0 + 0.9(100)$$

$$= 90$$

 $Q(s_2, \text{move up})$

$$= r(s_2, \text{move up})$$

$$\downarrow V^*(s_5)$$

$$= 0 + 0.9(100)$$

$$= 0 + 90 = 90$$



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

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

+

maximal discounted

cumulative score

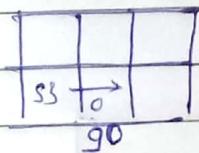
starting from s_3

to reach the goal G

$$= 0 + 0 \cdot g(s_3)$$

$$= [72 \cdot 9]$$

$$= 72$$



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

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

+

maximal discounted

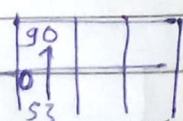
cumulative score

starting from s_3

to reach the goal G

$$= 0 + 0 \cdot g(90)$$

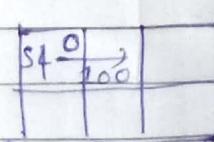
$$= 81 \cdot 0$$



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

$$= 0 + 0 \cdot g(90)$$

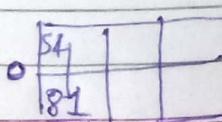
$$= 81 \cdot 0$$



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

$$= 0 + 0 \cdot g(100)$$

$$= 90$$



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

$$= 0 + 0 \cdot g(81)$$

$$= [72 \cdot 9]$$

$$= 72$$

90	55	
0		

 $Q(55, \text{turn left})$

$= 0 + 0 \cdot 9(90)$

$= 0 + 81 \cdot 0$

$= 81$

55	
0	
90	

 $Q(55, \text{Move down})$

$= 0 + 0 \cdot 9(90)$

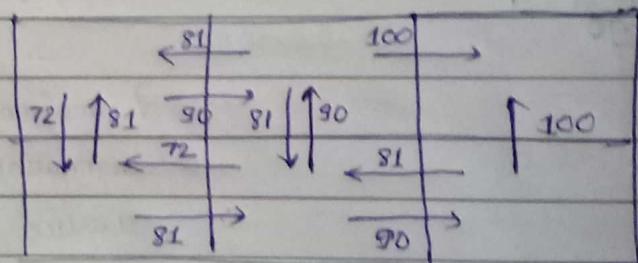
$= 81$

100	
55	0
0	

 $Q(55, \text{turn right})$

$= 100 + 0 \cdot 9(0)$

$= 100$

 $\therefore \text{we got}$ 

$Q(s, a) \equiv r(s, a)$

 $+$

$\gamma v^*(\delta(s, a))$

90	100	0
81	90	100

$v^*(s)$

Note that :

$$\max \{ q(s_1, \text{move up}), q(s_1, \text{turn left}) \}$$

$$\rightarrow \max(100, 81)$$

$$= 100 = V^*(s_1)$$

Surprise! Surprise!

$$\begin{aligned} \max \{ & q(s_2, \text{move up}), \\ & q(s_2, \text{turn left}), \\ & q(s_2, \text{turn right}) \} \end{aligned}$$

$$= \max \{ 90, 72, 90 \}$$

$$= 90 = V^*(s_2)$$

and in general

$$V^*(s) = \max q(s, a)$$

+ all
 valid actions
 a' that can
 be performed
 in state s

Using this relation,

$$Q(s, a)$$

$$= r(s, a)$$

+

$$\gamma V^*(\delta(s, a))$$

can be re-written as

$$Q(s, a) = r(s, a) +$$

$$\gamma \max \{ Q(\delta(s, a'), a') \}$$

for all

valid actions

a' that can

be performed

in ~~the~~

immediate

successor state

∴