Reinforcement Learning

position of helicopter ⟶ how to move control sticks

state s ⟶ action a

$x \cancel{\longrightarrow} y$

reward function ⟶ Positive reward : +1
negative " : −1

what to do

Controlling robots
factory optimization
financial trading
Playing games

Mars Rover Examp

	100	0	0	0	0	40
state	1	2	3	4	5	6

$\xleftarrow{\text{left}}$ $\xrightarrow{\text{right}}$

state	4	3	2	1		
	0	0	0	100		
	0	0	40			
	0	0	0	0	0	100

$(s, a, R(s), s')$

$(4, \leftarrow, 0, 3)$

Reward associated with state 4

Return in reinforcement learning

$$\text{Return} = 0 + 0(0.9) + 0(0.9)^2 + 100(0.9)^3 = 0.729 \times 100 = 72.9$$

$$= R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots$$

Discount factor $\gamma = 0.9$ $\quad 0.99 \quad 0.999$

$\gamma = 0.5$

0

$$\text{Return} = 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$$
$$= 12.5$$

100	50	25	12.5	6.25	40
100	0	0	0	0	40

$\gamma = 0.5$

100	2.5	5	10 →	20	40
100	0	0	0	0	40

$$\hookrightarrow \; 0 + (0.5)0 + (0.5)^2 40$$
$$= 10$$

Policy

State s $\xrightarrow[\pi]{}$ action a

$\pi(s) = a$

Policy Controller

$\pi(2) =$ ⇐

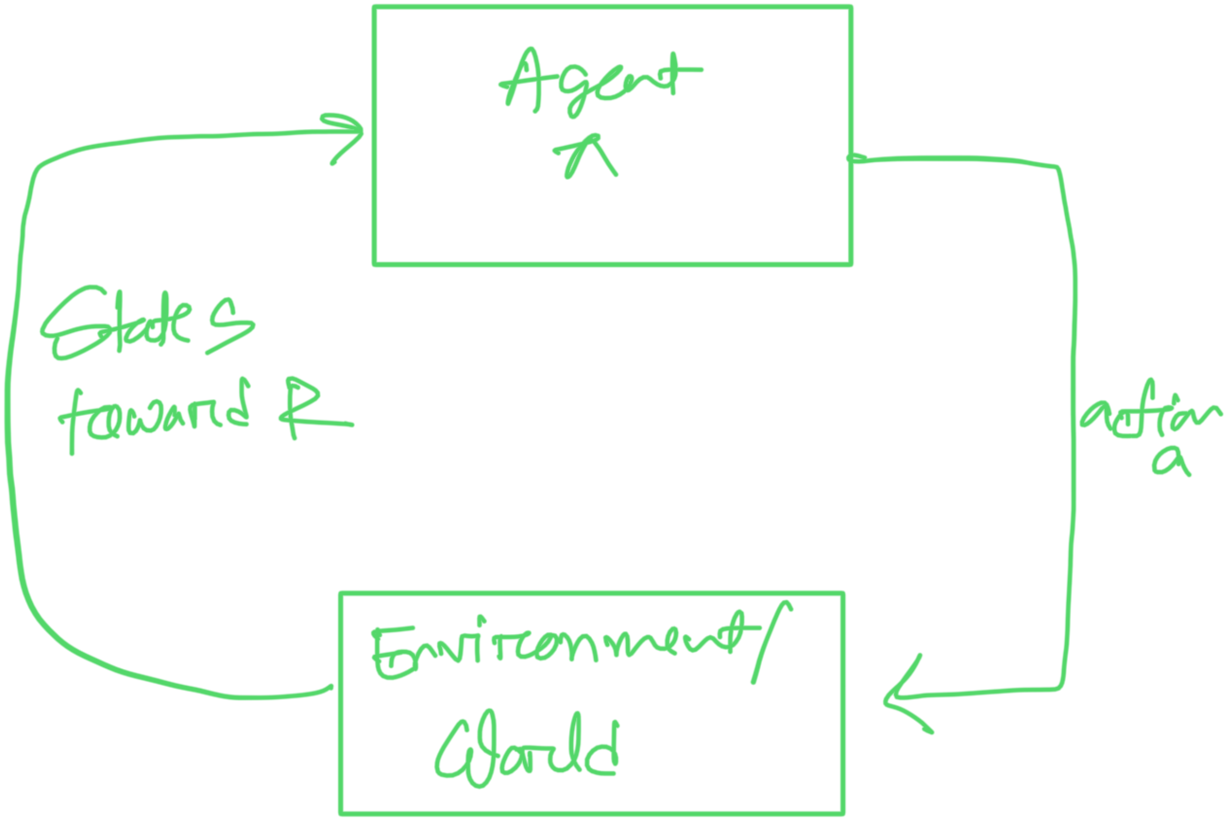
$\pi(3) =$ ⇐

$\pi(4) =$ ⇐

$\pi(5) =$ ⇒

Markov Decision Process (MDP)

	Mars rover 	Helicopter 	Chess 
states	6 states	position of helicopter	pieces on board
actions	← →	how to move control stick	possible move
rewards	100, 0, 40	+1, −1000	+1, 0, −1
discount factor γ	0.5	0.99	0.995
return	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
policy π	100 ← ← ← → 40	Find $\pi(s) = a$	Find $\pi(s) = a$



State action value function

$Q(s,a)$ = Return if you

- start in state s
- take action a (once)
- then behave optimally after that.

$$Q(2, \rightarrow) = 12.5$$

$$0 + 0(0.5) + (0.5)^2 0 + (0.5)^3 100$$

$$Q(2, \leftarrow) = 50$$

$$0 + (0.5) 100$$

$$Q(4, \leftarrow) = 12.5$$

$$\underset{12.5}{Q(4, \leftarrow)} \quad \underset{10}{Q(4, \rightarrow)}$$

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

$$\pi(s) = a$$

Bellman equation

$Q(s,a)$ = Return if you

⇨ ~~start~~ in state s

⇨ take action a (once)

⇨ Behave optimally

s: current state

a: " action

s': state you get to after taking action a

a'': action you take in state s'

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

$$Q(2, \rightarrow) = R(2) + 0.5 \max_{a'} Q(3, a')$$

$$= 0 + (0.5)\, 25 = 12.5$$

The best possible U from s' is $\max_{a'} Q(s',a')$

$$R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots$$

$$s \longrightarrow s'$$

Random (Stochastic) Environment

$$\text{Expected Return} = \text{avg}(R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots)$$

$$= E[R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots]$$

Bellman Equation: $Q(s,a) = R(s) + \gamma E[\max_{a'} Q(s',a')]$

$$Q(5, \leftarrow) = 0 + (0.5)0 + (0.5)^2 0 + (0.5)^4 100$$

$$= 0 + (0.25)0 + (0.25)^2 40$$

$$= 0 + 0.25 \cdot 0 + 0.25^2 0 + 0.25^3 40$$

Discrete Vs Continuous State

[...] helicopter

$$S = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

$$S = \begin{bmatrix} x \\ y \\ \alpha \\ \phi \\ \theta \\ \omega \\ \dot{x} \\ \dot{y} \\ \dot{\alpha} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}$$

Lunar Lander

action: do nothing
left thruster
main "
right "

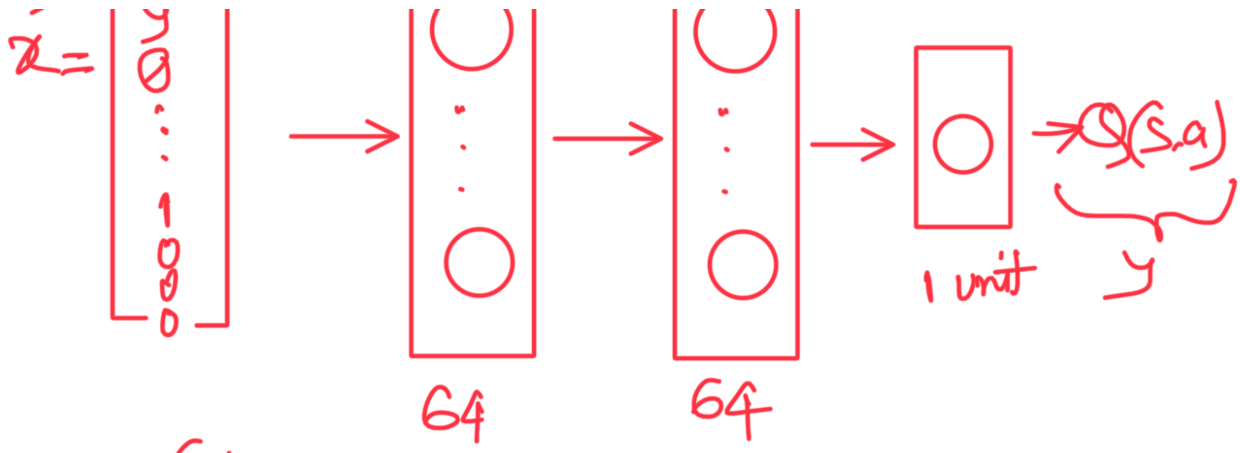
$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$$

0, 1

pick action $a = \pi(s) \longrightarrow$ maximize return

$\gamma = 0.985$

$\rightarrow \begin{bmatrix} x \\ \dot{} \end{bmatrix}$



$f_{w,b}(x) \approx y$

$(s, a, R(s), s')$

$(s^{(1)}, a^{(1)}, R(s^{(1)}), s'^{(1)})$

$(s^{(2)}, a^{(2)}, R(s^{(2)}), s'^{(2)})$

x	y
$x^{(1)} = (s^{(1)}, a^{(1)})$	$y^{(1)}$
8 4	
$x^{(2)} = (s^{(2)}, a^{(2)})$	$y^{(2)}$
x^{10000}	y^{10000}

$$y^{(1)} = R(s^{(1)}) + \gamma \max_{a'} Q(s'^{(1)}, a')$$

Learning algo

initialize NN randomly as guess $Q(s,a)$

Repeat {

Take action in lunar lander, Get $(s, a, R(s), s')$

Store 10k recent $(s, a, R(s), s')$ tuples

Train NN

create training set 10k examples

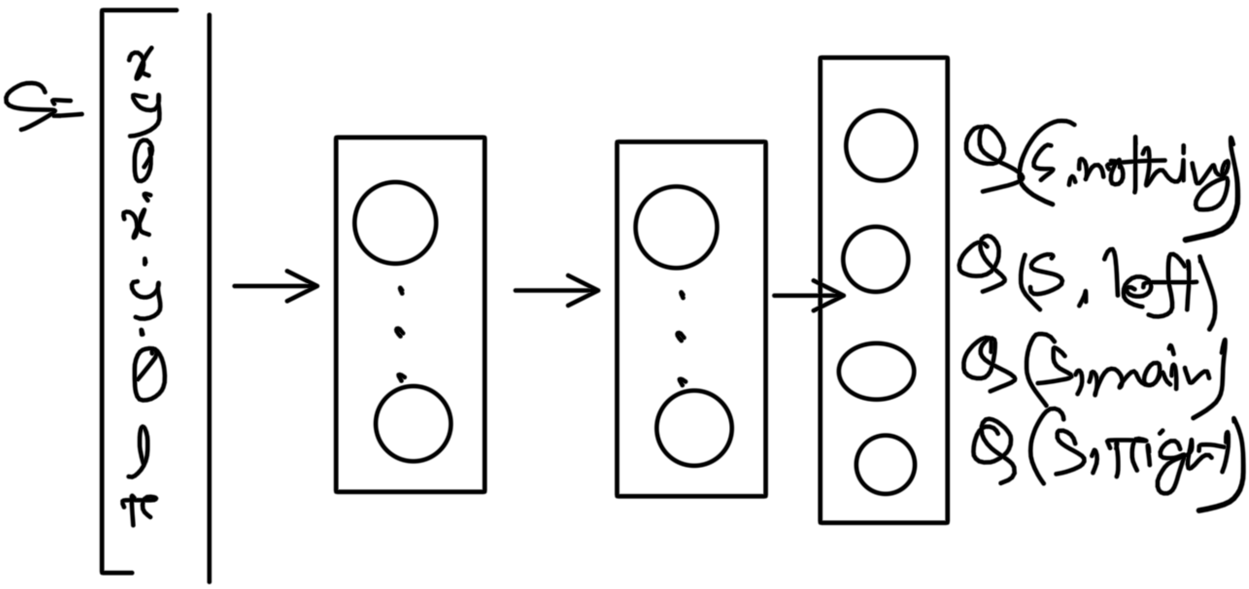
$x = (s,a)$ & $y = R(s) + \gamma \max_{a'} Q(s',a')$

Train Q_{new} such that $Q_{new}(s,a) \approx y$

set $Q = Q_{new}$

Deep Q Network

Continous State Space



How to choose actions while still learning

In some states

Option 1

pick the action a that maximizes $Q(s,a)$

Option 2

with probability 0.95, pick the action that maximizes $Q(s,a)$. Greedy, Exploitation

with prob 0.05, pick an action randomly "exploration"

ε-greedy policy

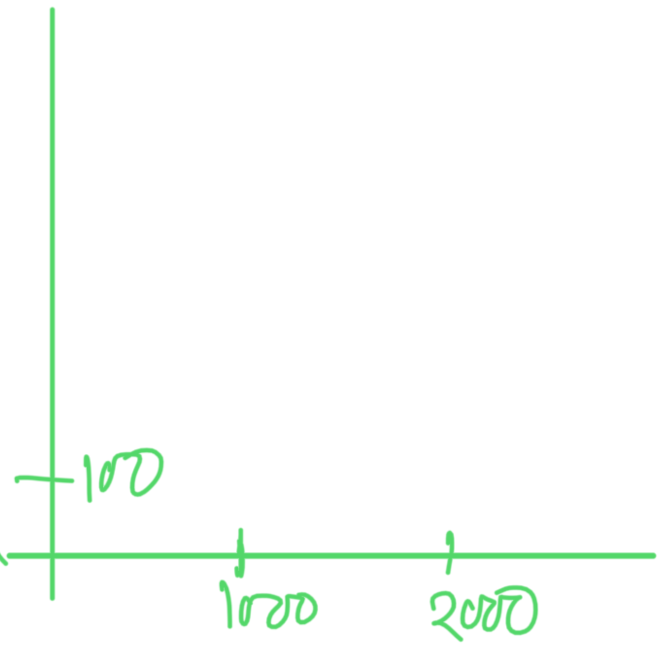
Start ε high
$1.0 \rightarrow 0.01$
gradually decrease

Algorithm refinement: Mini-batch & soft updates (optimal)

} mini batch 1

} mini batch 2

} mini batch 3



Batch learning

1000 instead of 10k

Soft update

$Q = Q_{new}$

$\uparrow$

w_{new}, B_{new}

$w = 0.01\, w_{new} + 0.99\, w$

$B = 0.01\, B_{new} + 0.99\, B$

Reinforcement learning