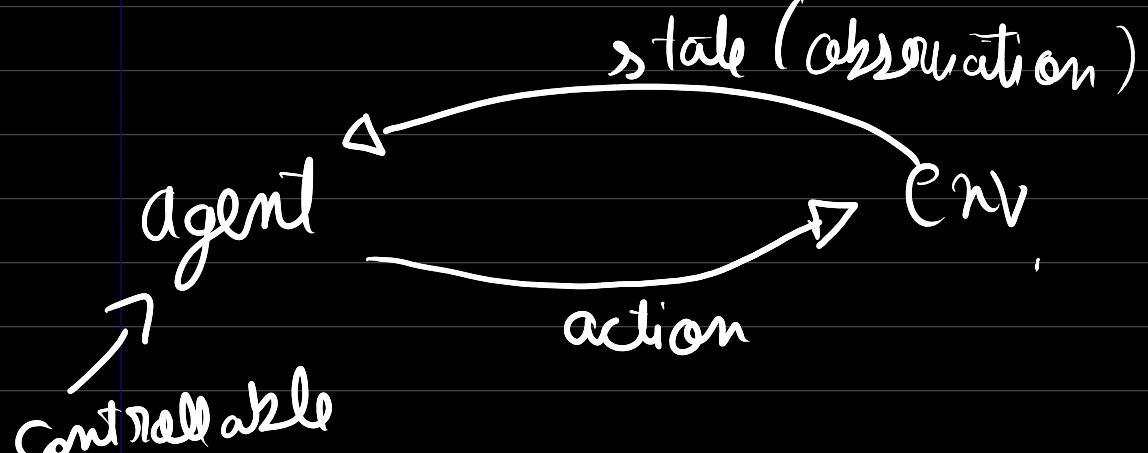
Reinforcement learning

Chap 1 : Makov Decision Process

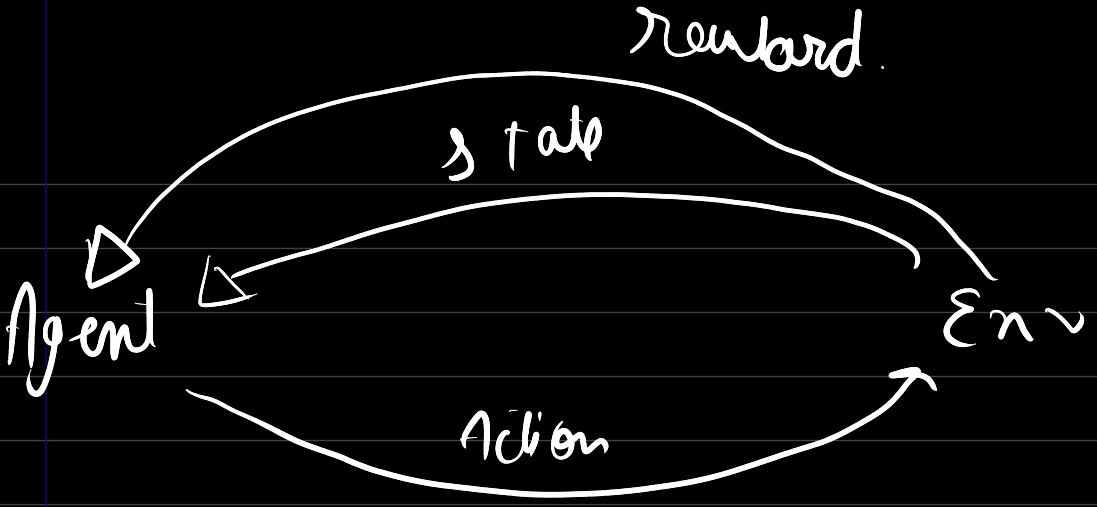


state - position
= velocity
= wind velocity
- temperature

action - motor speed
- direction

Eg. Driving car

program	agent	env	action	state
A	brain	limbs	neuron	limb position
B	human	car controls	limb movement	elements of car
C	car	road	car movement	road situation



MDP markov decision Process

$S - a - r$, $S - a - r$

single episode.

S_0, a_0, r_1 , S_1, a_1, r_2 + ...

state action reward

- Markov principle: state only dependent on immediate prev state

↳ go-to recent state

state can depend on more prev states

policy $\xrightarrow{(\pi)}$ $\pi : S \to a$

takes state → gives action

$\pi(a|s)$ = probability of action.

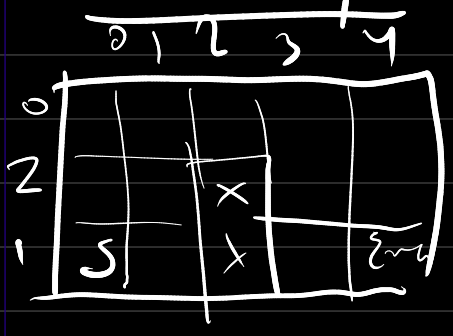
return G_t → reward

$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$

γ → decay factor

— Chap 2

→ Grid example:



→ How?

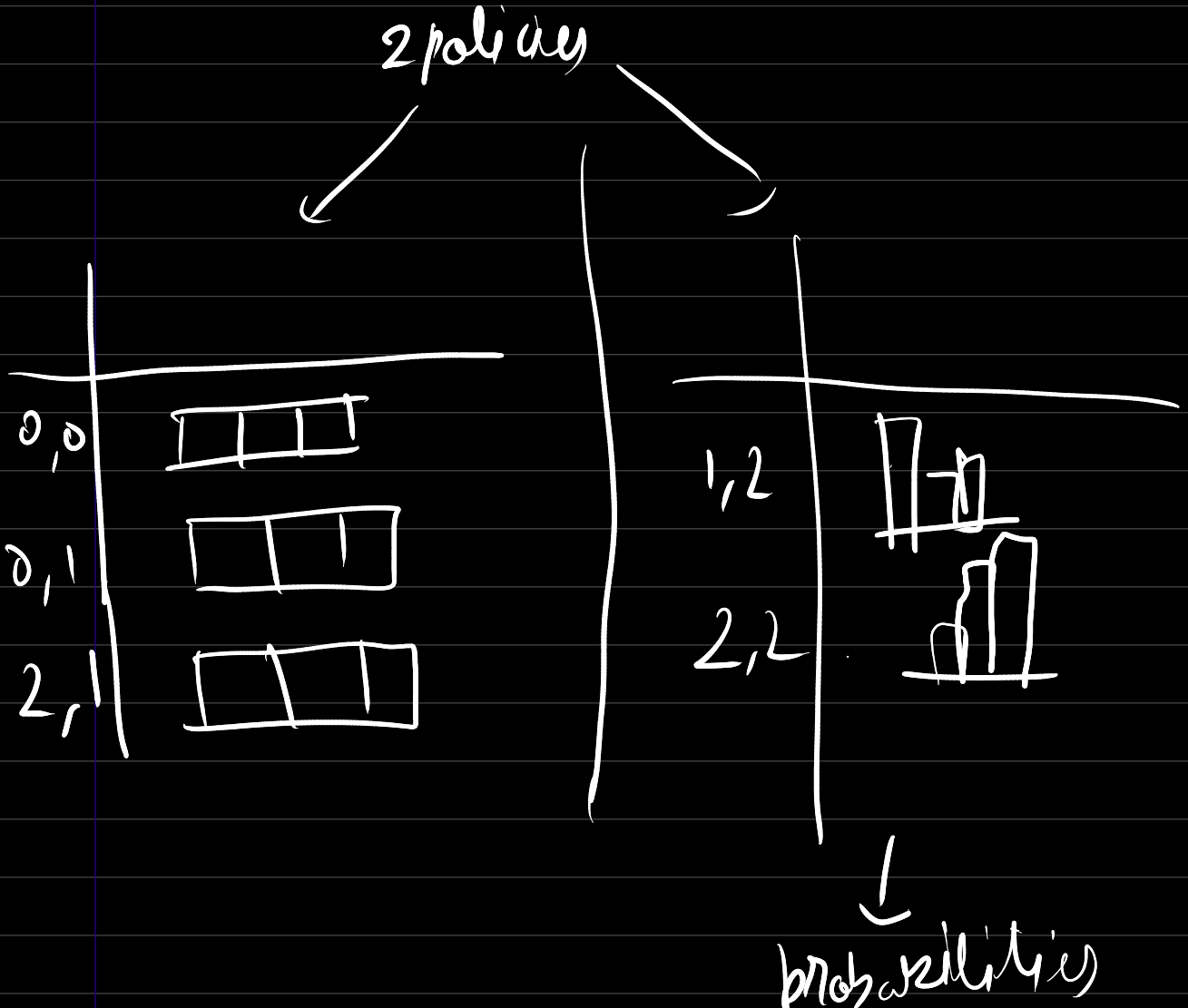
State : 2 nums - x, y coords.

action : 1 num - 0/1/2/3 || up/down/left/right

reward : 0 if in target cell
-1 otherwise

on
off] Model

↳ no access to world model



→ sampling → rollouts Sampled Trajectory

$\gamma = 0.8$ eg.

State	action	Next State	Reward	Return
				-7.33
3.0	1	3.1	-1	-1.8
3.1	2	4.1	-1	-1
4.1	1	4.2	0	0

$\gamma = 0.8$

It's successful one, when the agent reaches target

if it doesn't return is more negative

- Policy gradient method : change of policy
  - rate of change

State-value $V_\pi(s)$

action-value $Q_\pi(s,a)$

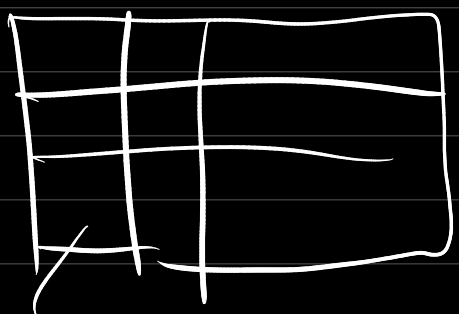
$V_\pi(s)$ $Q_{(\pi)}(s,a)$

↳ not as probability

- Policy gradient method
- Policy gradient method with value fn
- Just use action-value fn

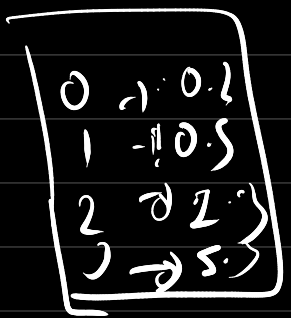
We Bendy Q function → that evaluates policy

Q_{fn}



↓

Better action

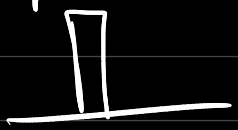


Q ——→ Q_* → Best case scenario

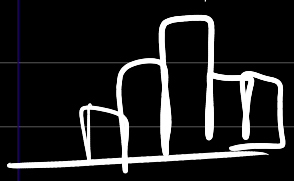
← maintaining balance between exploration and exploitation

exploration

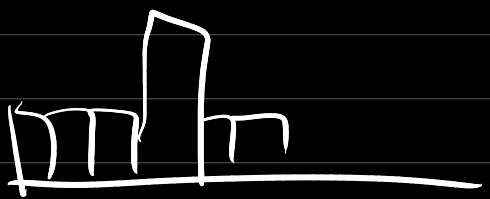
exploitation



policy gradient balance



Egreedy balance



0.9

Random + High probability

0.9 → 0.1

Monte Carlo method

→ Chapter 3 : Temporal difference.

Monte Carlo → works on episode basis

good

bad

→ need to wait for ep to end to evaluate

if long ep → problem

which action good?

" " bad?

→ impact of individual action ← Credit assignment problem

→ SARSA

Choosing Best

[How much better or worse relative to new state]

Q learning

max(action) all

[Highest estimated]

On policy — behavior policy = target policy

off policy — behavior policy ≠ target policy

↳ Q learning

{ sample efficient }

Q learning is better

Chapter - 4 : Deep Q Networks

Discrete states → continuous states ✓

Discrete actions → continuous actions ✗

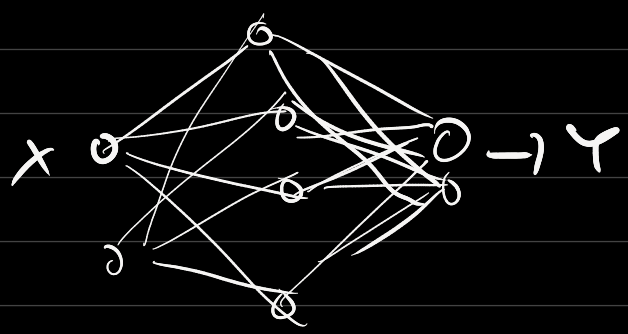
value based fn → DQN

Swap table with n-n

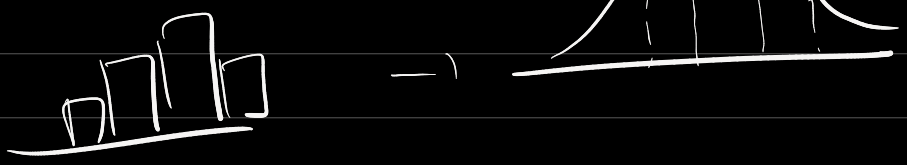
nn → fn approximator.

Policy Gradient : Chap 5

⇒



(, track of dist of probability



↓ loss fn

Inc probability of action acc to how good it is

→ Probabilities adjust acc. to definition

→ RL can be reliable.

Too much episodes