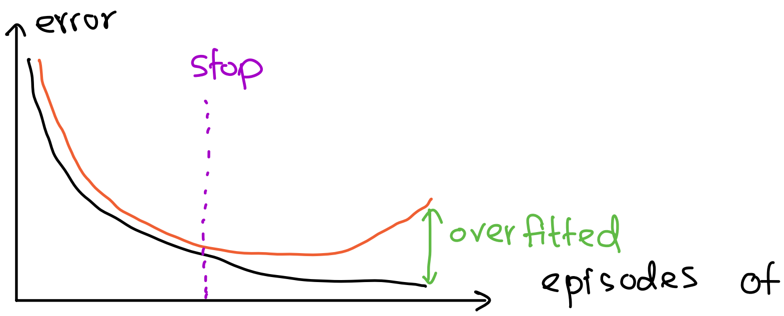
Overfitting in DL is a serious issue.

What we can do against it?

① Augmentation : translation, rotation, scaling

increase the size of training data

② Early stopping :



③ Dropout - Randomly remove neurons

* Network become more sensitive to individual neurons which leads to better Generalization.

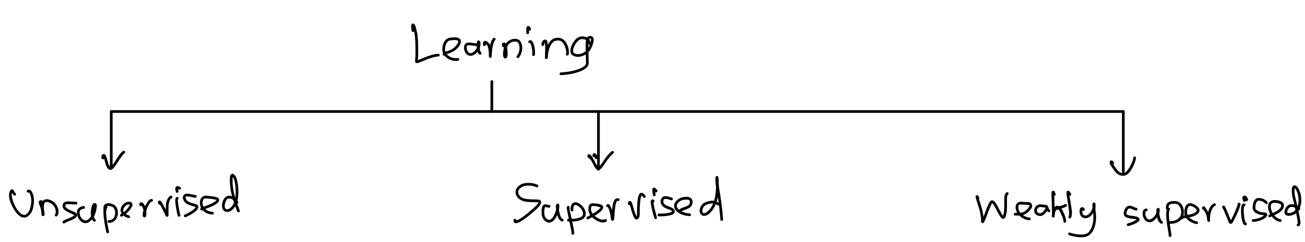
* Equvalent to averaging several models

④ Weight penalties using L_2 or L_1

$E^* = E_{total} + \|W\|$ regularizer

$E^* = E_{total} + \|W\|_2$

The model with smaller weights is better



[SOM]
no labelled data

[MLP, CNN, AEs]
labelled data

[Reinforcement Learni-ng]
Reinforcement signal

RL agents come from:

AI / control theory / operation Research / Psychology / cognitive science

RL is learning through interaction.

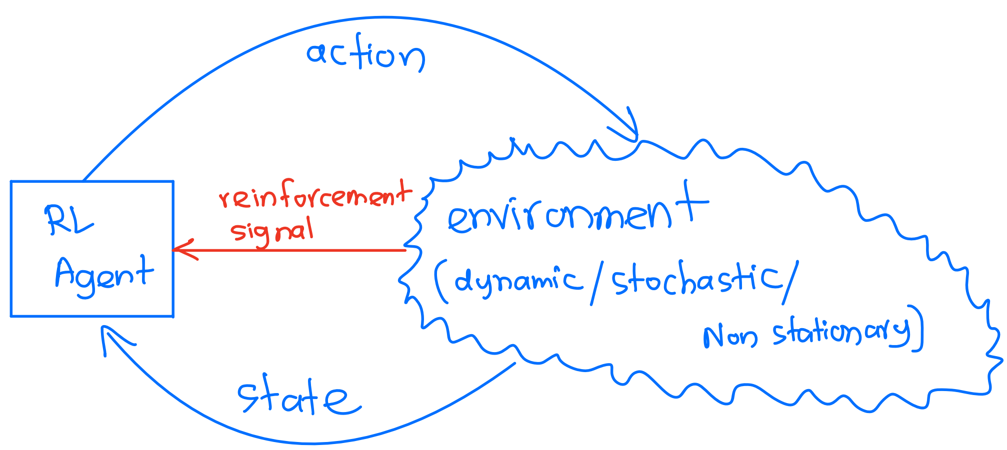
[learning by doing]

[learning from scratch]

RL goal is the control of large scale stochastic environment with partial knowledge

RL applications

- Sheduling (eg: dynamic channel allocation)
- Board games (eg: Backgammon, checkers, Go, chess)
- Robotics (eg: Robosoccer, elevator control)



Reinforcement signal = Reward or punishment

"controlling" any dynamic env means to find the trade off between "exploration" and "exploitation".

Look for knowledge   use knowledge

RL agent influences the state of distribution. Hence it make decisions/take actions to maximize reward or minimize punishment.

$s \in S$ states

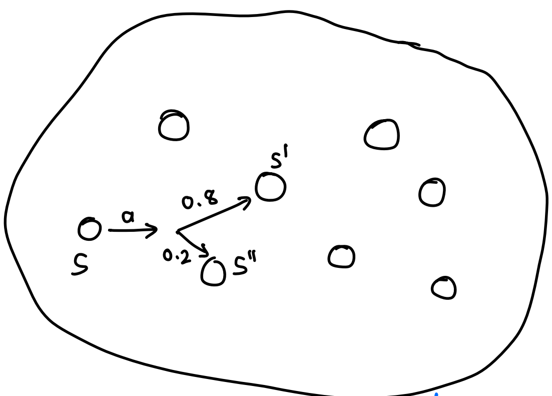
$a \in A$ actions

R reward function

Markov Decision Process (MDP)

δ transition probability $[\delta(s_1 \xrightarrow{a} s_2)]$

In state 1, then take action an go to state 2.



Stochastic environment

$P(s'|s,a) = 0.8$

$P(s''|s,a) = 0.2$

Taking action in state s causes the transition $\delta(s,a,s')$ with $p = 0.8$ and $\delta(s,a,s'')$ with $p = 0.2$.

$R(s,a)$ = reward for action a taken in state s.

(indirect guidance)

* There is no target to calculate the error

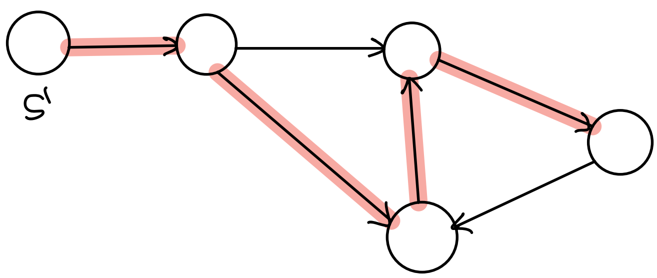
$R(s,a) = +10$ with $p = 0.1$

$R(s,a) = +5$ with $p = 0.3$

$R(s,a) = -5$ with $p = 0.6$

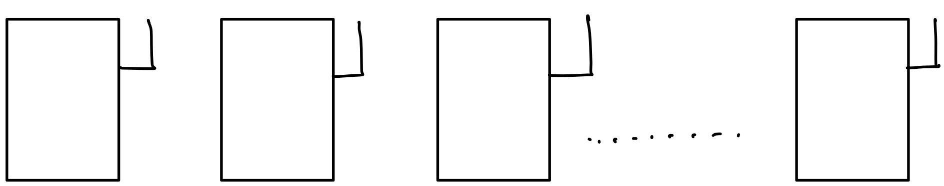
We have to follow trajectories.

$$s \xrightarrow[a_1]{} s' \xrightarrow[a_2]{} s'' \xrightarrow[a_3]{} s''' \ldots\ldots$$



terminal state (eg: winning S chess)

Classic example : N-armed bandits



Optimal strategy ?

How to maximize sum of wins (rewards)

Scenario A : Stick to the machine with high payout. (exploit)

Scenario B : frequently switch between machines (explore)

Challange : No supervision, but there is a reward

exploitation = short term reward

exploration = long term return

Also early vs late rewards : Are they same ?

(System) Environment endless or terminal ?

Driving force of learning in ANNs is the error.

Driving force of learning in RL is the return.

Return = (discounted) sum of rewards

weighted : late rewards have higher weights.

Return : ① Finite horizon (termina)

$$R_{tot} = \sum_{i=1}^{n} R(s_i, a_i)$$

② Infinite horizon (endless)

$$R_{tot} = \sum_{i=0}^{\infty} \gamma^i R(s_i, a_i)$$

$\gamma^i < 1$ discount weight

Condition : We have to measure / visit all states

Is environment observable ? (Are there hidden variables?)

We need to map states to action. This is called a policy. $\pi : S \longrightarrow a$

Tabular RL $\longleftarrow$

π	a_1	a_2	...	a_n
s_1				
s_2				
$\vdots$				
s_n				

Best action for all states = Optimal Policy π^*

RL goal : Find $\pi^* : s \longrightarrow a$ using $R(s,a)$ and a suitable return horizon

Do we know $\delta(s,a,s')$?

2 potential approaches.

① Find the value of state (difficult)

Dynamic programming

May be billions of states

② Find the value of action (easier)

and few actions.