▷ Reward hacking

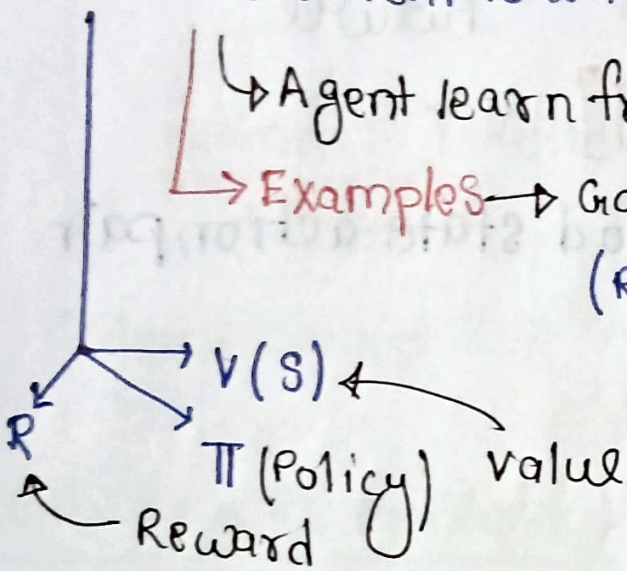
▷ In next class; RL algorithms and policies.

Lecture-20 (6-November-24) | Part A

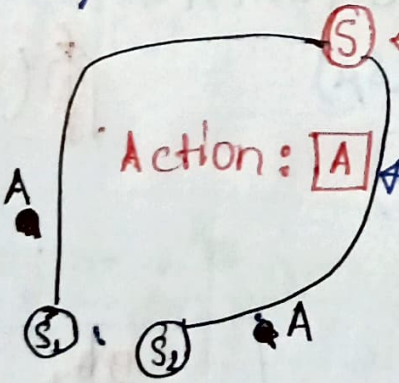
Reinforcement learning

↳ Agent learn from environment

→ Examples → Game playing
(RL gives values by self game playing)



Markov Decision Process (MDP)



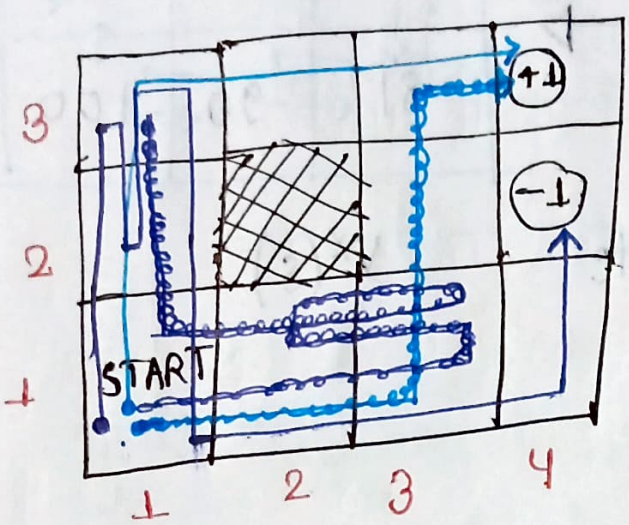
The out come is not deterministic (stochastic Environment)

▷ our assumption is that world is deterministic

Passive learning

→ Already has some particular policy 'π'

→ Agent is trying to evaluate the policy

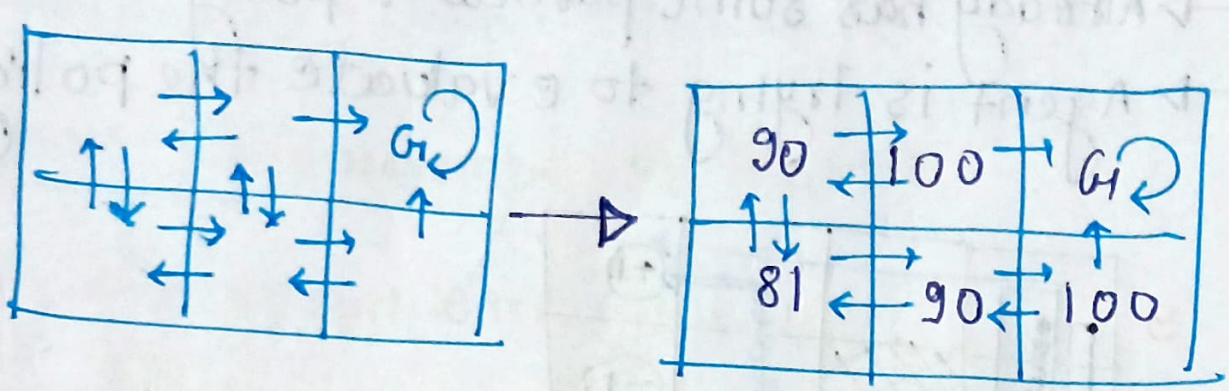


eg. (a) (1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3) [1]

(b) (1,1) → (1,2) → (1,3) → (1,2) → (1,3) → (1,2)
→ (1,1) → (2,1) → (3,1) → (4,-1) → (4,2) [-1]

n sequences and try to improve the policy by evaluating the policy

Epoch



$r(s, a)$: Immediate reward values

$V^*(s)$

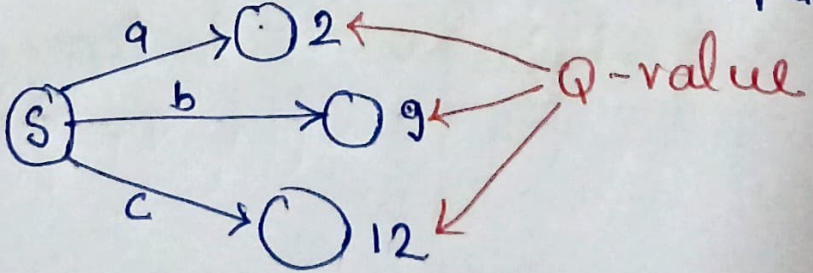
$$V^* s = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots$$

Assuming $\gamma = 0.9$

▷ Agent does not have idea about complete space state thats why q-learning come into picture

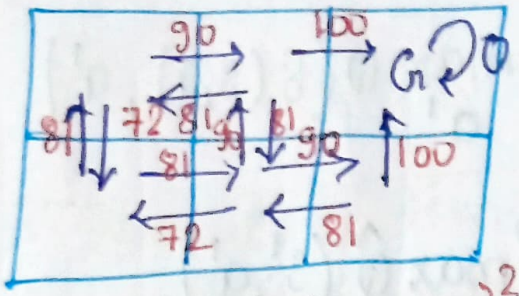
Q-Learning

Q-value: It will give a value/cost for each state-action pair



▷ Q-value provides goodness of the state-action pair.

▷ Q-learning is how we derive and use Q-value



$Q(S,a)$

$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2}$

$\pi^* s = \arg\max_a Q(s,a)$

$V^* s = \max_a Q(s,a)$

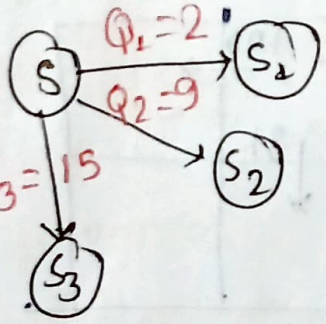
Now:

Immediate

Discount function

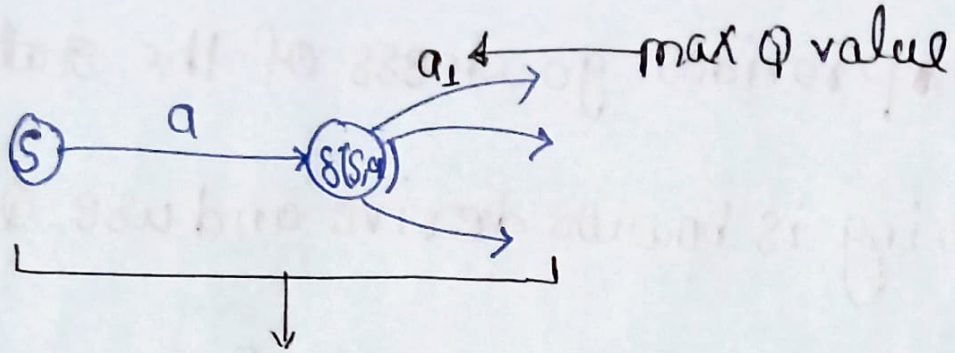
$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

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



This still depends on $r(s,a)$ and $\delta(s,a)$

Recursive definition



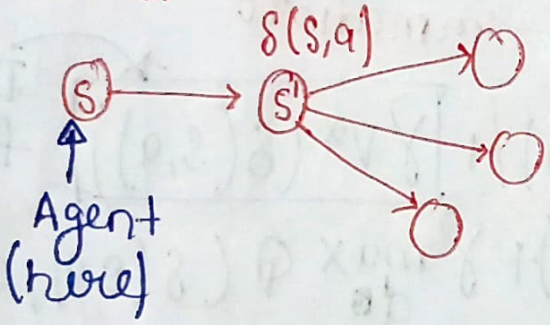
$$\gamma \max_{a'} Q(\delta(s,a), a')$$

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

where $s' = \delta(s,a)$

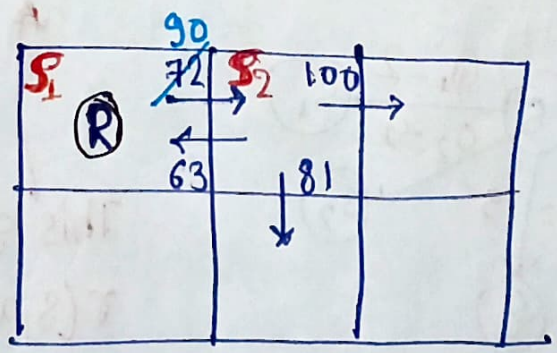
Bellman equation

▷ In Q-learning, we are looking 1 step ahead



▷ Modify Q(s,a) value

▷ Hyper parameter



initial state: s_1

$$\hat{Q}(s_1, \text{Right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$

γ: Gamma

$\gamma = 0.9$ (Discount factor)

$0 + 0.9 \times \max(63, 81, 100)$

$= 0.9 \times 100$

$= \boxed{90}$

- We will do 'n' number of times until the values are converging (reaching some values that does not change)
- Start with some initial Q-values

Similar to **Local Search**

- We might get stuck with some local optima (sub-optimal value)

We could use something like this

$P(a|s) \propto$.

- Simulated Annealing
- Prioritized sweeping
- Temporal Difference Learning

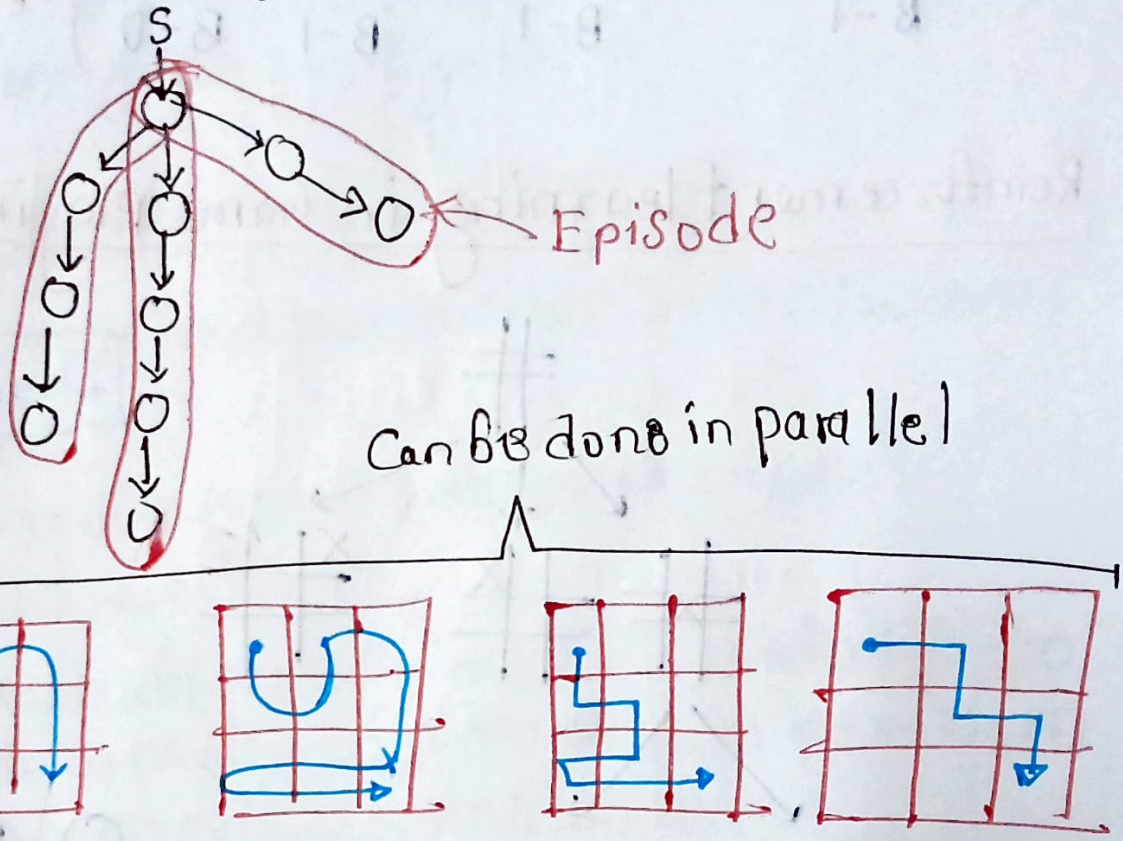
} Optimizations

- Deep Reinforcement learning
- Probabilities for stochastic environment

Lecture-20 (6/11/24) /Part-B

Monte Carlo Policy Evaluation

▷ We devise policy evaluation to different episode.

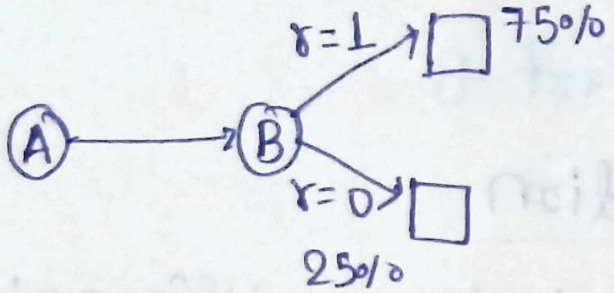


▷ Simulated environment is used.

▷ We update Q value after each episode.

Temporal Difference learning

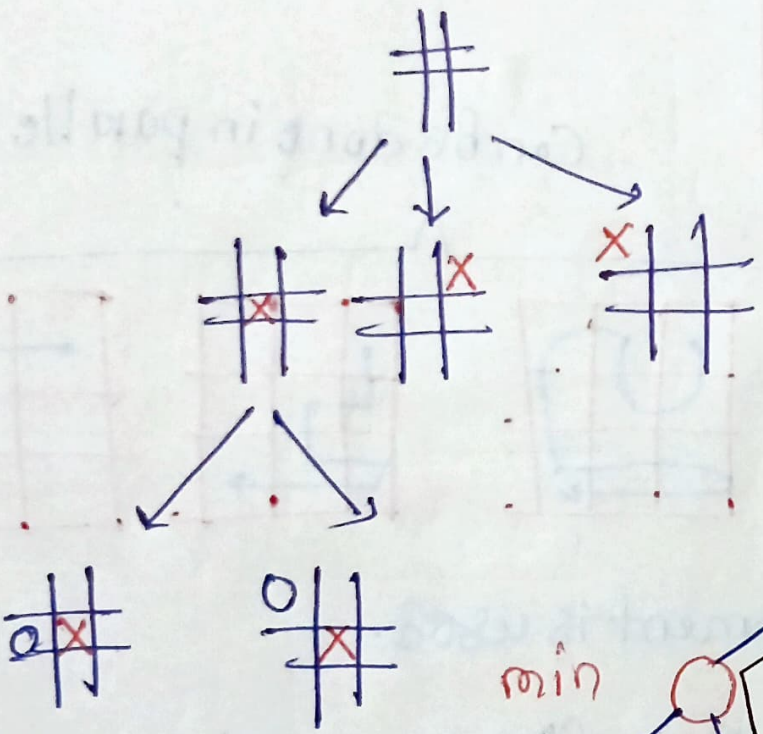
→ In Q learning we check only one step ahead
→ In Temporal difference learning, agent check upto 1 episode and update Q-value

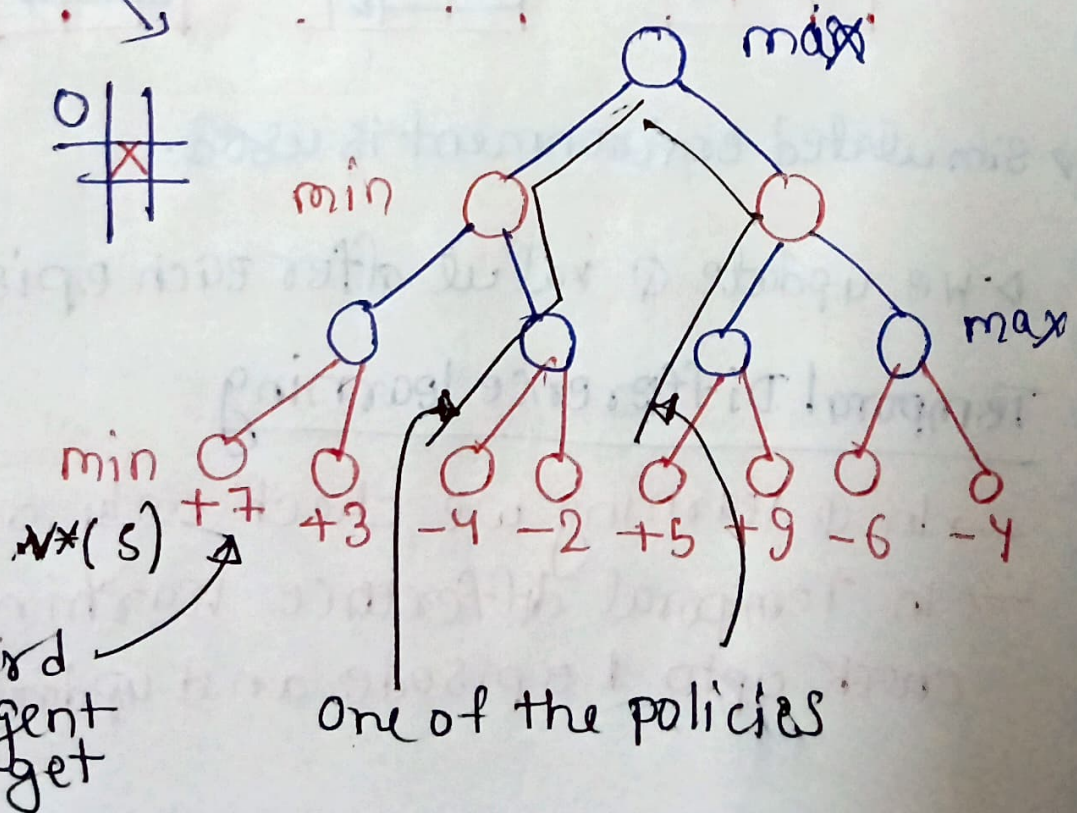


A-0, B-0 . B-1 B-1 B-1
B-1 B-1 B-1 B-0 } 8-episodes

Reinforcement learning in Game Playing

Daydreaming





- How to learn and survive in the environment?
- Q learning
- Markov Decision process (MDP
- Optimization
- Real world example
- Expected question: How to update Q-values given some grid?

Summary

Fundamentals About Neural Network

→ It is inspired by neurons in the human brain

→ Capacity of a model is ability to model varied functions

→ Generalization of a model is capacity of a model to perform well on previously unseen data

[Underfitting / Overfitting] → Low capacity: unable to obtain low error on training data

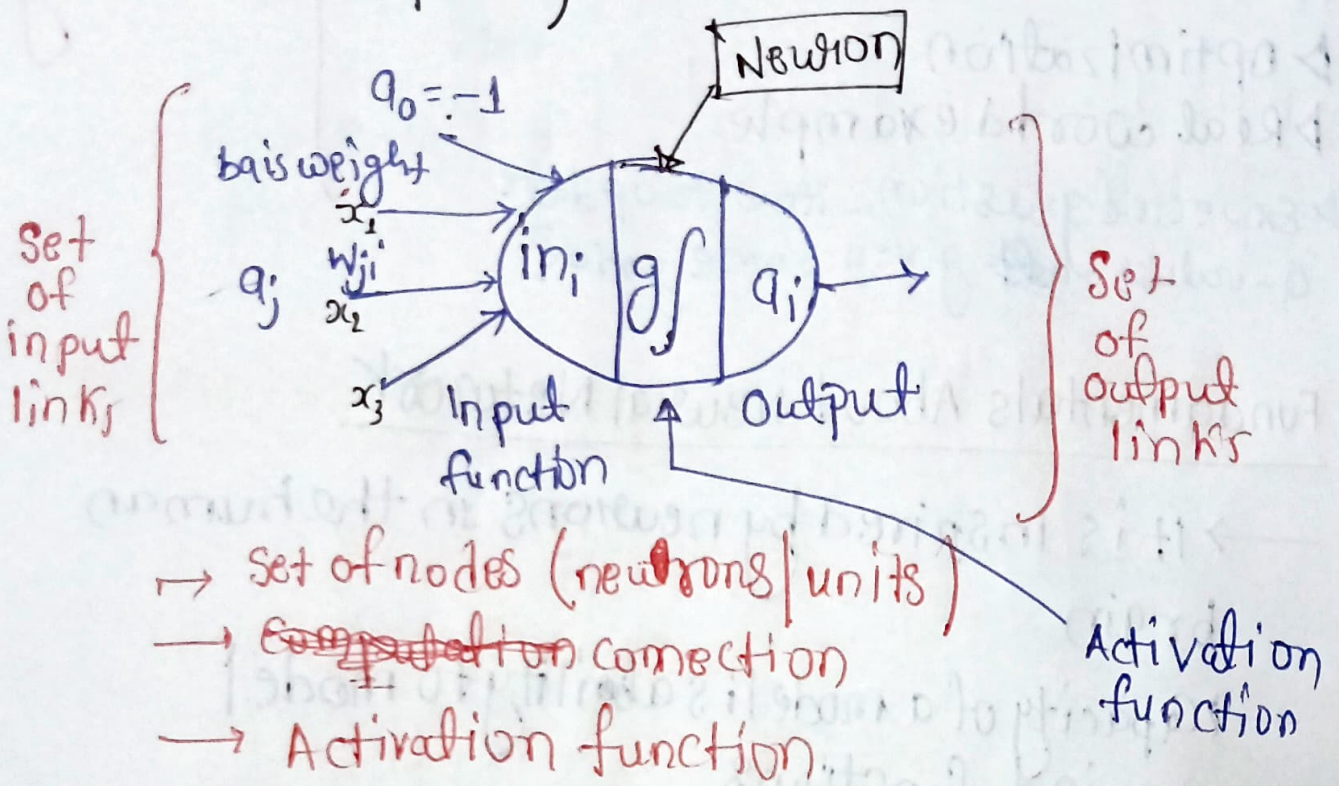
→ Learn very specific patterns

{Train Data / Test Data}

train error is very low but test error is quite high comparatively

Simple Neural Network

(Perceptron)



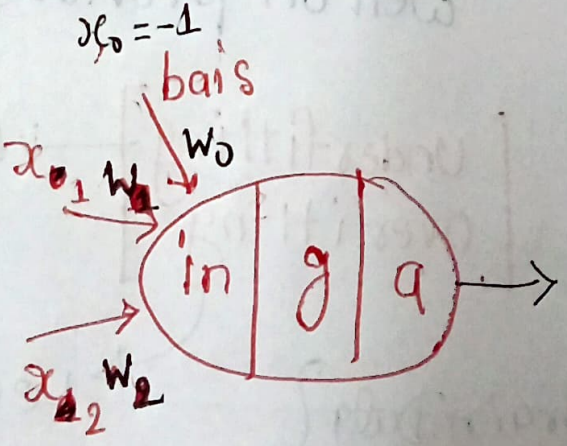
→ Set of nodes (neurons | units)

→ ~~Computation~~ connection

→ Activation function.

OR-Gate

x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	1



$in = \sum_{i=0}^{2} w_i x_i$ (weighted function for input value)

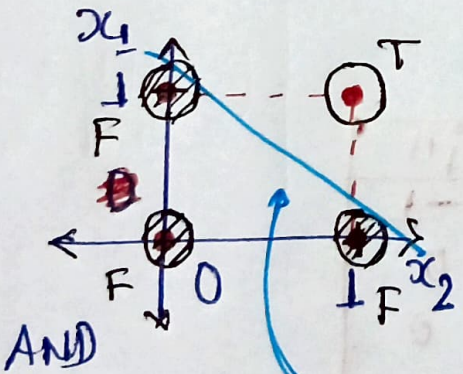
$a = g(in)$

$\hookrightarrow a = \begin{cases} 0 & \text{if } in \leq 0 \\ 1 & \text{if } in > 0 \end{cases}$

$in = -1 + w_1 x_1 + w_2 x_2 \quad (w_0 = 1, w_1 = 2, w_2 = 2)$

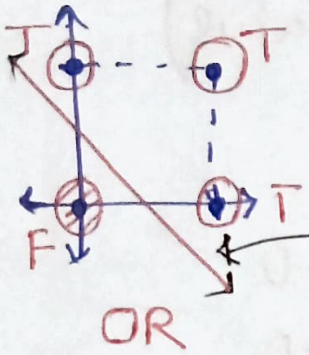
$= -1 + 2x_1 + 2x_2$

		a
$in = -1$	if $(x_1 = 0, x_2 = 0)$	0
1	if $(x_1 = 1, x_2 = 0)$	1
1	if $(x_1 = 0, x_1 = 1)$	1
3	if $(x_1 = 1, x_2 = 1)$	1



O : True
⊘ : False
} Levels

we want to draw some line so we can discriminate this two levels



can be represented using some linear function

AND - gate

x_1	x_2	output
0	0	0
0	1	0
1	0	0
1	1	1

$$a = \begin{cases} 0 \text{ if } in \leq 0 \\ 1 \text{ if } in > 1 \end{cases}$$

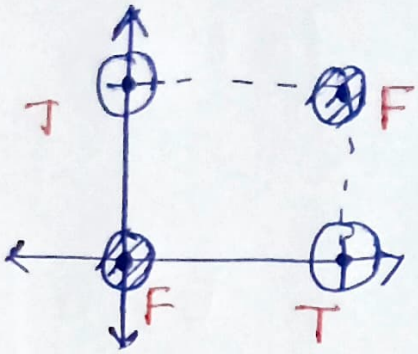
$w_0 = -1, w_1 = 1, w_2 = 1$

$in = -1 + x_1 + x_2$

x_1	x_0	in
0	0	-1
0	1	0
1	0	0
1	1	1

X-OR Gate

x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	0



No line can be drawn for distinguishing between 1 & 0 levels