In The Name Of ALLAH

Dive Into Deep Learning

Mohammad Hossein Amini
(mhamini@aut.ac.ir)
AAISS
Tehran Polytechnic

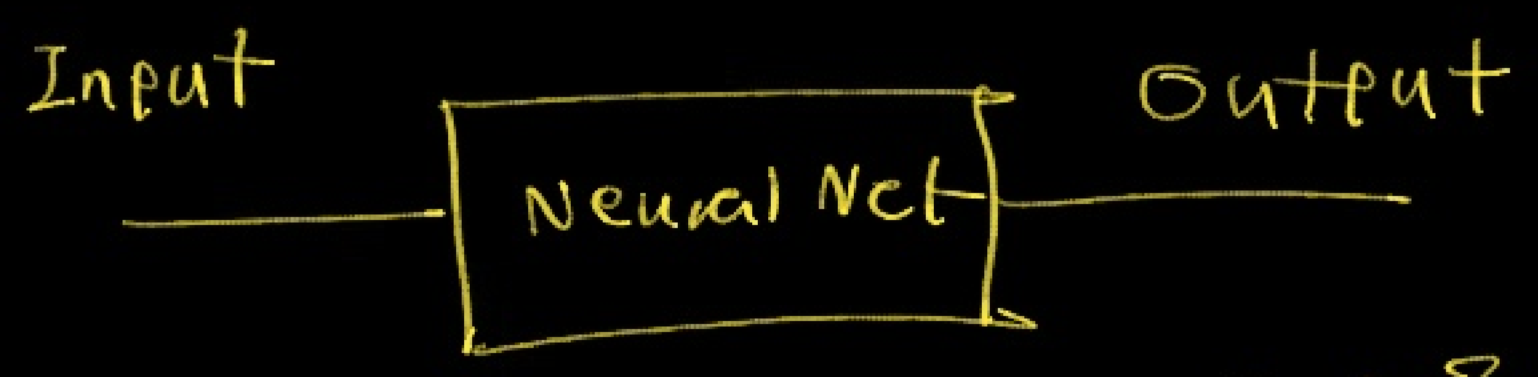
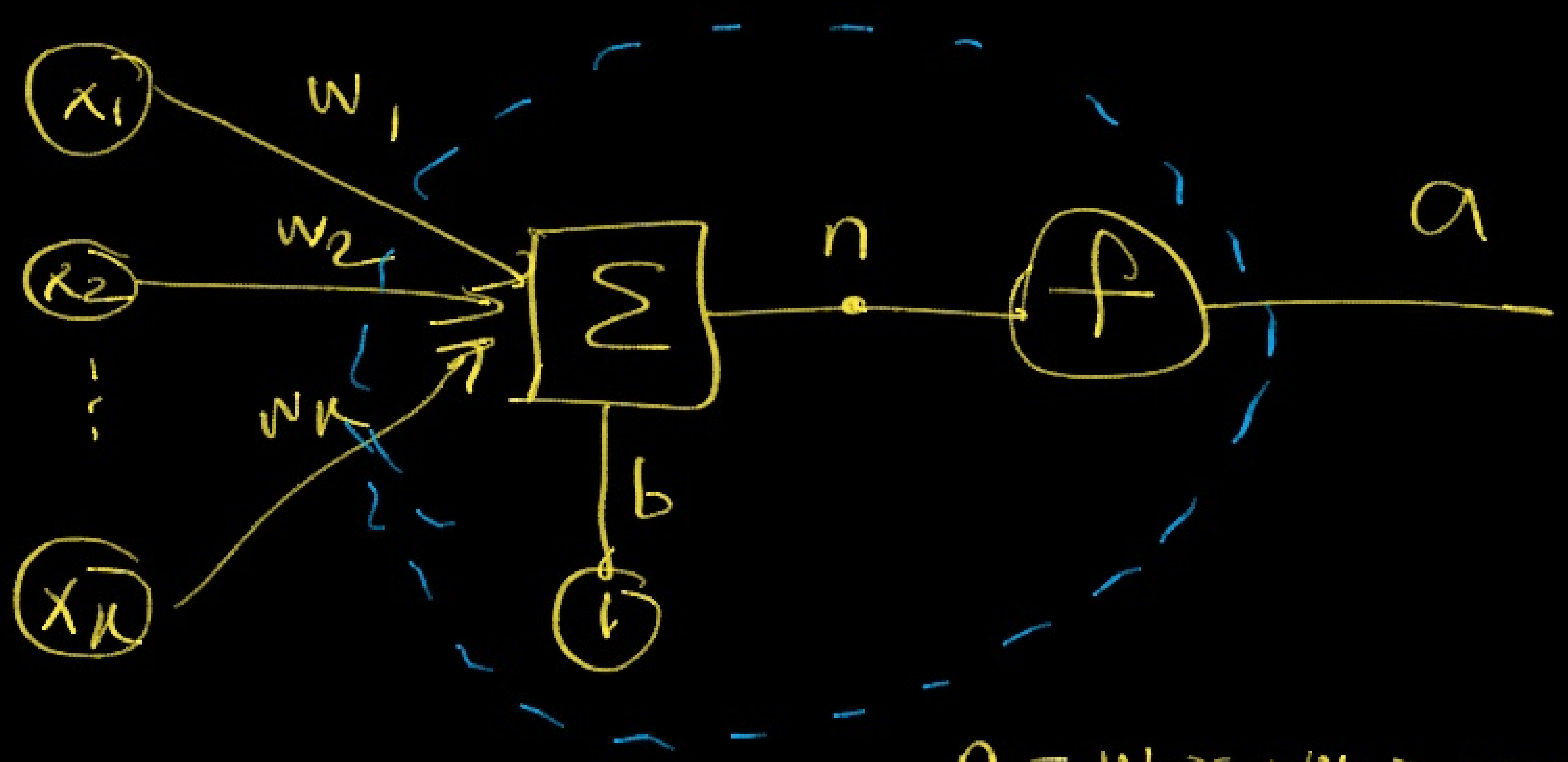


Image	Cat?
Speech	Text
Email	Spam?
Voltage	Fault?

Supervised Learning

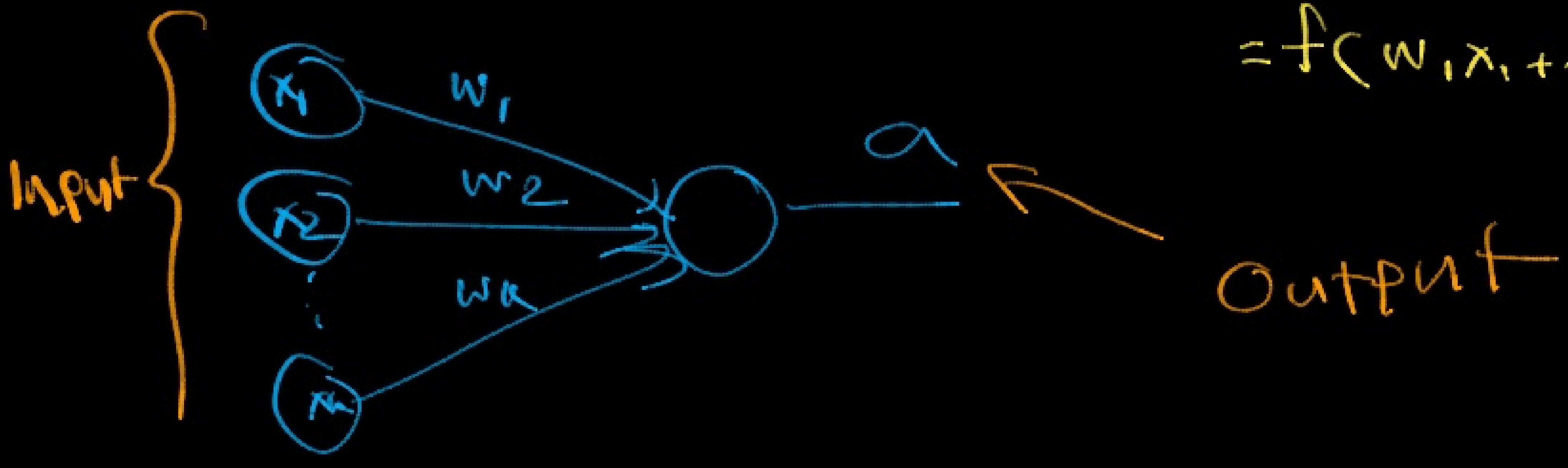
Neuron :



$$n = w_1x_1 + w_2x_2 + \dots + w_kx_k + b$$

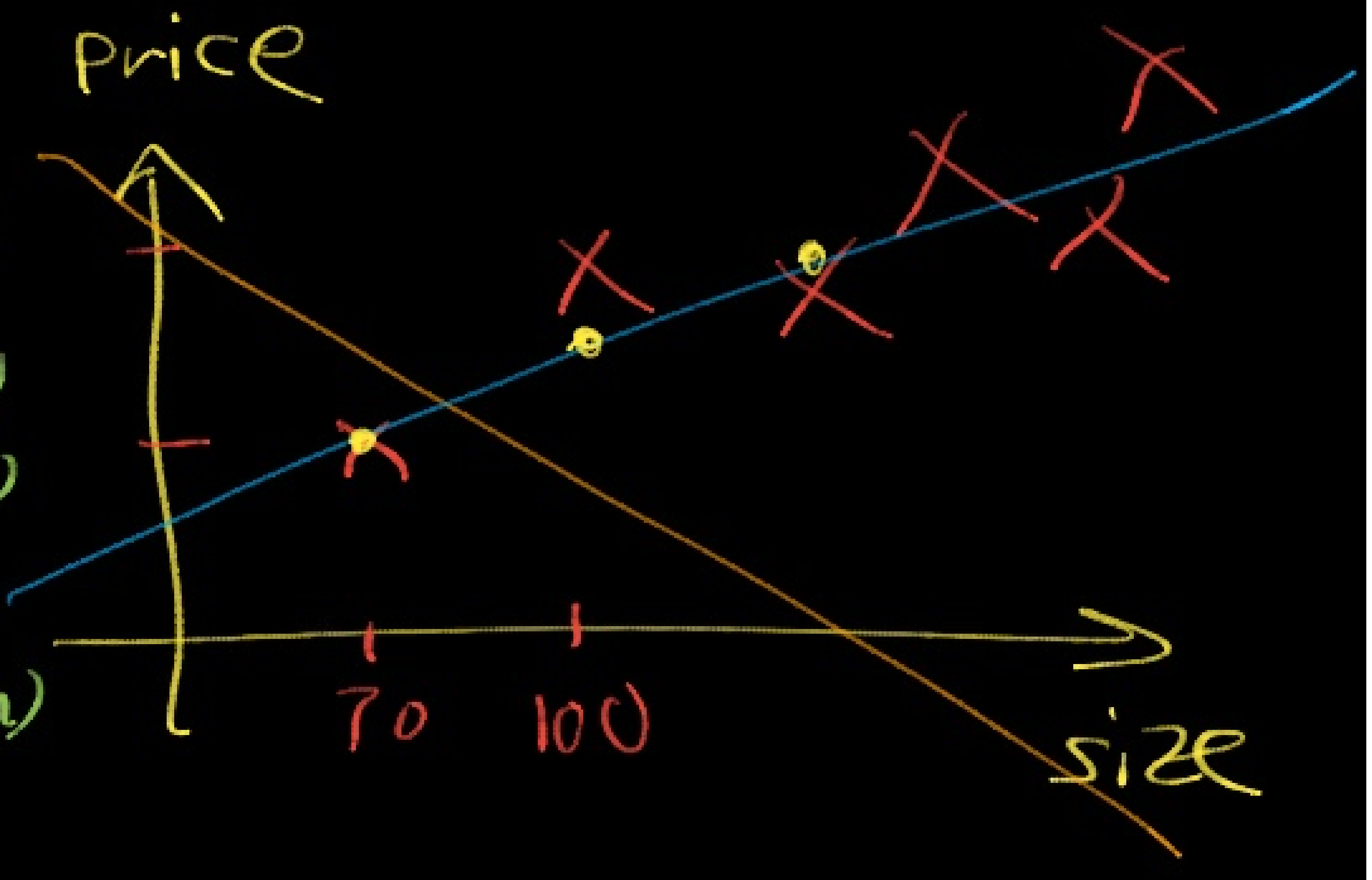
$$a = f(n)$$

$$= f(w_1x_1 + \dots + w_kx_k + b)$$

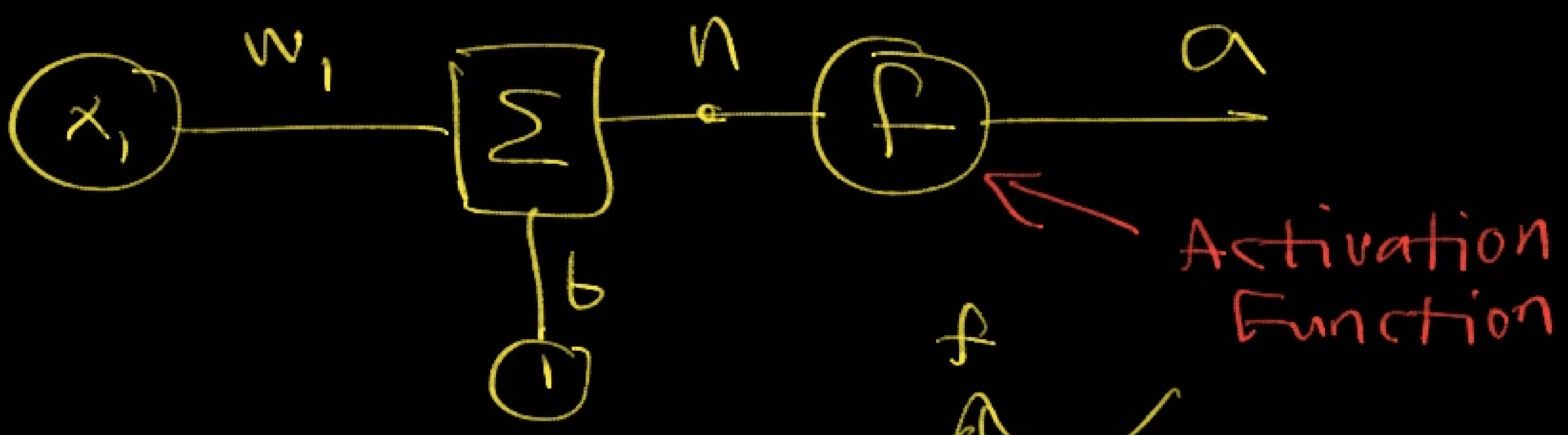


Housing Price Prediction:

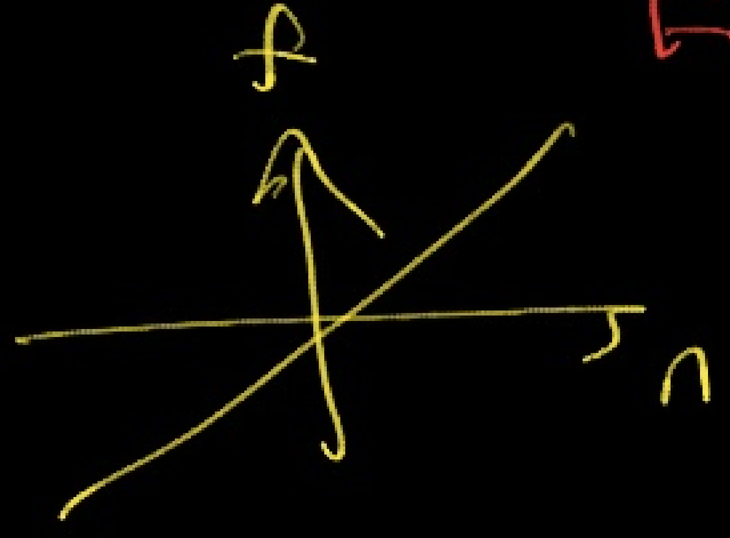
size	price
$x_1^{(1)} = 70$	$200 = y^{(1)}$
$x_1^{(2)} = 100$	$400 = y^{(2)}$
$x_1^{(3)} = 150$	$500 = y^{(3)}$
$\vdots$	$\vdots$
$x_1^{(m)} = \vdots$	$\vdots = y^{(m)}$
x_1	y



$$D = \{(x_1^{(1)}, y^{(1)}), \dots, (x_1^{(m)}, y^{(m)})\}$$



$$f(n) = n$$



$$n = w_1 x_1 + b$$

$$a = f(n) = n = w_1 x_1 + b$$

$$\underline{\underline{a = w_1 x_1 + b}}$$

Cost Function:

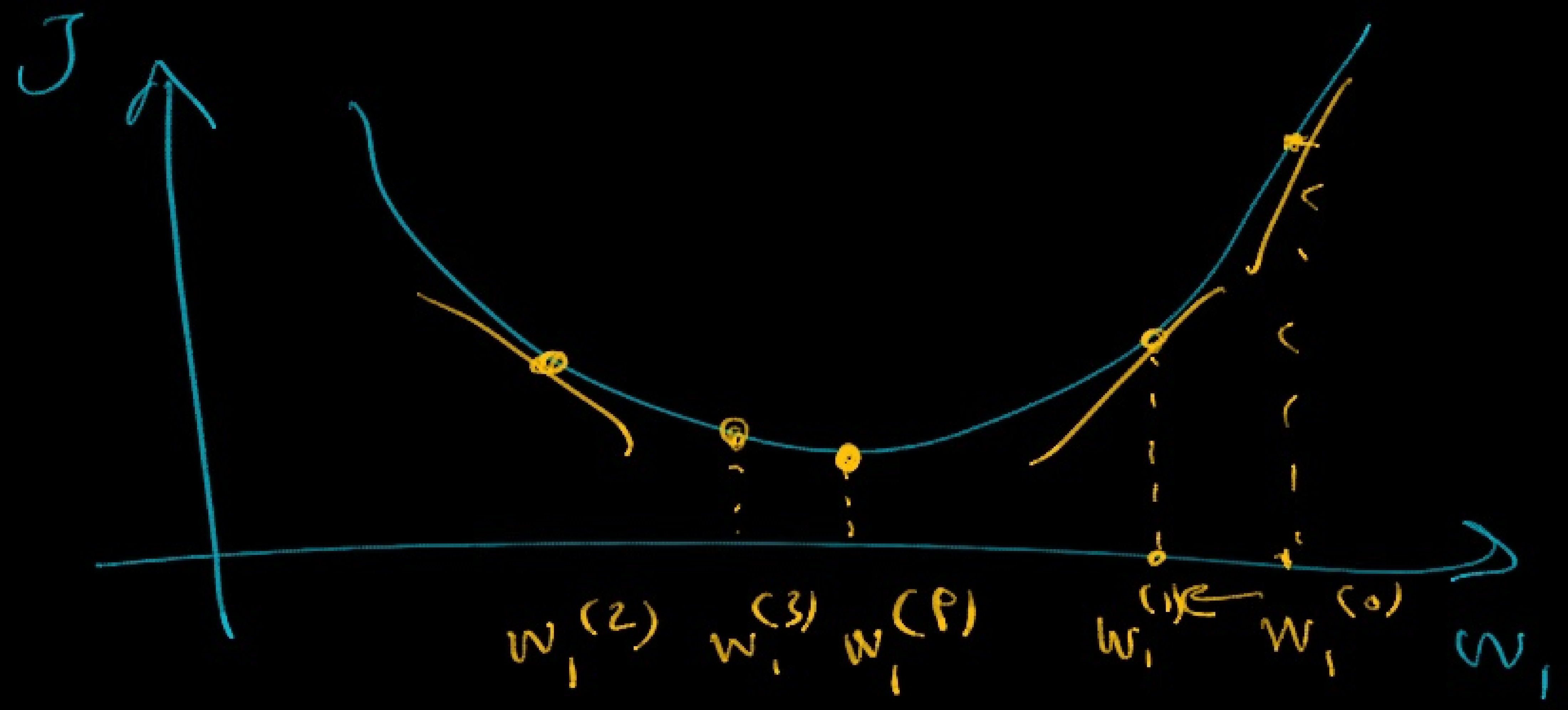
$$J = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - a^{(i)}\right)^2$$

Good Performance $\longrightarrow$ small J

Bad Performance $\longrightarrow$ Large J

Goal: $\min\limits_{w_1, b} \; J$

Gradient Descent:



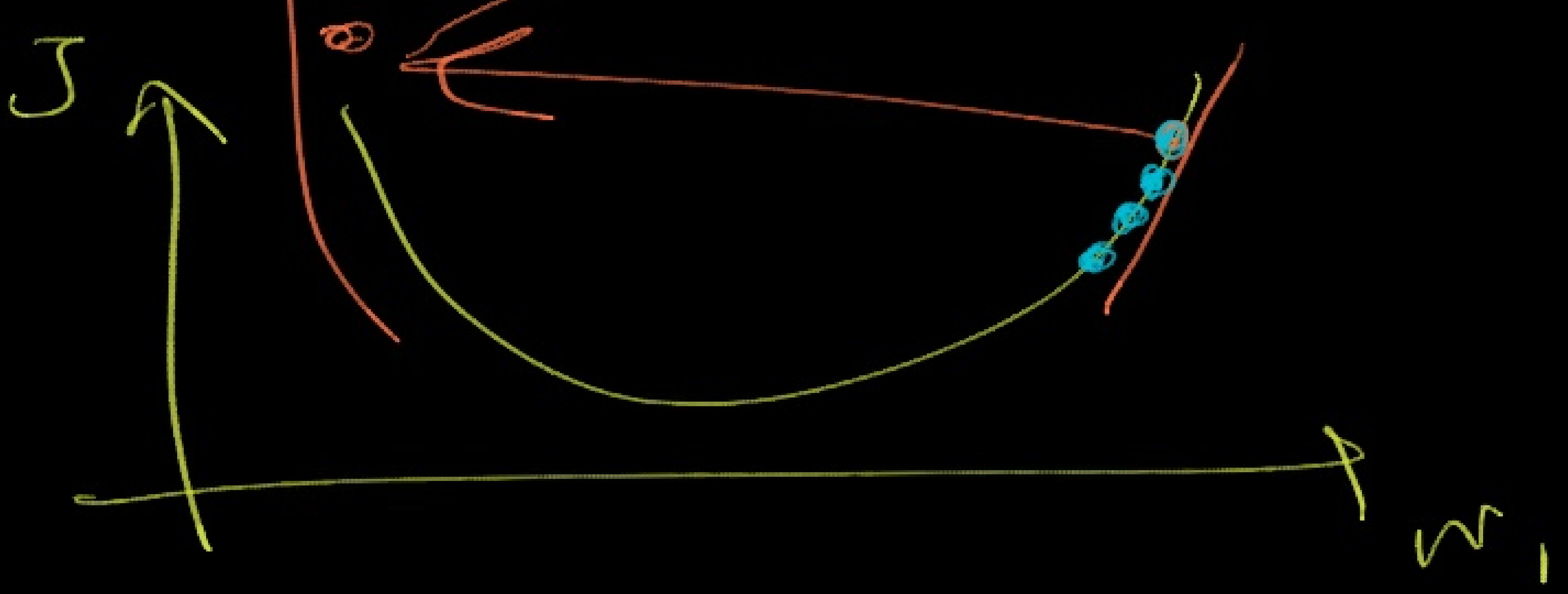
min J w.r.t w_1

In each iteration {

$$w_1^{(i)} = w_1^{(i-1)} - \underbrace{\alpha}_{\text{Learning Rate}} \frac{dJ}{dw_1}\Big|_{w_1^{(i-1)}}$$

}

Learning Rate:

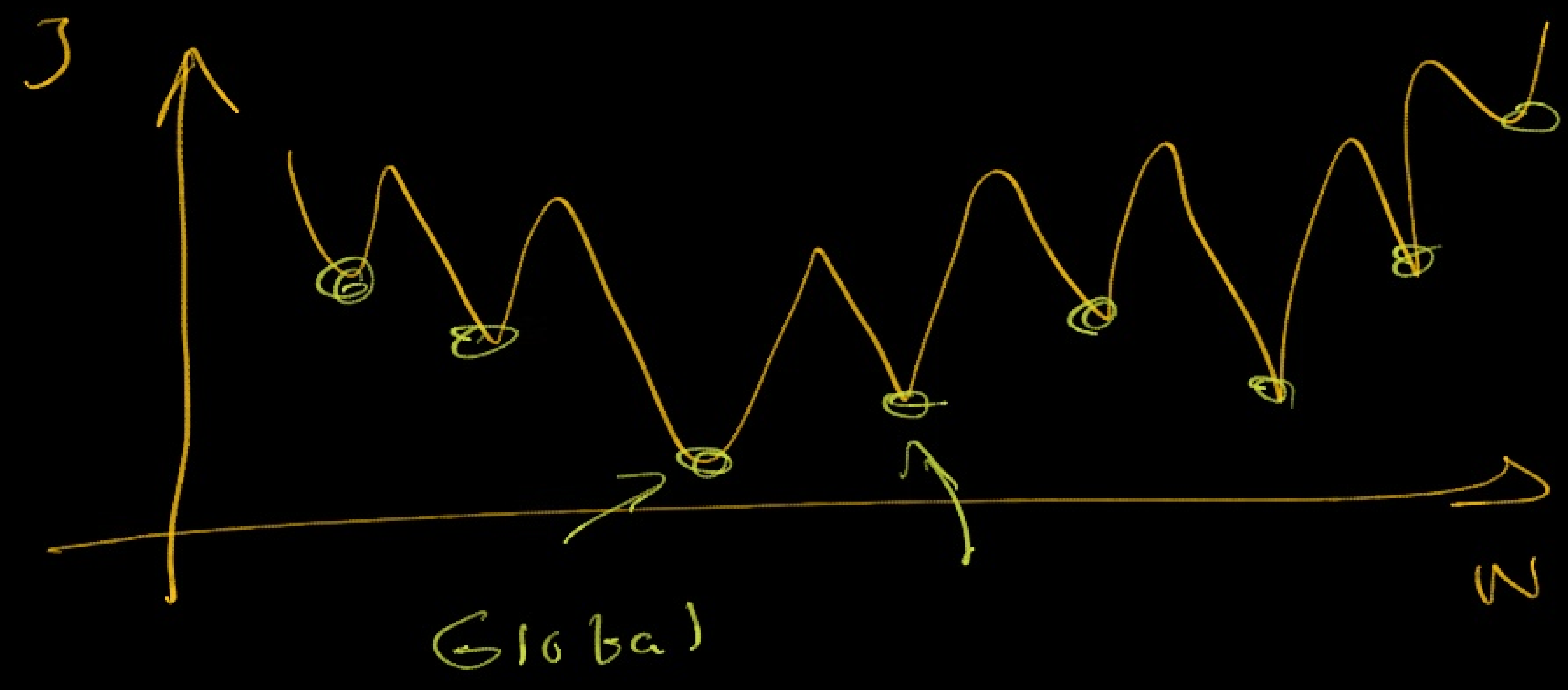


α { Large α : Divergence Problem
Small α : Slow Convergence

$$\min_{w_1,\dots,w_k} J$$

$$w_i^{(n)} = w_i^{(n-1)} - \alpha \left.\frac{\partial J}{\partial w_i}\right|_{w_i^{(n-1)}}$$

$$i = 1,\dots,k$$



$$J = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - a^{(i)}\right)^2$$

$$\min_{w_1,b} J \qquad a^{(i)} = w_1 x_1^{(i)} + b$$

$$w_1 := w_1 - \alpha\frac{\partial J}{\partial w_1}$$

$$b := b - \alpha\frac{\partial J}{\partial b}$$

$$J = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - a^{(i)}\right)^2$$

$$\frac{\partial J}{\partial w_1} = \sum_{i=1}^{m}\frac{\partial}{\partial a^{(i)}}\left(\frac{1}{2m}\sum\left(y^{(i)} - a^{(i)}\right)^2\right)\frac{\partial a^{(i)}}{\partial w_1}$$

$$= -\frac{2}{2m}\sum_{i=1}^{m}\left(y^{(i)} - a^{(i)}\right)x_1^{(i)}$$

$$\frac{\partial J}{\partial w_1} = \frac{-1}{m} \sum_{i=1}^{m} (y^{(i)} - a^{(i)}) x_1^{(i)}$$

$$\frac{\partial J}{\partial b} = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - a^{(i)})$$

$$\begin{cases} w_1 := w_1 + \frac{\alpha}{m} \sum_{i=1}^{m} (y^{(i)} - a^{(i)}) x_1^{(i)} \\ b := b + \frac{\alpha}{m} \sum_{i=1}^{m} (y^{(i)} - a^{(i)}) \end{cases}$$

1. Neural Network Structure

2. Define a Cost Function

3. Minimize the Cost Function

- $$y^{(i)} = a(x^{(i)}) + \underline{n^{(i)}}$$

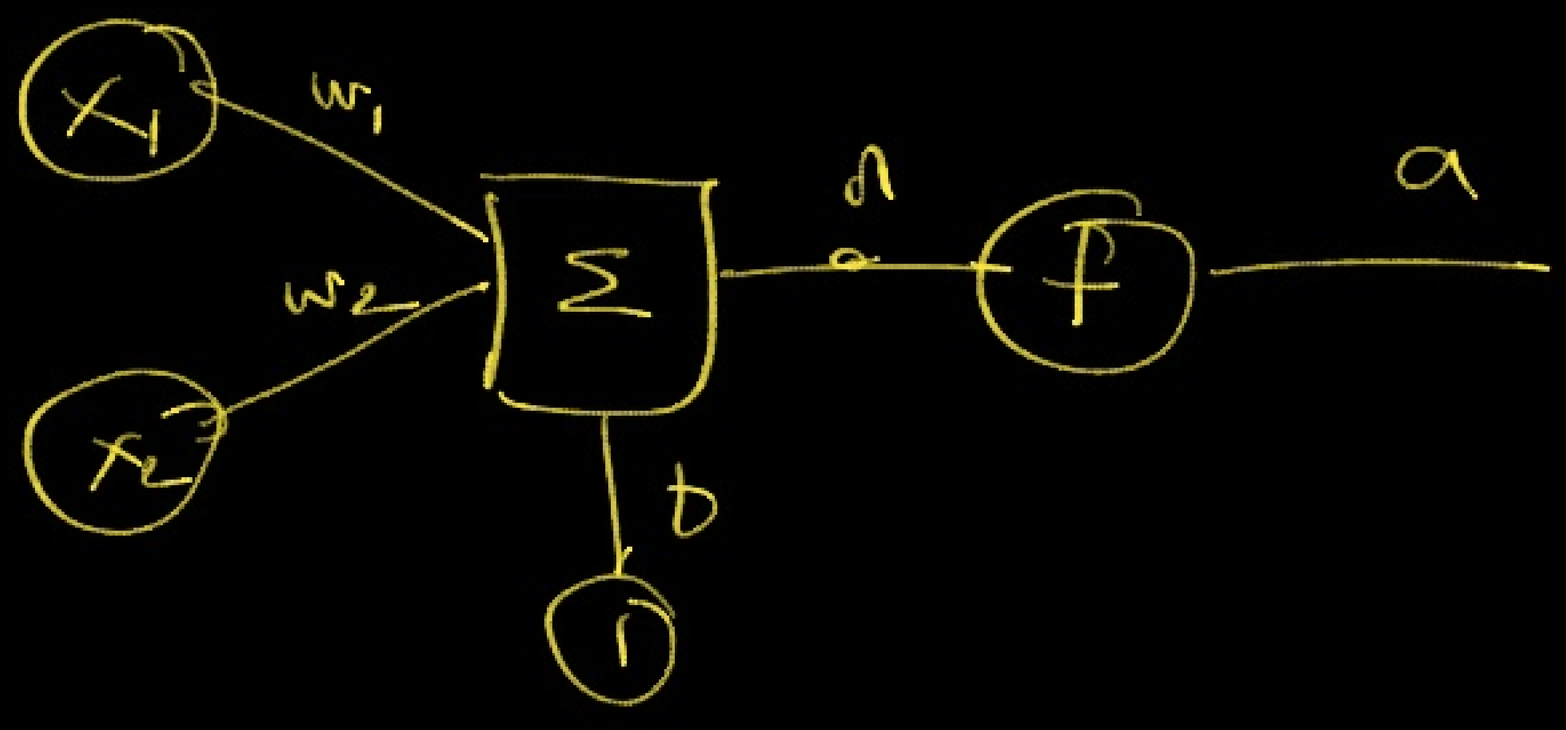
WGN

- "AND" Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

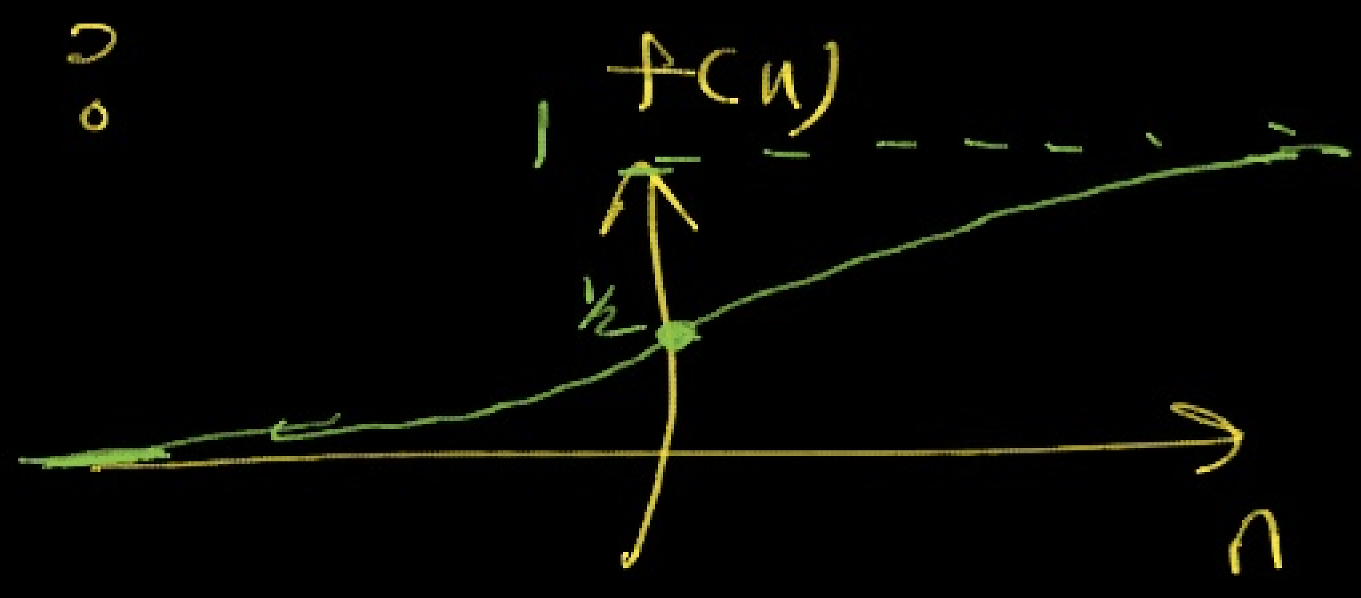
- "AND" Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



Sigmoid Function :

$$f(n) = \frac{1}{1+e^{-n}}$$



$\begin{cases} \text{Regression} \\ \text{Classification} \end{cases}$

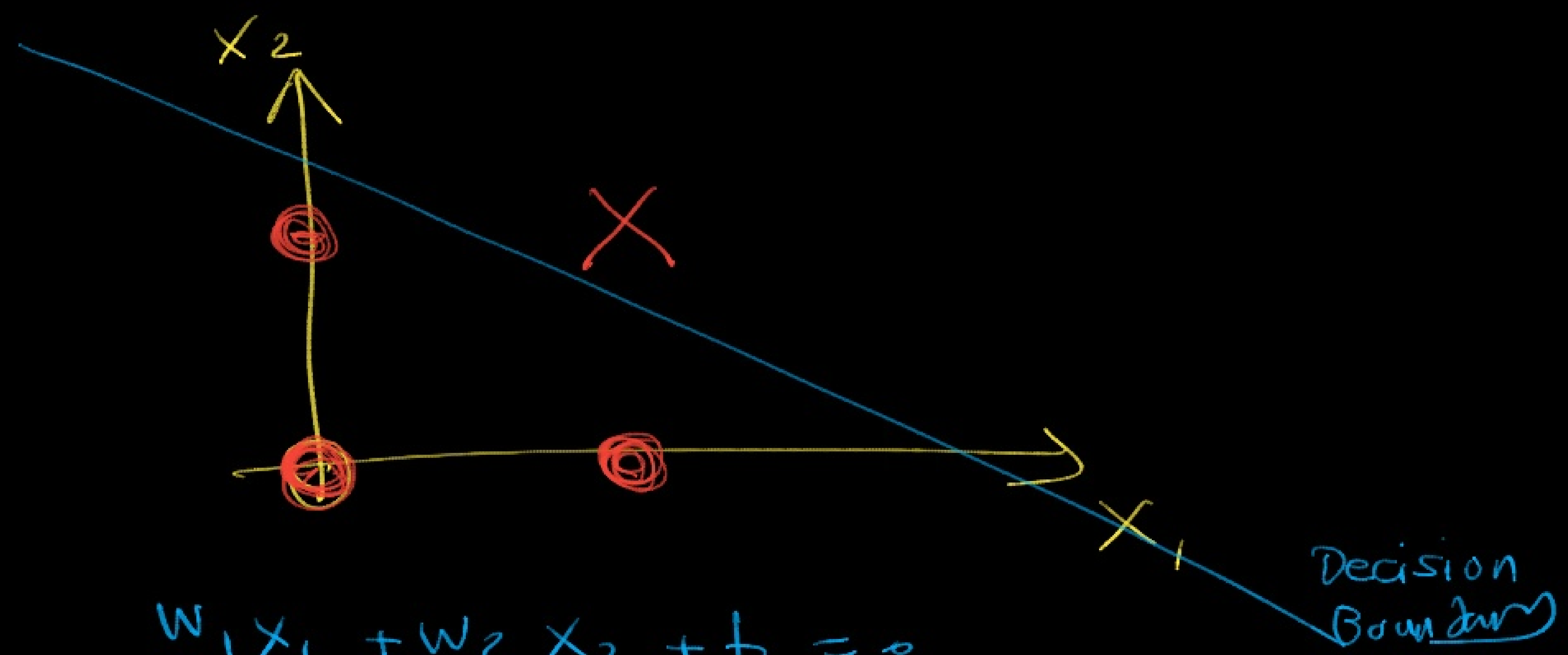
$$n = w_1 x_1 + w_2 x_2 + b$$

$$a = f(n)$$
$$= \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + b)}}$$

$$\begin{cases} a > 0.5 \rightarrow 1 \\ a \leq 0.5 \rightarrow 0 \end{cases}$$

$$a = 0.5 \Rightarrow n = 0$$

$\underline{\underline{w_1 x_1 + w_2 x_2 + b = 0}} \leftarrow$ Decision Boundary



$$w_1x_1 + w_2x_2 + b = 0$$

$$J = -\sum_{i=1}^{m} y^{(i)} \log a^{(i)} - \sum_{i=1}^{m} (1-y^{(i)}) \log (1-a^{(i)})$$

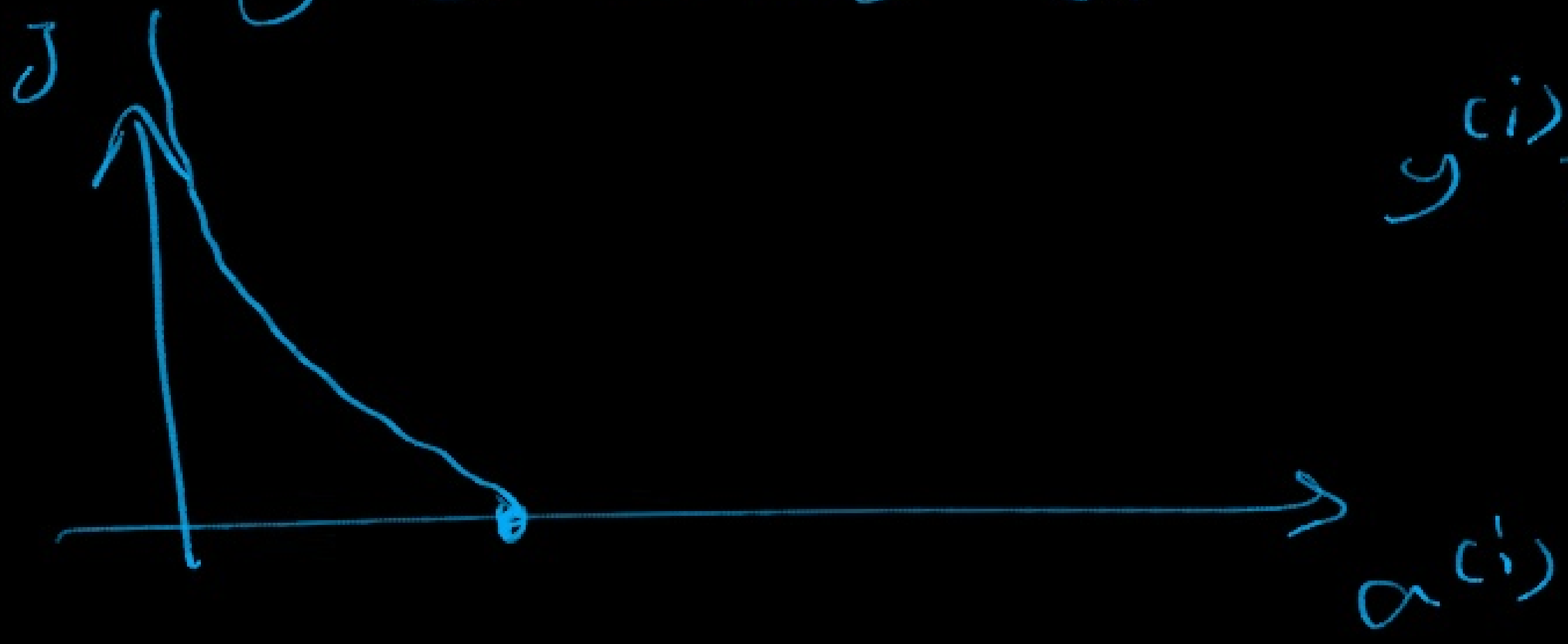
$$J' = -y^{(i)} \log a^{(i)} - (1-y^{(i)}) \log (1-a^{(i)})$$

If $y^{(i)} = 0$:

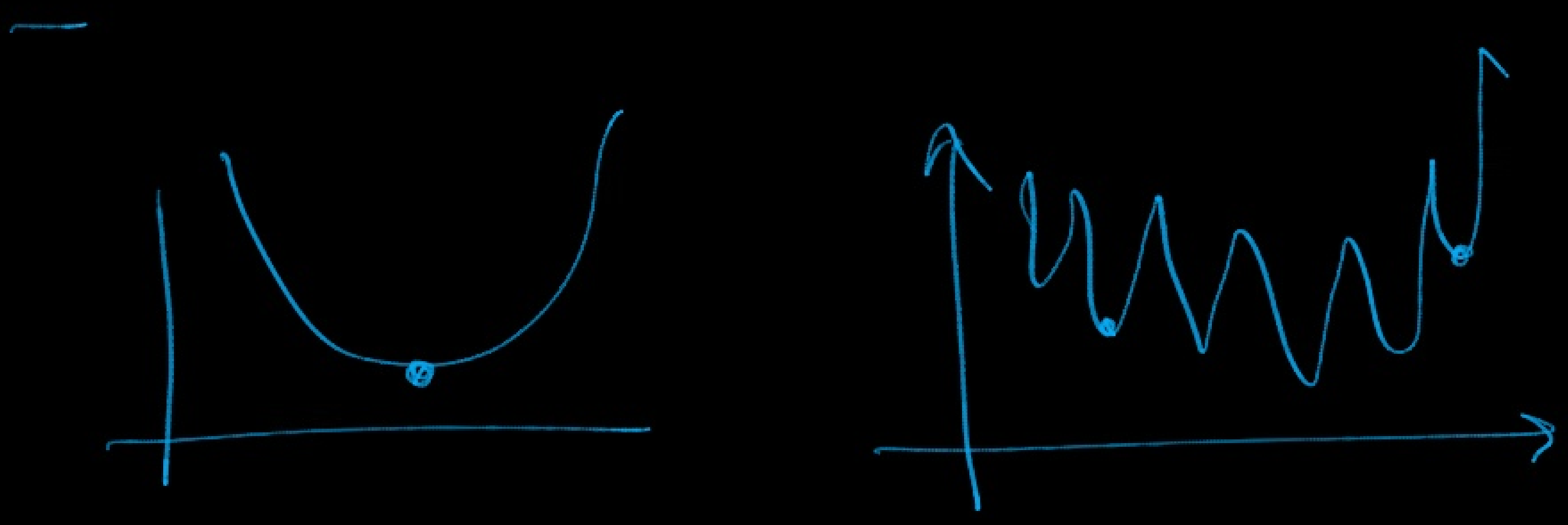
$$J' = -\log (1-a^{(i)})$$

If $y^{(i)} = 1$:

$$J' = -\log a^{(i)}$$



$y^{(i)} = 1$



Convex

Convex

Mean-Squared Error (MSE)

$$J = \frac{1}{2M} \sum_{i=1}^{m} \left(y^{(i)} - a^{(i)}\right)^2$$

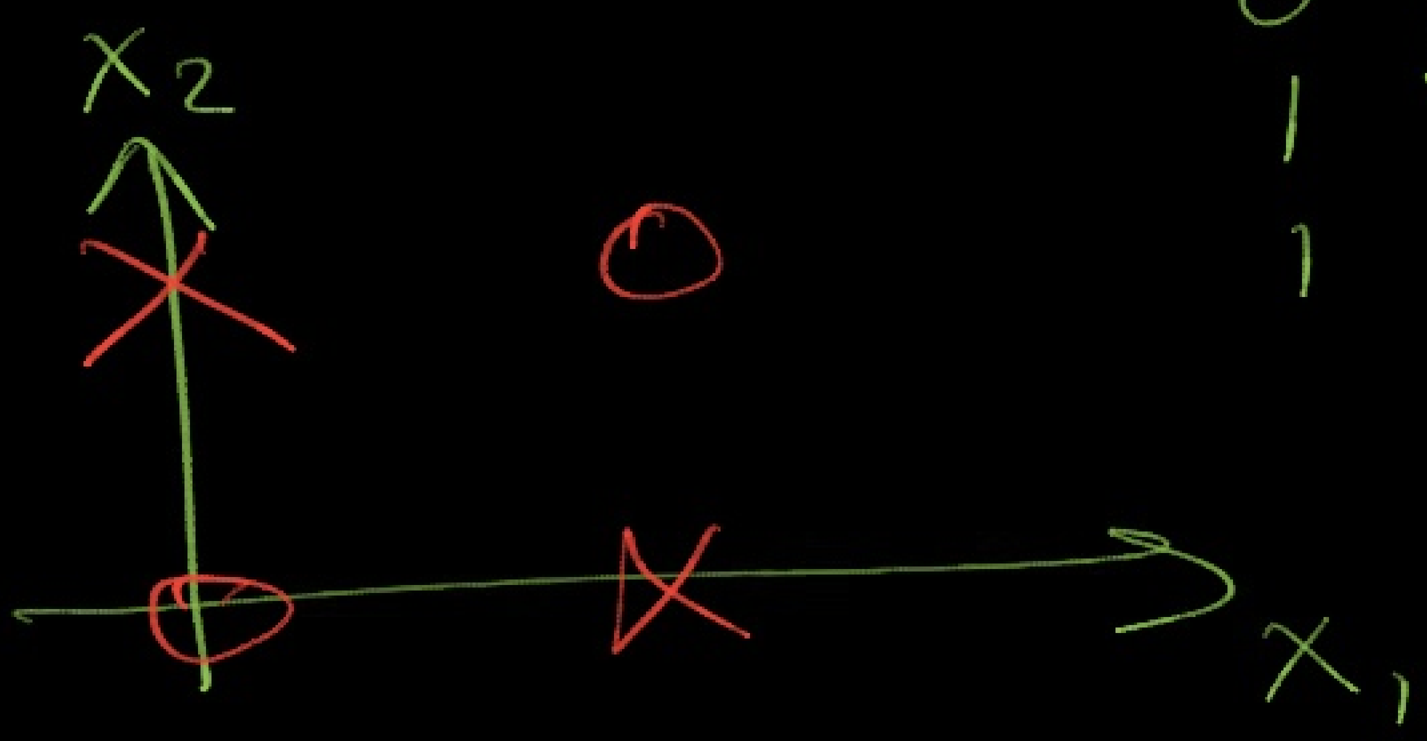
Binary Cross-Entropy :

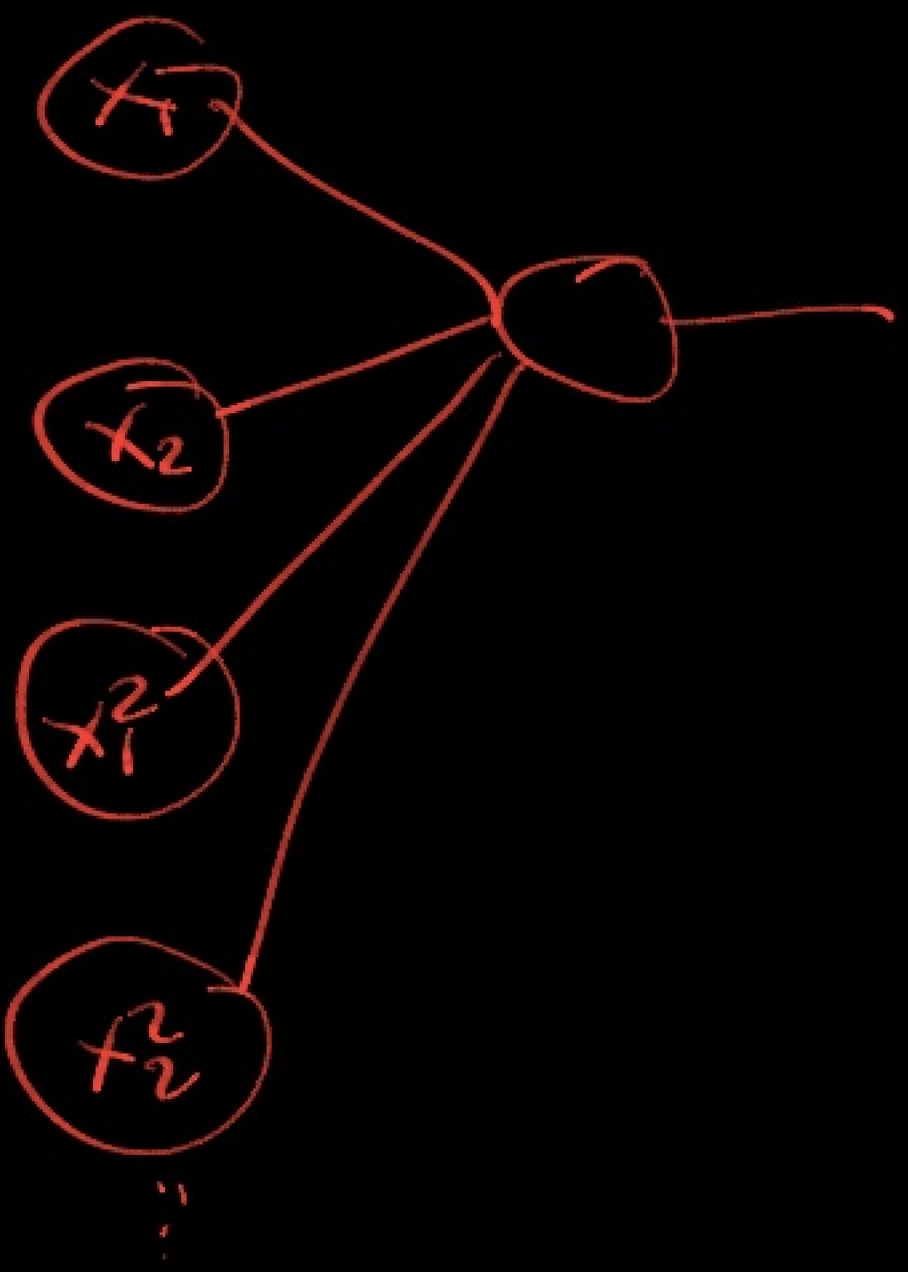
$$J = -\left[\sum_{i=1}^{m} y^{(i)} \log a^{(i)} + \sum_{i=1}^{m} (1-y^{(i)}) \log(1-a^{(i)}) \right]$$

- Batch Gradient Descent
- Mini-Batch Gradient Descent
- Stochastic Gradient Descent (SGD)

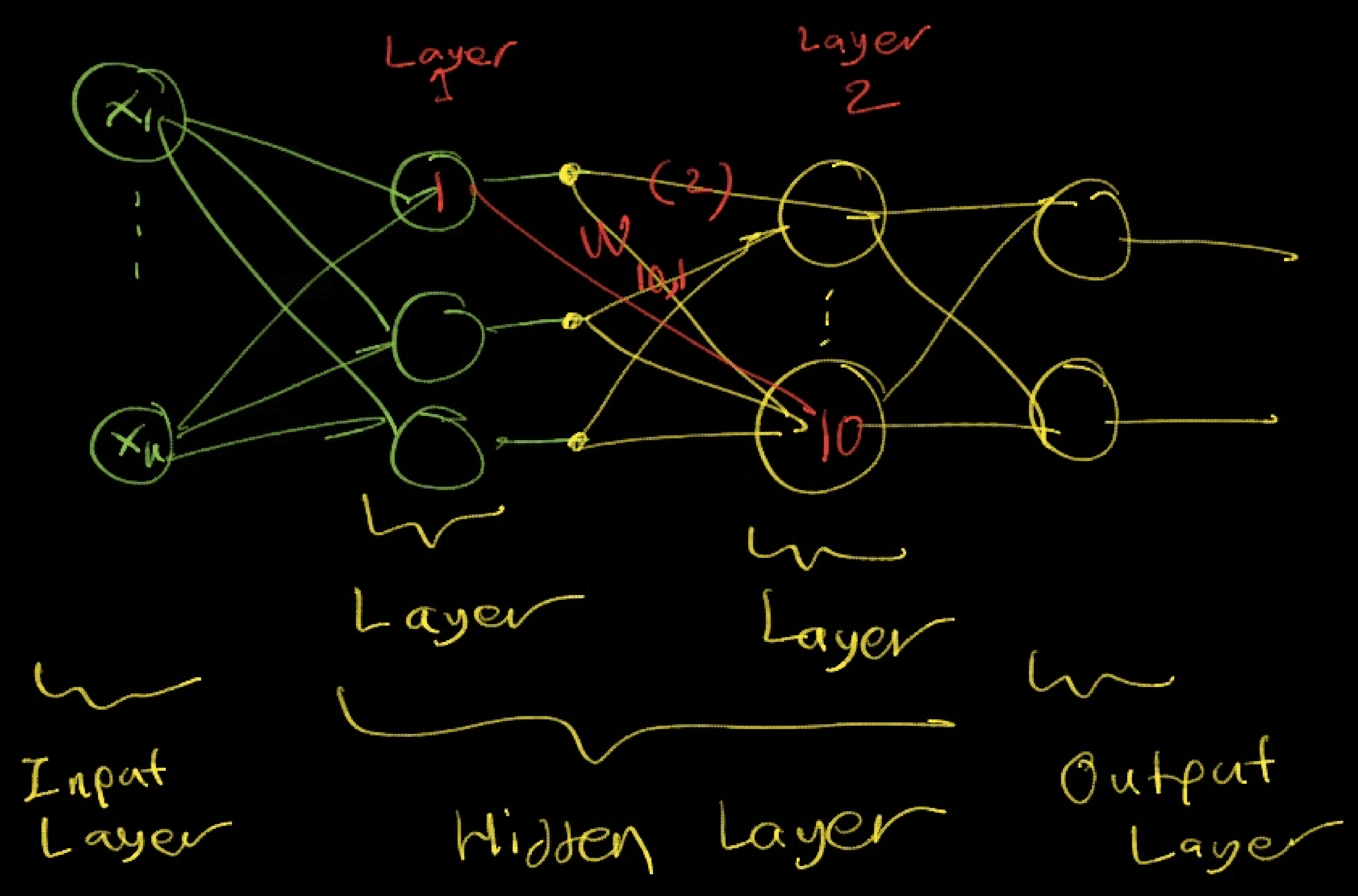
"XOR"

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0





Layer :



- $W_{ij}^{(k)}$: weight from neuron j of layer (k-1) into neuron i of layer (k)

$$W^{(k)} = [W_{ij}^{(k)}]$$

$\underline{p}^{(k)} \leftarrow$ Input vector

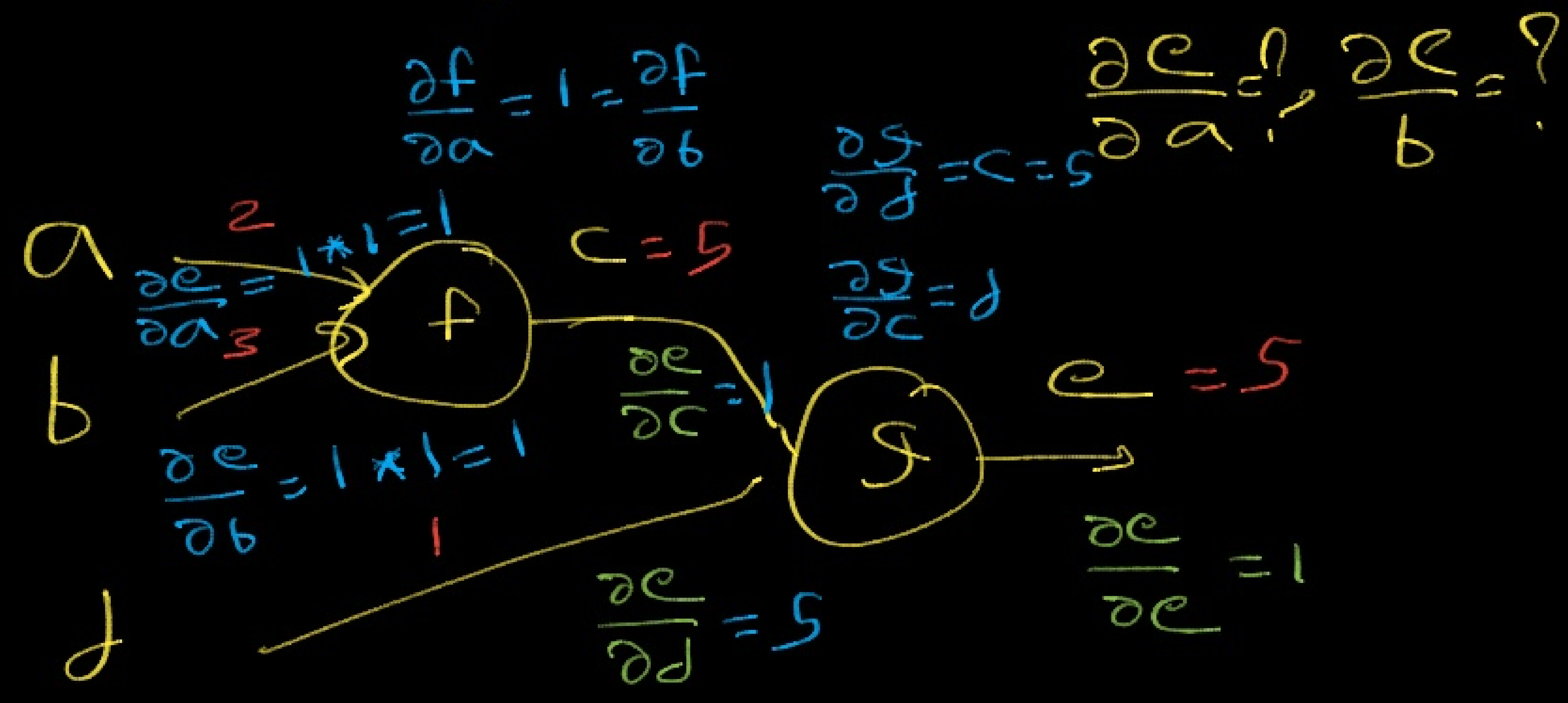
$$n^{(k)} = W^{(k)} \underline{p}^{(k)} + \underline{b}^{(k)}$$

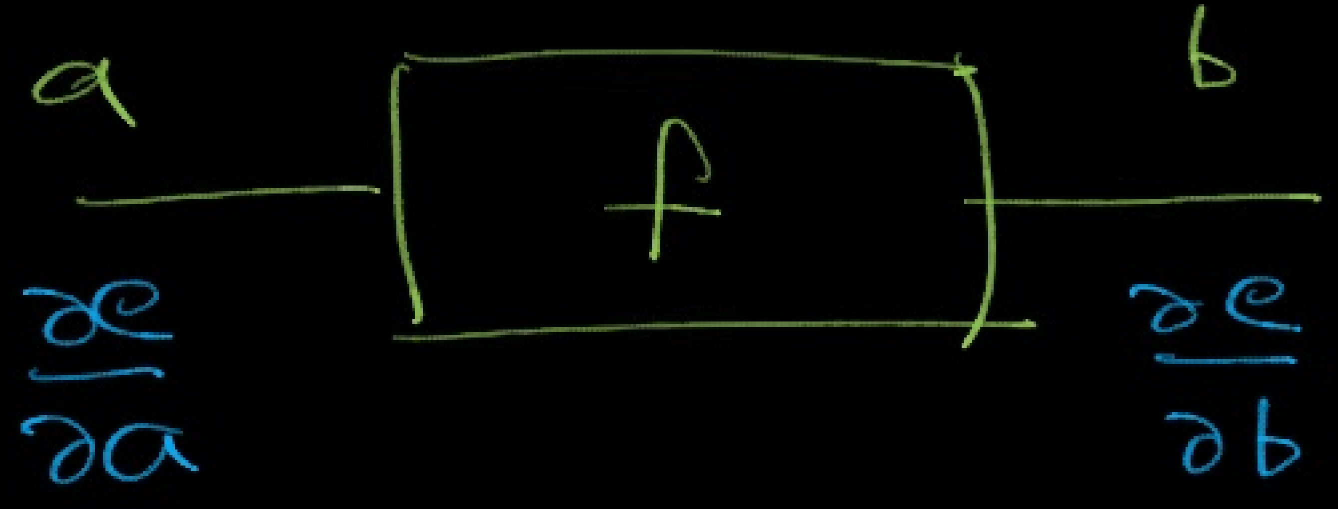
$$W_{ij}^{(k)} := W_{ij}^{(k)} - \alpha \frac{\partial J}{\partial W_{ij}^{(k)}}$$

Computational Graph :

$$c = f(a,b) = a+b$$

$$e = g(c,d) = c * d$$



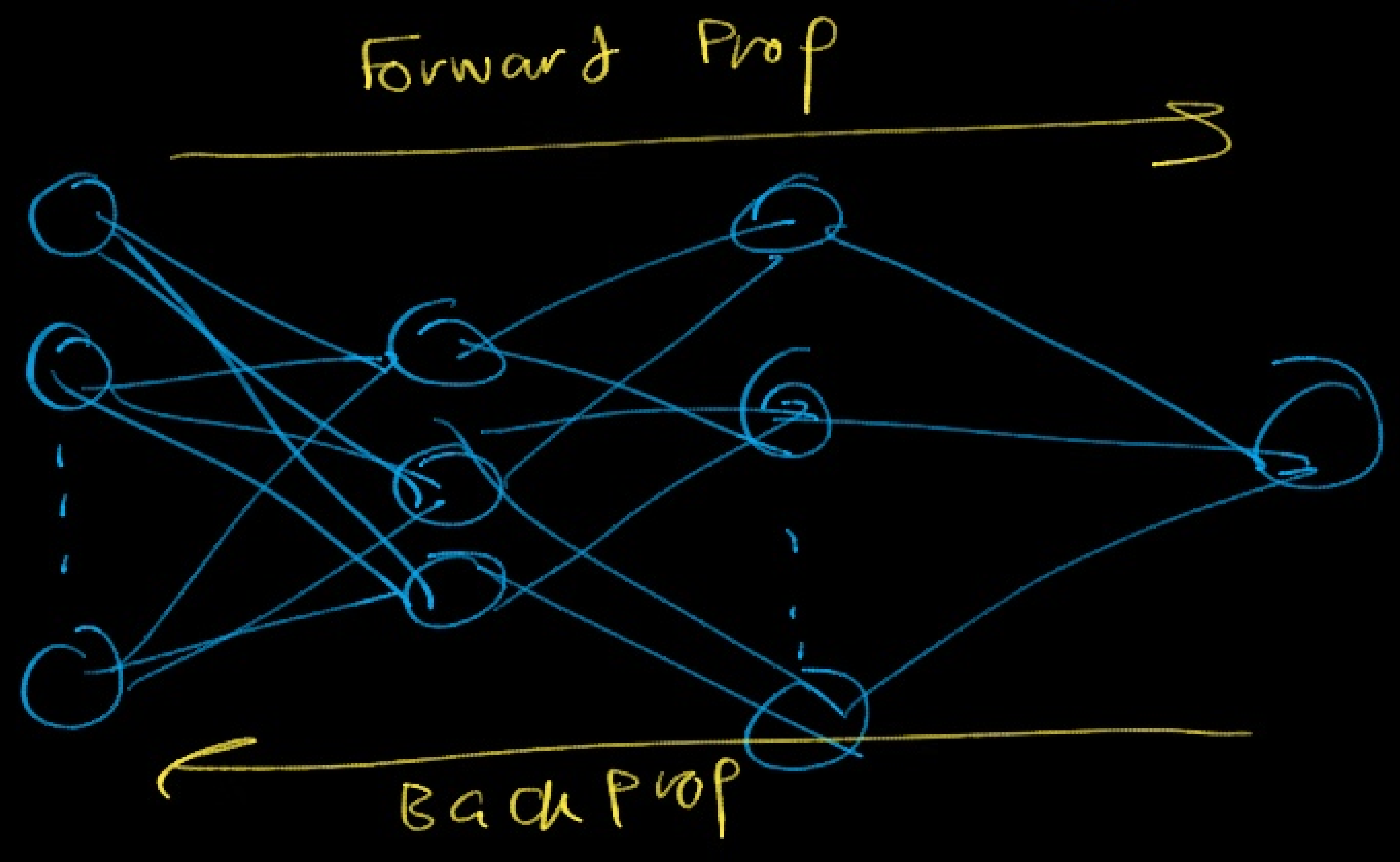


$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial b} \cdot f'$$

$$= \frac{\partial e}{\partial b} \cdot \frac{\partial b}{\partial a}$$

$$= \frac{\partial e}{\partial a}$$

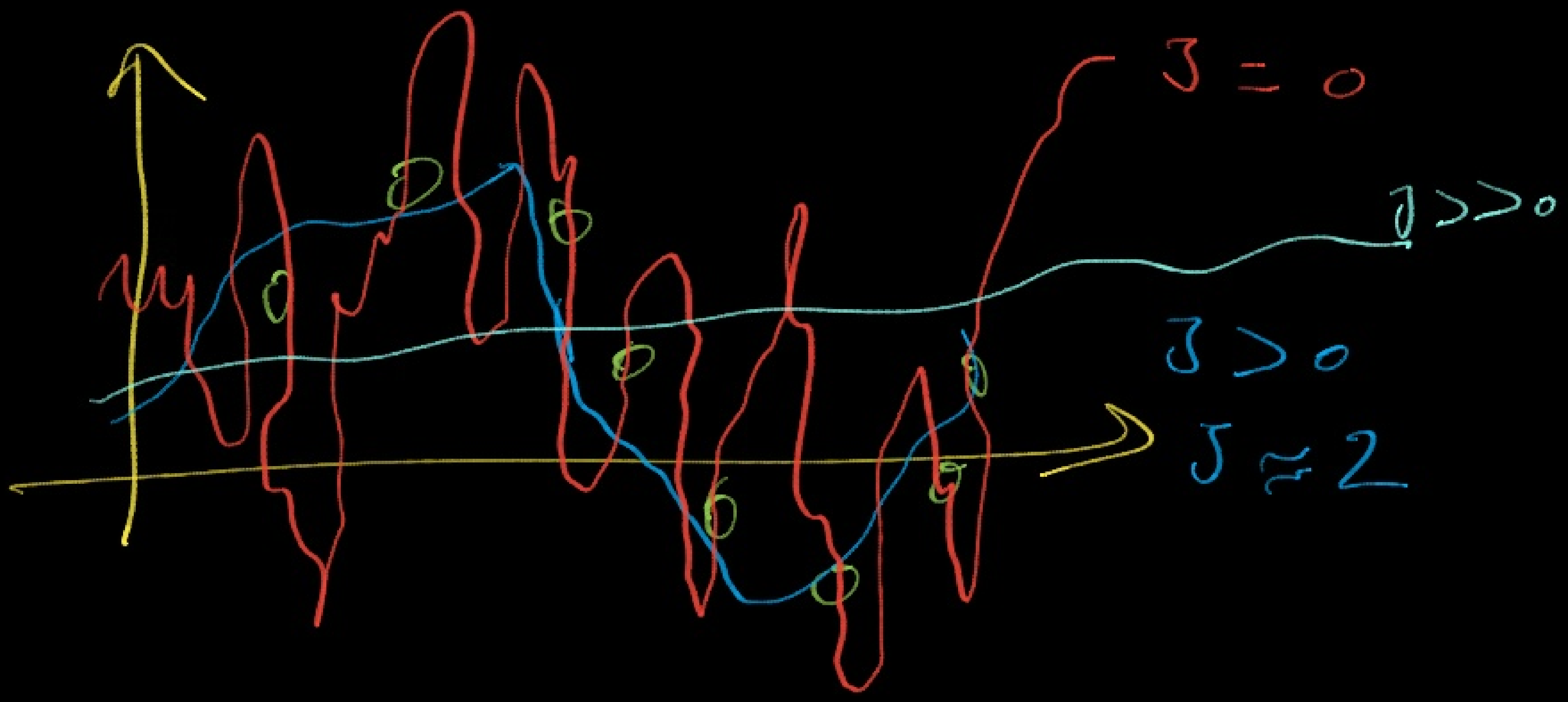
Backpropagation



Universal Approximation Theorem:

1 Hidden Layer - sigmoid

1 Output Layer - Linear



Overfit

Generalized well

Underfit

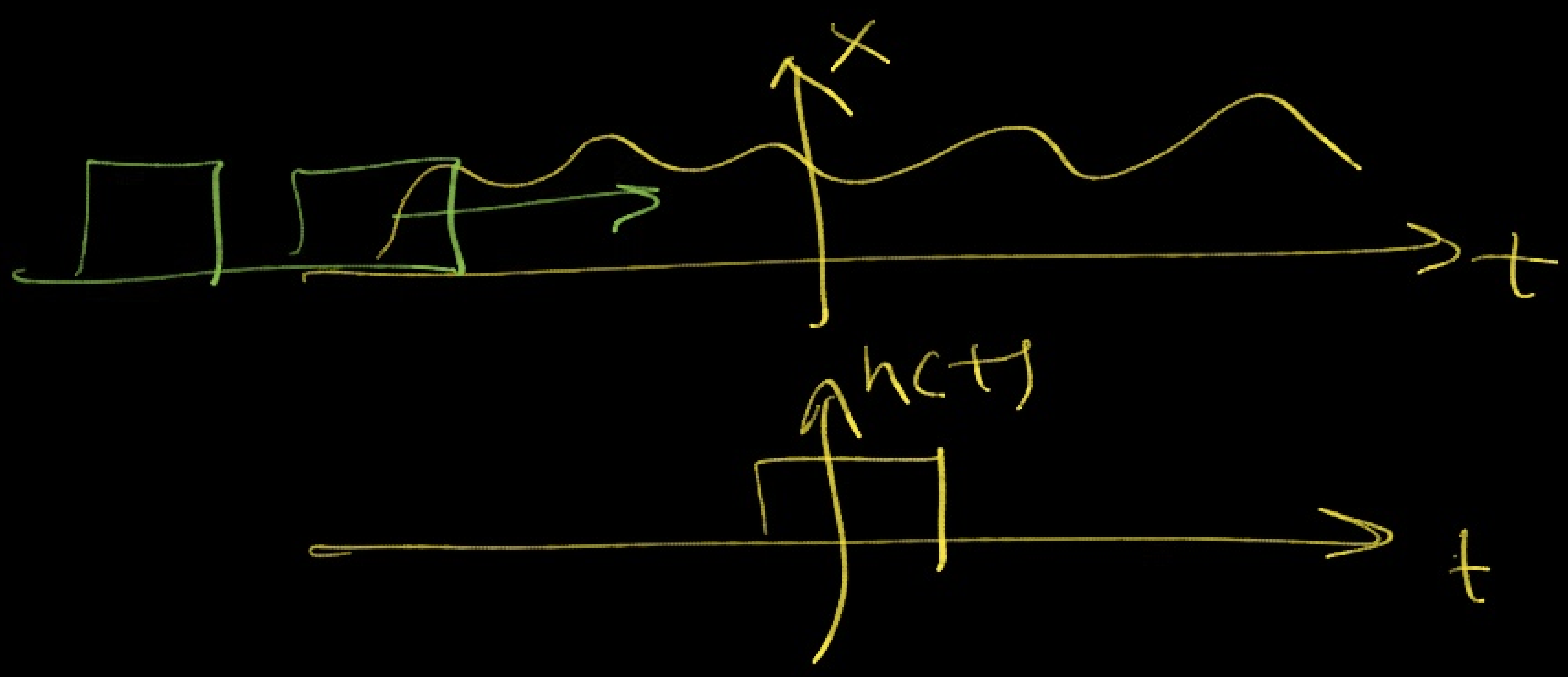
Convolutional Neural Nets :

CNN

$$x(t) \longrightarrow \boxed{LTI}_{\,h(t)} \longrightarrow y(t)$$

$$y(t) = x(t) * h(t)$$

$$= \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau$$



~~Image~~ (4x4)

	0	0	0	0
0	0	10	0	0
	1	0	0	255
	5	100	0	100
	0	28	6	0

4x4

h

1	0	-1
1	0	-1
1	0	-1

6	-345
6	-327

y

2x2

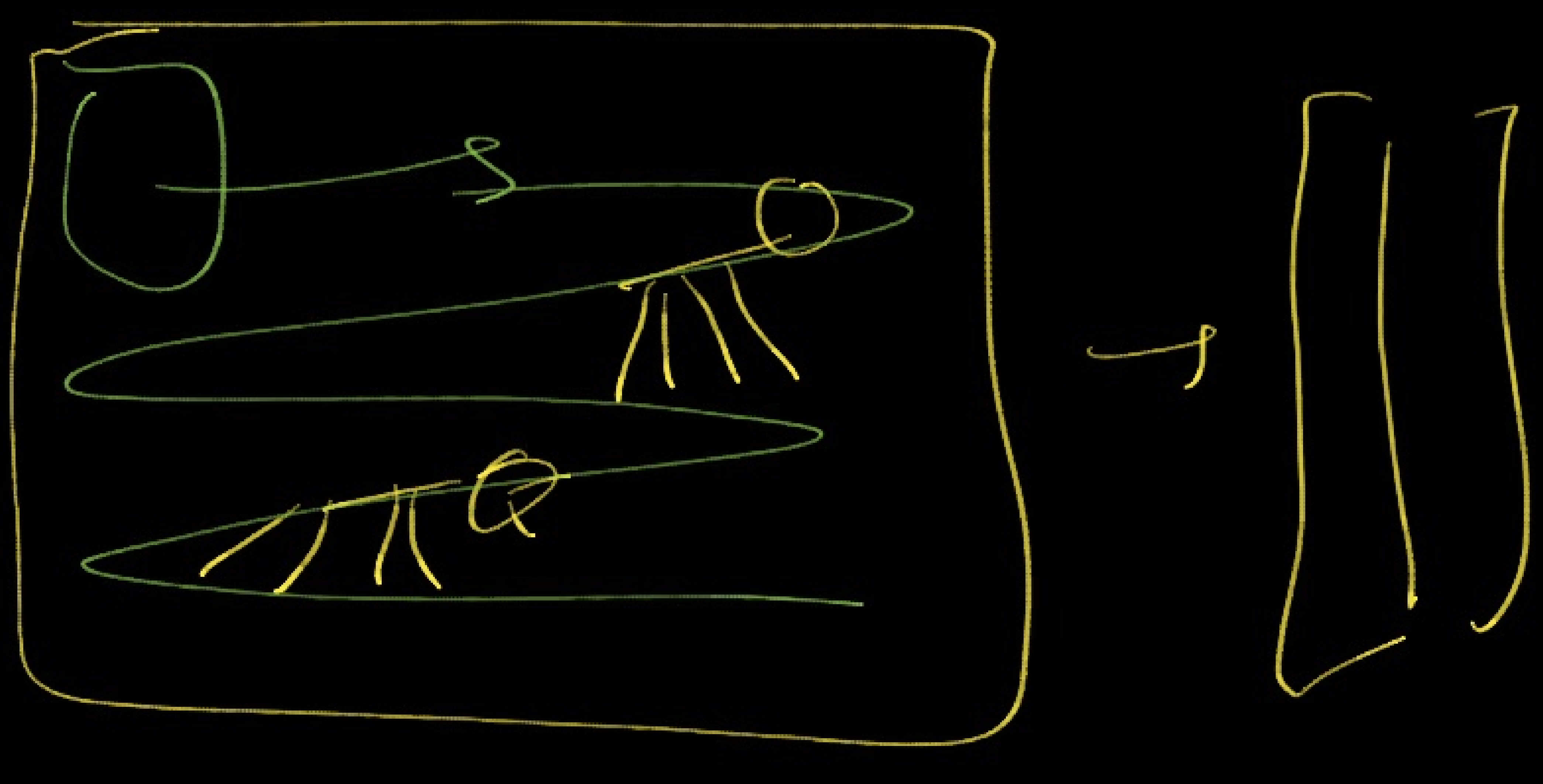
- Dense Layer
- Convolutional Layer

Convolution
Adding Nonlinearity
Pooling

10	15	20
0	-5	4
3	1	2

Max Pooling

15	20
3	4



Stride →

Padding → { same
valid

Pooling →