Machine Learning (COL 774)

Neural Networks: Basics

Mar 31, 2020

(1) Deep Learning Models. COL 774

Machine Learning

Mar 31, 2020

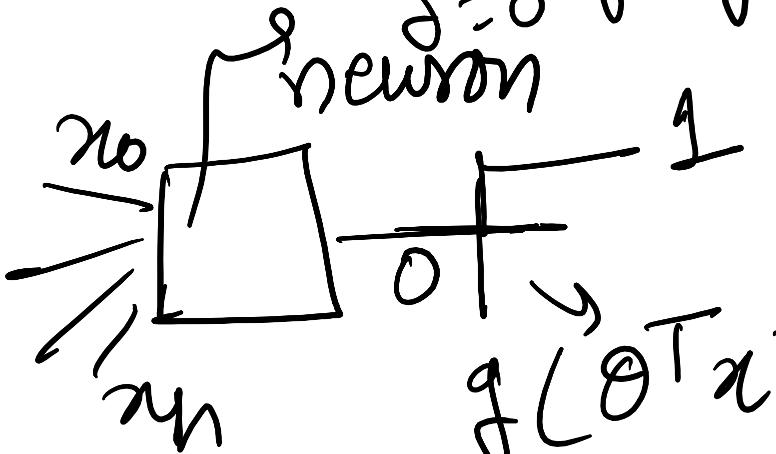
Neural Networks :-

? :- Brain :..

→ millions of neurons in human brain

→ billions of interconnections

$\sum_{j=0}^{n} \theta_j x_j \mid x_0 = 1$ intercept term



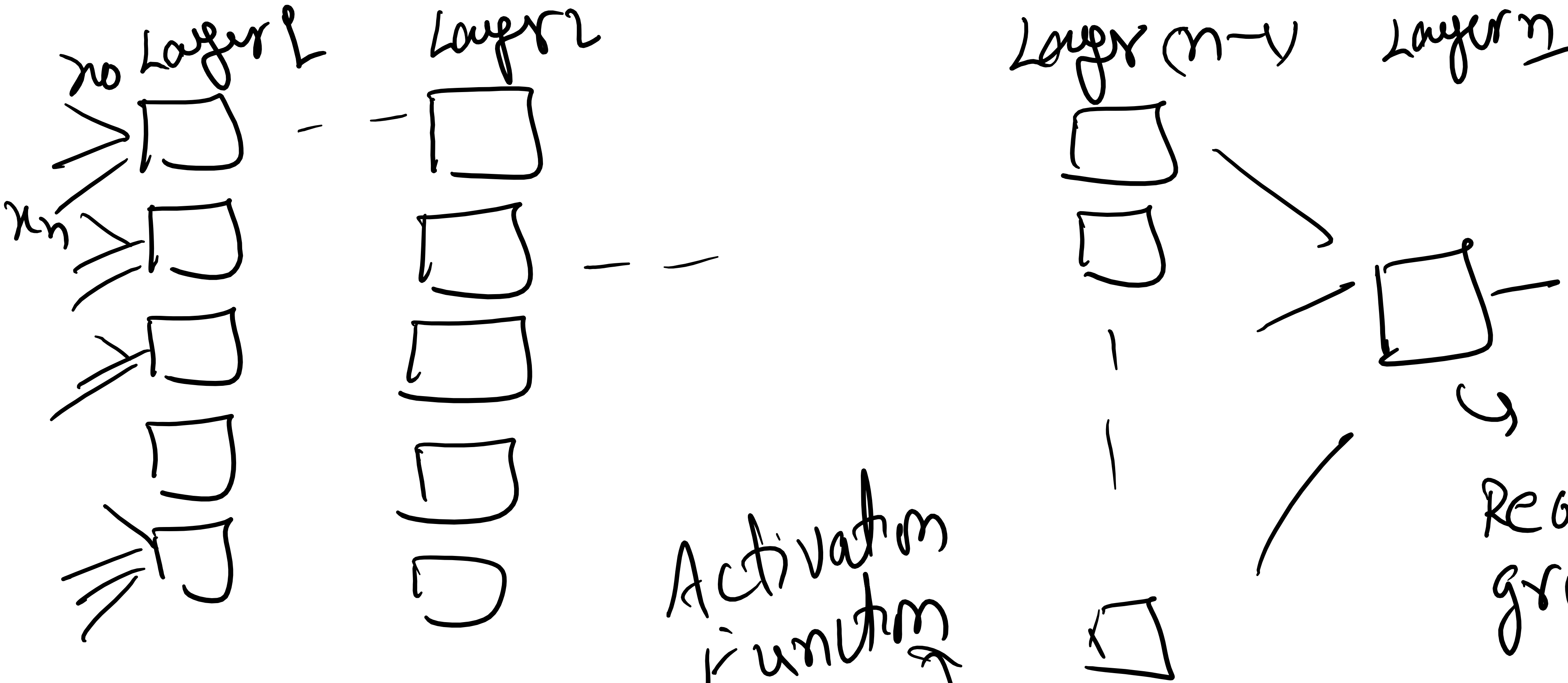
$g(\theta^T x)$

$\hookrightarrow \{ 1\{\theta^T x \geq 0\}$ $\boxed{x_0 = 1}$

$\sum_{j=1}^{n} \theta_j x_j \geq -\theta_0$

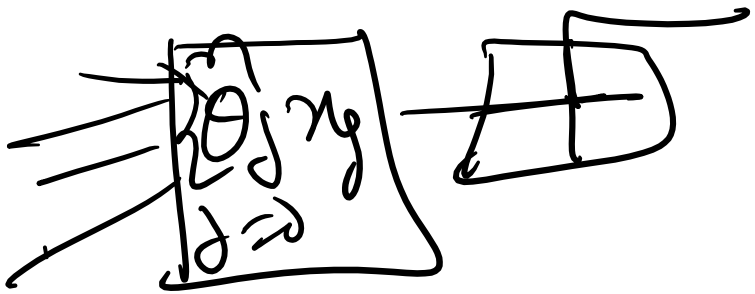
↳ th.

↳ neuron fires



Recognize grandmother

Activation function



$$g(z) = g(\theta^T x) = 1\{\theta^T x \geq 0\}$$

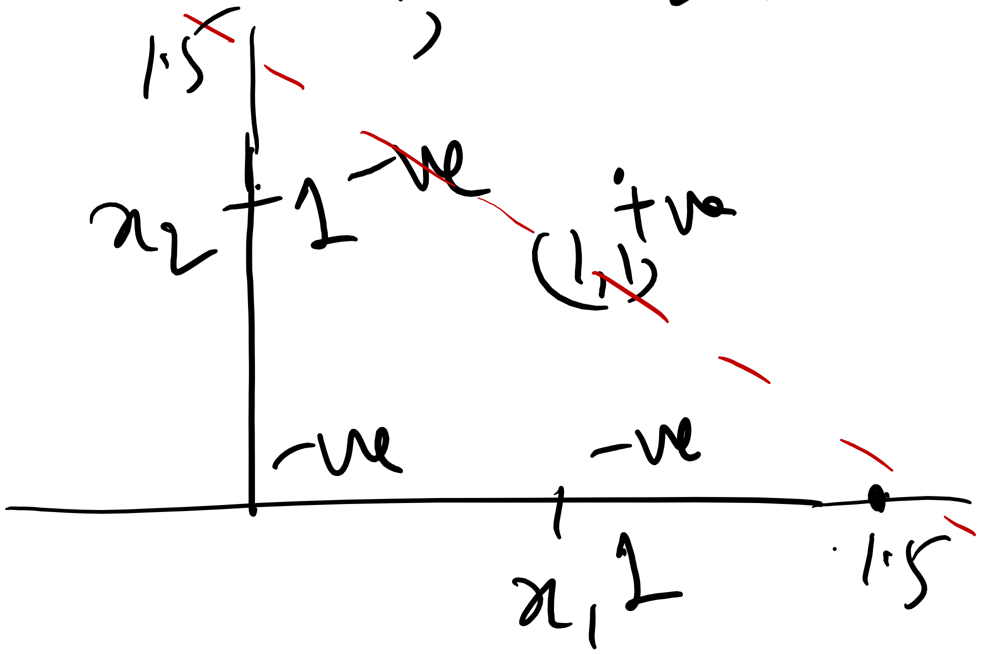
$$h_\theta(x) = g(\theta^T x)$$

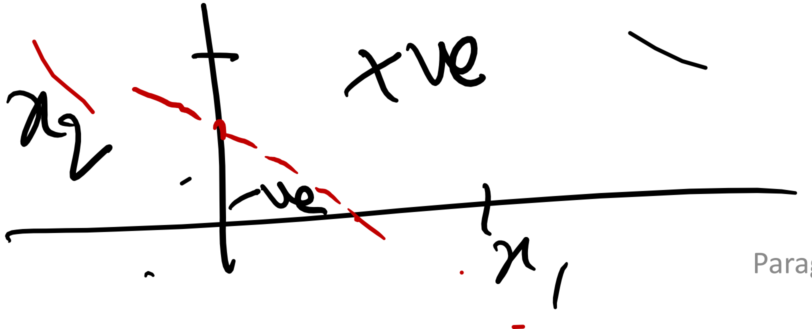
① Logistic Reg
② SVMs

Representing Boolean Functions :-

AND :- $f(x) = x_1 \wedge x_2$

$x_1, x_2 \in \{0, 1\}$





$h_\theta(x) = 1\{\theta_2 x_2 + \theta_1 x_1 + \theta_0 \geq 0\}$

$= 1$ if $x_2 = 1 \wedge x_1 = 1$

AND I

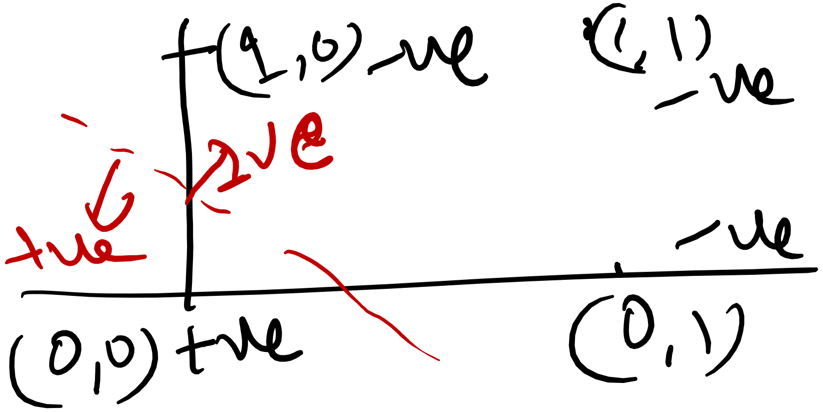
$\theta_2 = 1, \quad \theta_1 = 1, \quad \theta_0 = -1.5$

OR? I

$h_\theta(x) = 1\{\theta^T x \geq 0\}$

$\theta_2 = 1, \quad \theta_1 = 1, \quad \theta_0 = -0.5$

NAND :- $\neg(x_1 \wedge x_2)$



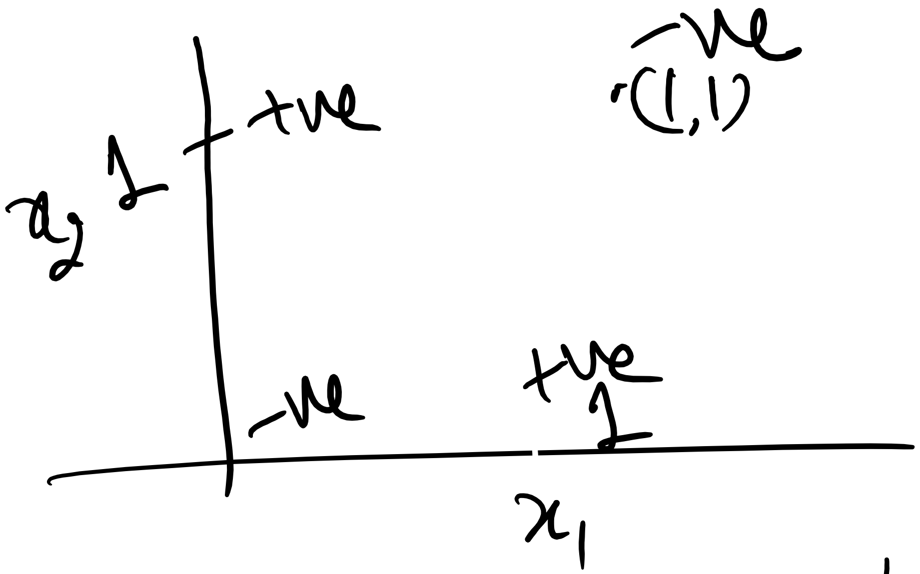
$h_\theta(x) = 1\{\theta^T x \geq 0\}$ NAND

$\theta_2 = -1, \ \theta_2 = -1, \ \theta_0 = 0.5$

HW:- $f(x) = \neg x_1$ $h_\theta(x) = 1\{\theta_1 x_1 + \theta_0 \geq 0\}$

↳ NOT

XOR:- $f(x) = x_1 \oplus x_2$ → 1 iff Exactly one of x_1 and $x_2 = 1$

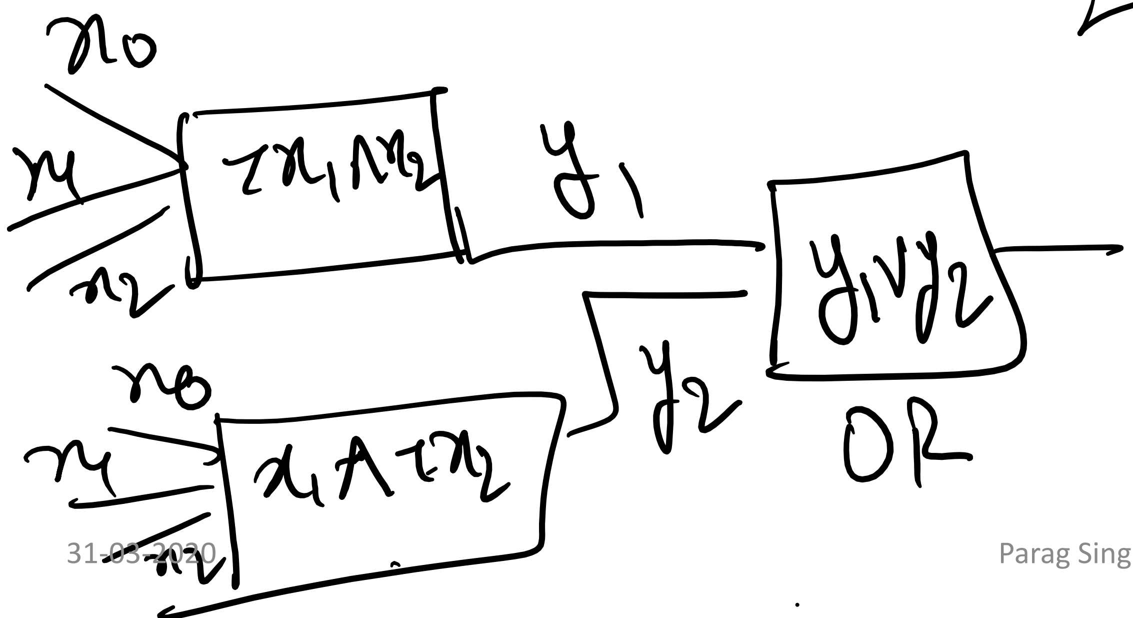


$f(x) = x_1 \oplus x_2$

can not be represented by a single perceptron.

$$\hookrightarrow \underbrace{(\neg x_1 \wedge x_2)}_{\text{AND}} \vee \underbrace{(x_1 \wedge \neg x_2)}_{\text{AND}}$$

OR



$\neg x_1 \wedge x_2$:- $h_\theta(x) = \mathbb{1}\{\theta^T x \geq 0\}$

↳ $\theta_2 = 1, \ \theta_1 = -1, \ \theta_0 = 0.5$

Similarly $x_1 \wedge \neg x_2$:-

XOR :- $\equiv \ y_1 \vee y_2$ where $y_1 = \neg x_1 \wedge x_2$

$y_2 = x_1 \wedge \neg x_2$

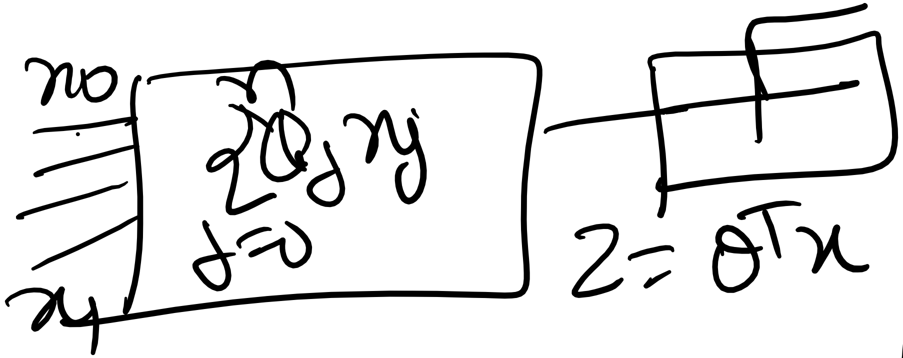
↳ Min # of perceptrons :- 3

Learning the parameters of a perceptron :-

Delta Rule :-

$g(z) = \{1 \text{ iff } \theta^T x \geq 0\}$

→ Activation Units.



↳ can not pass gradient

1. Linearly separable Data
2. η :- (learning rate) sufficiently small.

---

1. Not very principled
2. Does not work when data is NOT Linearly separable

$\theta^{(0)} = \text{init}();$

$t = 0;$

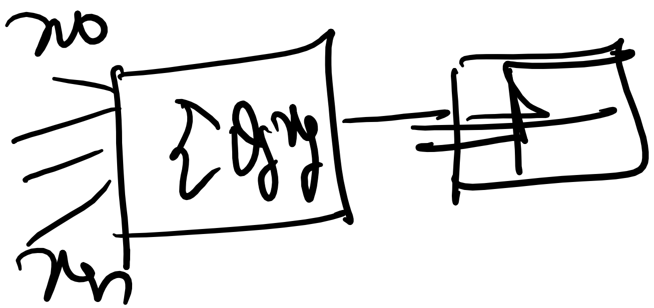
$\text{while } (\exists i : y^{(i)} \neq h_{\theta^{(t)}}(x^{(i)}))$

$\{$

$\forall j :- \theta_j^{(t+1)} \leftarrow \theta_j^{(t)} + \eta [y^{(i)} - h_{\theta^{(t)}}(x^{(i)})] x_j^{(i)}$

$t \leftarrow t+1;$

$\}$



$g(z) = 1\{\theta^T x \geq 0\}$

↳ step fn

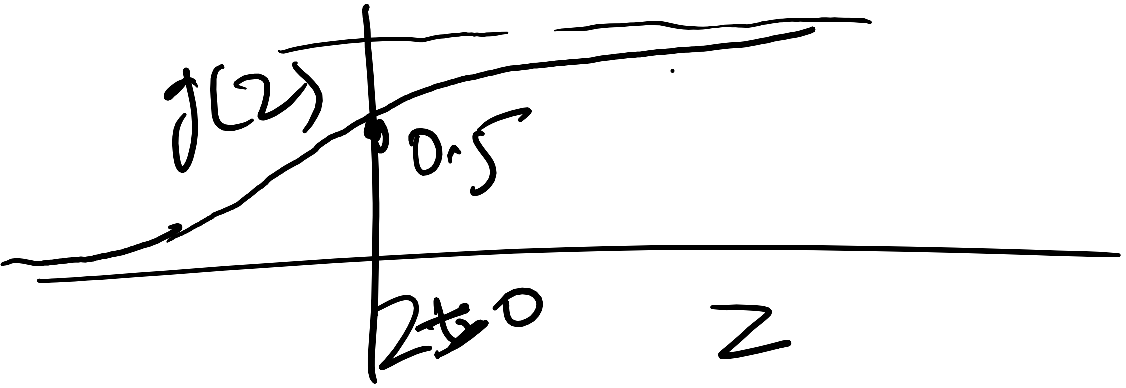
$z = \theta^T x$

$h_\theta(x) = g(\theta^T x)$

$= \dfrac{1}{1+e^{-\theta^T x}}$

very similar

logistic



$g(z) = \dfrac{1}{1+e^{-z}}$

No probabilistic interpretation

For $z > 0$

$z \rightarrow \infty$ $g(z) \rightarrow 1$

For $z < 0$

$g(z) \rightarrow 0$

Learn using gradient descent

Logistic:- $\underline{-LL(\theta)}$

Error Function

$\arg\min_\theta -LL(\theta)$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} [y^{(i)} - h_\theta(x^{(i)})]^2$$

Contrast with objective fn. for logistic ~~regression~~

$$\nabla_\theta J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} 2(y^{(i)} - h_\theta(x^{(i)})) \cdot \nabla_\theta h_\theta(x^{(i)}) (-1)$$

$\rightarrow$ derivative of sigmoid fn.

$$h_\theta(x^{(i)}) = g(\theta^T x^{(i)})$$

$\because \nabla_z g(z) = g(z)(1-g(z))$

$$\nabla_\theta g(\theta^T x^{(i)}) = g(\theta^T x^{(i)})(1-g(\theta^T x^{(i)})) \cdot \boxed{\nabla_\theta (\theta^T x^{(i)})} \; x^{(i)}$$

$$J(\theta) = \frac{1}{2m}\underbrace{\sum_{i=1}^{m}\left[y^{(i)} - h_\theta(x^{(i)})\right]^2}_{} \rightarrow \text{highly non convex}$$

$$h_\theta(x^{(i)}) = g(\theta^T x^{(i)}) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$$

$$\nabla_\theta J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left[y^{(i)} - h_\theta(x^{(i)})\right](-1)\underbrace{\left[h_\theta(x^{(i)})\,(1-h_\theta(x^{(i)}))\right]}_{} x^{(i)}$$

↳ extra term compared to Logistic Reg.

→ initialization

{ Gradient descent update:-

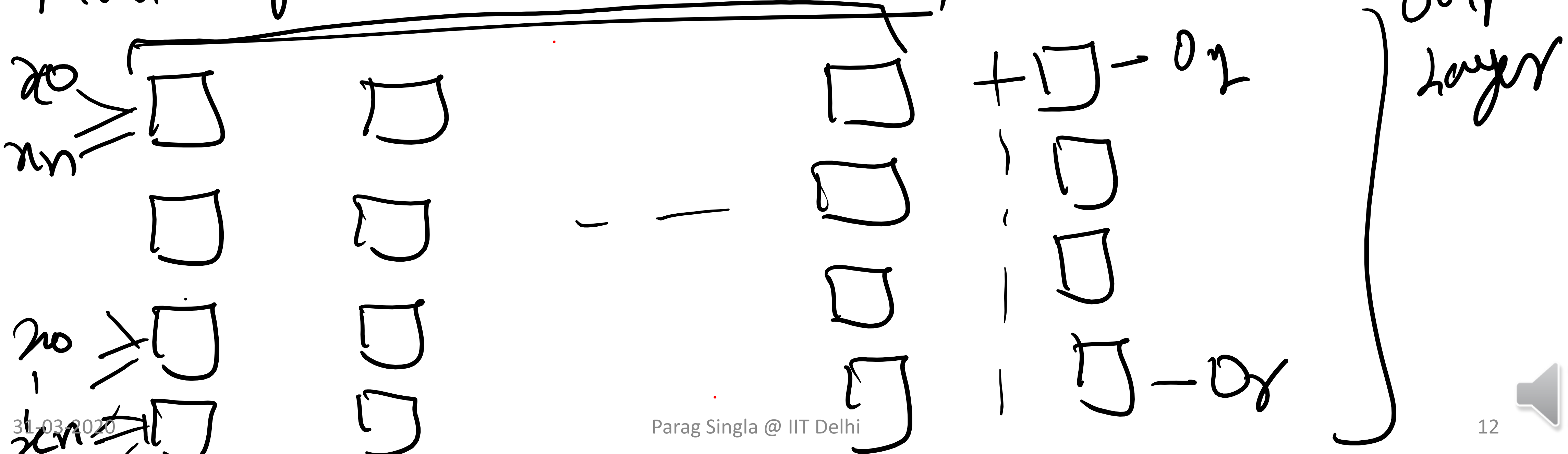
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \underbrace{\eta}_{\text{Learning rate}} \cdot \nabla_\theta J(\theta)\big|_{\theta^{(t)}}$$

} $t \leftarrow t+1$

while(! converged)

Completes learning for a perceptron

Multilayered Neural Network → Hidden Layers



$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\sum_{l=1}^{r}\left(y_l^{(i)} - O_l\right)^2 \quad [r \text{ output units}]$$

$J(\theta)$ ↓ entire set of parameters

Assume:- $\underline{\overline{m=1}}$ (Only to simplify the notation)

$\nabla_\theta J(\theta)$ — parameters in the output layer / parameters in the hidden layers

$$J(\theta) = \sum_{l=1}^{r}(y_l - O_l)^2 \quad [m=1]$$