1. Structure of an artificial neuron

a. Linear combination

b. Nonlinear activation

Also used in logistic regression

dendrites

$$\sigma(z) = \frac{1}{1+\exp(-z)} = \frac{1}{1+\exp(-\sum w_i x_i)}$$

x_1

$\vec{x}$ ↓ Vector

x_2

x_3

$\sum w_i x_i$

$z = \sum w_i x_i$

↓ Scalar

Linear Combination

Linear Combinatio

$\int$

→ Nonlinearity

g ↓ Sigmoid tanh ReLU

Nonlinearity

$g(z)$

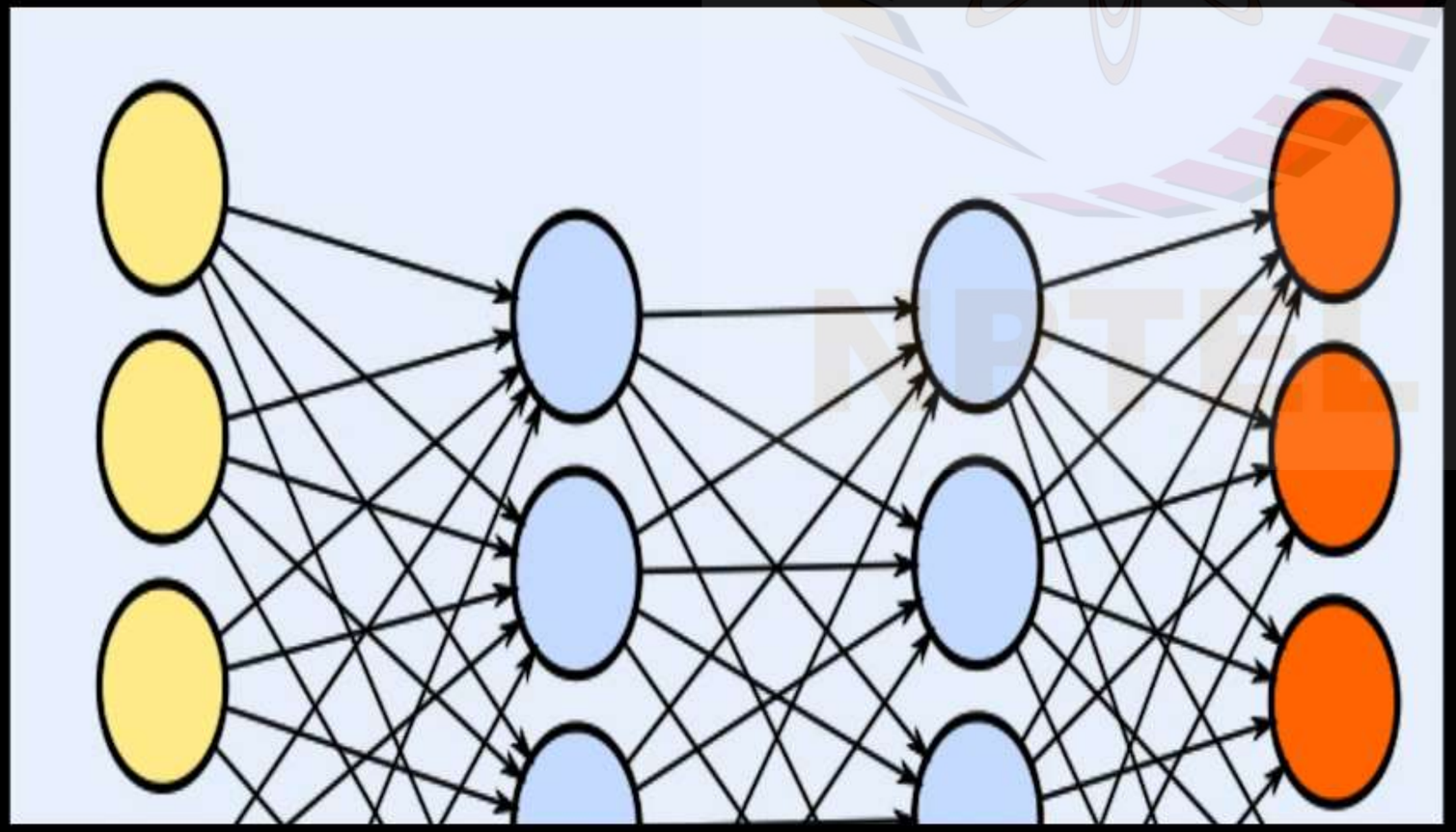
$\equiv a$ (activation)

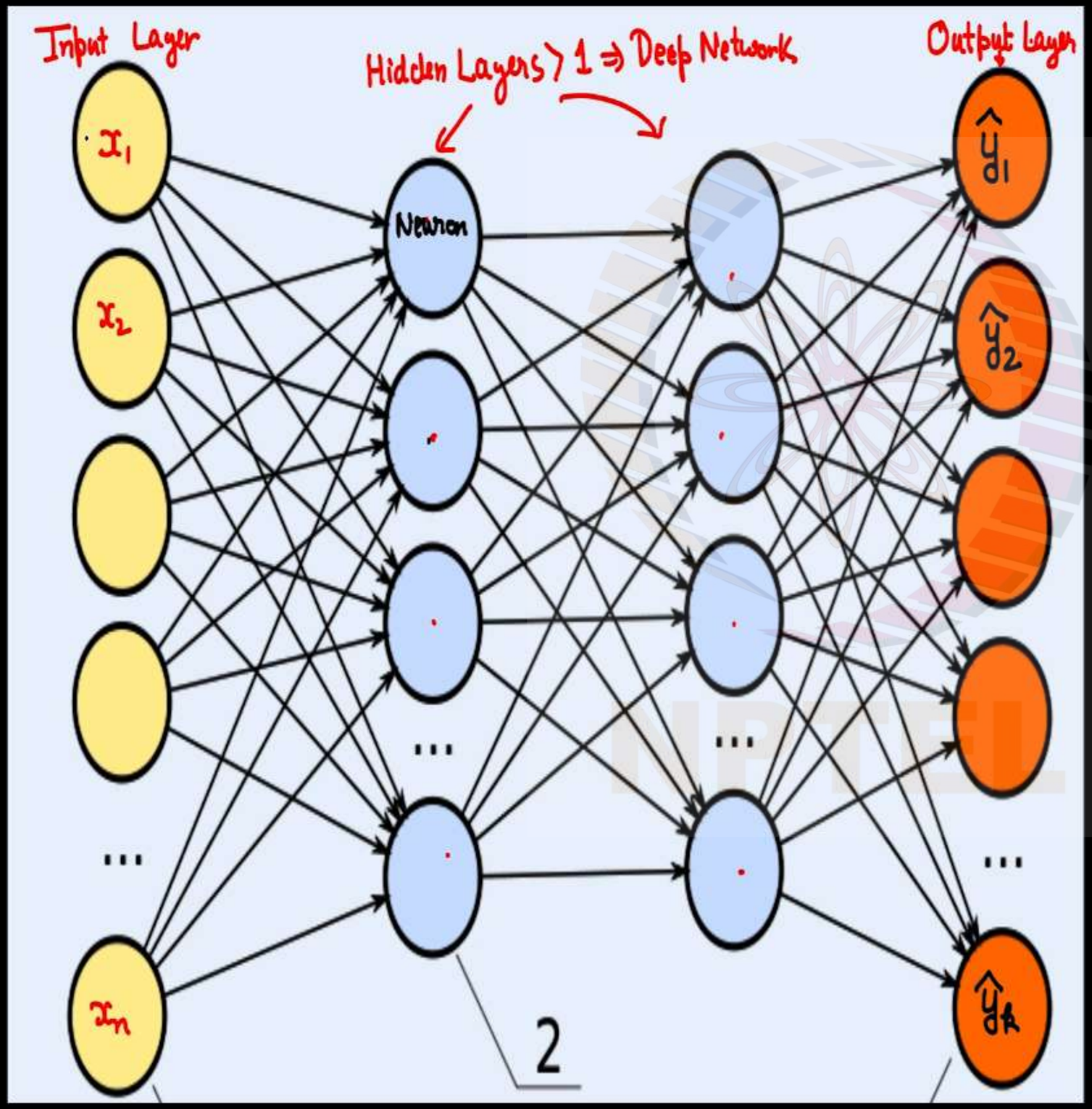
$\hat{y} = a = g(z)$

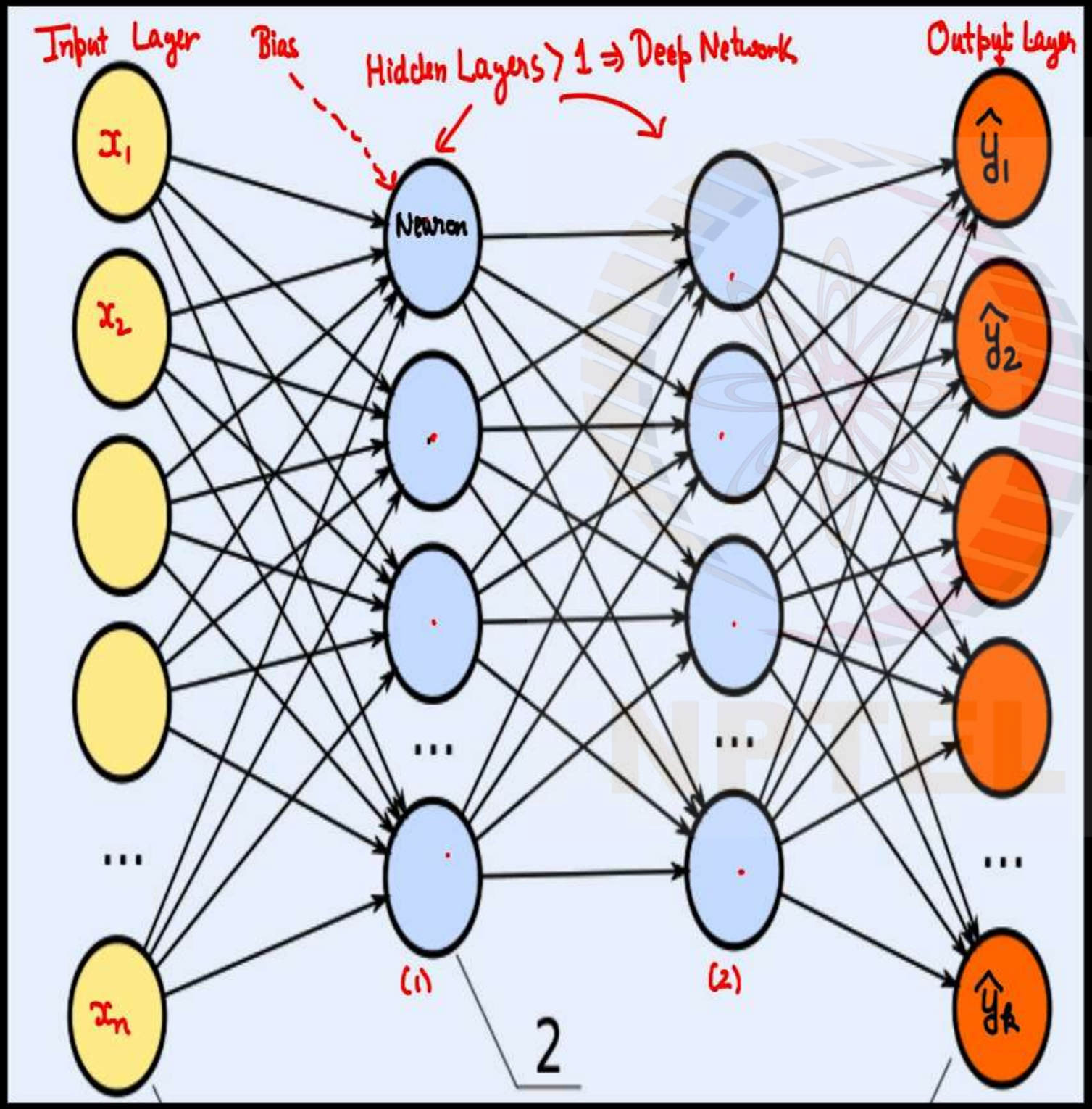
Feed forward Network

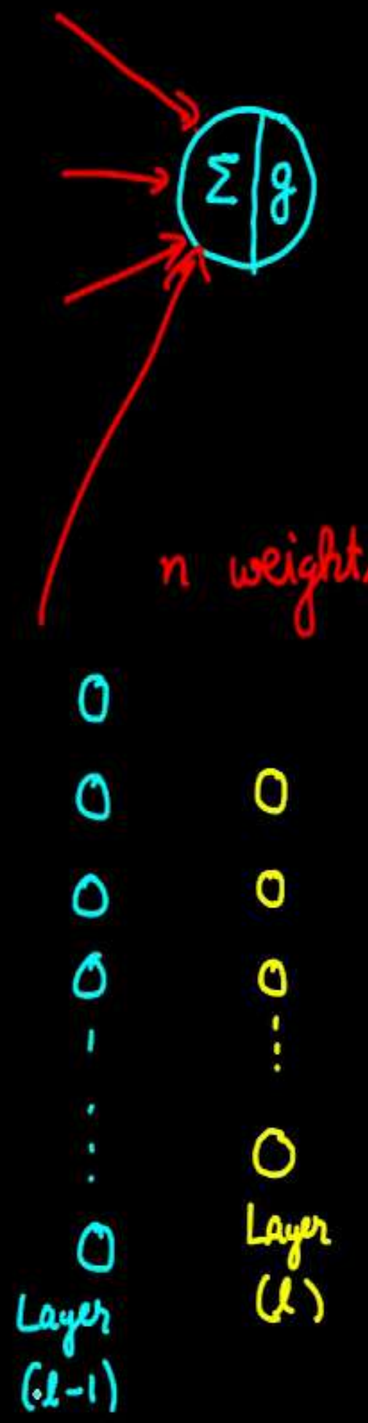
Neural Network

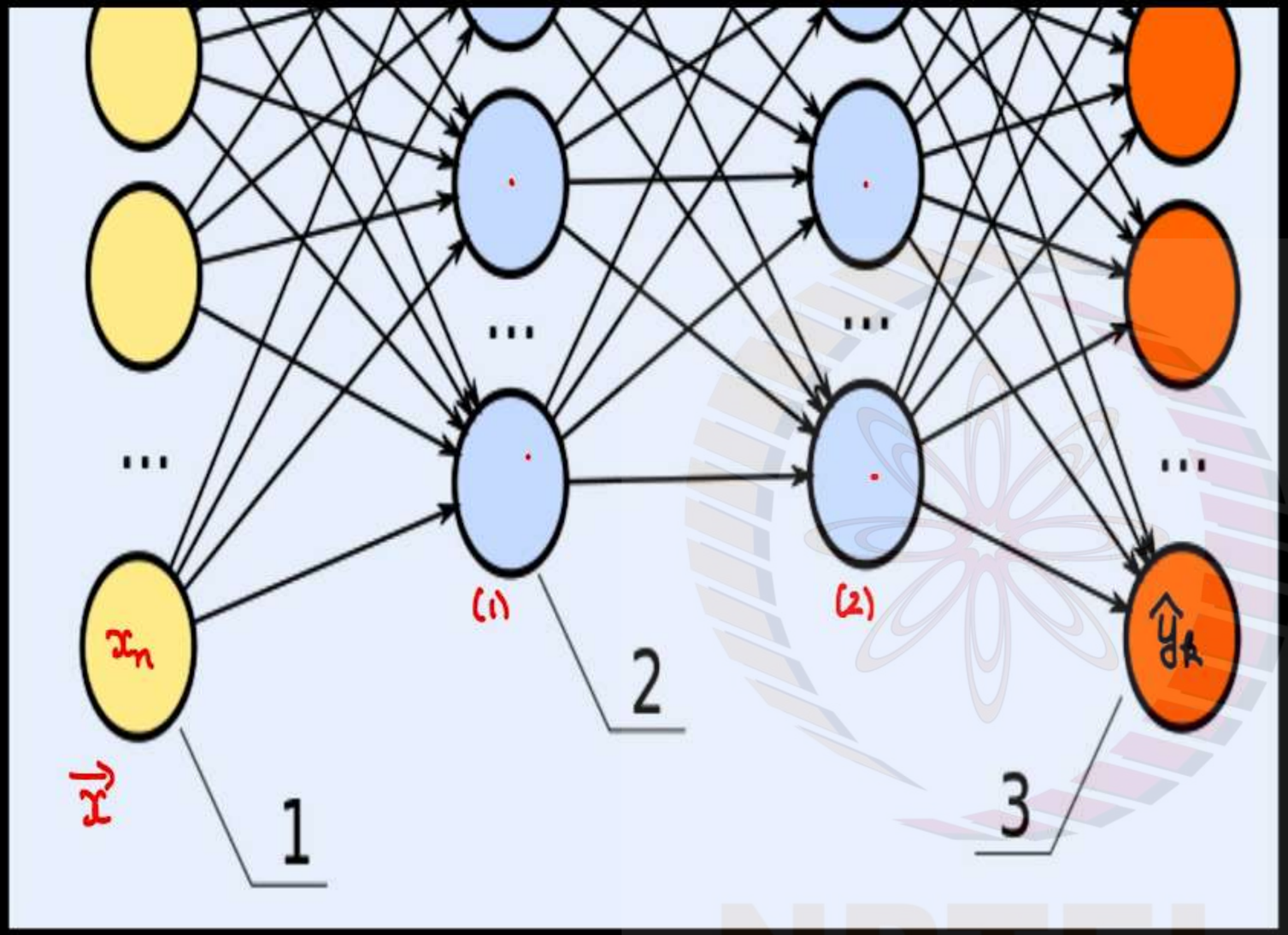
Note Title




Input Layer
Hidden Layers > 1 ⇒ Deep Network
Output Layer
x_1
x_2
x_n
Neuron
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_k$
2


Input Layer
Bias
Hidden Layers > 1 ⇒ Deep Network
Output Layer
x_1
x_2
x_n
Neuron
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_k$
(1)
(2)
2


Σ
g
n weights
Layer (l-1)
Layer (l)

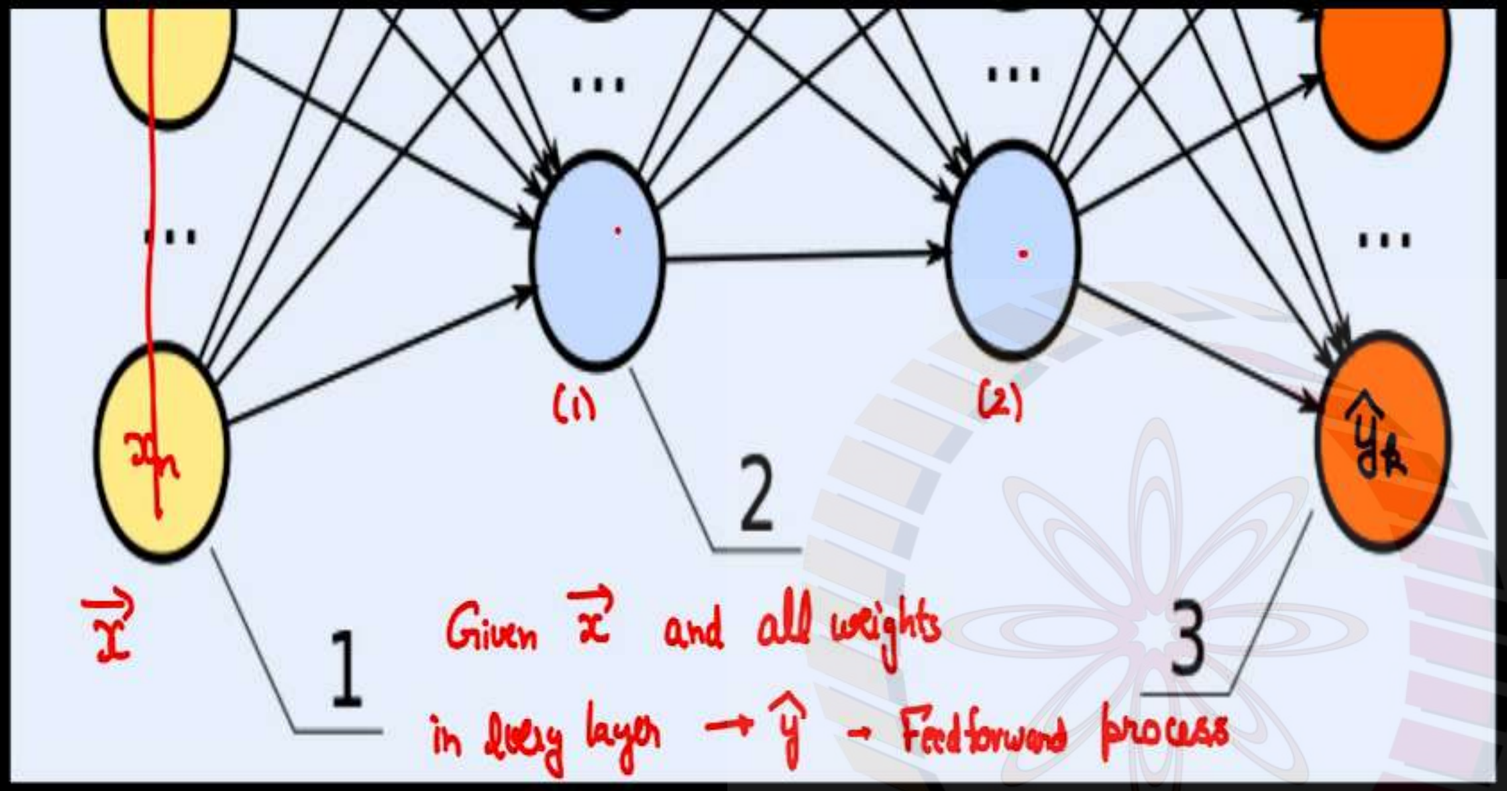


n weights

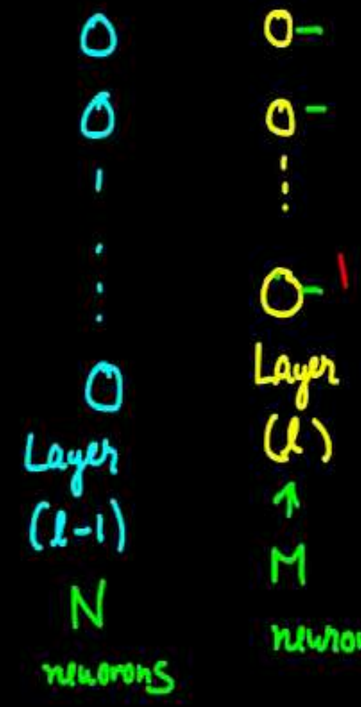
Layer (l-1): N neurons

Layer (l): M neurons

⇒ Total weights required assuming every neuron is connected to every other neuron = N*M + Bias.



Fully Connected network { assuming

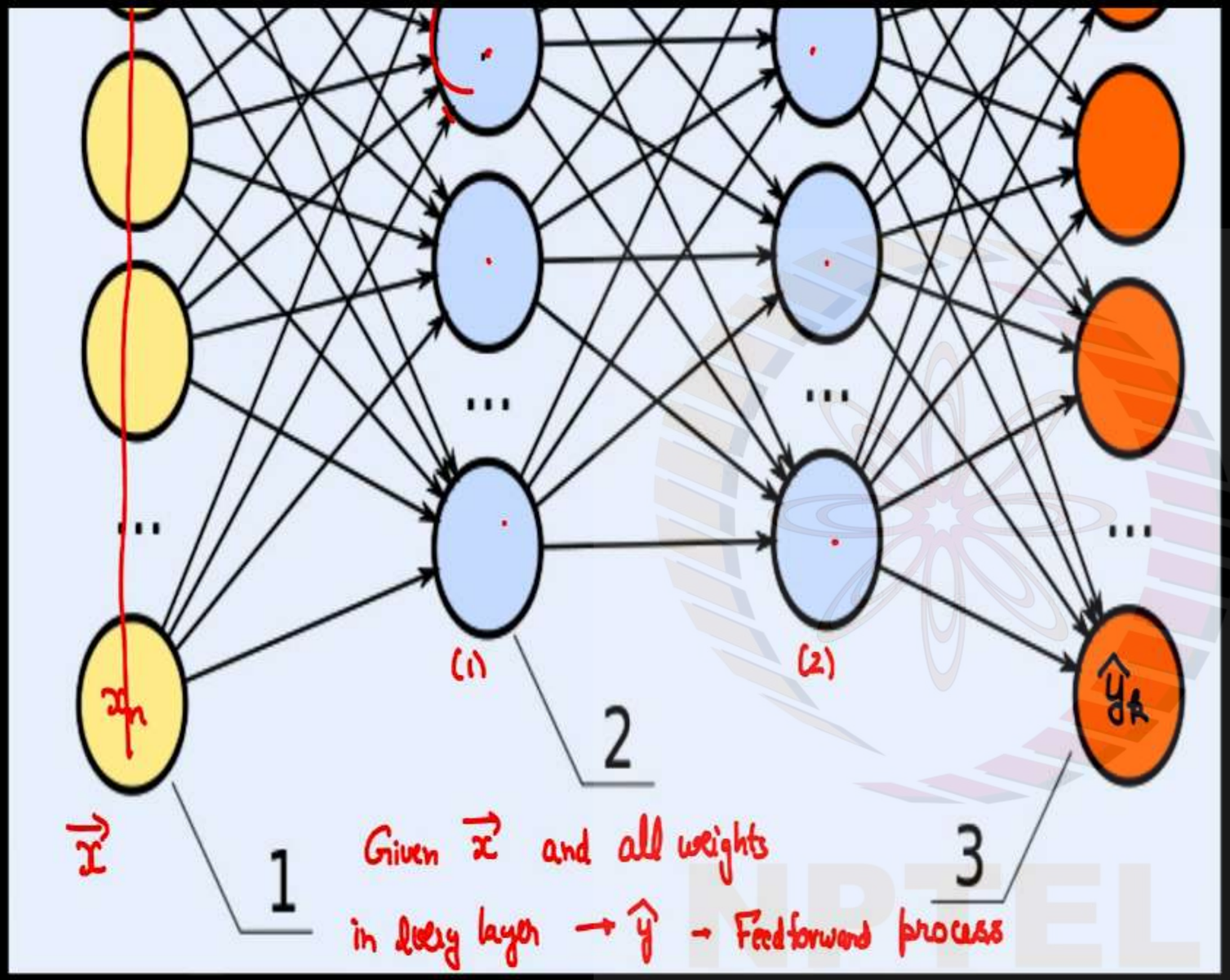


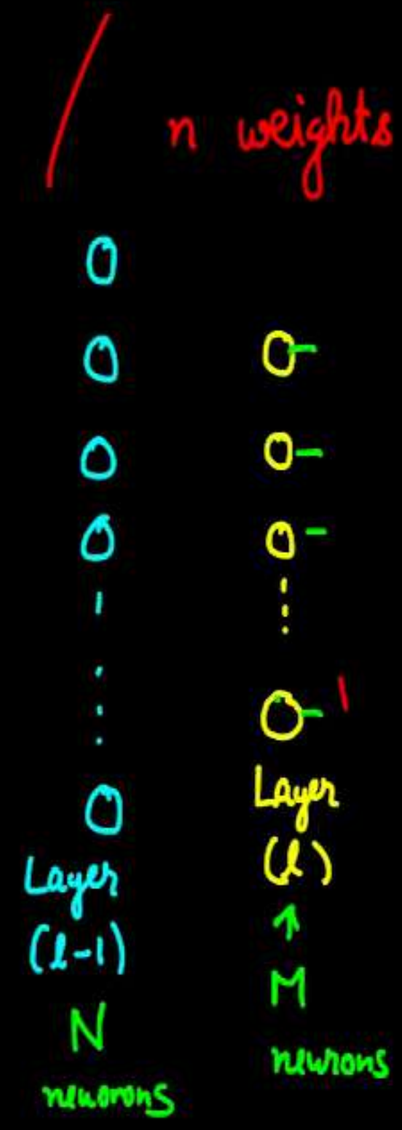
every neuron is connected to every other neuron

⇒ Total weights required

$= N*M + \text{Bias} \rightarrow M$

$= N*M + M$



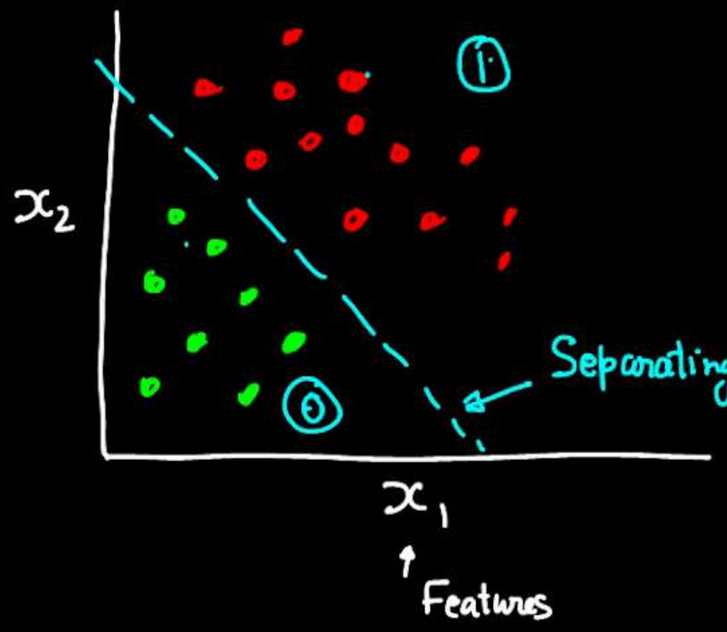


⇒ Total weights required

Fully Connected networks (FC) { assuming every neuron is connected to every other neuron = N*M + Bias → M

= N*M + M

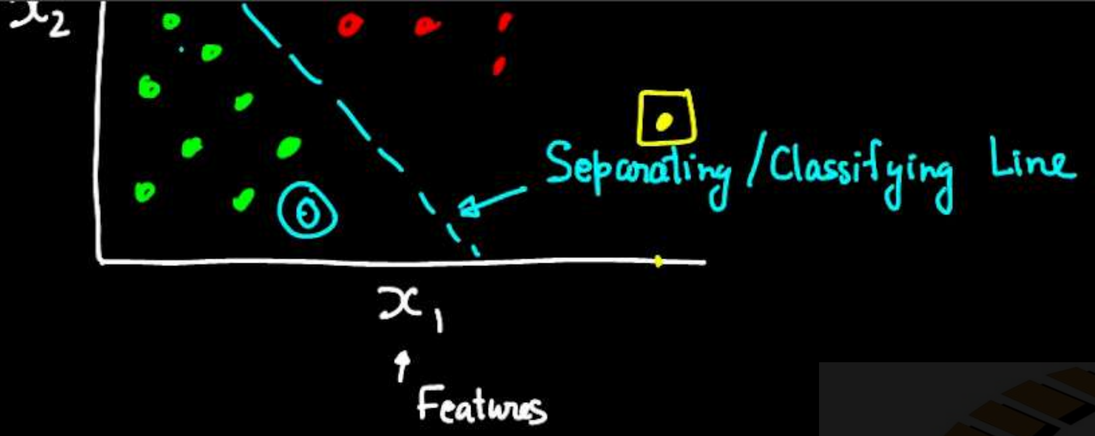
Note Title



Binary Classification → Two classes

1 0 ← Labels

Example	Input $(x_1, x_2) = \vec{x}$	Output y	$\hat{y}$ ← Model
1	$x_1^{(1)}, x_2^{(1)}$	0 or 1	
2	$x_1^{(2)}, x_2^{(2)}$	0 or 1	
⋮		⋮	
m	$x_1^{(m)}, x_2^{(m)}$		



1	$x_1^{(1)}, x_2^{(1)}$	0 or 1
2	$x_1^{(2)}, x_2^{(2)}$	0 or 1
⋮		⋮
m	$x_1^{(m)}, x_2^{(m)}$	⋮

If we try simple linear regression

$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$

Will not lie between 0 and 1

↳ $\hat{y} = w_0 + w_1 x_1 + w_2 x_2$

↑

Will not lie between 0 and 1

Simple idea : "Squish" all data $\rightarrow [0, 1]$

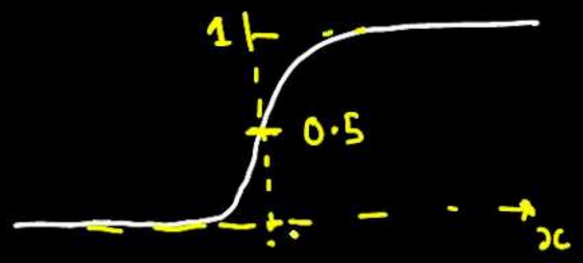
$\sigma \leftarrow$ Sigmoid function

$$\sigma(z) \equiv \frac{1}{1 + \exp(-z)}$$

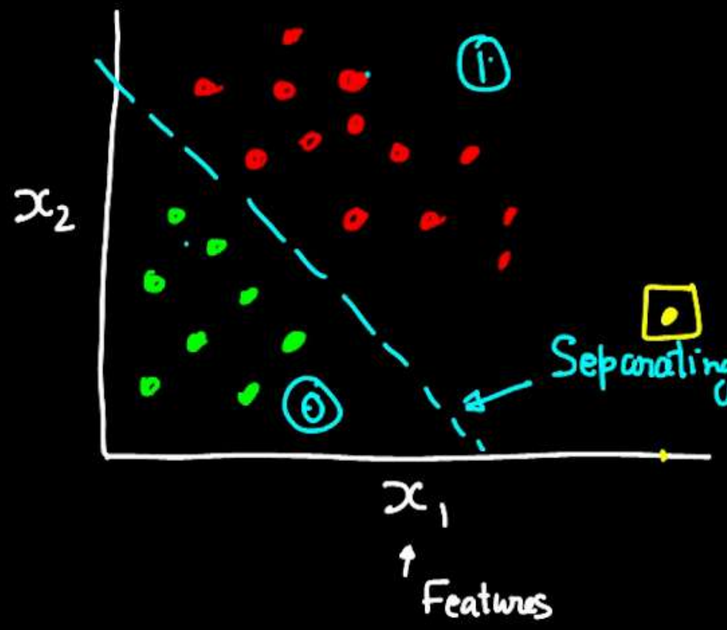
- As $z \rightarrow \infty$, $\sigma(z) \rightarrow 1$
- As $z \rightarrow -\infty$, $\sigma(z) \rightarrow 0$
- At $z = 0$, $\sigma(z) = 0.5$

↑

Monotonic function



Note Title



Binary Classification → Two classes

1 0 ← Labels

Example	Input $(x_1, x_2) = \vec{x}$	Output y	$\hat{y}$ ← Model
1	$x_1^{(1)}, x_2^{(1)}$	0 or 1	
2	$x_1^{(2)}, x_2^{(2)}$	0 or 1	
⋮	⋮	⋮	
m	$x_1^{(m)}, x_2^{(m)}$	⋮	

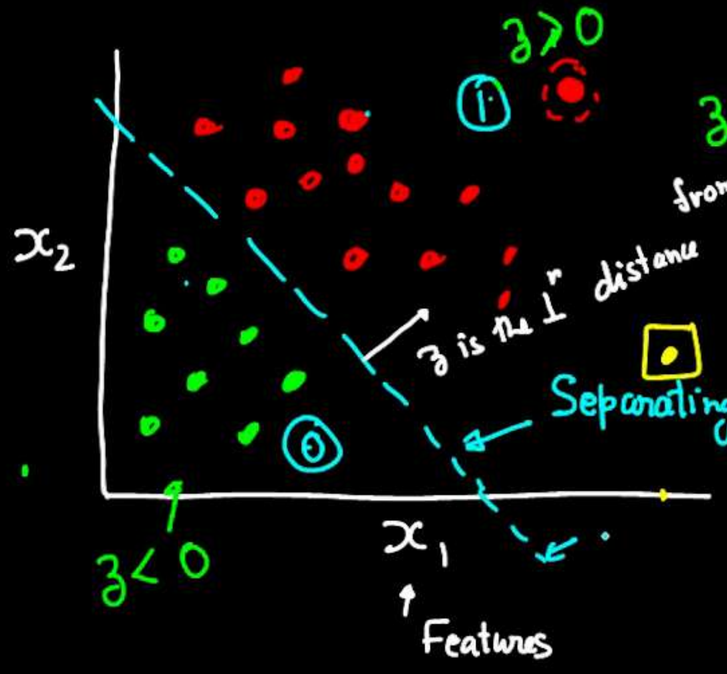
If we try simple linear regression

↳ $\hat{y} = w_0 + w_1 x_1 + w_2 x_2$

Logistic Regression

(Classification)

Note Title



$z = w_0 + w_1 x_1 + w_2 x_2$

from classifying line

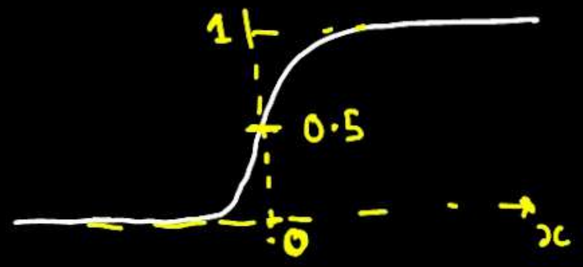
Binary Classification → Two classes

1 ← Labels

0

$\hat{y}$ ← Model

Example	Input $(x_1, x_2) = \vec{x}$	Output y	$\hat{y}$
1	$x_1^{(1)}, x_2^{(1)}$	0 or 1	
2	$x_1^{(2)}, x_2^{(2)}$	0 or 1	
⋮	⋮	⋮	
m	$x_1^{(m)}, x_2^{(m)}$		



9

Monotonic function

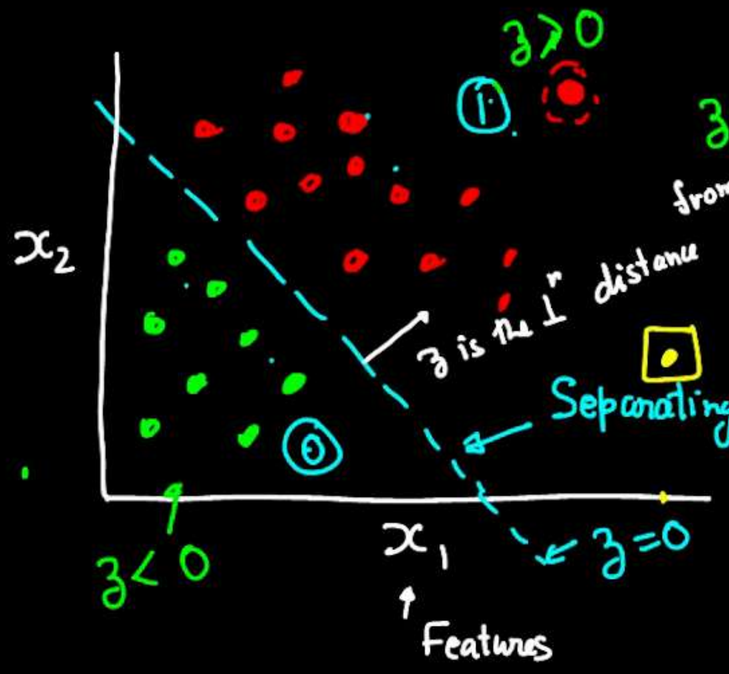
$$\hat{y} = \sigma(\underbrace{w_0 + w_1 x_1 + w_2 x_2}_{z}) \quad \Rightarrow \hat{y} \in [0,1]$$

Interpret $\hat{y} = p(y=1|x)$

$$\hat{y} = \sigma(z) = p(y=1|x)$$

(Classification)

Note Title



$z = w_0 + w_1 x_1 + w_2 x_2$ from classifying line

Binary Classification → Two classes

1 0 ← Labels

$\hat{y}$ ← Model

Example	Input $(x_1, x_2) = \vec{x}$	Output y	$\hat{y}$
1	$x_1^{(1)}, x_2^{(1)}$	0 or 1	
2	$x_1^{(2)}, x_2^{(2)}$	0 or 1	
⋮	⋮	⋮	
m	$x_1^{(m)}, x_2^{(m)}$	⋮	

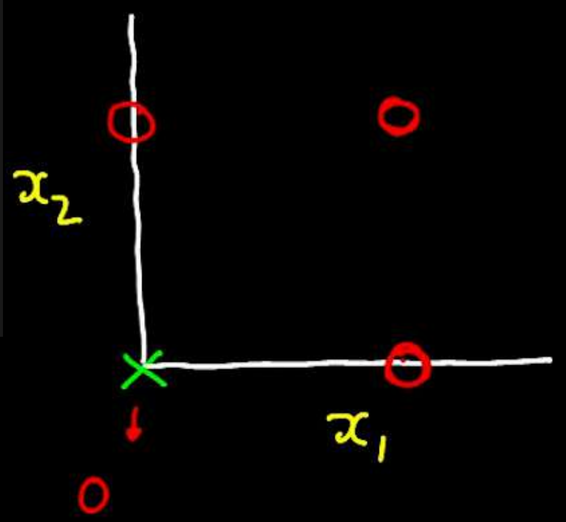
If we try simple linear regression

OR gate through classification

Note Title

OR gate $OR(x_1, x_2)$

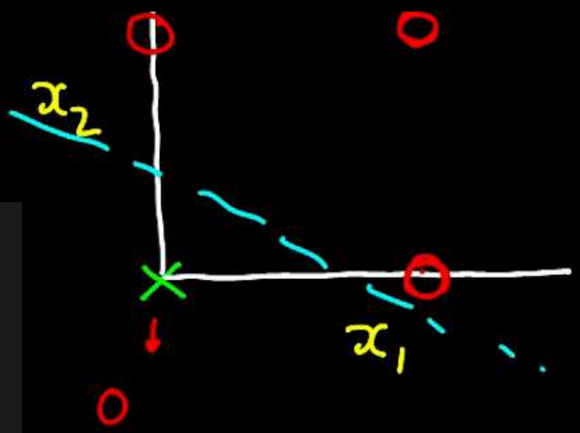
$\vec{x}$	y
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	1



$0 \rightarrow 1$

$x \rightarrow 0$

$\vec{x}$	y
$(0, 0)$	0
$(0, 1)$	1
$(1, 0)$	1
$(1, 1)$	1



$1 \rightarrow 0$

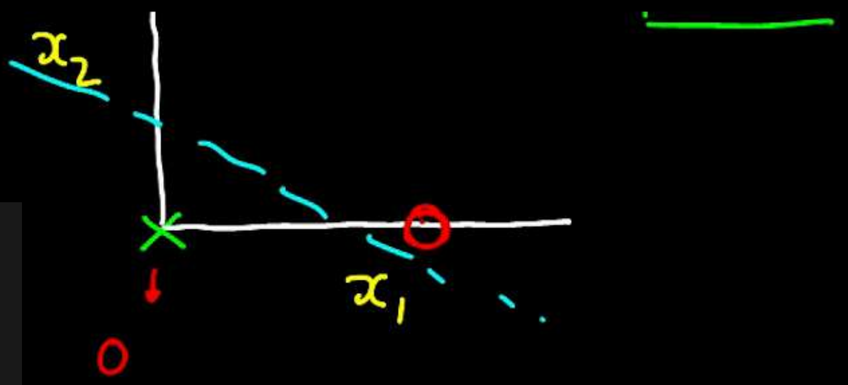
$z = w_1 x_1 + w_2 x_2 + w_0$

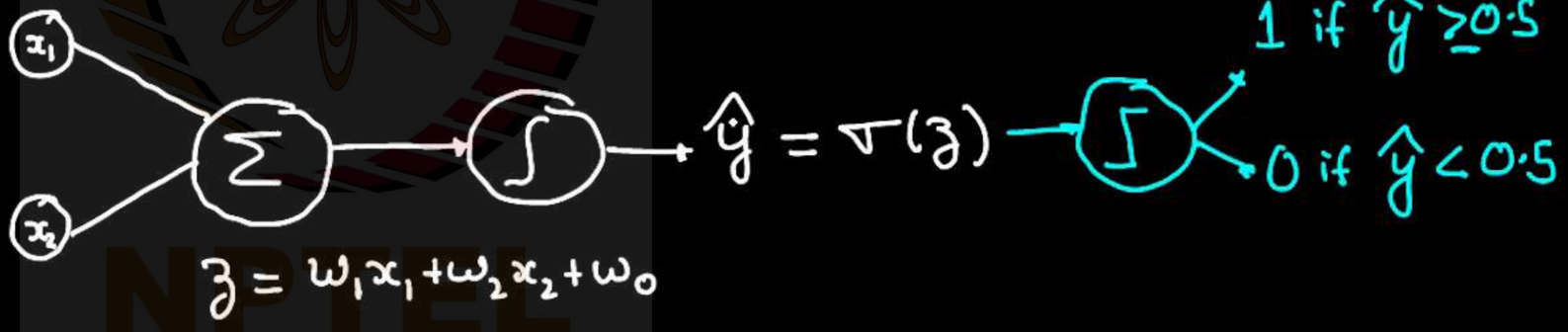
$\hat{y} = \sigma(z)$

1 if $\hat{y} \geq 0.5$

0 if $\hat{y} < 0.5$

(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	1





What weights w_0, w_1, w_2 $(\vec{w})$ will classify same as OR gate

Note Title

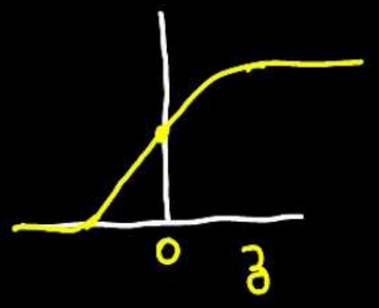
OR gate $OR(x_1, x_2)$

$\vec{x}$	y	z
(0, 0)	0	-ve
(0, 1)	1	+ve
(1, 0)	1	+ve
(1, 1)	1	+ve

$w_0 = -1$

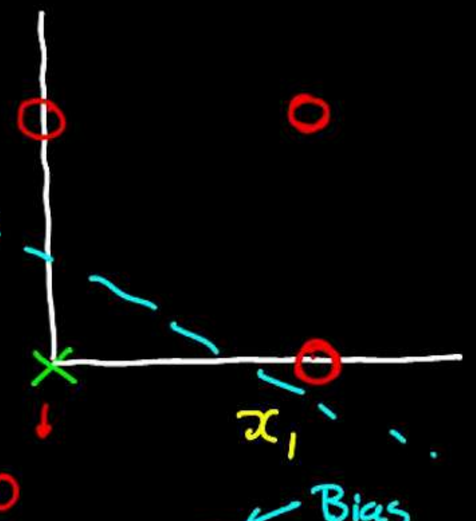
$w_2 = 2$

$w_1 = 2$



$z = w_1 x_1 + w_2 x_2 + w_0$

$\hat{y} = \sigma(z)$

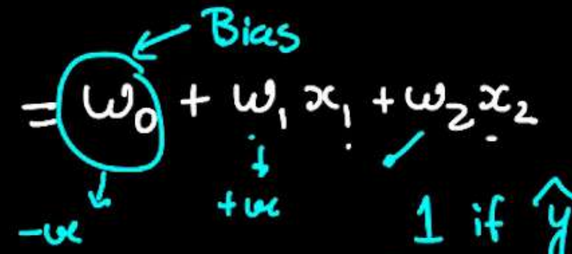


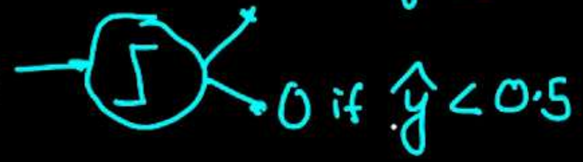
0 → 1

x → 0

$w_0 + w_2 > 0$

$-1 \quad 2$

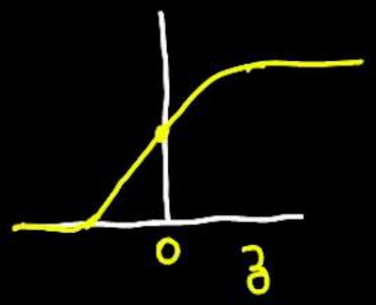


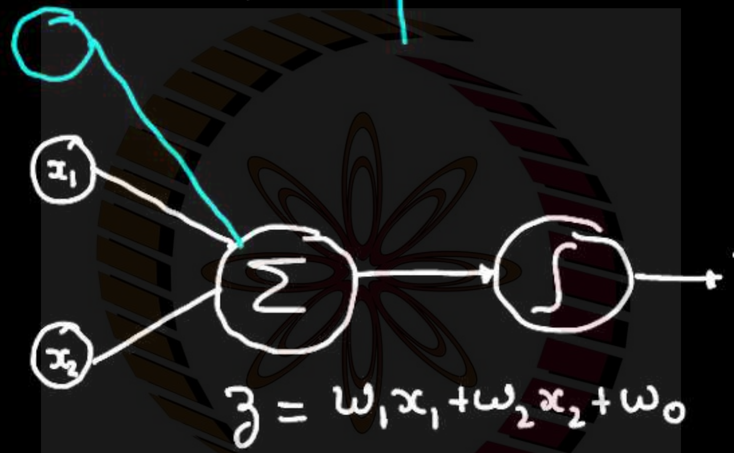


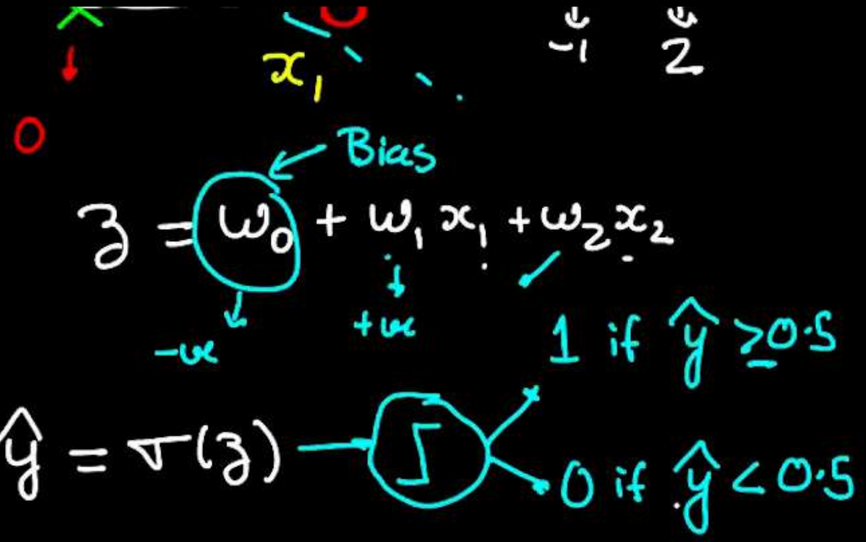
$w_2 = 2$

$w_1 = 2$

(1, 0)	1	+ve
(1, 1)	1	+ve







What weights w_0, w_1, w_2 $(\vec{w})$ will classify same as OR gate

NOTE : (1) w_0 is essential → Need for bias unit

(2) w_0, w_1, w_2 are <u>not unique</u>

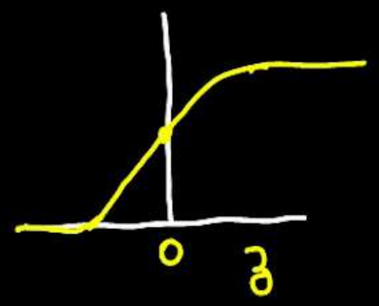
↳ The results of logistic regression

$w_0 = -1$

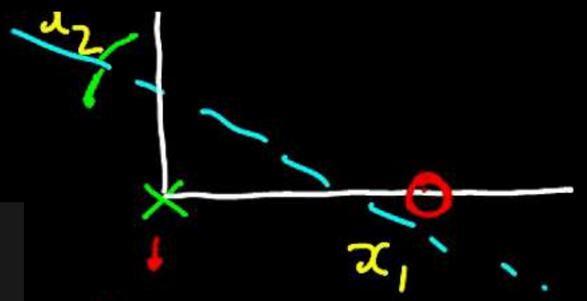
$w_1 = 2$

$w_2 = 2$

(0, 0)	0	-ve
(0, 1)	1	+ve
(1, 0)	1	+ve
(1, 1)	1	+ve

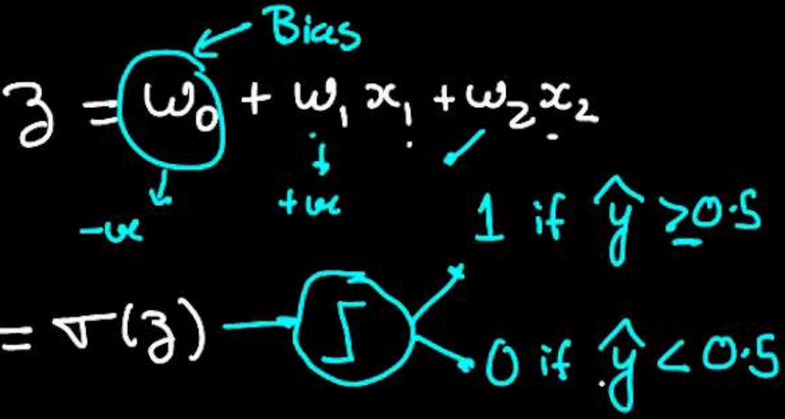


$z = w_1 x_1 + w_2 x_2 + w_0$



$w_0 + w_2 > 0$

$-1 \quad 2$



What weights w_0, w_1, w_2 $(\vec{w})$ will classify same as OR gate

NOTE : (1) w_0 is essential → Need for bias unit

$z = w_1 x_1 + w_2 x_2 + w_0$

What weights w_0, w_1, w_2 $(\vec{w})$ will classify same as OR gate

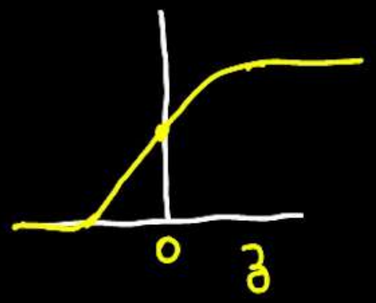
NOTE : (1) w_0 is essential → Need for bias unit

(2) w_0, w_1, w_2 are <u>not unique</u>

↳ The results of logistic regression need not be unique

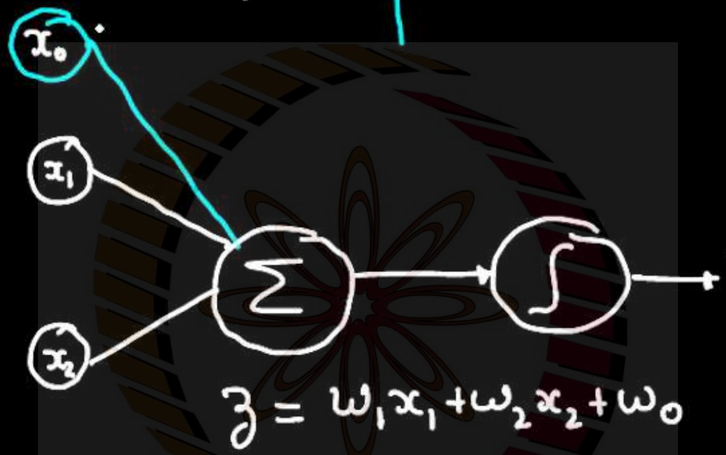
$w_0 + w_1 x_1 + w_2 x_2 = z$.

$w_1 = 2$

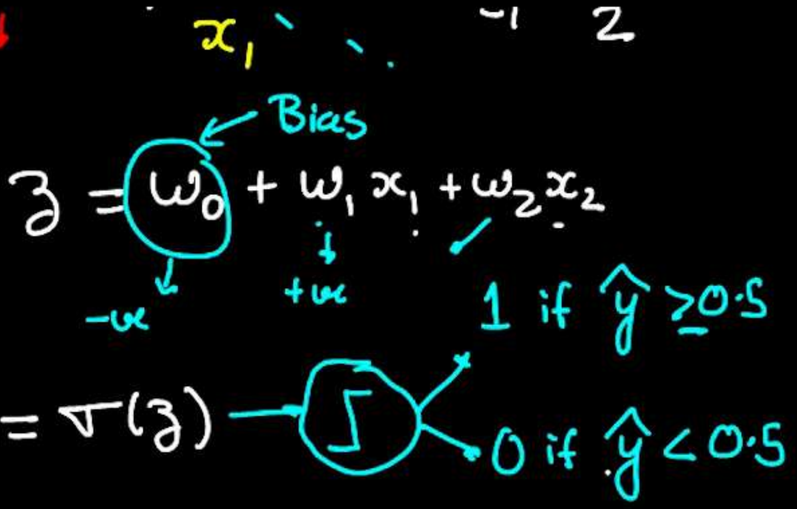


(1, 0) | 1 | +ve

(1, 1) | 1 | +ve



$\hat{y} = \sigma(z)$



What weights w_0, w_1, w_2 $(\vec{w})$ will classify same as OR gate

NOTE : (1) w_0 is essential → Need for bias unit

(2) w_0, w_1, w_2 are <u>not unique</u>

↳ The results of logistic regression

NOR, AND, NAND gates

Note Title

NOR = NOT(OR)

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

$W_0 = -1, W_1 = 2, W_2 = 2$

OR gate

Note Title

NOR = NOT (OR)

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

$W_0 = -1, W_1 = 2, W_2 = 2$

OR gate

$W_0 = 1, W_1 = -2, W_2 = -2$

NOR gate

AND

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

$W_0 = -3, W_1 = 2, W_2 = 2$

AND gate

$W_0 = 3, W_1 = -2, W_2 = -2$

NAND gate

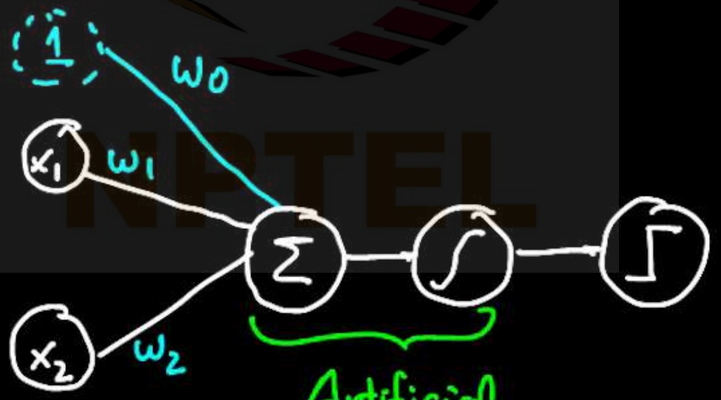
1, 0	0
1, 1	0

$W_0 = 1, W_1 = -2, W_2 = -2$

NOR gate

AND

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



$W_0 = -3, W_1 = 2, W_2 = 2$

AND gate

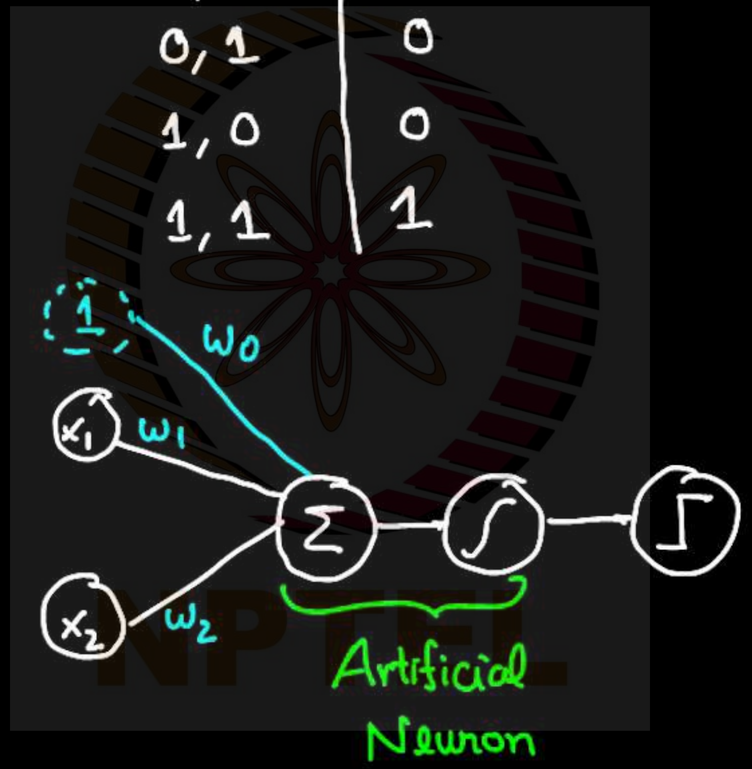
$W_0 = 3, W_1 = -2, W_2 = -2$

NAND gate

Can all logic circuits/gates be represented by this simple network architecture?

AND

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



$\underbrace{w_0 = -3,\ w_1 = 2,\ w_2 = 2}_{\text{AND gate}}$

$\underbrace{w_0 = 3,\ w_1 = -2,\ w_2 = -2}_{\text{NAND gate}}$

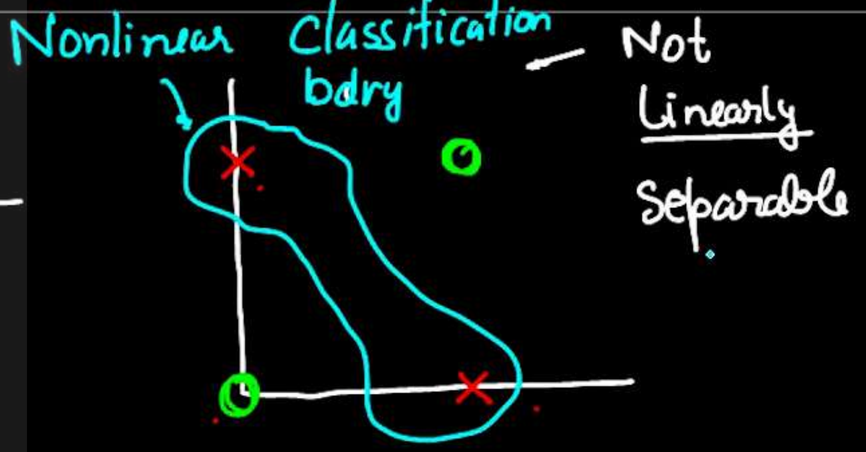
Can all logic circuits/gates be represented by this simple network architecture?

XOR Gate

Note Title

XOR

$\vec{x}$	XOR y
(0,0)	0
(0,1)	1
(1,0)	1
(1,1)	0

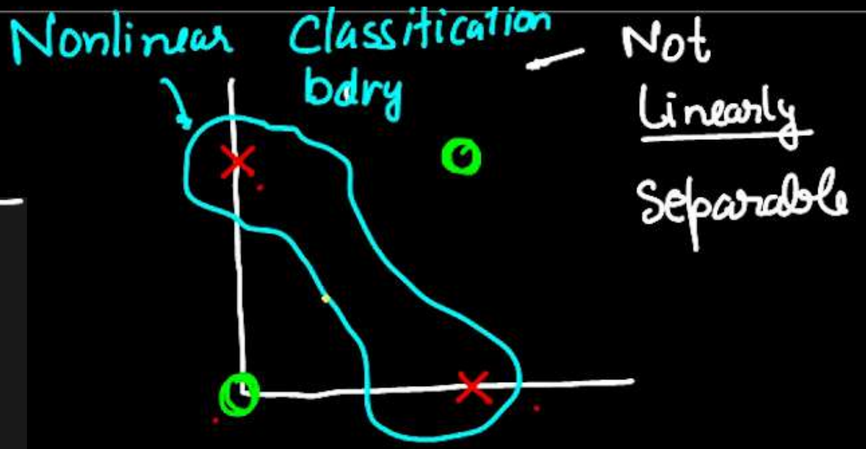


$z = 0 = w_0 + w_1 x_1 + w_2 x_2$

XOR

$\vec{x}$	XOR y
(0,0)	0
(0,1)	1
(1,0)	1
(1,1)	0

Not linealy Separable $\Rightarrow$ Logistic regression will not work directly



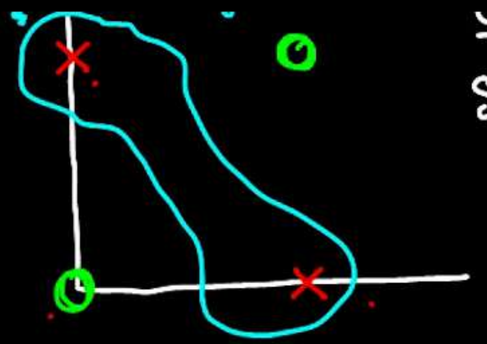
$z = 0 = w_0 + w_1 x_1 + w_2 x_2$

Two possible solutions

(1) Nonlinear features → "Kernel" trick

$\vec{x}$	XOR y
(0,0)	0
(0,1)	1
(1,0)	1
(1,1)	0

Not linearly Separable $\Rightarrow$ Logistic regression will not work directly



Separable

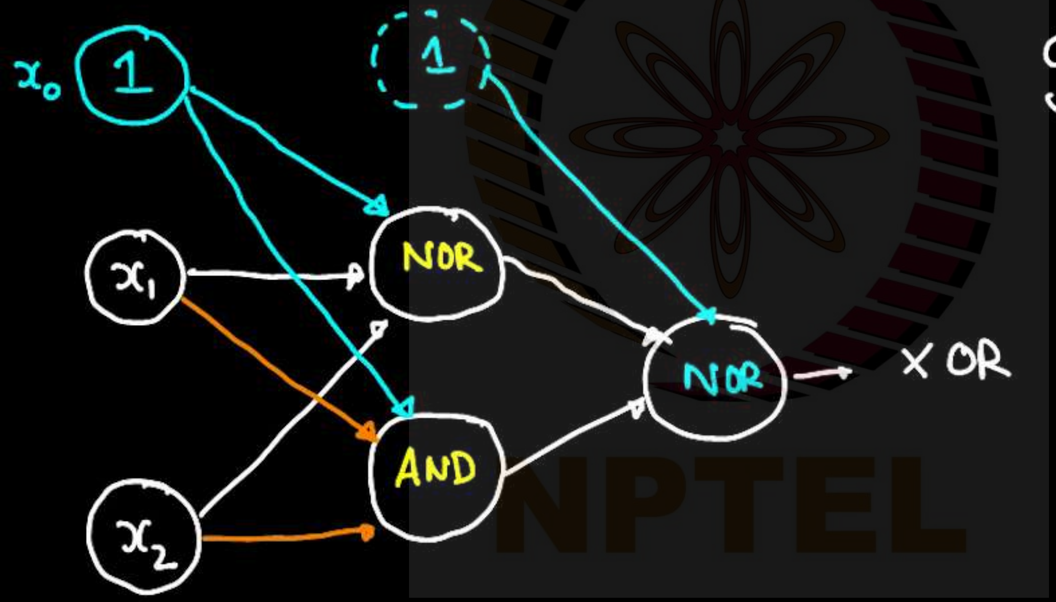
$z = 0 = w_0 + w_1 x_1 + w_2 x_2$

Two possible solutions

1. Nonlinear features → "Kernel" trick
2. Add extra layers → "Deep" networks.

(2) Add extra layers → "Deep" networks.

$$XOR(x_1, x_2) = NOR\left(NOR(x_1, x_2), AND(x_1, x_2)\right)$$

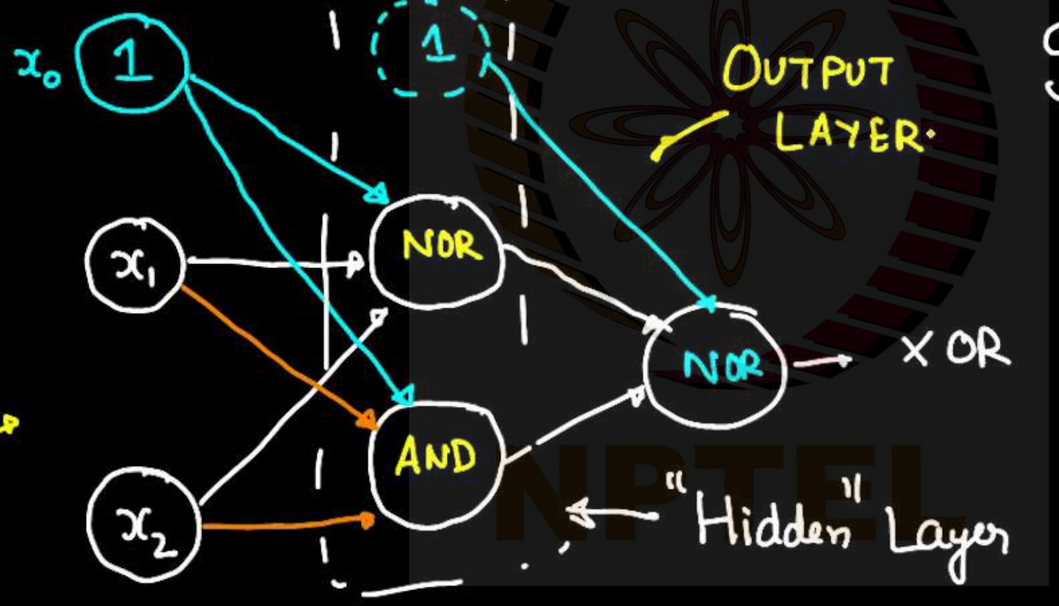


9 weights

(x_1, x_2)	(NOR, AND)
0, 0	1,
1, 0	0,
0, 1	0,
1, 1	0,

(2) Add extra layers → "Deep" networks.

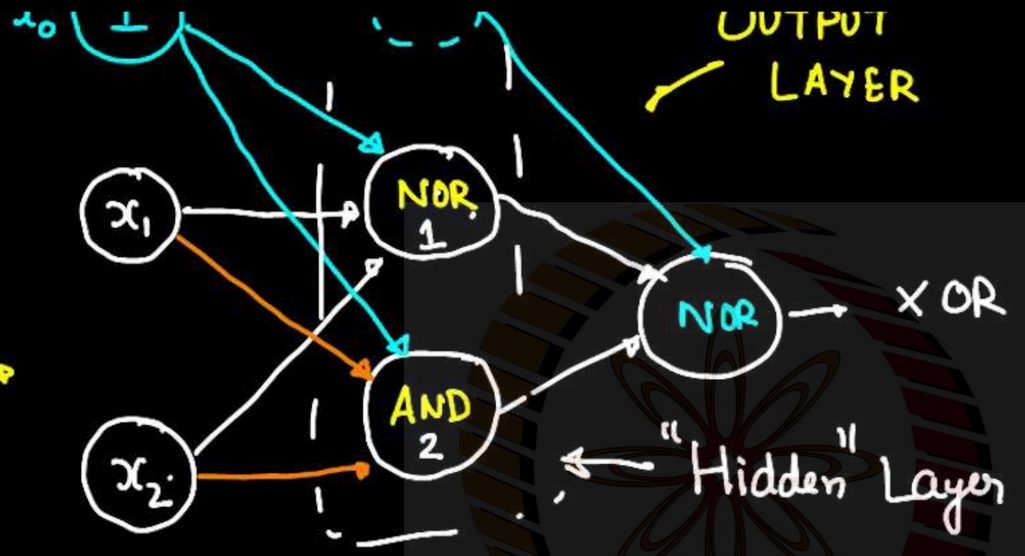
$XOR(x_1, x_2) = NOR\ (NOR(x_1, x_2),\ AND(x_1, x_2))$



Neural Diagram

9 weights

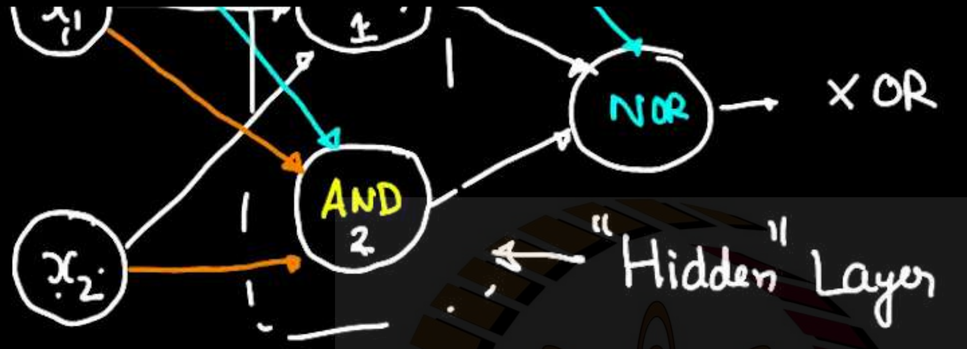
(x_1, x_2)	y_1, y_2 (NOR, AND)	$NOR(y_1, y_2) = XOR$
0, 0	1, 0	0
1, 0	0, 0	1
0, 1	0, 0	1
1, 1	0, 1	0



(X_1, X_2)	y_1, y_2 (NOR, AND)	NOR(y_1, y_2) = XOR
0, 0	1, 0	0
1, 0	0, 0	1
0, 1	0, 0	1
1, 1	0, 1	0

(NOR) $W_{01} = 1,\ W_{11} = -2,\ W_{21} = -2$

(AND) $W_{02} = -3,\ W_{12} = 2,\ W_{22} = 2$



INPUT LAYER

Neural Diagram

0, 0	1, 0	0
1, 0	0, 0	1
0, 1	0, 0	1
1, 1	0, 1	0

Layer

(NOR) $W_{01}^{(1)} = 1$, $W_{11}^{(1)} = -2$, $W_{21}^{(1)} = -2$

(AND) $W_{02}^{(1)} = -3$, $W_{12}^{(1)} = 2$, $W_{22}^{(1)} = 2$

} Layer (1) weights
6 weights in total

(FINAL NOR) $W_{01}^{(2)} = 1$, $W_{11}^{(2)} = -2$, $W_{21}^{(2)} = -2$

(NOR) $W_{01}^{(1)} = 1, \quad W_{11}^{(1)} = -2, \quad W_{21}^{(1)} = -2$

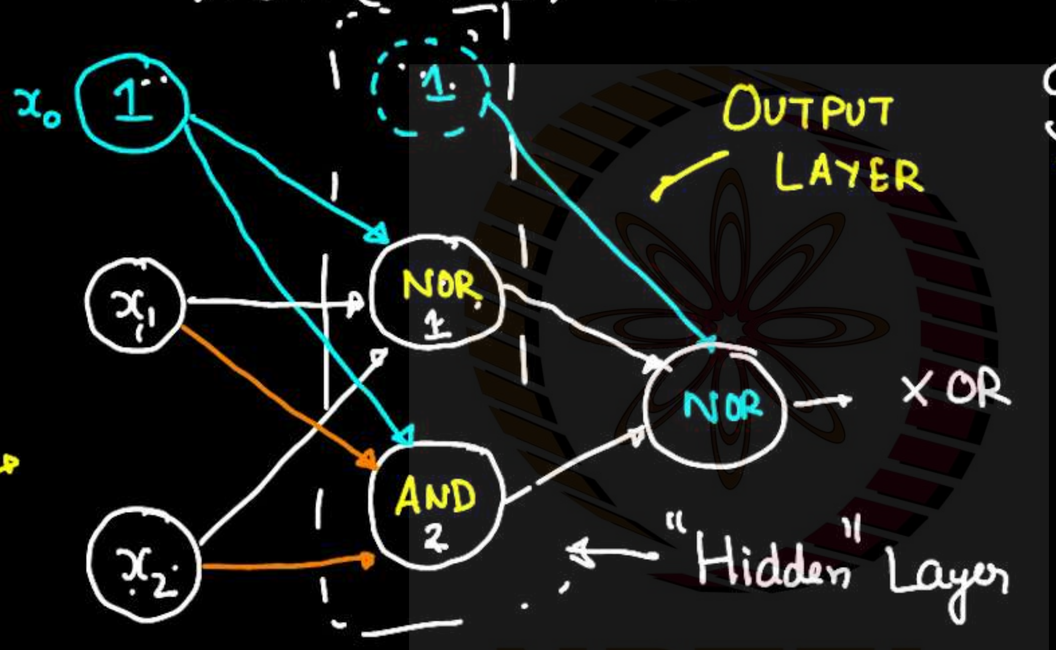
(AND) $W_{02}^{(1)} = -3, \quad W_{12}^{(1)} = 2, \quad W_{22}^{(1)} = 2$

} Layer (1) weights
6 weights in total

(FINAL NOR) $W_{01}^{(2)} = 1, \quad W_{11}^{(2)} = -2, \quad W_{21}^{(2)} = -2$

Universal Approximation Thm: Any function can be expressed as a neural network with one hidden layer to desired accuracy

$$XOR(x_1, x_2) = NOR\left(NOR(x_1, x_2),\ AND(x_1, x_2)\right)$$



Neural Diagram

9 weights

(x_1, x_2)	y_1, y_2 (NOR, AND)	$NOR(y_1, y_2) = XOR$
0, 0	1, 0	0
1, 0	0, 0	1
0, 1	0, 0	1
1, 1	0, 1	0

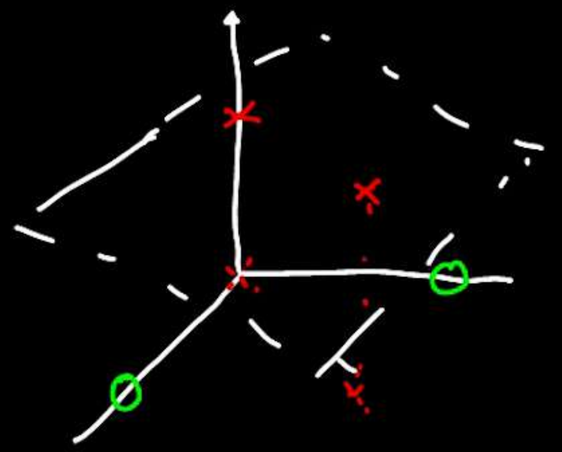
Layer

(NOR) $W_{01}^{(1)} = 1,\ W_{11}^{(1)} = -2,\ W_{21}^{(1)} = -2$

(AND) $W_{02}^{(1)} = -3,\ W_{12}^{(1)} = 2,\ W_{22}^{(1)} = 2$

} Layer (1) weights
6 weights in total

Note Title



Linearly separable in a higher dimension.

XOR

$\vec{x}$	XOR y
(0,0)	0
(0,1)	1
(1,0)	1
(1,1)	0

Not linearly Separable $\Rightarrow$

Nonlinear Classification bdry — Not Linearly Separable

$z = 0 = w_0 + w_1 x_1 + w_2 x_2$

Logistic regression will not work directly

Two possible solutions

Classification

Note Title

Multinomial Classification, $K > 2$

K ↓ No of classes

$0, 1, \ldots, 9$

$K = 10$ class

.

Multinomial Classification, $K > 2$

K → No of classes

$0, 1, \ldots, 9$

$K = 10$ class

What we need

horses, cats, dogs

(1) How do we represent the output class (label)?

↳ One hot vector

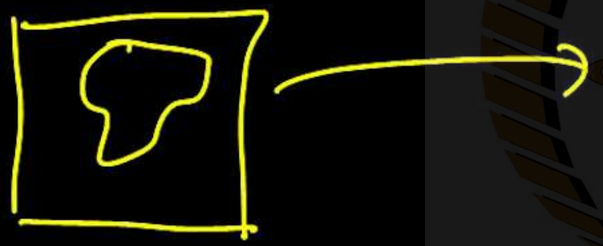
(2) What happens in the final layer for nonlinearity?

↳ Softmax

(3) Loss function

Note Title

Multinomial Classification



Cat → 1

Dog → 0

Cats, Dog, Horses

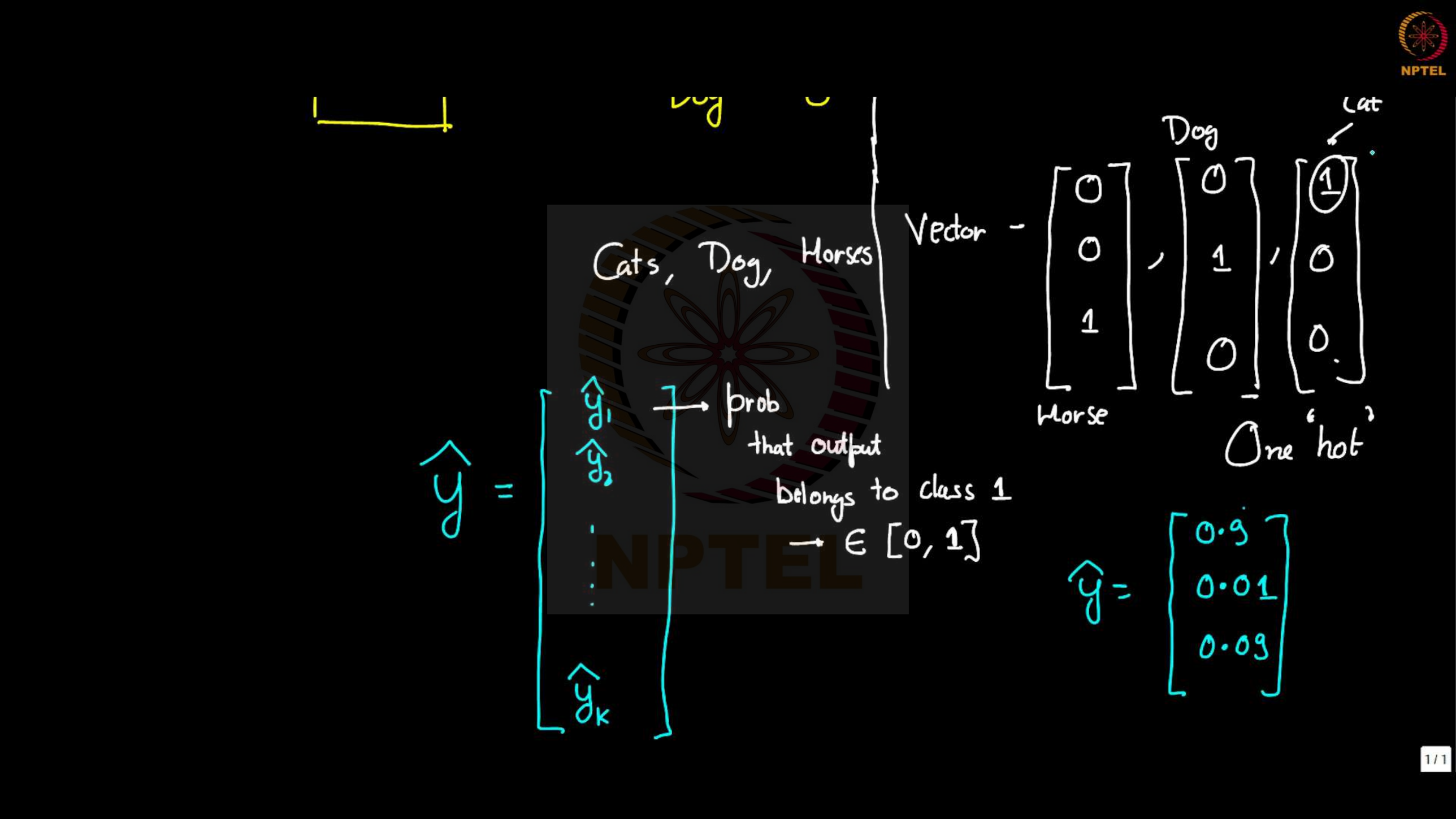
Vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ Cat

$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ Cat, Dog

Dog

Vector - $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Horse Dog Cat


Cats, Dog, Horses
Vector -
Dog
Cat
Horse
One 'hot'
$\hat{y}$ =
$\hat{y}_1$ → prob that output belongs to class 1 → ∈ [0, 1]
$\hat{y}_2$
$\hat{y}_k$
$\hat{y}$ =
0.9
0.01
0.09

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_k \end{bmatrix}$$

that output

belongs to class 1

$\rightarrow \in [0, 1]$

One hot

$$\hat{y} = \begin{bmatrix} 0.9 \\ 0.01 \\ 0.09 \end{bmatrix}$$

$$\hat{y}_k = P(\text{belongs to class } k)$$

$$\hat{y}_k = P(y_k = 1 \mid x)$$

Image / Input

Differentiating the Sigmoid

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

$$\frac{d\sigma}{dz} = \sigma'(z) = \frac{-1}{(1+\exp(-z))^2} \frac{d\,e^{-z}}{dz}$$

$$= \frac{1}{(1+\exp(-z))^2} e^{-z}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

$$\frac{d\sigma}{dz} = \sigma'(z) = \frac{-1}{(1+\exp(-z))^2} \frac{d\,e^{-z}}{dz}$$

$$= \frac{1}{(1+\exp(-z))^2} e^{-z}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$\boxed{\sigma'(z) = \sigma(z)\,[1-\sigma(z)]}$$

$\hat{y} = \sigma(\dot{w} \cdot x) \rightarrow [1 \quad x_1 \quad .. \quad x_n]$

$J = ?$ $\qquad x \longrightarrow \hat{y}$

y - Ground Truth } Cost

What about least squares? → For classification?

$$J^{LS} = \frac{1}{2}(y - \hat{y})^2$$

0 ↑ Say $\hat{y} = 1$

x	y	$\hat{y}$
	0, 1	[0, 1]

→ Not a good cost function

Cost incurred for misclassification is low

$J = \frac{1}{2}$ 0 0 Say $\hat{y} = 1$

→ Not a good cost function

Cost incurred for misclassification is low

We use Binary Cross Entropy Cost Function

$$J = -\{ y \ln \hat{y} + (1-y) \ln (1-\hat{y}) \}$$

Desirable properties : (1) $J = 0$ if $y = \hat{y}$

(2) J should be very high for misclassification

Consistency → (3) $J \geq 0$

y is either 0 or 1

We use Binary Cross Entropy Cost Function

$$J = -\{ y \ln \hat{y} + (1-y) \ln (1-\hat{y}) \}$$

$\hat{y} \in [0,1]$

Desirable properties :

(1) $J = 0$ if $y = \hat{y}$

(2) J should be very high for misclassification

Consistency → (3) $J \geq 0$

$y = 0$, $\hat{y} = 0$ ← close to 0.

$$J = -\{y \ln \hat{y} + (1-y) \ln (1-\hat{y})\}$$

Desirable properties : (1) $J = 0$ if $y = \hat{y}$ ✓

(2) J should be very high for misclassification

Consistency → (3) $J \geq 0$ ✓

$y = 0$, $\hat{y} = 0$ ← close to 0. $\Rightarrow J \approx 0$

$y = 1$, $\hat{y} \approx 1$ $\Rightarrow J \approx 0$

is low

$\hat{y}$ is either 0 or 1

We use Binary Cross Entropy Cost Function

$$J = -\{ y \ln \hat{y} + (1-y) \ln (1-\hat{y}) \}$$

$\hat{y} \in [0,1]$; $(1-y) \approx 1$; $\ln(1-\hat{y}) \approx 0$

$y = 0$, $\hat{y} \approx 1$

Desirable properties :

(1) $J = 0$ if $y = \hat{y}$

(2) J should be very high for misclassification

Consistency → (3) $J \geq 0$

$y = 0$, $\hat{y} = 0$ ← close to 0. $\Rightarrow J \approx 0$

$y = 1$, $\hat{y} \approx 1$ $\Rightarrow J \approx 0$

Consistency → (3) $J \geq 0$

$y = 0$, $\hat{y} = 0$ ← close to 0. $\Rightarrow J \approx 0$

$y = 1$, $\hat{y} \approx 1$ $\Rightarrow J \approx 0$

$y = 0$, $\hat{y} \approx 1$ $\Rightarrow J \to \infty$

Check $y = 1$, $\hat{y} \approx 0$ $\Rightarrow J \to \infty$

sert Actions Tools Help

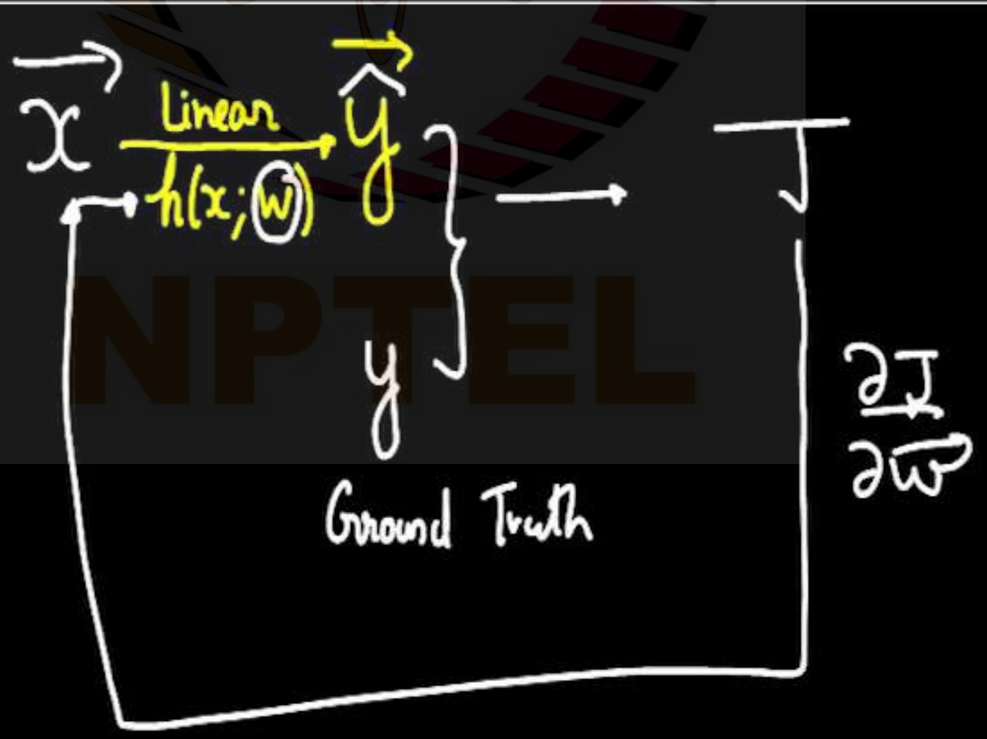
NPTEL

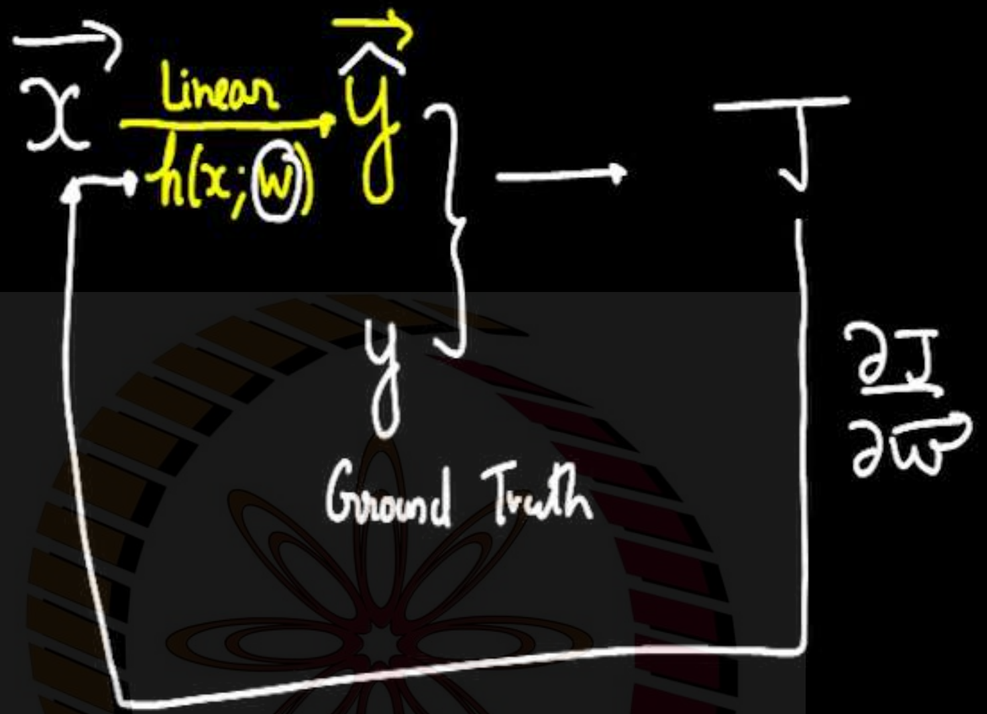
Introduction to

Week 5

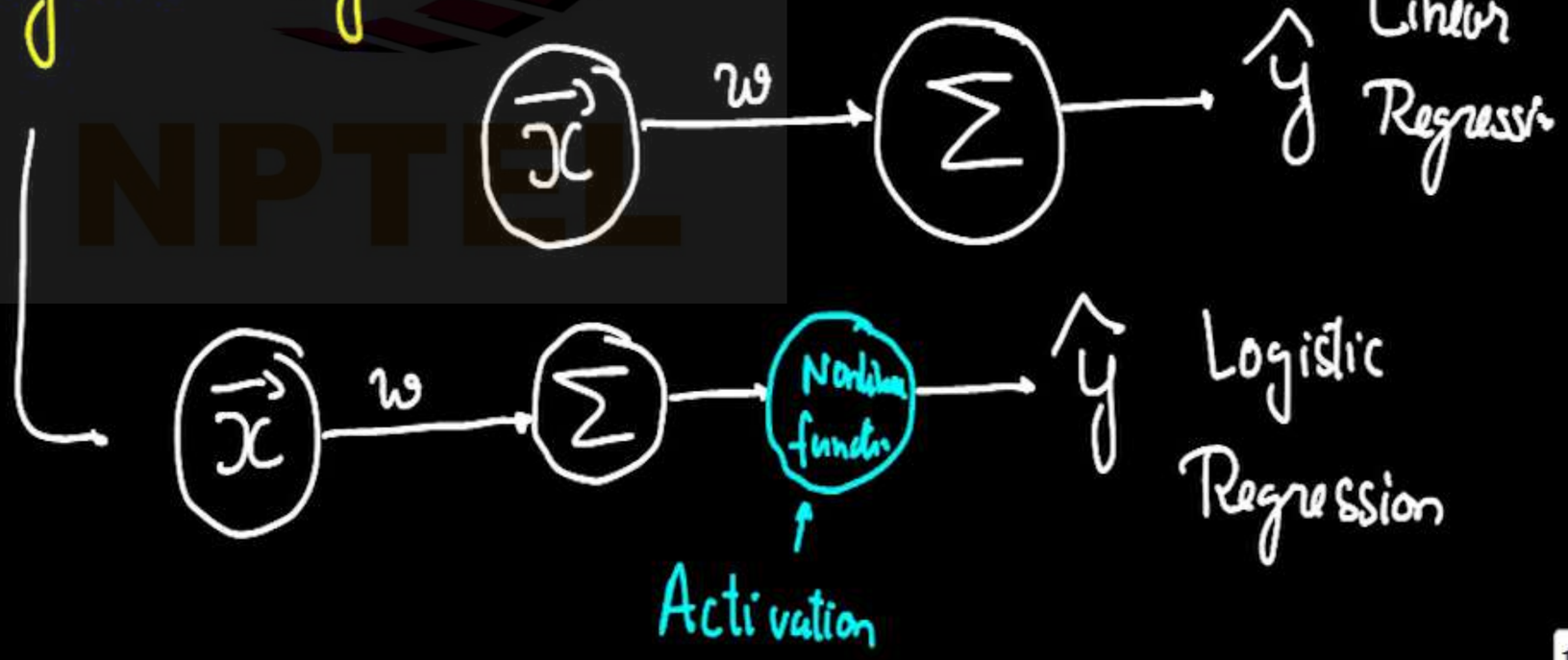
(Deep Learning)

te Title

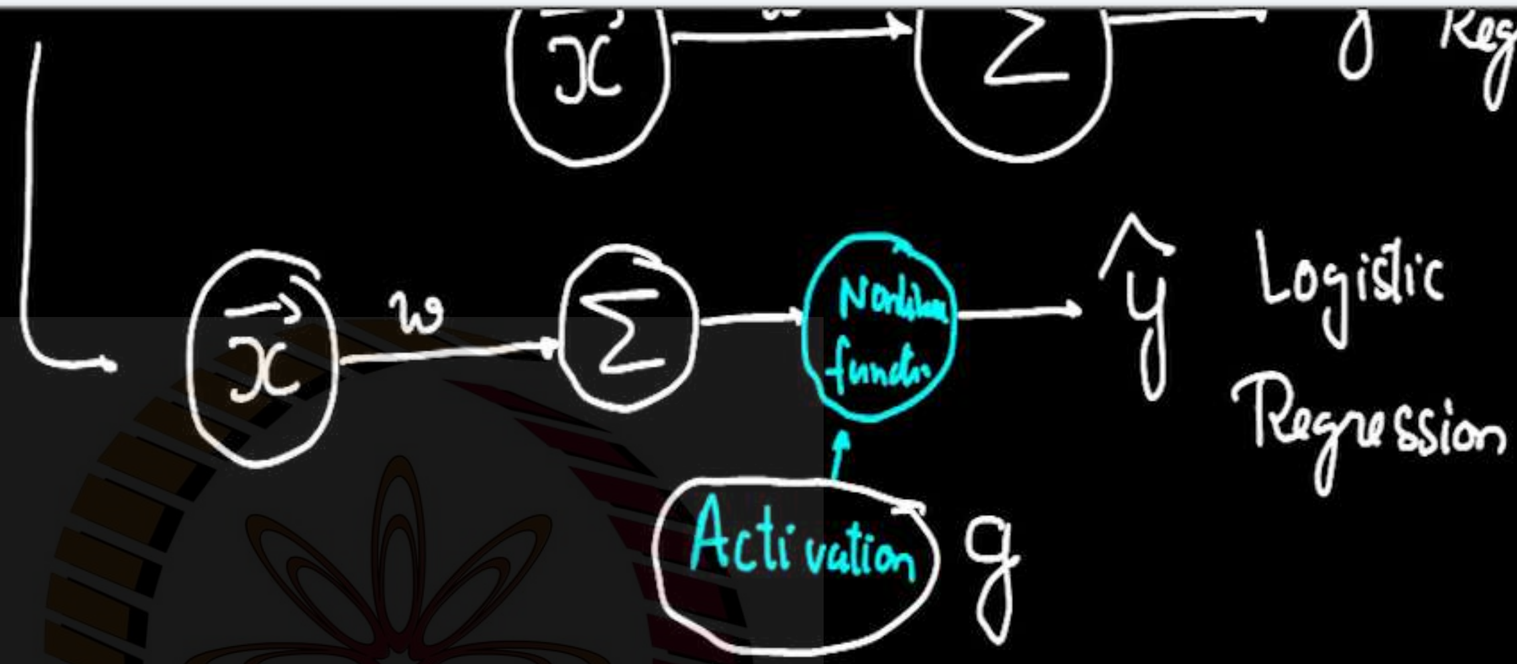


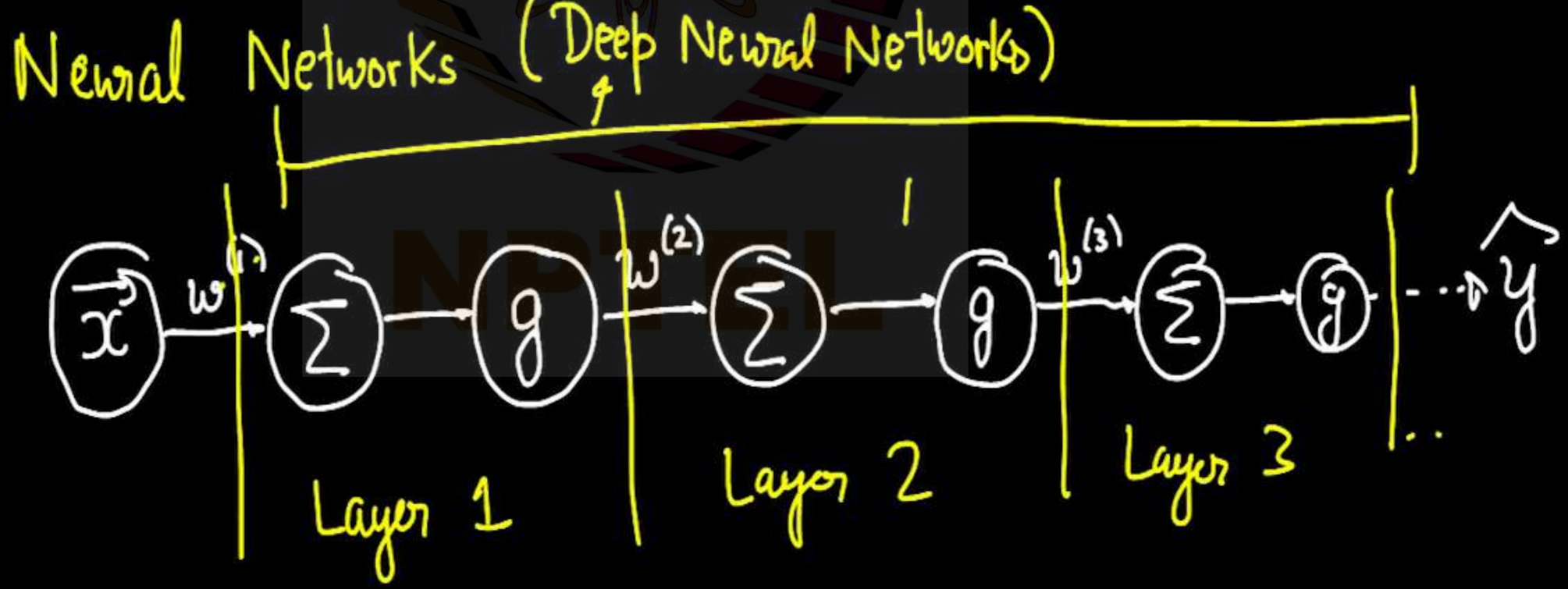


Logistic Regression → Classification



sert Actions Tools Help

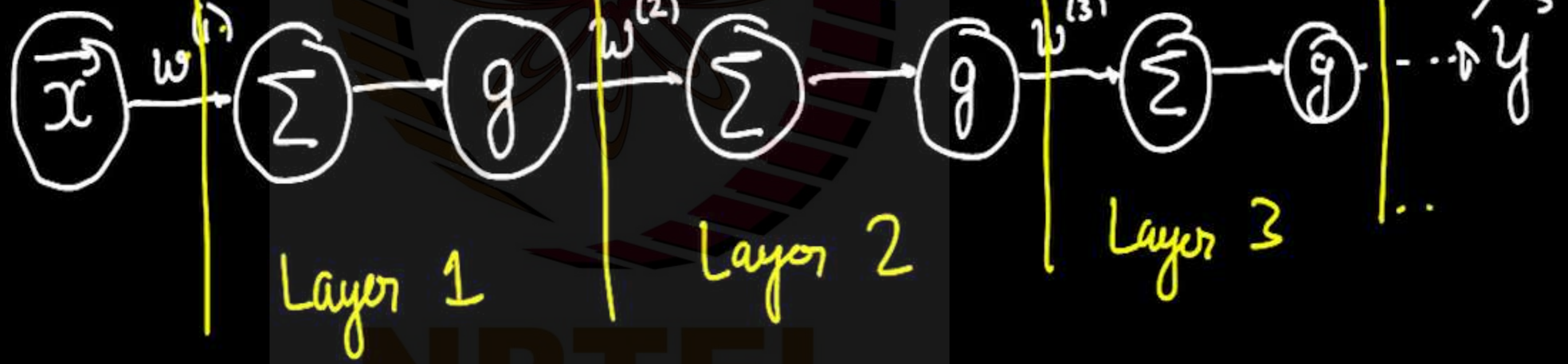


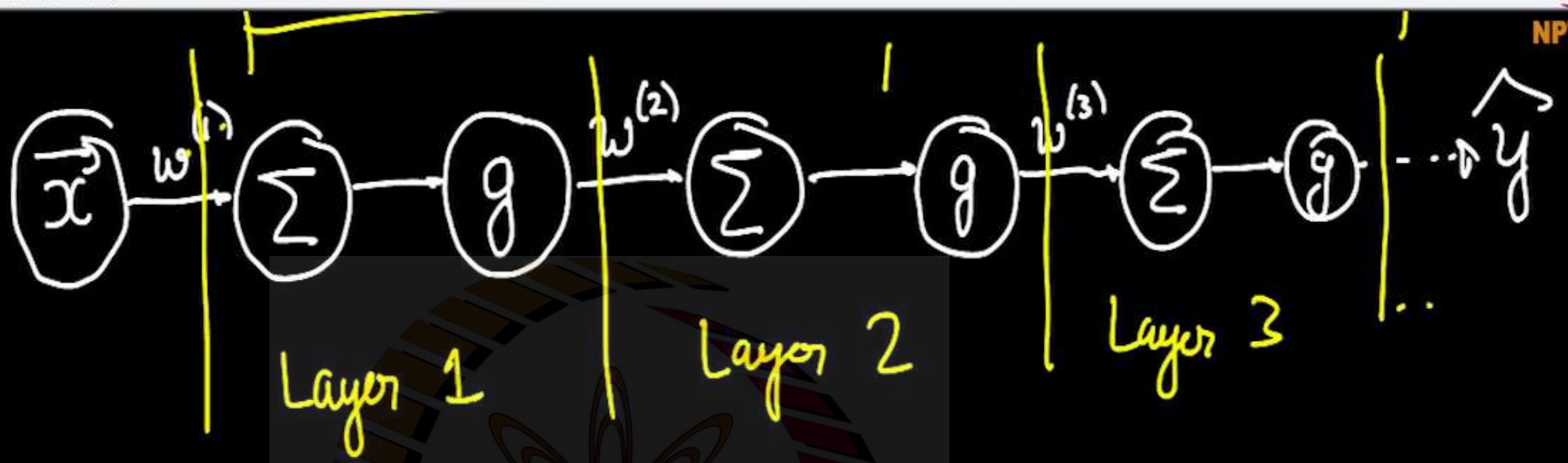


1/1

Activation g

Neural Networks (Deep Neural Networks) — More than one layer → Deep Network





Pay attention to $\hat{y}$

a) How do we characterize the output $y, \hat{y}$ — How do we assign numbers?

$x \xrightarrow[w]{h(x)} \hat{y}$ b) What is the forward model? [Which nonlinear function? g]

$\hat{y}, y \rightarrow J$ c) What is the loss function J?

sert Actions Tools Help

NPTEL

| Layer 1 | Layer 2 | Layer 3

Pay attention to ⌐ How do we assign numbers ?

$\hat{y}$ a) How do we characterize the output $y, \hat{y}$

$x \xrightarrow[w]{h(x)} \hat{y}$ b) What is the forward model ? [Which nonlinear function? g

$\hat{y}, y \to J$ c) What is the loss function J?

$\frac{\partial J}{\partial w}$ d) How do we calculate $\frac{\partial J}{\partial w}$? Gradient

e) How do we use $\frac{\partial J}{\partial w}$ to find better w

Pay attention to

$\hat{y}$ a) How do we characterize the output $y, \hat{y}$ (Representation)

$x \xrightarrow[w]{h(x)} \hat{y}$ b) What is the forward model? [Which nonlinear function? g

$\hat{y}, y \to J$ c) What is the loss function J?

$\frac{\partial J}{\partial w}$ d) How do we calculate $\frac{\partial J}{\partial w}$? Gradient

e) How do we use $\frac{\partial J}{\partial w}$ to find better w } optimization

$$w \leftarrow w - \alpha \frac{\partial J}{\partial w}$$

Logistic Regression

te Title

$K > 2$

Represent $\hat{y}$ ⟶ One hot vector

$K = 3$

$$\hat{y} = \begin{bmatrix} 0.75 \\ 0.1 \\ 0.15 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

sert Actions Tools Help

Represent $\hat{y}$ — One hot vector

$K = 3$

$$\hat{y} = \begin{bmatrix} 0.75 \\ 0.1 \\ 0.15 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

What is the nonlinearity that will achieve classificati?

$K = 3$

$$\hat{y} = \begin{bmatrix} 0.75 \\ 0.1 \\ 0.15 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

What is the nonlinearity that will achieve classificati?

$\vec{x}$ is an image

60×60 grayscale imag

$\begin{bmatrix} 0.15 \end{bmatrix}$ $\begin{bmatrix} 0 \end{bmatrix}$

What is the nonlinearity that will achieve classificatio?

Cat, dog, horse

$\vec{x}$ is an image

60x60 grayscale imag

$\vec{x} \rightarrow \Sigma$

$\begin{bmatrix} x_1 \\ \vdots \\ x_{3600} \end{bmatrix}$

x_1 ... x_n

$\hat{y}$ - 0 0 0

What is the nonlinearity that will achieve classificati?

Cat, dog, horse

$\vec{x}$ is an image

60×60 grayscale imag

x_1

x_n

$\vec{x} \rightarrow \Sigma$

$\hat{y}$ - 0 – prob
0
0

$K > 2$

Represent $\hat{y}$ — One hot vector

$K = 3$

$$\hat{y} = \begin{bmatrix} 0.75 \\ 0.1 \\ 0.15 \end{bmatrix} \begin{matrix} \text{Cat} \\ \text{horse} \\ \text{dog.} \end{matrix}$$

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

What is the nonlinearity that will achieve classificati?

Cat, dog, horse

$\vec{x}$ is an image

60×60 grayscale imag

$$\begin{bmatrix} x_1 \\ \vdots \\ \end{bmatrix}$$

x_1

$\hat{y}$ - 0 — prob
0

What is the nonlinearity that will achieve classification?

Cat, dog, horse

$\vec{x}$ is an image

60×60 grayscale image

$\vec{x} \rightarrow \Sigma$

x_1

x_n

$\hat{y}$ - 0 – prob
0
0

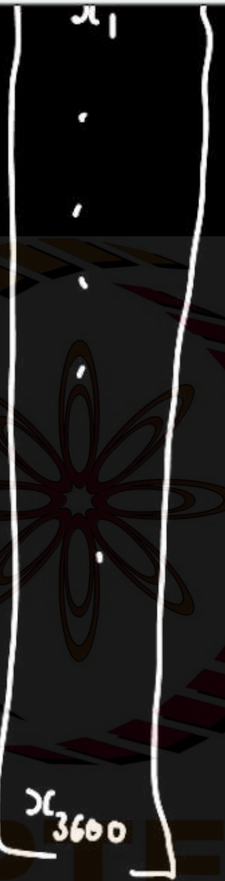
$$\sum_{k=1}^{K} \hat{y}_k = 1$$ – Required

Cat, dog, horse

$\vec{x}$ is an image

60x60 grayscale imag

$\vec{x} \rightarrow (\Sigma)$



x_1

x_n

$\hat{y}$ - 0 - prob
0
0

① $\sum_{k=1}^{K} \hat{y}_k = 1$ → Required

② All $\hat{y}_k \in [0, 1]$

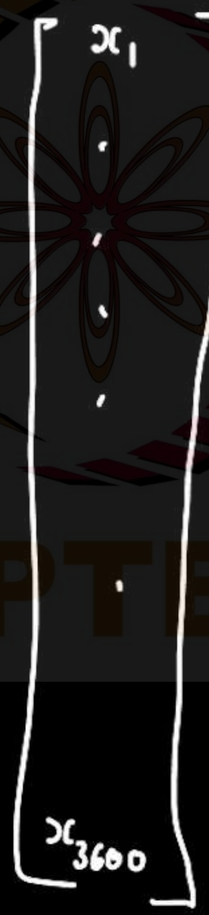
What is the nonlinearity that will achieve classificati?

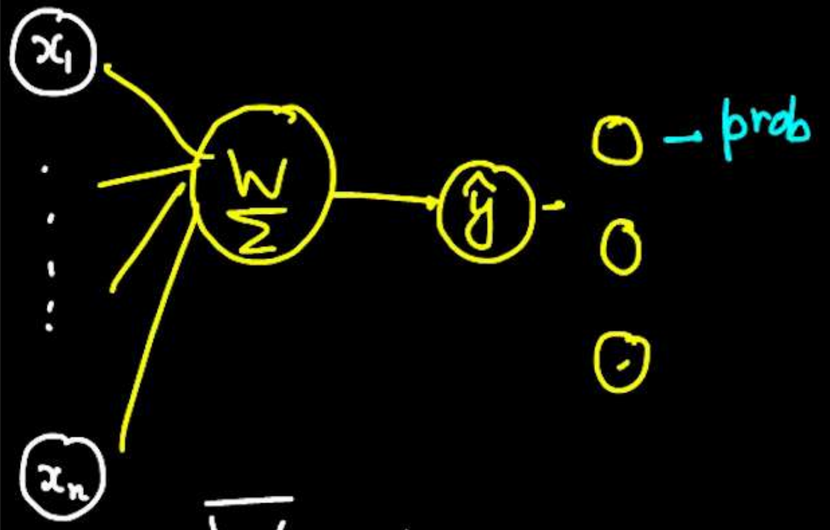
Cat, dog, horse

$\vec{x}$ is an image

60×60 grayscale imag

$\vec{x} \rightarrow \Sigma$



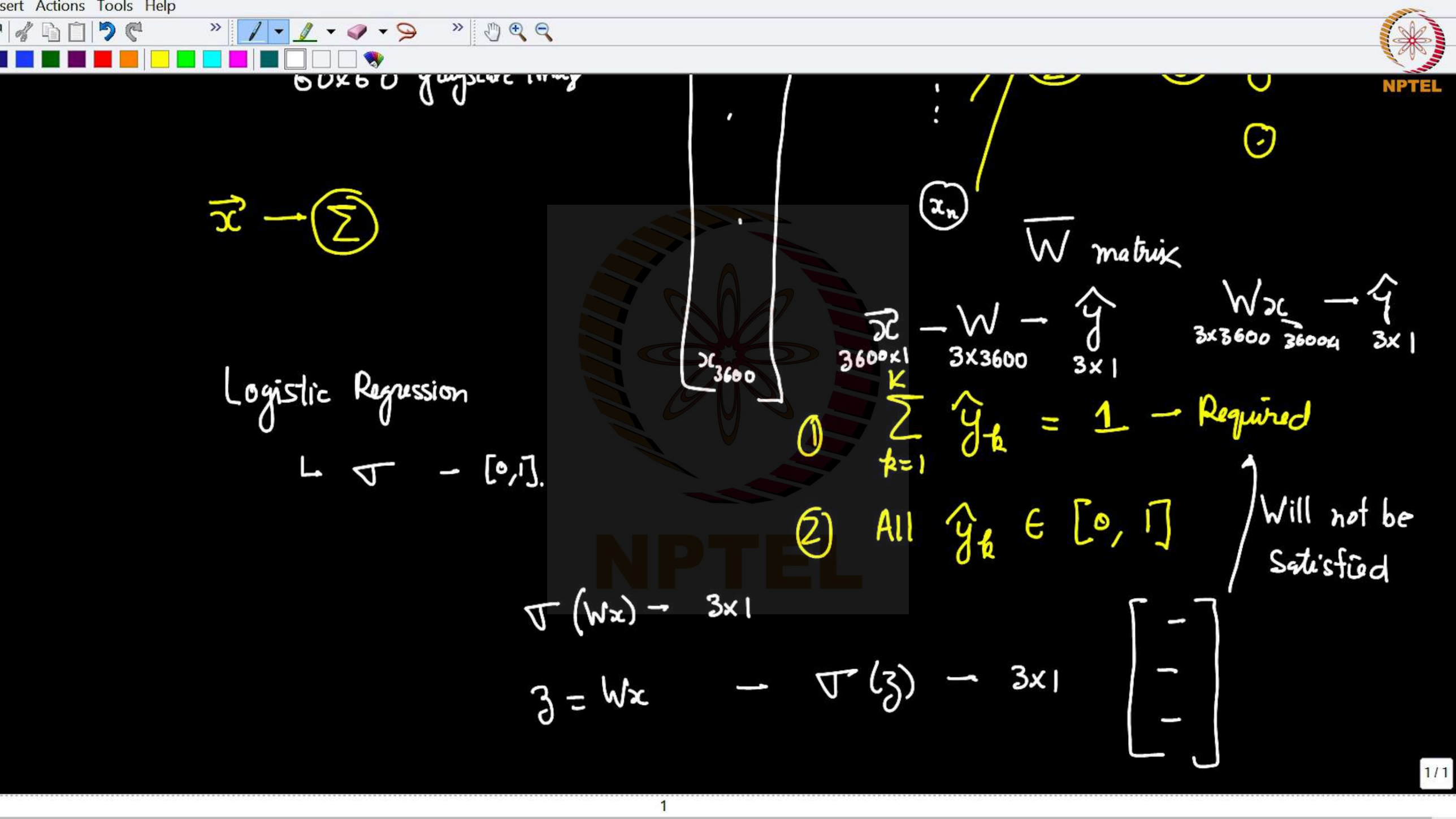


W matrix

$\vec{x}_{3600\times1} - W_{3\times3600} - \hat{y}_{3\times1}$

$W_{3\times3600} \; \vec{x}_{3600\times1} \rightarrow \hat{y}_{3\times1}$

$\sum_{k}^{} \hat{y}_k = 1$ → Required


$\vec{x} \rightarrow \Sigma$
x_{3600}
x_n
$\bar{W}$ matrix
$\vec{x}$ 3600×1 — W 3×3600 — $\hat{y}$ 3×1
Wx 3×3600 3600×1 $\hat{y}$ 3×1
Logistic Regression
σ — $[0,1]$.
① $\sum_{k=1}^{K} \hat{y}_k = 1$ — Required
② All $\hat{y}_k \in [0,1]$
Will not be Satisfied
$\sigma(Wx) \rightarrow 3\times1$
$z = Wx$ — $\sigma(z)$ — 3×1
$\begin{bmatrix} - \\ - \\ - \end{bmatrix}$

$\begin{bmatrix} x_1 \\ \vdots \\ x_{3600} \end{bmatrix}$

$\vec{x}_{3600\times 1} - W_{3\times 3600} - \hat{y}_{3\times 1}$

$W_{3\times 3600}\, x_{3600\times 1} \rightarrow \hat{y}_{3\times 1}$

Logistic Regression

σ — $[0,1]$.

① $\sum_{k=1}^{K} \hat{y}_k = 1$ — Required

② All $\hat{y}_k \in [0,1]$

Will not be Satisfied

$\sigma(Wx) \rightarrow 3\times 1$

$z = Wx \quad - \sigma(z) \rightarrow 3\times 1$

$\begin{bmatrix} - \\ - \\ - \end{bmatrix}$.

Softmax function.

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_{i}^{K} \exp(z_i)}$$

$$\sum_{j=1}^{3} \exp(z_j)$$

$$x \xrightarrow[Wx]{W} z$$

x: 3600×1 $\quad$ z: 3×1

$$\hat{y} = \text{Softmax}\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}\right) = \begin{bmatrix} \text{Softmax}(z_1) \\ \text{Softmax}(z_2) \\ \text{Softmax}(z_3) \end{bmatrix} = \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \end{bmatrix} \frac{1}{D}$$

$$\sum_{j=1}^{3} \exp(z_j)$$

$$x \xrightarrow[Wx]{W} z$$

x: 3600×1, z: 3×1

$$\hat{y} = \text{Softmax}\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}\right) = \begin{bmatrix} \text{Softmax}(z_1) \\ \text{Softmax}(z_2) \\ \text{Softmax}(z_3) \end{bmatrix} = \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \end{bmatrix} \frac{1}{D}$$

$$D = e^{z_1} + e^{z_2} + e^{z_3}$$

$$\sum_{k=1}^{3} \hat{y}_k = 1$$

sert Actions Tools Help

NPTEL

Binary Logistic Regression $K = 2$

Sigmoid

$\hat{y}$ (0 or 1)

$J = -[y \ln \hat{y} + (1-y) \ln(1-\hat{y})]$.

Multinomial " " $K > 2$ $\hat{y}$ (one-hot vector)

Softmax

Binary Logistic Regression $K = 2$ $\hat{y}$ (0 or 1)

$$J = -[y \ln \hat{y} + (1-y) \ln (1-\hat{y})]$$

Sigmoid

Multinomial " " $K > 2$ $\hat{y}$ (one-hot vector)

Softmax

Loss function for $K > 2$?

$$J = -\sum_{k=1}^{K} y_k \ln \hat{y}_k$$

$K = 2$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$\hat{y}_2 = (1 - \hat{y}_1)$ $\qquad$ $y_2 = (1 - y_1)$

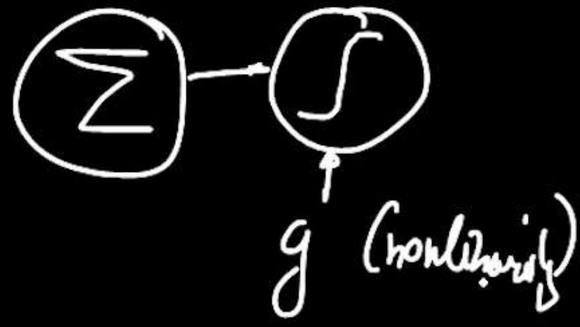
$$J = -\sum_{k=1}^{2} y_k \ln \hat{y}_k = -\left[y_1 \ln \hat{y}_1 + y_2 \ln \hat{y}_2 \right]$$

$$= -\left[y_1 \ln \hat{y}_1 + (1 - y_1) \ln (1 - \hat{y}_1) \right]$$

↑

Binary Cross-Entropy

Multinomial " " $K > 2$ $\hat{y}$ (one-hot vector)



Softmax

General

Loss function for $K > 2$?

$$J = -\sum_{k=1}^{K} y_k \ln \hat{y}_k$$

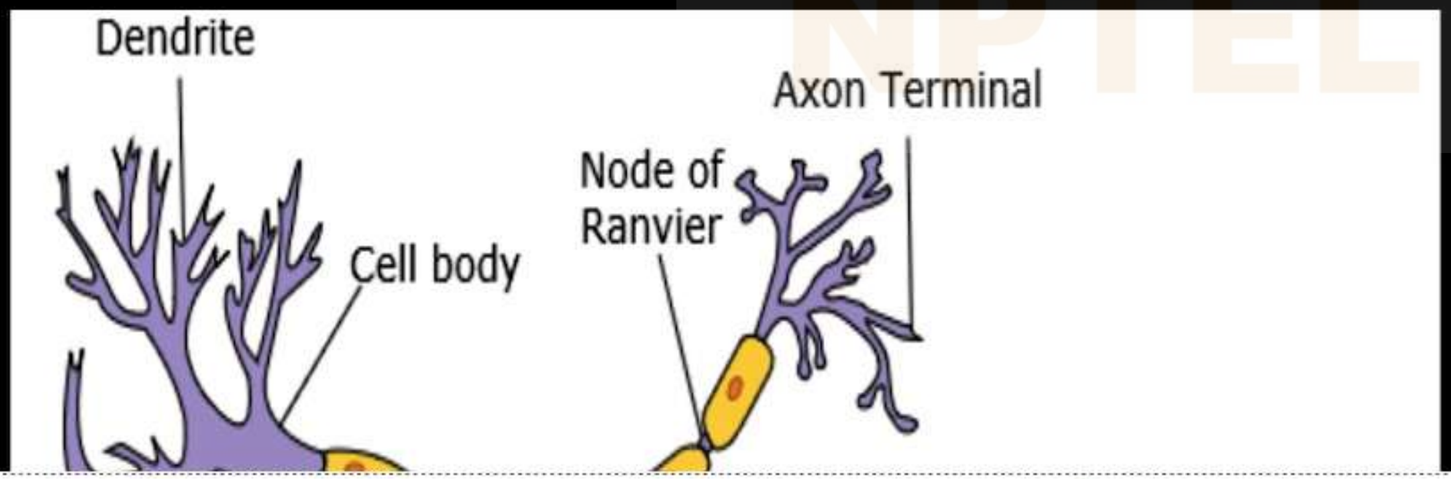
Cross - Entropy Loss function. for $K > 2$.

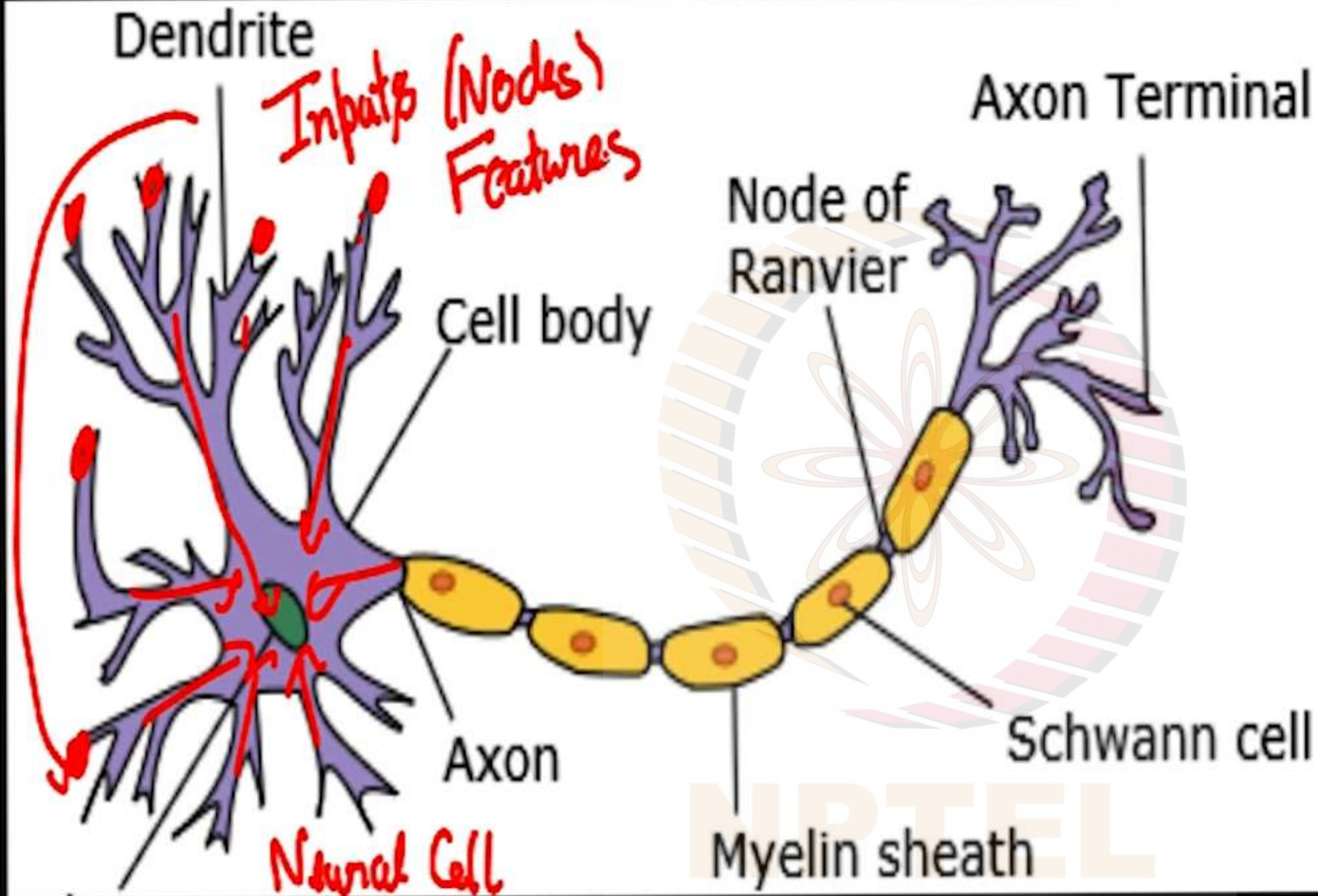
$K = 2$

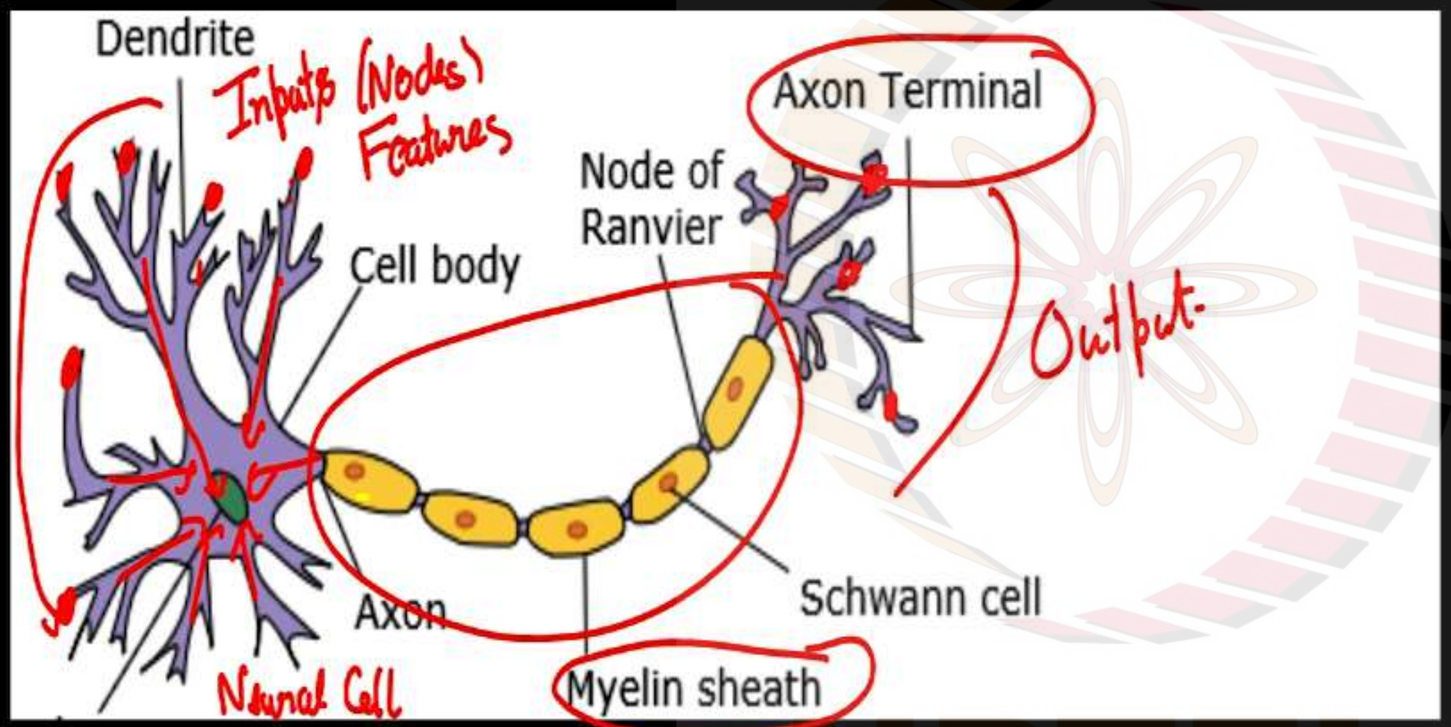
$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

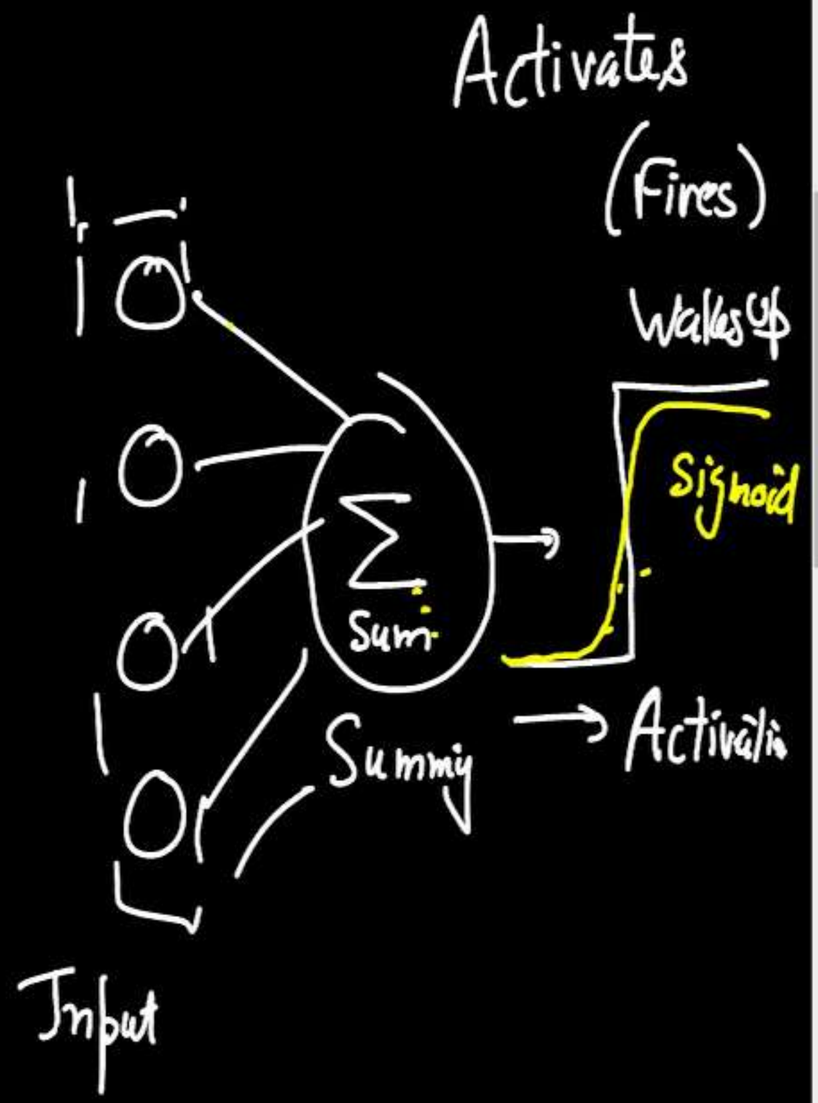
Biological Neuron

Note Title




Dendrite
Inputs (Nodes)
Features
Axon Terminal
Node of
Ranvier
Cell body
Axon
Schwann cell
Myelin sheath
Neural Cell

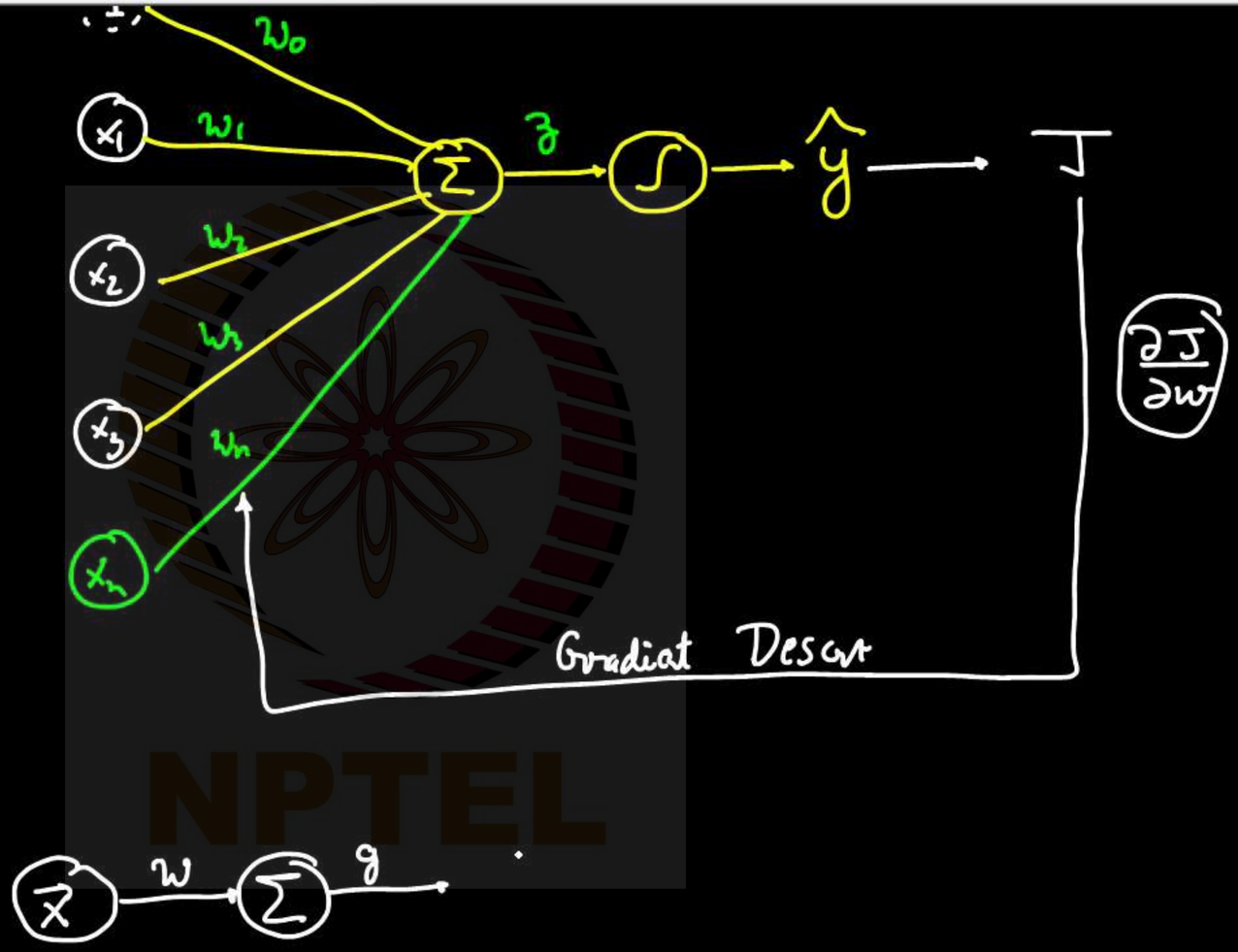

Dendrite
Inputs (Nodes) Features
Axon Terminal
Node of Ranvier
Cell body
Output
Axon
Schwann cell
Neural Cell
Myelin sheath


Activates (Fires)
Wakes up
Sigmoid
Sum
Summig
Activatio
Input

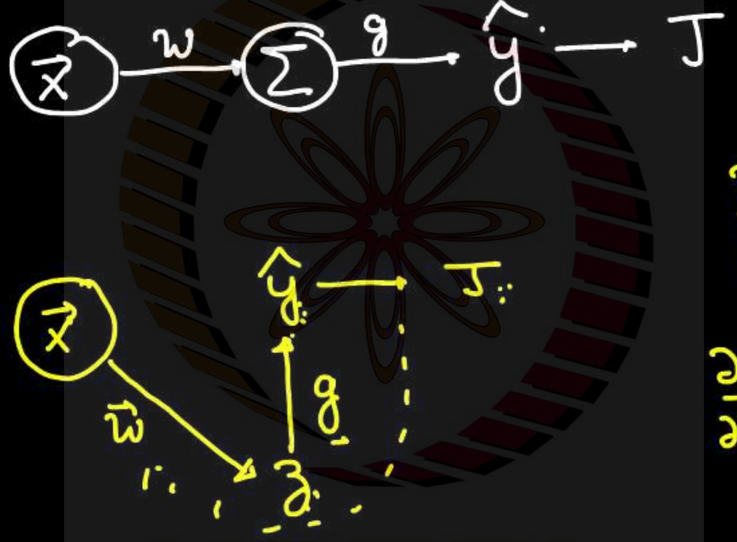
Gradient of Logistic Regression

Note Title

OR, AND, etc → Guessing for weights



Gradiat Descr



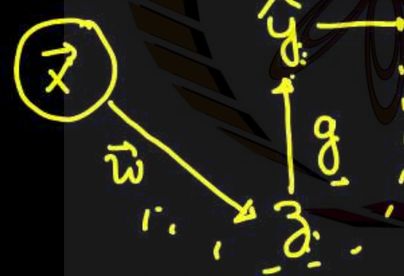
$$\frac{\partial J}{\partial w} = ?$$

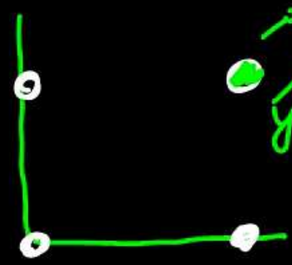
$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w}$$ → Chain Rule.

Gradiat Descr

Ground Truth ↓ Prediction = $h(x;w)$ ↙

Exaple	$\vec{x}$	y	$\hat{y}$	$J^{(i)}$
(1)	$x_1^{(1)}, x_2^{(1)}, \ldots x_n^{(1)}$	$y^{(1)}$	$\hat{y}^{(1)}$	$J^{(1)}$
(2)	$x_1^{(2)}, x_2^{(2)}, x_n^{(2)}$	$y^{(2)}$	$\hat{y}^{(2)}$	$J^{(2)}$
(m)	$x_1^{(m)}, x_2^{(m)}, \ldots x_n^{(m)}$	$y^{(m)}$	$\hat{y}^{(m)}$	$J^{(m)}$





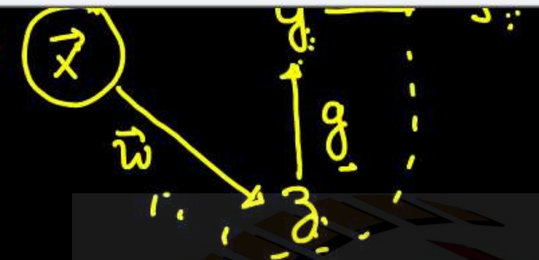
$$\frac{\partial J}{\partial w} = ?$$

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w} \rightarrow \text{Chain Rule}$$

$$J = -\sum_{i=1}^{m} y^{(i)} \ln \hat{y}^{(i)} + (1-y^{(i)}) \ln (1-\hat{y}^{(i)})$$

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

(2)	$x_1^{(2)}, x_2^{(2)}, \ldots x_n^{(2)}$	$y^{(2)}$	$\hat{y}^{(2)}$	$J^{(2)}$
(m)	$x_1^{(m)}, x_2^{(m)}, \ldots x_n^{(m)}$	$y^{(m)}$	$\hat{y}^{(m)}$	$J^{(m)}$



$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w} \rightarrow \text{Chain Rule.}$$

$$J = -\sum_{i=1}^{m} y^{(i)} \ln \hat{y}^{(i)} + (1-y^{(i)}) \ln (1-\hat{y}^{(i)})$$

$$J = \sum_{i=1}^{m} J^{(i)}$$

$$\frac{\partial J}{\partial w} = \sum_{i=1}^{m} \frac{\partial J^{(i)}}{\partial w}$$

$$\frac{\partial J^{(i)}}{\partial w} = \frac{\partial J^{(i)}}{\partial \hat{y}^{(i)}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w}$$

$$\frac{\partial J}{\partial w} = \sum_{i=1} \frac{\partial J^{(i)}}{\partial w}$$

$$\frac{\partial J^{(i)}}{\partial w} = \underbrace{\frac{\partial J^{(i)}}{\partial \hat{y}^{(i)}} \frac{\partial \hat{y}^{(i)}}{\partial z}} \frac{\partial z}{\partial w}$$

$$\hat{y} = \sigma(z)$$

$$J = -\{y \ln \hat{y} + (1-y) \ln (1-\hat{y})\}$$

$$\frac{\partial J}{\partial \hat{y}} = -\left\{\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right\}$$

$$\frac{\partial \hat{y}}{\partial z} = \frac{\partial\, \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

$$= \hat{y}(1-\hat{y}).$$

$$J = -\{y \ln \hat{y} + (1-y) \ln (1-\hat{y})\}$$

$$\frac{\partial J}{\partial \hat{y}} = -\left\{\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right\} \qquad \frac{\partial \hat{y}}{\partial z} = \frac{\partial\, \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

$$= \hat{y}(1-\hat{y})$$

$$\frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} = -\left\{\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right\} \hat{y}(1-\hat{y})$$

$$\frac{\partial J}{\partial z} = -\{y - \hat{y}\} \quad \text{(Check!)}$$

$-$ Error

$$\frac{\partial J}{\partial w} = \sum_{i=1} \frac{\partial J^{(i)}}{\partial w}$$

$$\frac{\partial J^{(i)}}{\partial w} = \frac{\partial J^{(i)}}{\partial \hat{y}^{(i)}} \frac{\partial \hat{y}^{(i)}}{\partial z} \frac{\partial z}{\partial w}$$

$$\hat{y} = \sigma(z)$$

$z = \vec{w} \cdot \vec{x} \rightarrow [1 \; x_1 \; \cdots \; x_n]$.

Includes w_0

$= w^T x = x^T w$.

$$J = -\{y \ln \hat{y} + (1-y) \ln (1-\hat{y})\}$$

$$\frac{\partial J}{\partial \hat{y}} = -\left\{\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right\} \qquad \frac{\partial \hat{y}}{\partial z} = \frac{\partial\, \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

$$= \hat{y}(1-\hat{y})$$

$$\frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} = -\left\{\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right\} \hat{y}(1-\hat{y})$$

$$\frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} = -\left\{ \frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})} \right\} \hat{y}(1-\hat{y})$$

$$\frac{\partial J}{\partial z} = -\underbrace{\{y - \hat{y}\}}_{-\text{Error}} \quad \text{(Check !)}$$

$\frac{\partial z}{\partial \vec{w}}$ where $z = x^T w$.

$$\Rightarrow \boxed{\frac{\partial z}{\partial \vec{w}} = \vec{x}}$$

$$\frac{\partial J}{\partial z} = -\{y - \hat{y}\} \quad \text{(Check!)}$$

$-$ Error

$$\frac{\partial z}{\partial \vec{w}} = \vec{x}$$

$$\Rightarrow \frac{\partial J}{\partial \vec{w}}^{(i)} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial \vec{w}} = -\{y^{(i)} - \hat{y}^{(i)}\}\vec{x}$$

$$= -\text{Error} \cdot \text{Input}$$

Exactly the Same as Linear Reg with Least Squares

$$\frac{\partial J}{\partial \vec{w}} = -\sum_{i=1}^{m} \{y^{(i)} - \hat{y}^{(i)}\}\vec{x}.$$

Logistic Regression with Binary Cross-Entropy

$= -$ Error · Input

Exactly the Same as Linear Reg with Least Squares

$$\frac{\partial J}{\partial \vec{w}_j} = -\sum_{i=1}^{m} \{ y^{(i)} - \hat{y}^{(i)} \} \vec{x}_j^{(i)}$$

Logistic Regression with Binary Cross-Entropy

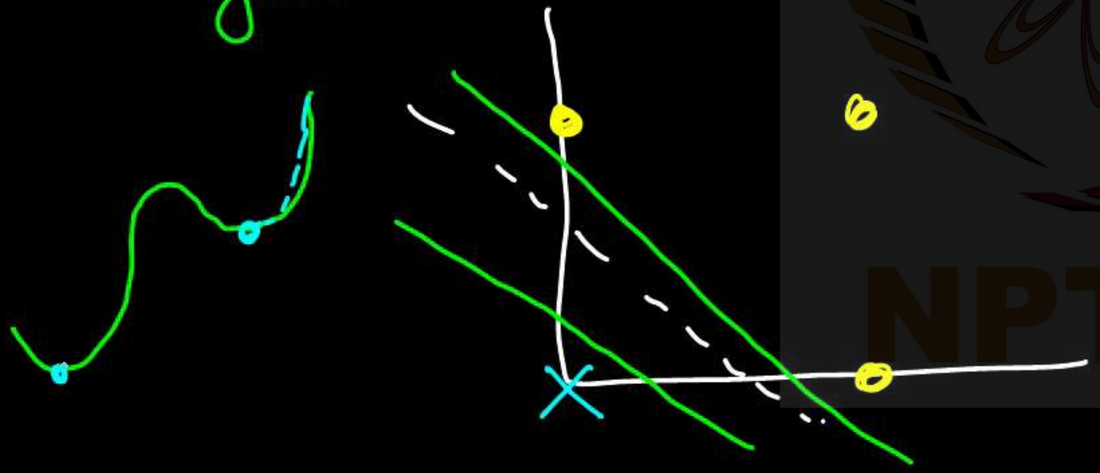
For Logistic Regression, the loss function J is not Convex

i.e. the minimum is, in general, not unique

For Logistic Regression, the loss function J is not Convex

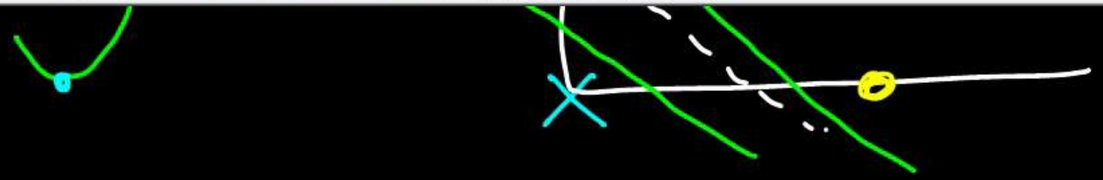
i.e. the minimum is, in general, not unique

Does not necessarily give global optimum



Many possible classifying lines

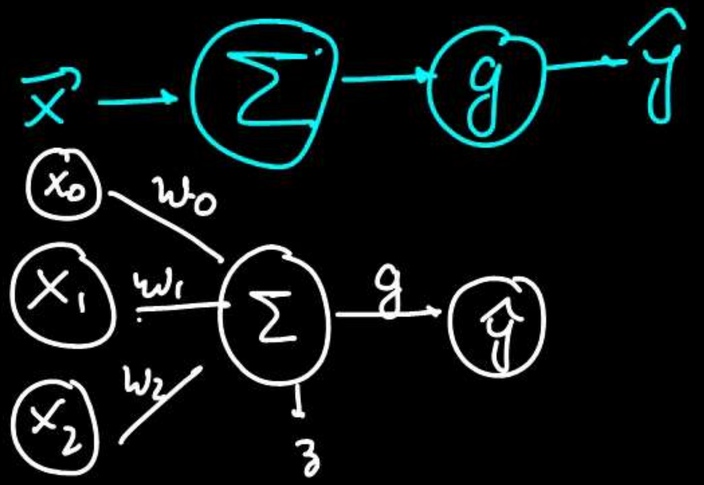
– Logistic Regression does not give unique solutions



- Logistic Regression does not give unique solutions

OR gate

Eqn	$\vec{x}$	y
1	(0,0)	0
2	(0,1)	1
3	(1,0)	1
4	(1,1)	1



$z = \vec{w} \cdot \vec{x}$ } vectorized expression.

$\hat{y} = g(z)$.

x_1^2, x_2^2

Inbox (28) - akshai@nptel.iitm.ac | MATLAB Online R2018b

https://matlab.mathworks.com

HOME | PLOTS | APPS | LIVE EDITOR | INSERT | VIEW

Search Documentation | Balaji

New | Save | Find Files | Go To | Find | Text | Heading 1 | Code | Section Break | Run Section | Run and Advance | Run to End | Run | Step | Stop

FILE | NAVIGATE | TEXT | CODE | SECTION | RUN

/ > MATLAB Drive >

CURRENT FOLDER

Name

- NPTEL
- Published *(my site)*
- GeneralizedLinearRegression.ml
- LinearRegressionPolyFit.mlx
- LinearRegressionSimple.mlx

WORKSPACE

NAME | VALUE | SIZE | CLASS

LogisticModel.m | ORgate.m | GeneralizedLogisticRegression.mlx

Generalized Function for Logistic Regression

This function takes in the data and fits it using linear regression for some given learning rate and stopping criterion

Inputs

x : The input vector/matrix. We assume that x is a vector of size n i.e $x = [x_1, x_2, ..., x_n]$

y: The output vector (Ground truth)

alpha : The learning rate α

ep : The stopping criterion ϵ

Output

w : The weight vector $w = [w_0, w_1, w_2, ..., w_n]$

COMMAND WINDOW

>>

Note1.jnt

Show all

ENG 12:37

NAVIGATE BREAKPOINTS RUN

LogisticModel.m | ORgate.m | GeneralizedLogisticRegression.mlx

```
1 -  clear; close all;
2
3 -  x = [0 0;0 1; 1 0; 1 1];
4 -  y = [0 ; 1 ; 1 ; 1];
5 -  alp = 1 ; ep = 1.e-3;
6 -  w = LogisticModel(x, y, alp, ep);
```

COMMAND WINDOW

>>

TEXT CODE SECTION RUN

LogisticModel.m | ORgate.m | GeneralizedLogisticRegression.mlx

```
function w = GeneralizedLogisticRegression(x, y, alp, ep)
```

Determine size of incoming data

m : Number of examples

n : Number of features

```
[m,n] = size(x);
```

Make initial guess for the w vector $w = [w_0, w_1, w_2, ..., w_n]^T$

```
w = rand(n+1,1);
```

Iterating for w using gradient descent

The hypothesis function is now

COMMAND WINDOW

>>

TEXT CODE SECTION RUN

LogisticModel.m | ORgate.m | GeneralizedLogisticRegression.mlx

The hypothesis function is now

$$\hat{y} = \sigma(w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n) = w \bullet x = \sigma(w^T x)$$

Define the sigmoid first

```
4  sigmoid = @(z) 1./(1+exp(-z));
5
6  for i = 1:m
7      z = [1 x(i,:)]*w; %[1 x] is augmenting x with x0 = 1 to acheive
8      yh(i,1) = sigmoid(z);
9  end
```

Exercise : You can also try out an alternate, vectorized form of the above calculation without the for loop

```
10 J(1) = -1*sum(y.*log(yh)+(1.-y).*log(1.-yh)); %Initial value of J -
```

err is a variable which denotes the appropriate error variable which we wish to

COMMAND WINDOW

>>

TEXT CODE SECTION RUN

LogisticModel.m | ORgate.m | GeneralizedLogisticRegression.mlx

The gradient has (n+1) components. That is $\nabla_w J$ is a (n+1)x1 vector

$$\frac{\partial J}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})$$

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

...

$$\frac{\partial J}{\partial w_r} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})x_r^{(i)}$$

...

$$\frac{\partial J}{\partial w_n} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})x_n^{(i)}$$

```
16      pred_err = yh - y;
```

COMMAND WINDOW

>>

TEXT | CODE | SECTION | RUN

LogisticModel.m | ORgate.m | GeneralizedLogisticRegression.mlx

```
29      err = abs(J(iter)-J(iter-1));
30      %err = norm(alp*DJ);
31
```

Create Plots

```
32      subplot(211)
33       plot(x(:,1),y,'o',x(:,1),yh,'rx');
34      xlabel('x')
35      ylabel('y')
36      legend({'Data','Logistic'},'Location', 'NorthWest')
37      drawnow;
38      subplot(212)
39      plot([1:iter],J([1:iter]),'-o');
40      xlabel('iterations');
41      ylabel('J')
42      drawnow;
43
44
```

COMMAND WINDOW

>>

Show

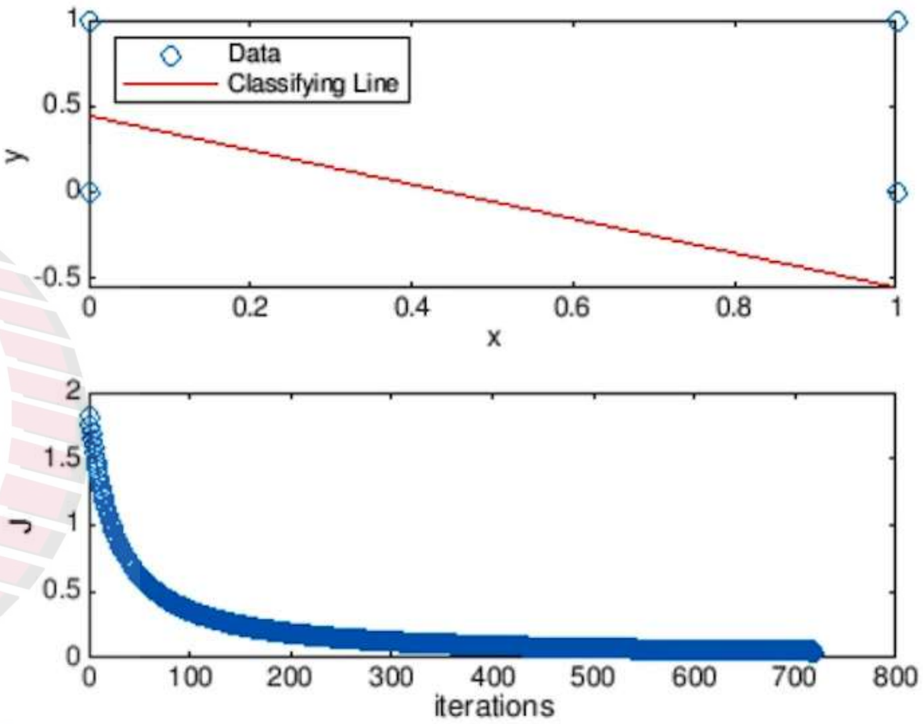
NAVIGATE BREAKPOINTS RUN

NPTEL > Week 4

LogisticModel.m | ORgate.m | GeneralizedLogisticRegressio

```
1 -  clear; close all;
2
3 -  x = [0 0;0 1; 1 0; 1 1];
4 -  y = [0 ; 1 ; 1 ; 1];
5 -  alp = 1 ; ep = 1.e-3;
6 -  w = LogisticModel(x, y, alp, ep);
```

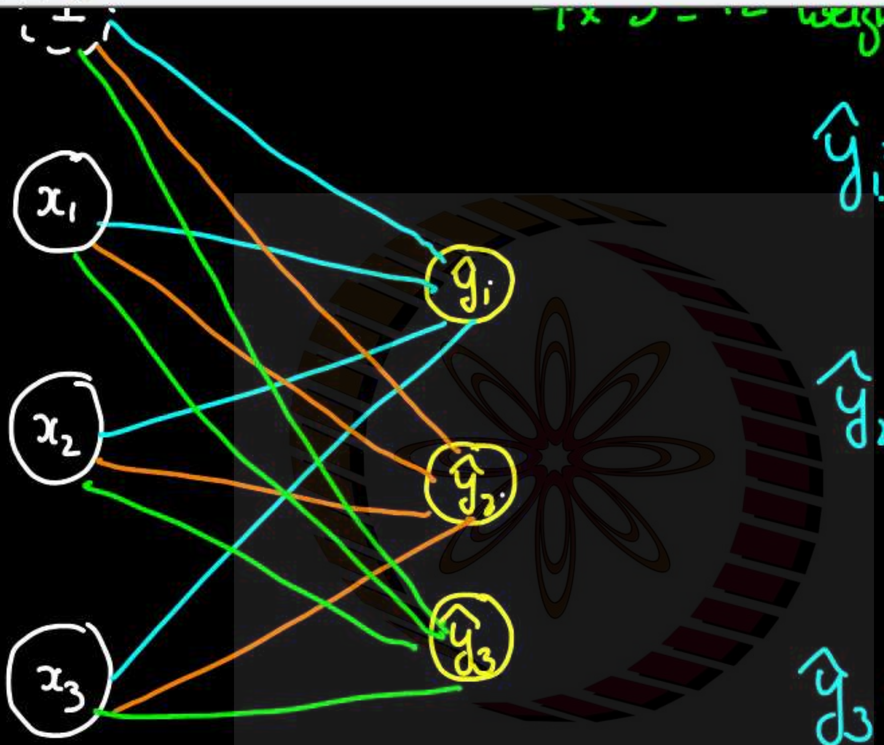
Figure 1



COMMAND WINDOW

```
>> ORgate
```

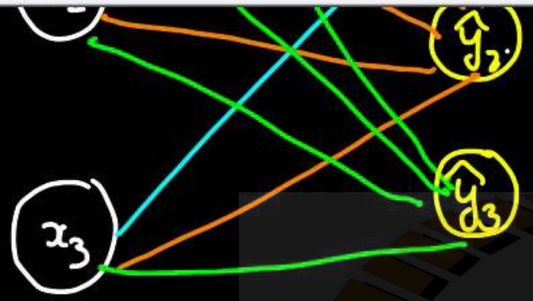
Show



$$\hat{y}_1 = \text{Softmax}\left(w_{01} + w_{11}x_1 + w_{21}x_2 + w_{31}x_3\right)$$

$$\hat{y}_2 = \text{Softmax}\left(w_{02} + w_{12}x_1 + w_{22}x_2 + w_{32}x_3\right)$$

$$\hat{y}_3 = \text{Softmax}\left(w_{03} + w_{13}x_1 + w_{23}x_2 + w_{33}x_3\right)$$



$+ w_{22} x_2 + w_{32} x_3$

$\hat{y}_3 = \text{Softmax}(w_{03} + w_{13} x_1 + w_{23} x_2 + w_{33} x_3)$

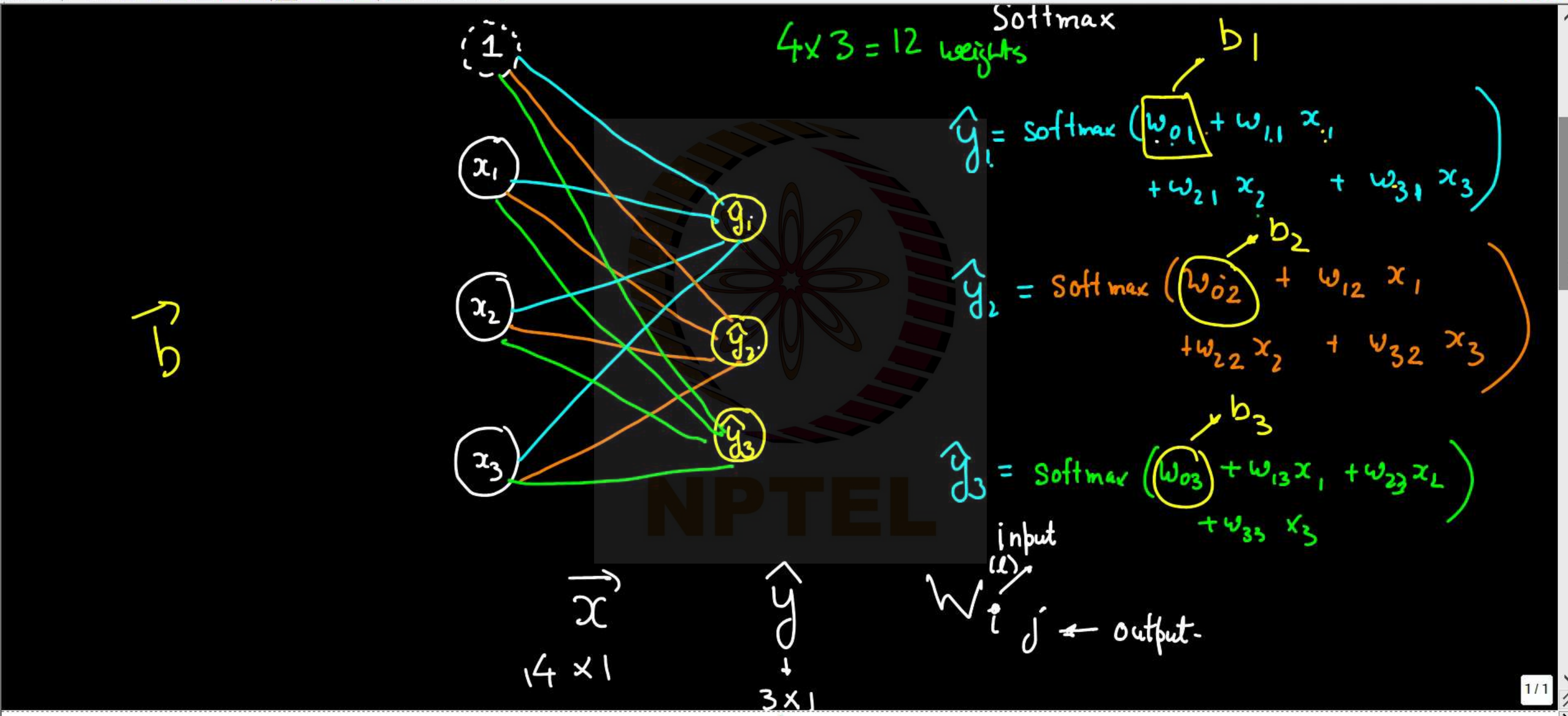
$\vec{x}$ 14×1

$\hat{y}$ 3×1

W (input (1)) i j ← output.

$\hat{y} = W x$

$3 \times 1 \quad 3 \times 4 : 4 \times 1$


Softmax
4x3 = 12 weights
1
x_1
x_2
x_3
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\vec{b}$
b_1
$\hat{y}_1 = \text{Softmax}(w_{01} + w_{11} x_1 + w_{21} x_2 + w_{31} x_3)$
b_2
$\hat{y}_2 = \text{Softmax}(w_{02} + w_{12} x_1 + w_{22} x_2 + w_{32} x_3)$
b_3
$\hat{y}_3 = \text{Softmax}(w_{03} + w_{13} x_1 + w_{23} x_2 + w_{33} x_3)$
$\vec{x}$
14 x 1
$\hat{y}$
3 x 1
$W^{(1)}$
input
i
j ← output

Note4 - Windows Journal

File Edit View Insert Actions Tools Help

NPTEL

Introduction to

Backprop.

Note Title

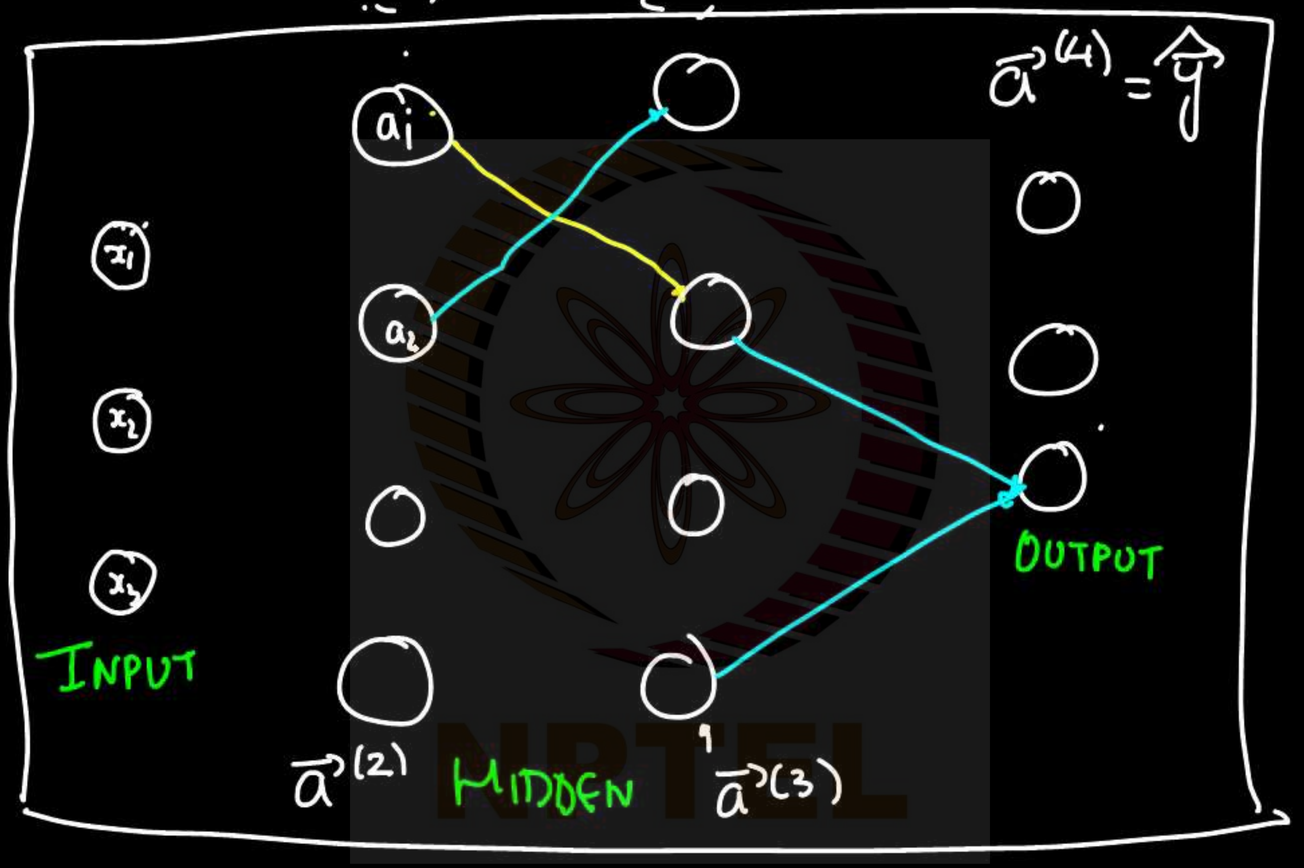
1/1

1

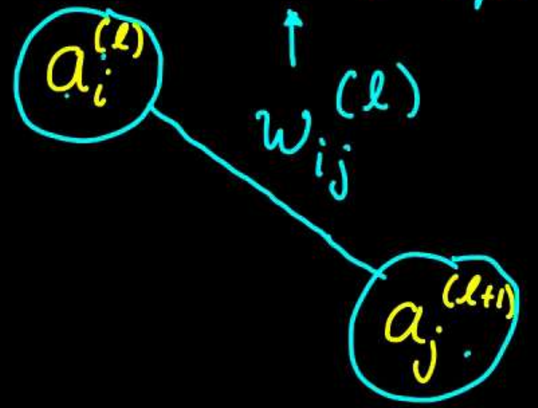
ENG 13:04

Not shown bias units

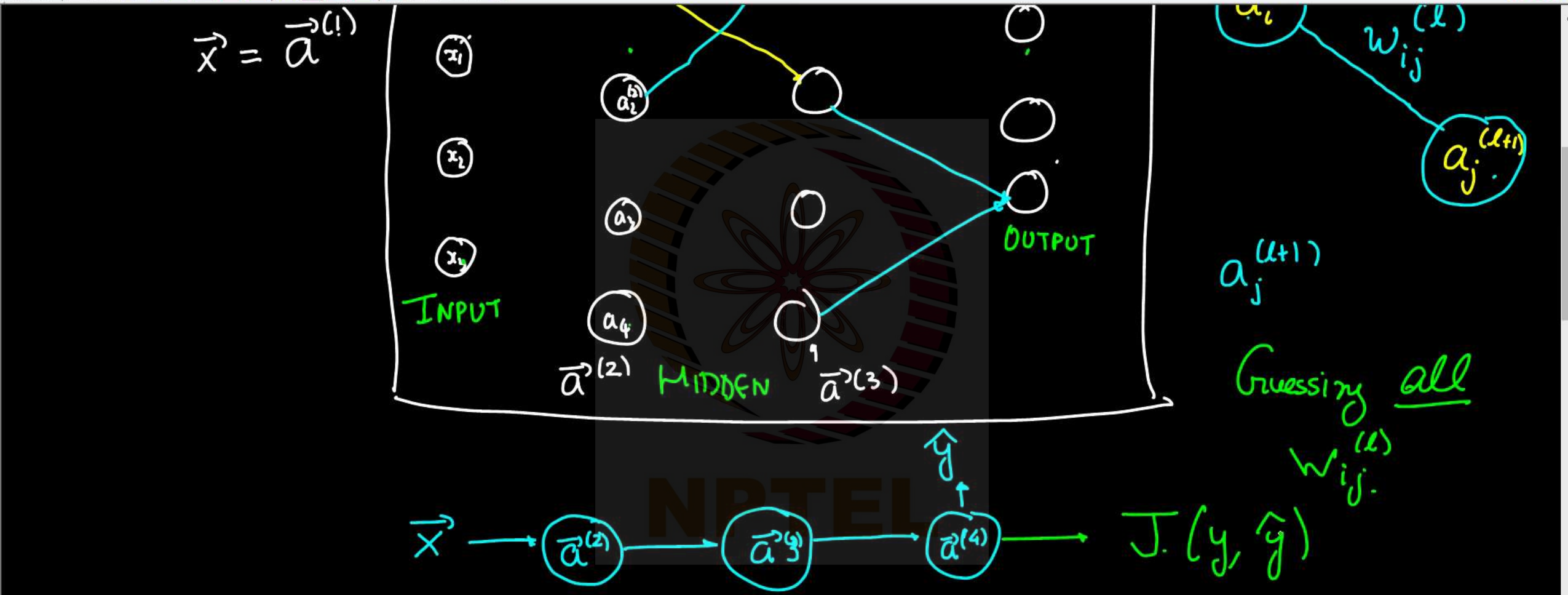
$\vec{x} = \vec{a}^{(1)}$



Weight at the l^{th} layer



$a_j^{(l+1)}$


$\vec{x} = \vec{a}^{(1)}$
INPUT
HIDDEN
OUTPUT
$\vec{a}^{(2)}$
$\vec{a}^{(3)}$
$a_j^{(l+1)}$
$w_{ij}^{(l)}$
Guessing all
$w_{ij}^{(l)}$
$\hat{y}$
$J(y, \hat{y})$

$a_j^{(l+1)}$

Guessing all $w_{ij}^{(l)}$

Redistributy $\vec{w}$ using $\frac{\partial J}{\partial \vec{w}}$ is called Backprop for NN

Redistributing $\vec{w}$ using $\left(\frac{\partial J}{\partial \vec{w}}\right)$ is called Backprop for NN

Simple Method

Coding

① Guess $\vec{w}$ —Forward→ $\hat{y}$. Calculate $J(\vec{w})$

② Guess $\vec{w} + \Delta\vec{w}$ (Perturbation) —Forward→ $\hat{y} + \Delta\hat{y}$. " $J(\vec{w} + \Delta\vec{w})$

$$\frac{\partial J}{\partial w_1} \approx \frac{J(\vec{w} + \Delta\vec{w}) - J(\vec{w})}{\Delta w_1}$$: Finite Difference Method

Expensive

Simple Method

Coding

(1) Guess $\vec{w}$ —Forward→ $\hat{y}$. Calculate $J(\vec{w})$

(2) Guess $\vec{w} + \Delta\vec{w}$ (Perturbation) —Forward→ $\hat{y} + \delta\hat{y}$. " $J(\vec{w} + \delta\vec{w})$

$$\frac{\partial J}{\partial w_1} \approx \frac{J(\vec{w} + \Delta\vec{w}) - J(\vec{w})}{\Delta w_1}$$: Finite Difference Method

Very Expensive

Backprop (Hinton) — Automatic Differentiation

Chain Rule.

$$\frac{\partial J}{\partial w_1} \approx \frac{J(\vec{w} + \Delta\vec{w}) - J(\vec{w})}{\Delta w_1}$$: Finite Difference Method

Very Expensive

Backprop (Hinton) Automatic Differentiation

Chain Rule.

Tensorflow, MATLAB have backprop routines

Very Expensive

Automatic Differentiation

Backprop (Hinton)

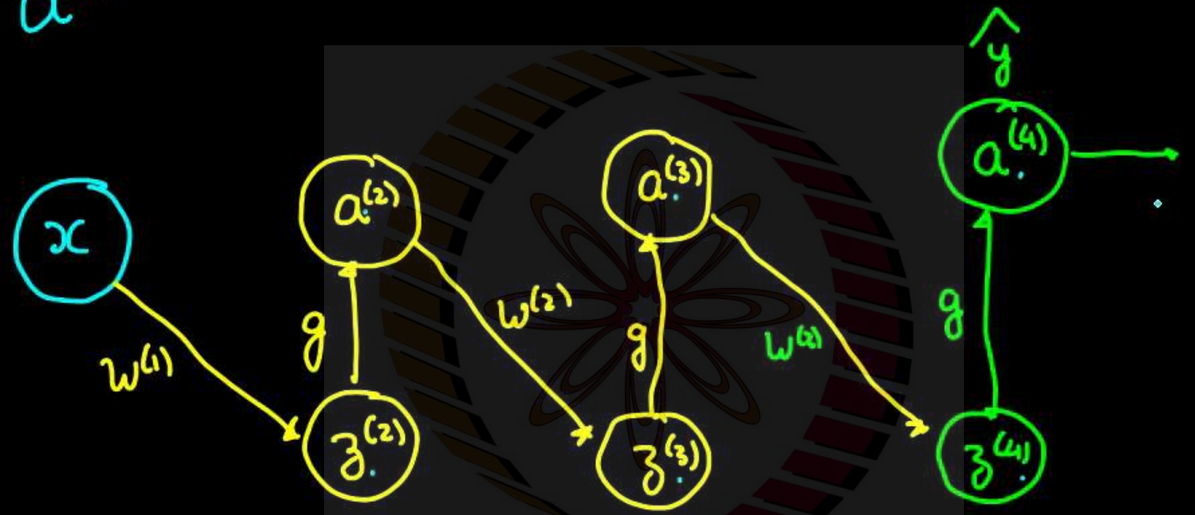
Chain Rule.

Tensorflow, MATLAB have backprop routines

Toy derivation (Simplified derivation)

Ignoring the bias term

$a^{(1)}$



$a^{(1)} = x$

$z^{(2)} = w^{(1)} a^{(1)}$ $\quad a^{(2)} = g(z^{(2)})$

$z^{(3)} = w^{(2)} a^{(2)}$, $\quad a^{(3)} = g(z^{(3)})$

$z^{(4)} = w^{(3)} a^{(3)}$ $\quad a^{(4)} = g(z^{(4)})$

$\hat{y} = a^{(4)}$

$\frac{\partial J}{\partial w^{(1)}} = ?$ $\quad \frac{\partial J}{\partial w^{(2)}} = ?$ $\quad \frac{\partial J}{\partial w^{(3)}} = ?$

$z^{(2)}$ $z^{(3)}$ $z^{(4)}$

$z^{(4)} = w^{(3)} a^{(3)}$ $a^{(4)} = g(z^{(4)})$

$\hat{y} = a^{(4)}$

$J = -\{y \ln \hat{y} + (1-y) \ln(1-\hat{y})\}$

g = sigmoid

$\frac{\partial J}{\partial w^{(1)}} = ?$ $\frac{\partial J}{\partial w^{(2)}} = ?$ $\frac{\partial J}{\partial w^{(3)}} = ?$

$$\frac{\partial J}{\partial w^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial w^{(3)}}$$

$$\frac{\partial J}{\partial z^{(4)}} = -\{y - \hat{y}$$

$\frac{\partial}{\partial w^{(1)}}$ $\frac{\partial}{\partial w^{(2)}}$ $\frac{\partial}{\partial w^{(3)}}$

$$J = -\{y \ln \hat{y} + (1-y) \ln(1-\hat{y})\}$$

g = sigmoid

$$\frac{\partial J}{\partial w^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial w^{(3)}}$$

$\frac{\partial z^{(4)}}{\partial w^{(3)}} \rightarrow a^{(3)}$

$$\frac{\partial J}{\partial z^{(4)}} = -\{y - \hat{y}\}.$$

Denotes Error in activation

$$\frac{\partial J}{\partial z^{(4)}} \equiv \delta^{(4)} \quad ; \quad \frac{\partial J}{\partial z^{(l)}} \equiv \delta^{(l)}$$

$$\frac{\partial J}{\partial w^{(3)}} = -\{y - a^{(4)}\} a^{(3)} \quad ;$$

Error in output activation

$\frac{\partial J}{\partial w^{(3)}}$

$z^{(3)} = w^{(2)} a^{(2)}$

$$\frac{\partial J}{\partial w^{(2)}} = \underbrace{\frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}}}_{\frac{\partial J}{\partial z^{(3)}} \equiv \delta^{(3)}} \frac{\partial z^{(3)}}{\partial w^{(2)}}$$

$\frac{\partial z^{(3)}}{\partial w^{(2)}} \rightarrow a^{(2)}$

$$\frac{\partial J}{\partial w^{(2)}} = \delta^{(3)} a^{(2)} \quad (**)$$

$$\boxed{\frac{\partial J}{\partial w^{(l)}} = \delta^{(l+1)} a^{(l)}}$$

l^{th}

We Know $\delta^{(l)}$ — $\frac{\partial}{\partial w^{(l)}}$

Don't " $\delta^{(3)}$, $\delta^{(2)}$

L — Number of levels.

We Know $\delta^{(L)}$

We need to find other $\delta^{(l)}$ s.

$$\delta^{(4)} = \frac{\partial J}{\partial z^{(4)}}$$

$$= \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}}$$

$$\delta^{(3)} = \frac{\partial J}{\partial z^{(3)}}$$

$$= \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial}$$

We need to find other δ's.

$$\delta^{(4)} = \frac{\partial J}{\partial z^{(4)}}$$

$$= \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}}$$

$$\delta^{(3)} = \frac{\partial J}{\partial z^{(3)}}$$

$$= \underbrace{\frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}}}_{\delta^{(4)}} \boxed{\frac{\partial z^{(4)}}{\partial a^{(3)}}} \frac{\partial a^{(3)}}{\partial z^{(3)}}$$

$W^{(3)}$

$$\delta^{(3)} = \delta^{(4)} w^{(3)} g'(z^{(3)})$$

Exact to Machine precision

$a^{(4)} \xrightarrow{y} \delta^{(4)}$

$\delta^{(4)} = -(y - a^{(4)})$

In general

$$\delta^{(l)} = \delta^{(l+1)} w^{(l)} g'(z^{(l)})$$

Backprop

$$\frac{\partial J}{\partial w^{(l)}} = \delta^{(l+1)} a^{(l)}$$

Chain Rule

All gradients can be computed.

General, vectorized expressions.

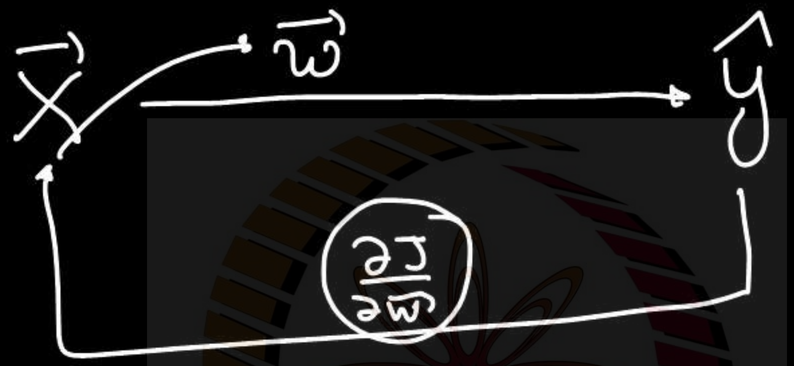
$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \delta_j^{(l+1)} a_i^{(l)}$$

$\frac{\partial J}{\partial w^{(l)}} = \delta^{(l+1)} a^{(l)}$

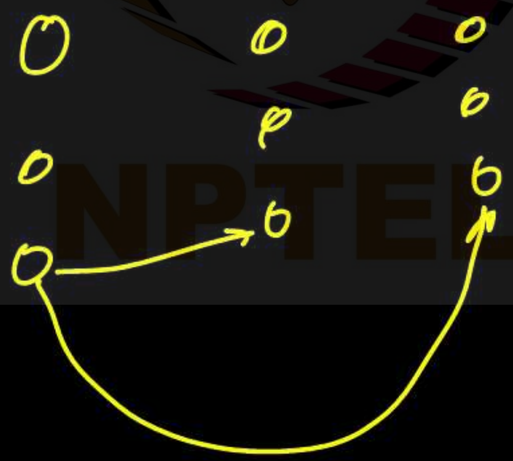
$$\vec{\delta}^{(l+1)} = \left[W^{(l)} \vec{\delta}^{(l)} \right] \odot \left[g'(\vec{z}^{(l)}) \right]$$

Vector = [Matrix · Vector] ⊙ Vector

Hadamard Product (Element wise Multiplication)



Graph



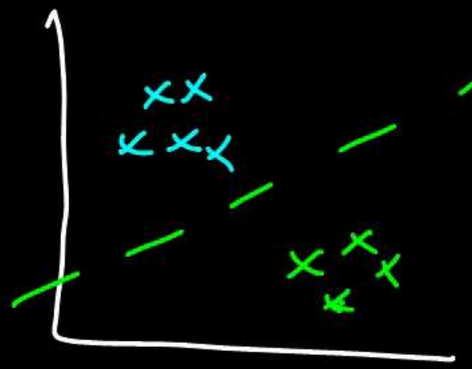
Week 5

Note Title

Linearly Separable Case

Logistic Regression → Two classes

Multinomial " " → K classes



$\hat{y} = \sigma(\vec{w} \cdot \vec{x})$ ← J = Binary cross

Softmax $(\vec{w} \cdot \vec{x})$

$$\sum_{i=1}^{m} -\{y - \hat{y}\} x$$

Multinomial " " ↳ K classes

$\hat{y} = \sigma(\vec{w} \cdot \vec{x})$ ⟵ J = Binary cross

$\sum_{i=1}^{m} -\{y - \hat{y}\} x$

Softmax $(\vec{w} \cdot \vec{x})$

XOR ⟶ Extra Layer. ⟶

↓

Universal Approx Thm.

Neural Networks ⟶ More than one hidden layer

'Deep' Learning > 1 " "

$i=1$

Softmax ($W \cdot x$)

XOR — Extra Layer. → (x) → (y)

Train Deep Networks

↓

Universal Approx Thm.

Neural Networks — More than one hidden layer

'Deep' Learning > 1 " "

No Training.

Saturate /

Back prop

$\delta^{(3)} \approx 0.1\, \delta^{(4)}$

$\delta^{(2)} \approx 0.1\, \delta^{(3)}$

$\delta^{(l+1)} = \delta^{(l)} W^{(l)} g'(z^{(l)})$

Small

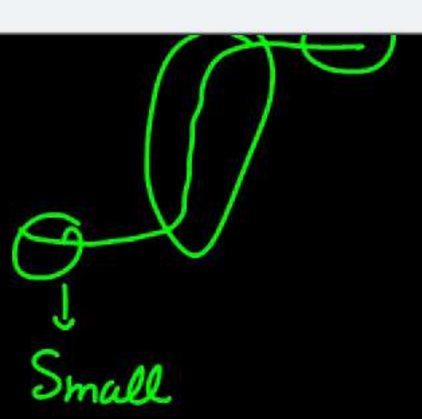
Saturate !

Back prop

$\delta^{(3)} \approx 0.1\,\delta^{(4)}$

$\delta^{(2)} \approx 0.1\,\delta^{(3)}$

$\delta^{(l+1)} = \delta^{(l)} W^{(l)} g'(z^{(l)})$



Did not Cover

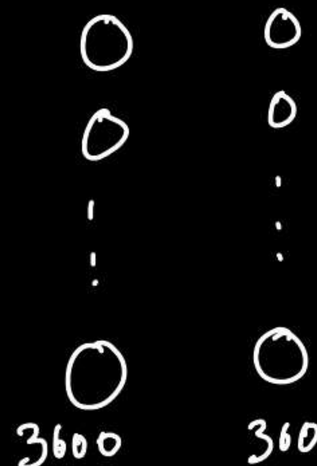
1. How do we initialize $\vec{W}$?
2. How do we determine # of layers # of neurons — Hyperparameters
3. Nonlinearity? Sigmoid, tanh, Linear, ReLU

Hyperparameters

(1) How do we initialize $\vec{w}$?

(2) How do we determine # of layers # of neurons

(3) Nonlinearity? Sigmoid ∇ tanh ReLU Linear



$(3600)^2$ weights

Convolutional Neural Networks (CNNs)

ANNs for vision problems