NN- lecture 4

N-layer Neural Networks

→ Non-linear decision boundary

→ Activation function

→ Forward Prog. N-layer.

(Optional) → Math code for N-layer

→ Backprogation code for N-layer.

Softmax Classifier - Layers.

Activation of layer = 0

a_{i1}^{0}

a_{i2}^{0}

x_{i1}

x_{i2}

W_{11}, W_{21}, W_{12}, W_{22}, W_{13}, W_{23}

b_1, b_2, b_3

f_1^{1} → $\hat{y}_1^{1} = a_1^{1}$

f_2^{1} → $\hat{y}_2^{1} = a_2^{1}$

f_3^{1} → $\hat{y}_3^{1} = a_3^{1}$

L-0

L-1

— NN layer.

Represent ⇓ Superscript.

L-1

a_1^{1}

a_2^{1}

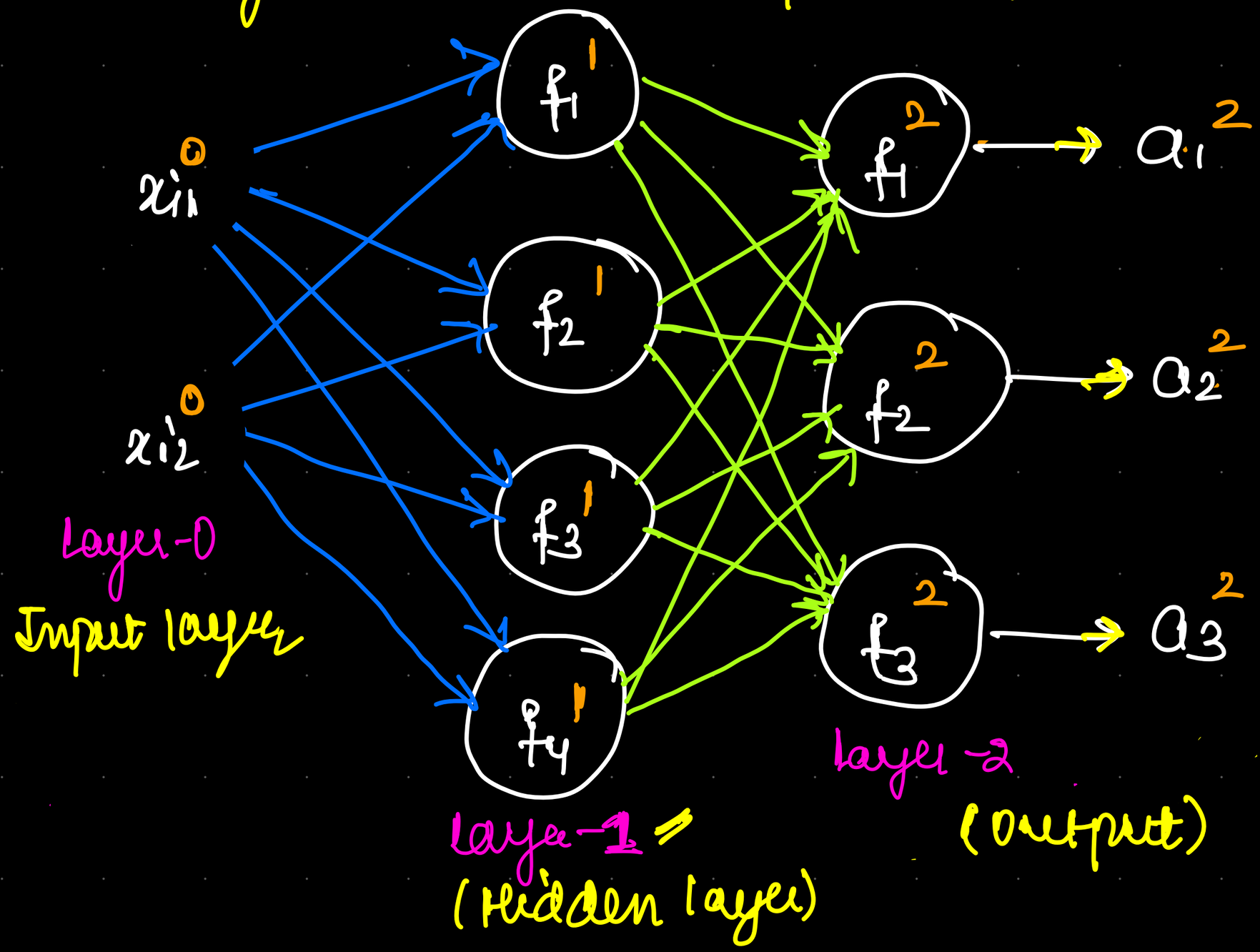
a_3^{1}

Output = Activation

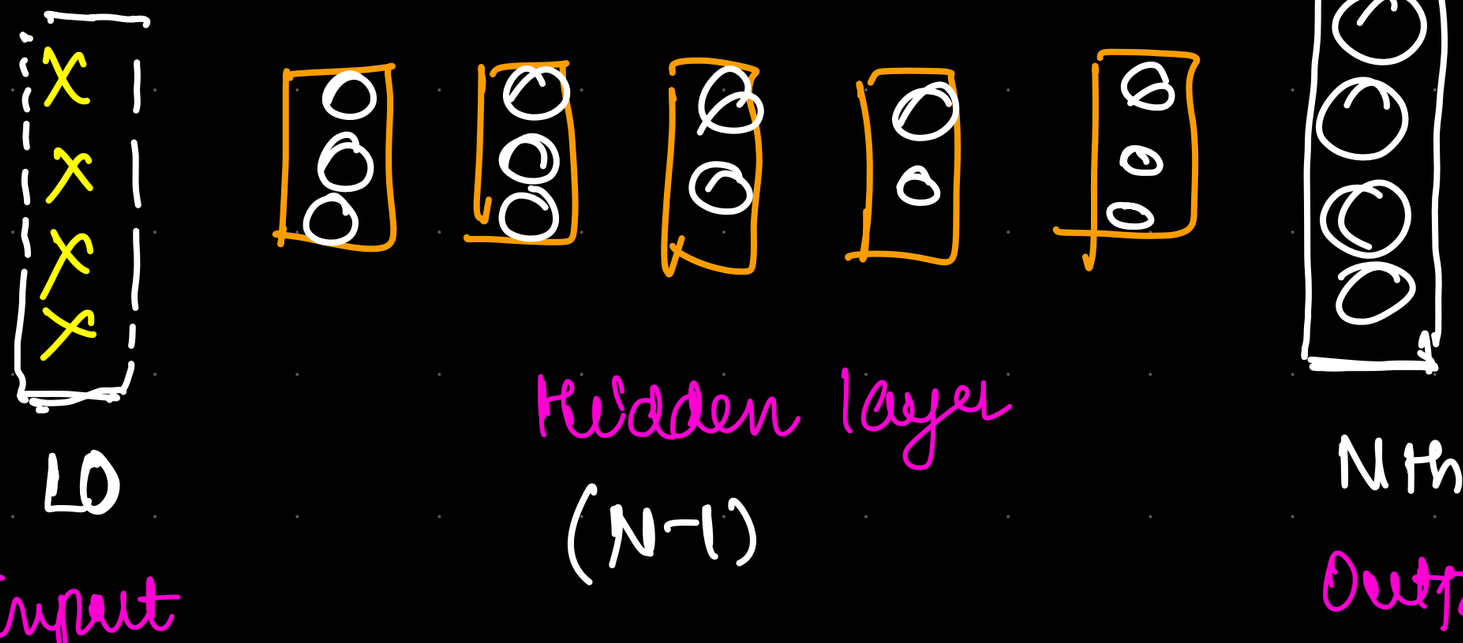
Output last L = Activation / $\hat{y}$

Multi-layer NN (Perceptron)

N+1 layer -



N layer NN.



$$0 + (N-1) + (1) = \text{N-1 layer NN}$$

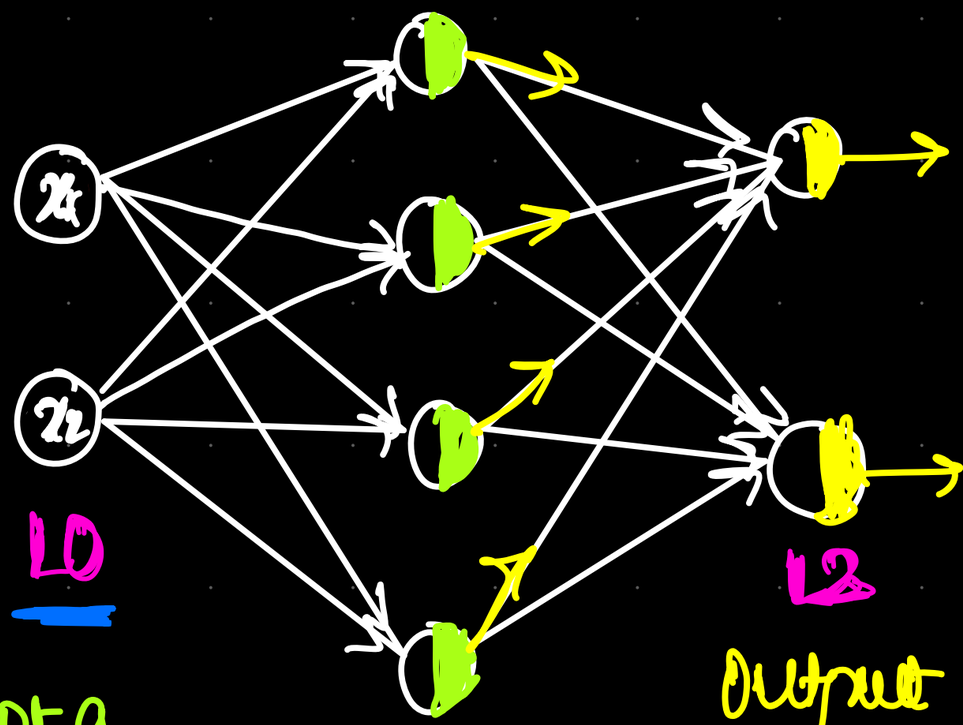
Hidden Output

What will be the activation of L-1?

f(x) Activation Function

Activation

Softmax



Not a layer ✗

?????

Output Classification

Softmax.

Drop. Non-linear function.

Suggestion - Step Function. Sigmoid. ReLU, L.ReLU, tanh.

Why not a linear function?

$f(x) = 2x + 1$

$g(x) = 3x + 2$

100 layers NN
≡ 1 layer NN. ✗

$g(f(x)) = 3(2x+1) + 2$

$= 6x + 3 + 2$

$= \boxed{6x + 5}$

Composition of two linear function is a linear function

Meant using 1-NN.

Why a non-linear function?

$$f(x) = 2x + 1$$

$$g(x) = x^2 + 1$$

$$gf(x) = 2(2x+1)^2 + 1$$

$$= 2(4x^2 + 1 + 4x) + 1$$

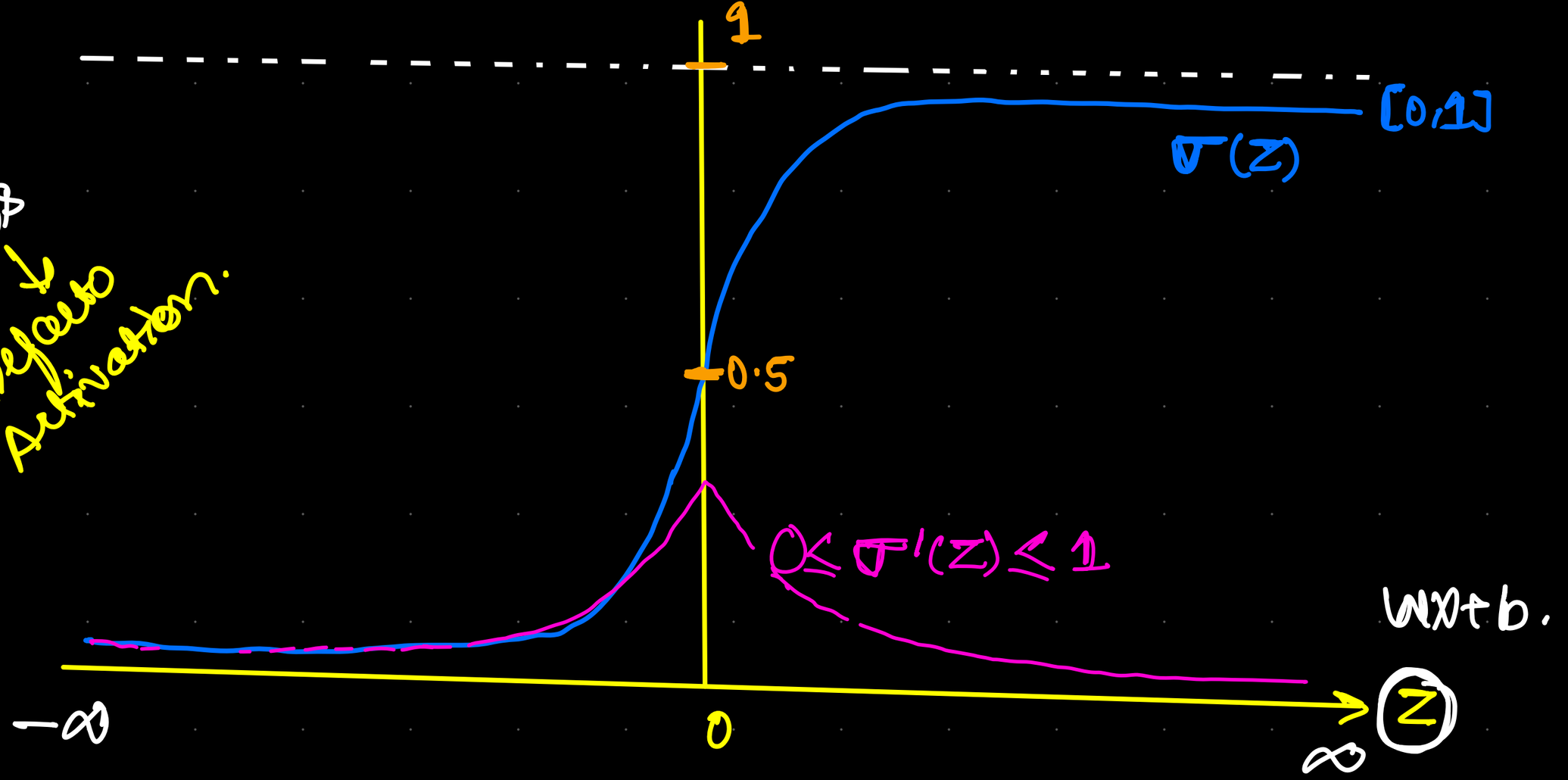
$$= 8x^2 + 2 + 8x + 1$$

$$= \boxed{8x^2 + 8x + 3} \longrightarrow \text{Non-Linear func.}$$

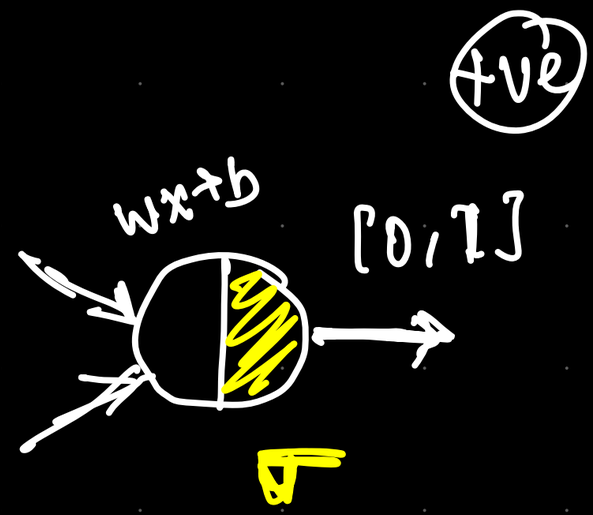
Activation function - Sigmoid ★★★

Adds non-linear

1970s
1980s
↓
Default
activation.



Sigmoid :



Logistic Regression

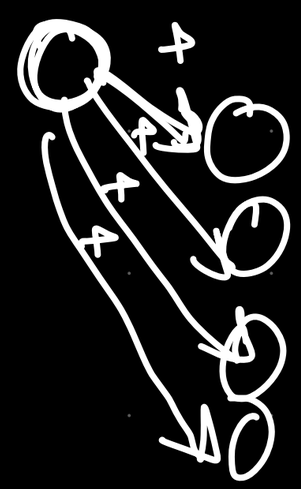
Output of Binary classifier

$Z \longrightarrow p$

for converting z into prop.

$[-\infty, \infty] \longrightarrow [0, 1]$

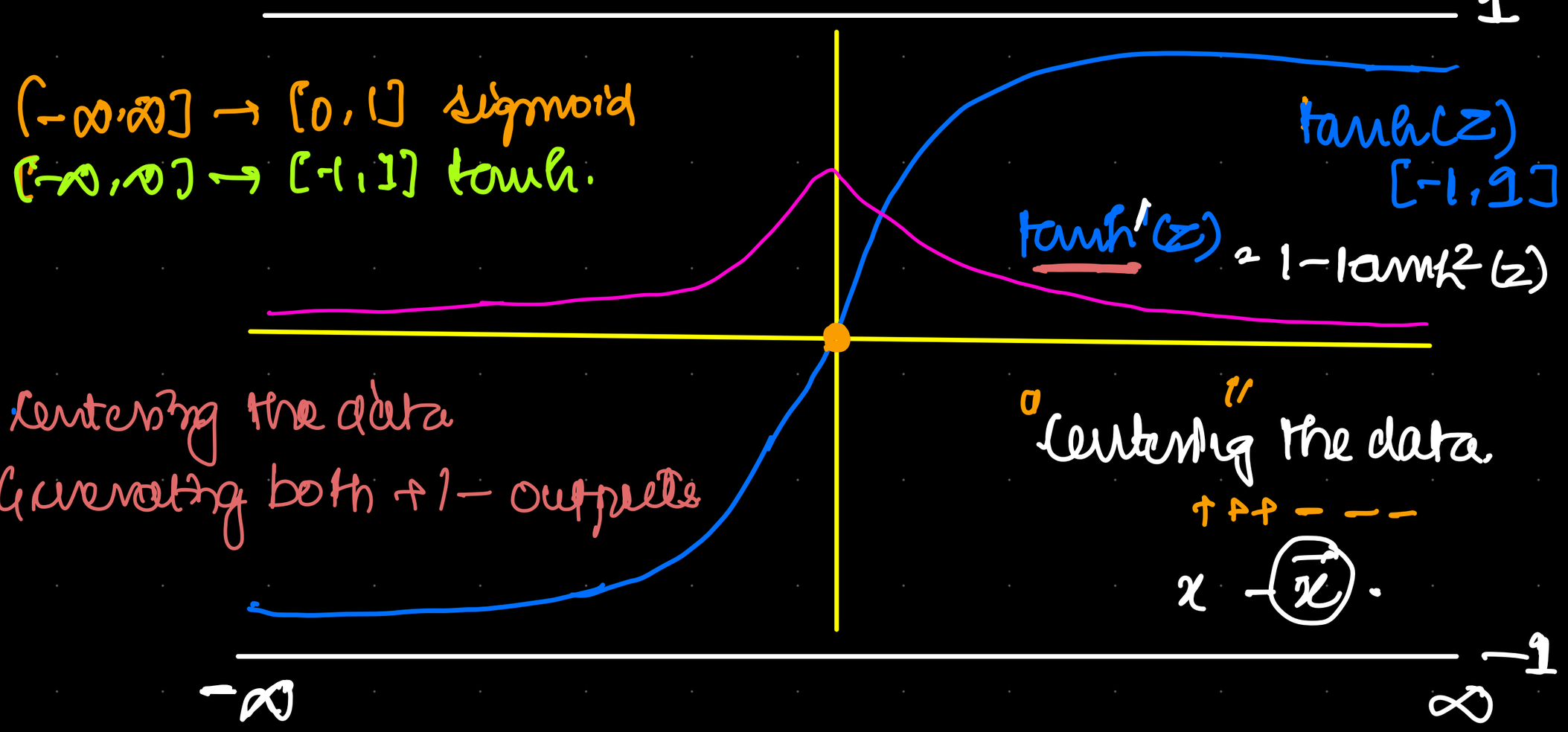
"Non-Linearity" to the system



Activation function - tanh

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$ ...

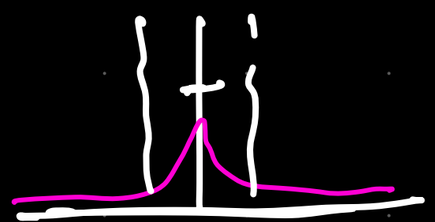
why similar/better?



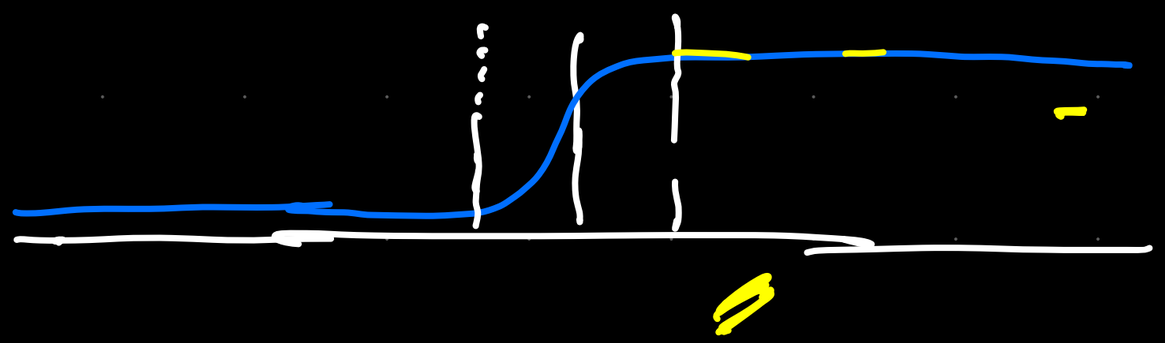
Issue with sigmoid / tanh

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial p} \quad \frac{\partial p}{\partial z} \quad \frac{\partial z}{\partial w}$$

$[0,1]$ $[0,1]$ $[0,1]$



- All derivatives $0 \leq \partial < 1$



For most $z \rightarrow$ ∂ will $\rightarrow 0$

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

Very small

VANISHING GRADIENT

Dynamic in a very small range $g \rightarrow 0$ for most values z

$0.0004 \times 0.004 \times 0.04 \times 0.4 = 0$

Ideal properties of activation function:

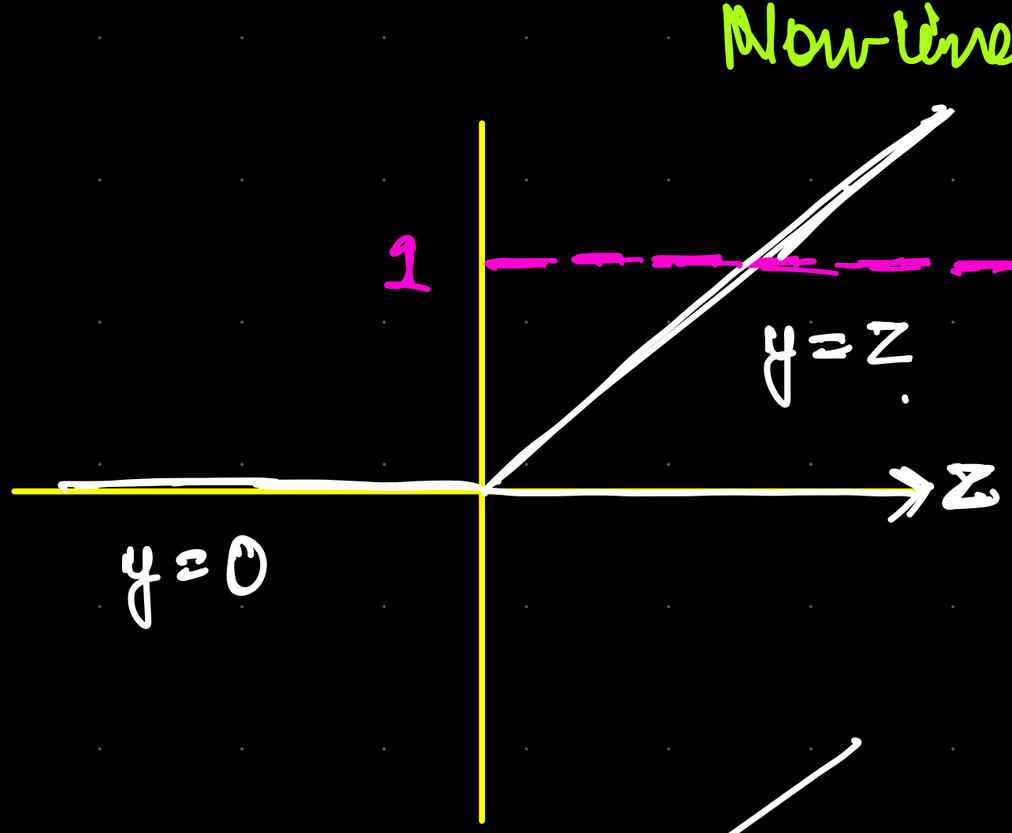
- → Non-linear
- → Differentiable

→ Gradient ≥ 1 for almost a big range z
→ Easy to calculate.

→ Not strict requirement

⟹ $[0, 1]$, $[-1, 1]$ → Atleast in that range

How shall we deal with this?



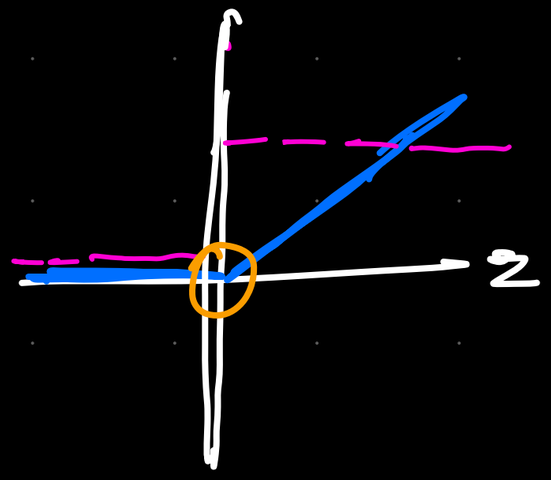
$$\text{ReLU} \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Rectified linear Unit

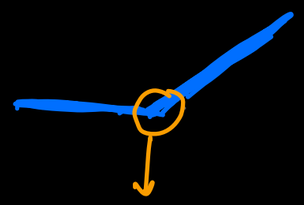
simple and fast

ReLU → max(z, 0)

Problems with ReLU activation



→ Not differentiable at z=0



Not differentiable.

→ still gradient is zero for half of the range.

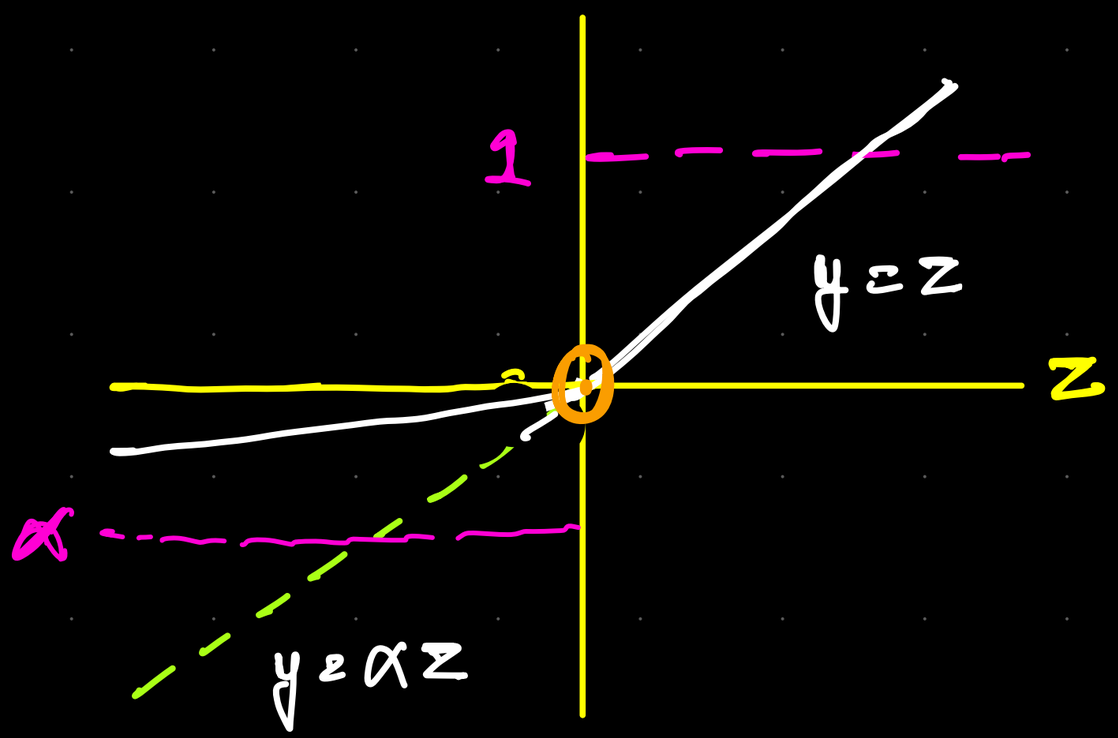
1 x 1 x 1 x 1 x 1 = 1

1 x 1 x 1 x x 0 = 0 → No update

Another simple activation.

Leaky ReLU - $\begin{cases} z & \text{if } z \geq 0 \\ \alpha z & \text{if } z \leq 0 \end{cases}$

α ← Hyperparameter.



---- Increase.

$L'ReLU = \begin{cases} 1 & \text{if } z \geq 0 \\ \alpha & \text{if } z \leq 0 \end{cases}$

Not differentiable.

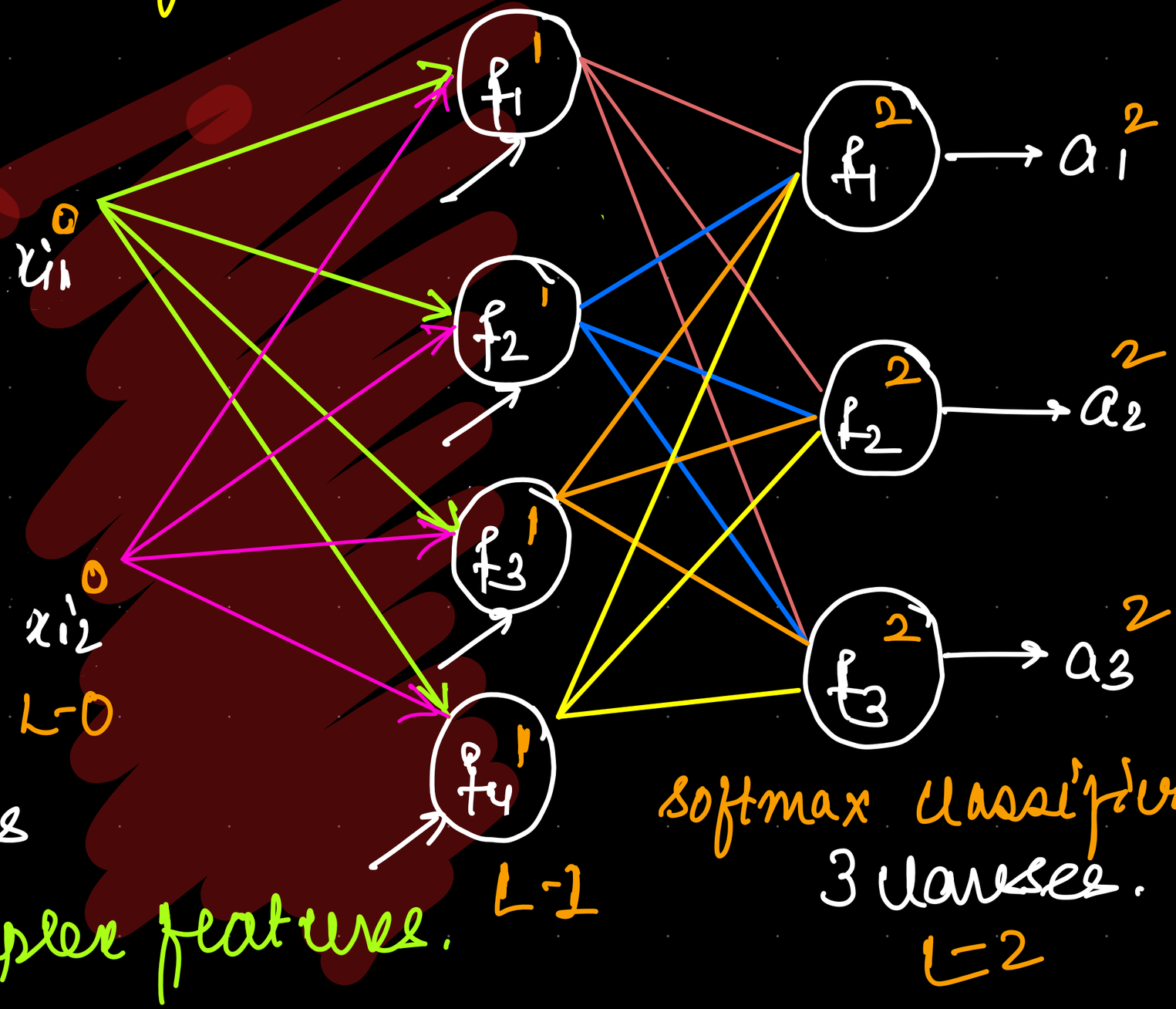
$z \rightarrow [-\infty, \infty]$

$z \rightarrow z + \Delta$

$-\infty$ ——— $z = 0$ ——— ∞

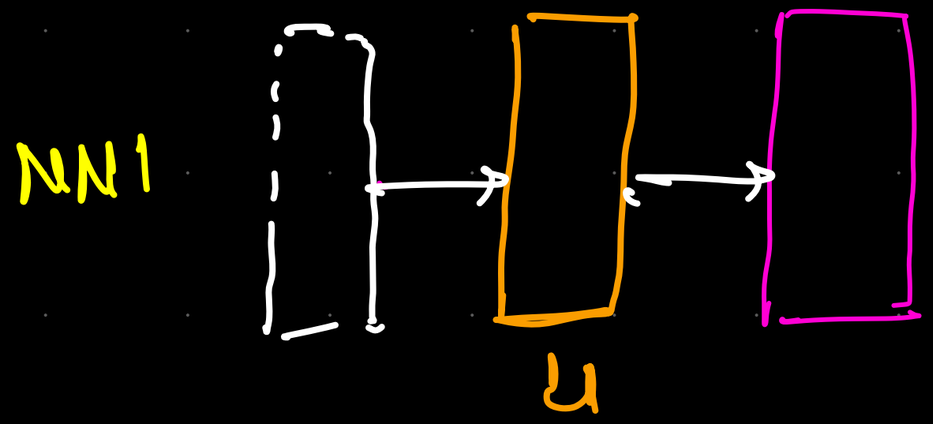
$z - 0.00000001$

Multi-layer NN (Notations)



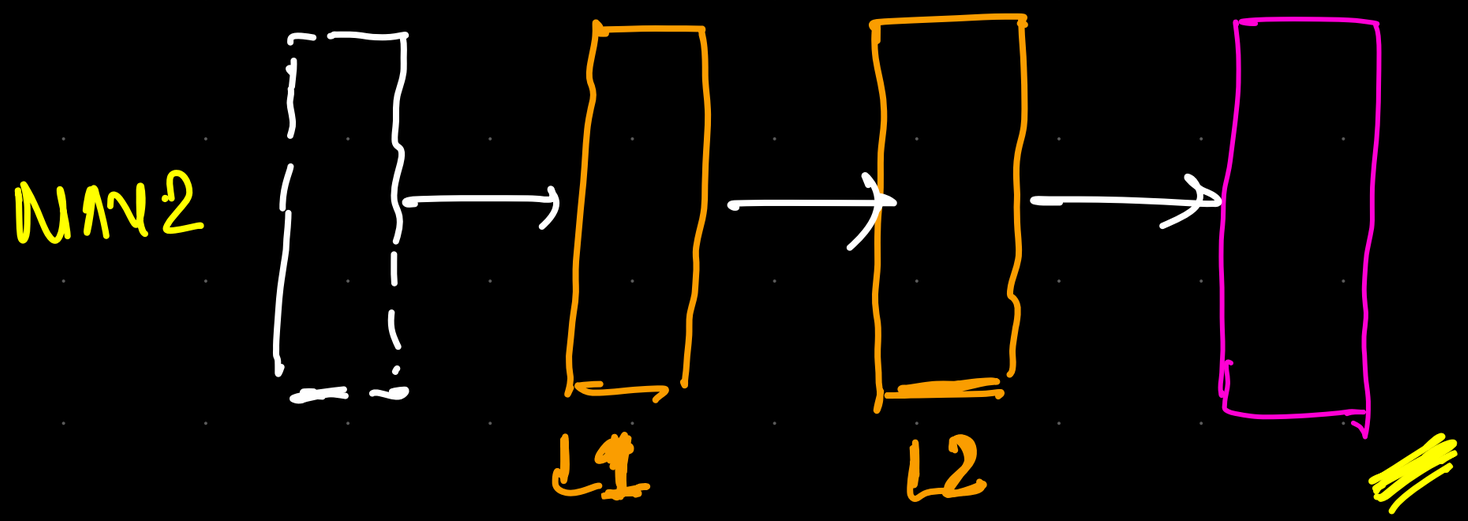
4 features
- 4 complex features.

Creation of complex features.



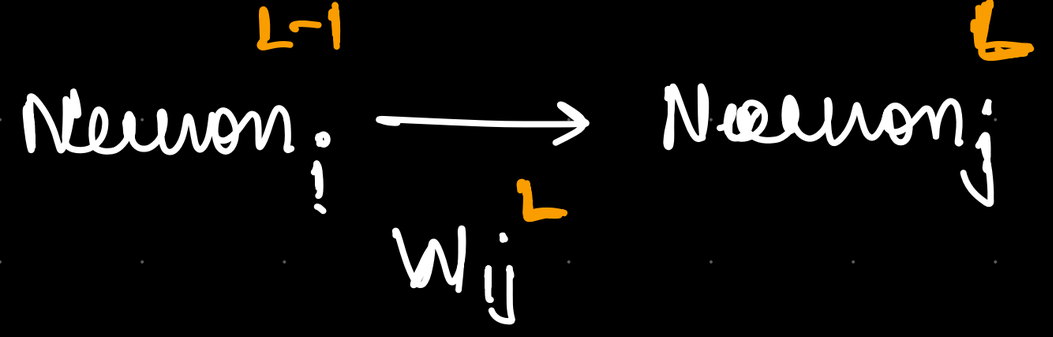
NN2 > NN1

N-layer NN
(Deeper)



Notations

w_{ij}^{L}



b_j^{L}

Neuron j^{L}

Notations

W^1

$$W^1 = \begin{matrix} & N_1' & N_2' & N_3' & N_4' \\ I_1 & \\ I_2 & \end{matrix} \begin{bmatrix} W_{11}' & W_{12}' & W_{13}' & W_{14}' \\ W_{21}' & W_{22}' & W_{23}' & W_{24}' \end{bmatrix}$$

2×4

inputs # neurons

b^1

$$b^1 = \begin{matrix} N_1' & N_2' & N_3' & N_4' \end{matrix} \begin{bmatrix} b_1' & b_2' & b_3' & b_4' \end{bmatrix}$$

1×4

neurons.

Matrices

W^2

$$\begin{array}{c} \\ I_1^2 \\ I_2^2 \\ I_3^2 \\ I_4^2 \end{array} \begin{array}{c} \begin{array}{ccc} N_1^2 & N_2^2 & N_3^2 \end{array} \\ \begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 \\ w_{41}^2 & w_{42}^2 & w_{43}^2 \end{bmatrix} \end{array}_{4\times3}$$

b^2

$$\begin{bmatrix} b_1^2 & b_2^2 & b_3^2 \end{bmatrix}_{1\times3}$$

Pending.

→ Fwd. Prop.

→ Backpropagation.

→ How initialize weight with random value?

→ Keras and Tensorflow

→ EDA of Business case.