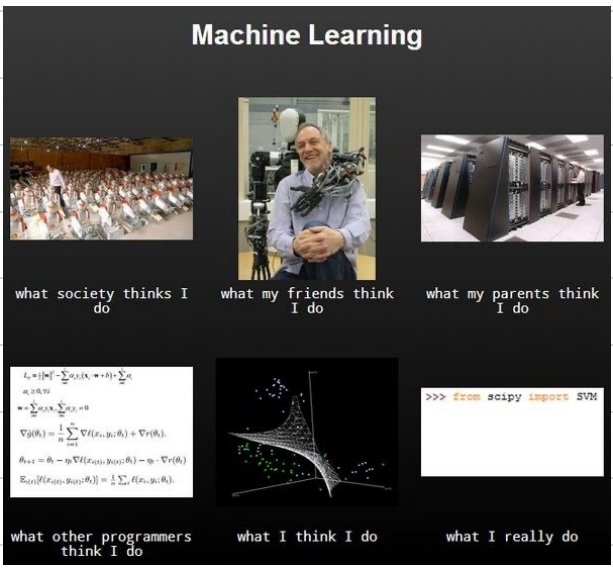
Session 2

Neural Networks -2

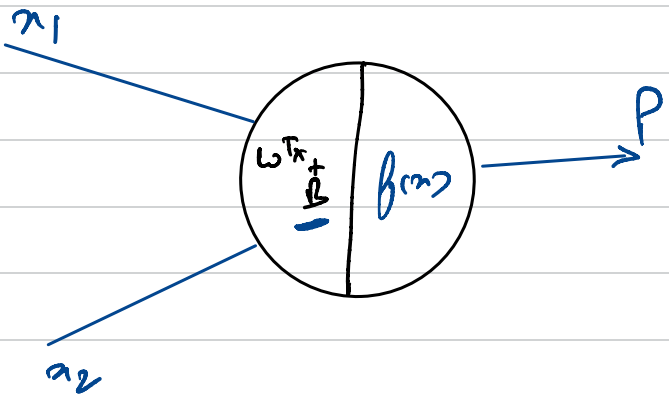
Feb 07, 2024



Agenda

1. Why we need non-Linearity
2. Forward Propagation
3. Why is NN universal function approximator?
4. Softmax
5. Cross Entropy.

NEURON



$f(x) \rightarrow$ Activation func

if $f(x) == \sigma$

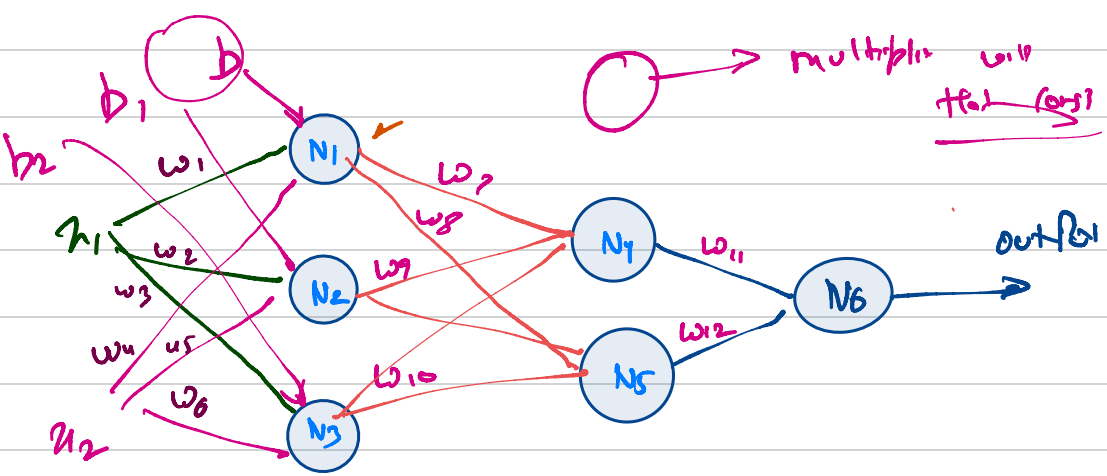
↓

LRU

↓

Logistic regression unit

if $P > 0.5 \rightarrow 1$, else 0



$N_2 = w_2 x_1 + w_5 x_2 + b$

$N_1 \Rightarrow \underbrace{(w_1 x_1 + w_4 x_2 + b)}_{z_1}$

$f(z_1) \rightarrow a_1$

Suppose to be non-linear → tanh, Relu, sigmoid

$\rightarrow K$

$N_1 \Rightarrow (w_1 x_1 + w_4 x_2 + b)$

z_1

$\Rightarrow K w_1 x_1 + K w_4 x_2 + K b \rightarrow a n_1$

$N_2 = (w_2 x_1 + w_5 x_2 + b) * g$

$= g w_2 x_1 + g w_5 x_2 + g b \rightarrow a n_2$

$N_4 \leftarrow w_7 * a n_1 + w_8 * a n_2$

$w_7 * (K w_1 x_1 + K w_4 x_2 + K b) + w_8 (g w_2 x_1 + g w_5 x_2 + g b)$

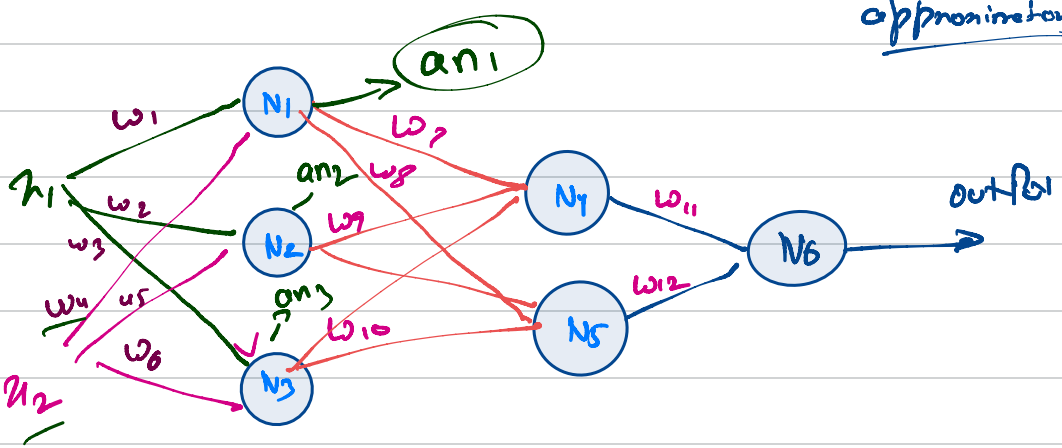
$w_7 (k w_1 x_1 + w_7 k w_4 x_2 + w_7 k b$

$+$

$w_8 g w_2 x_1 + w_8 g w_5 x_1 + w_8 g b$

$x_1 (w_7 k w_1 + w_8 g w_2)$

-> Why is NN -> called universal function approximator



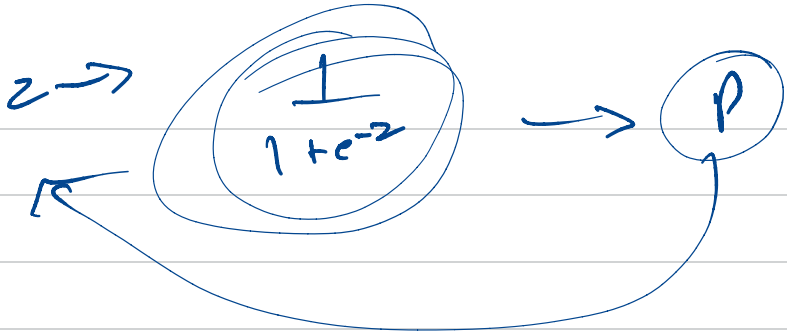
$a_{n_1} = f(w_1 x_1 + w_4 x_2)$

- non-linear combination of my input feature.

4 -> 12

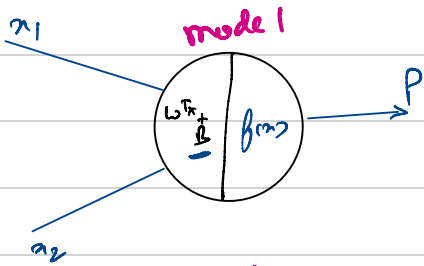
2

0 -> 1

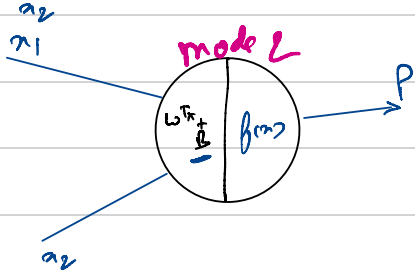


* epoch → my model has trained for 1 epoch

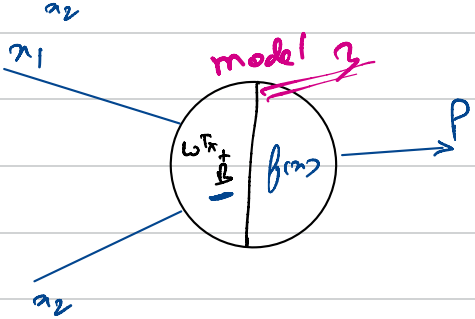
Trained through entire dataset once



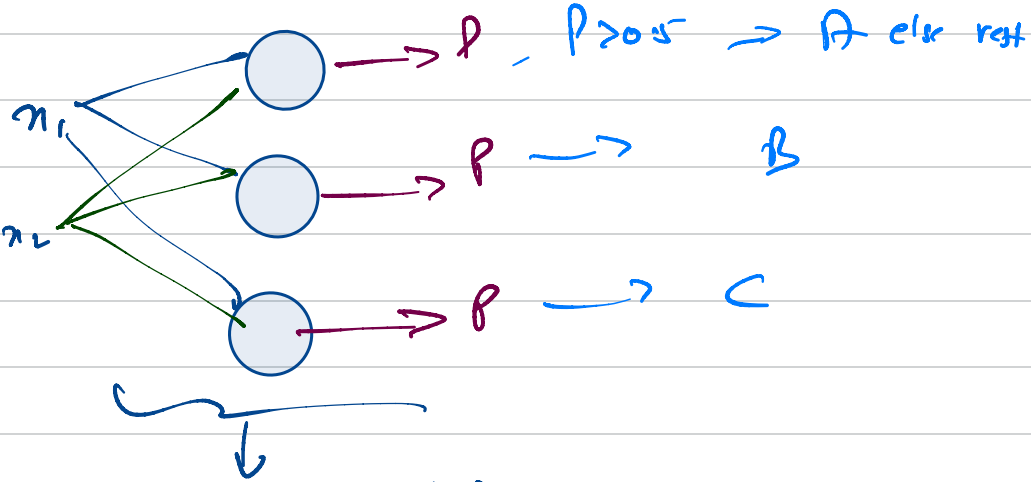
p > 0.5 → 1 (A) elder
else rest



→ p > 0.5 → 1 (B) like
else rest

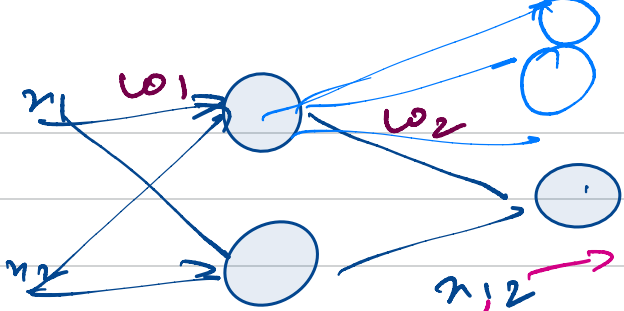


p > 0.5 → 1 (C) red
else rest



In back Propagation

1. How to modify the weight vector
2. How the loss Propagate → throw network



x_{12} → 2nd Feature

x_{12} → i^{th} sample.

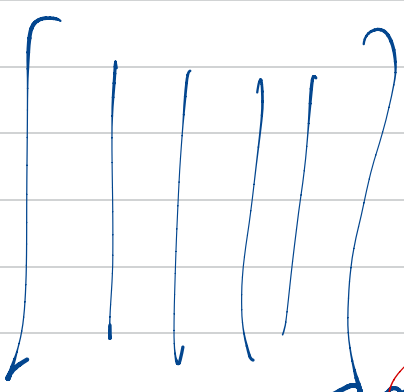
Height	weight
x_{11}	x_{12}
x_{21}	x_{22}
⋮	⋮
x_{m1}	x_{m2}

$m \times 2$ → rows of data

→ randomly initialize

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$$

$2 \times n$ → no of neuron in the next layer



→ 1st neuro

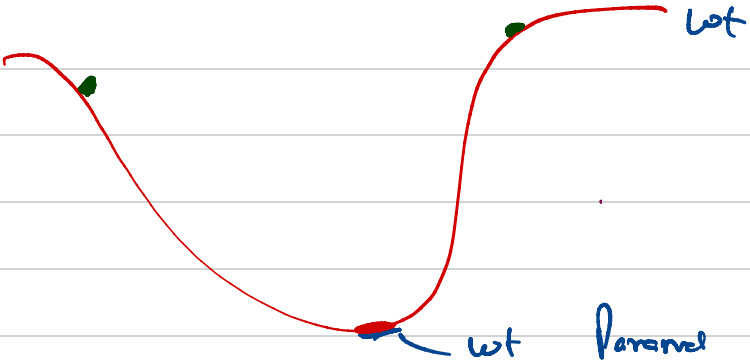
→ 2nd neuro

→ 200×2

$200 = m \times 2$

max → only for next layer

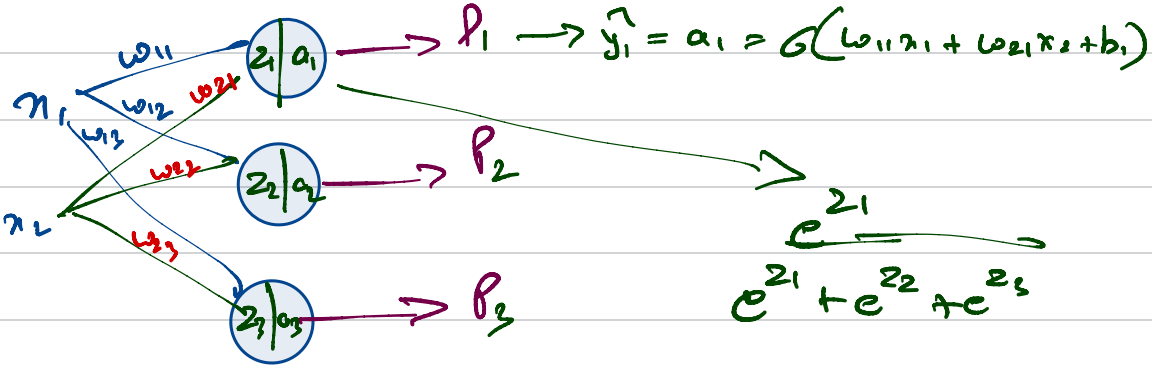
; $n = 2$ as I've 2 neurons



rows(m) Feature

200×4 , 4×5 → 5 neurons in my next layer

→ 200×5



$P_1 \& P_2 > 0.5$

$\frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$

$\frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$

SOFTMAX

z_1

$$p_i = \frac{e^{z_i}}{\sum_{i=0}^{N} e^{z_i}}$$

exponential function have nice property

1. nice differential
2. Output can be interpreted as log likelihood not Probability

Input pixels, $\mathbf{x}$ | Feedforward output, $\mathbf{y}_i$ | Softmax output, $\mathbf{S}(\mathbf{y}_i)$

Forward propagation →

cat	dog	horse
5	4	2
4	2	8
4	4	1

Softmax function →

cat	dog	horse
0.71	0.26	0.04
0.02	0.00	0.98
0.49	0.49	0.02

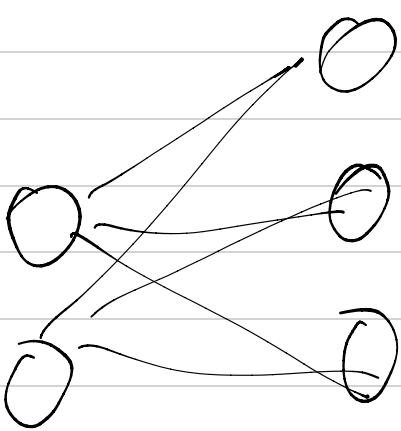
Shape: (3, 32, 32) | Shape: (3,) | Shape: (3,)

for training

log(softmax)

Don't treat softmax output as probability

log loss function



Cross Entropy

$$y_i \times \log(\hat{y}_i) + (1-y_i) \cdot \log(1-\hat{y}_i)$$

$\hat{y}_i$ → p_1 ; $1-\hat{y}_i$ → p_2

- y_i: actual label
- $\hat{y}_i$: Prob of the label

$$y_i \times \log(p_1) + (1-y_i) \times \log(p_2)$$

↓

multi-class

$$CE = \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \log(p_{ij})$$

Replace $K \rightarrow 2$

$$y_{i1} \times \log(p_{i1}) + y_{i2} \times \log(p_{i2})$$

- Quantifies the discrepancy between

Predicted Prob dist

&

Actual " "