Multilayer perceptron — Backpropagation, Weight,

Activation Function, Regularization, BatchNormalization,

Dropout & early stopping

Code — Reg + Classification — Tensorflow + Keras — done → Google

→ Image — " " " "

→ Pytorch — Facebook / Meta

Loss ≈ zero

Backpropogation / Back-prop - V.V. Imp - Interview

Prof. G. Hinton

2006

① "Chain Rule"

② "Memoization"

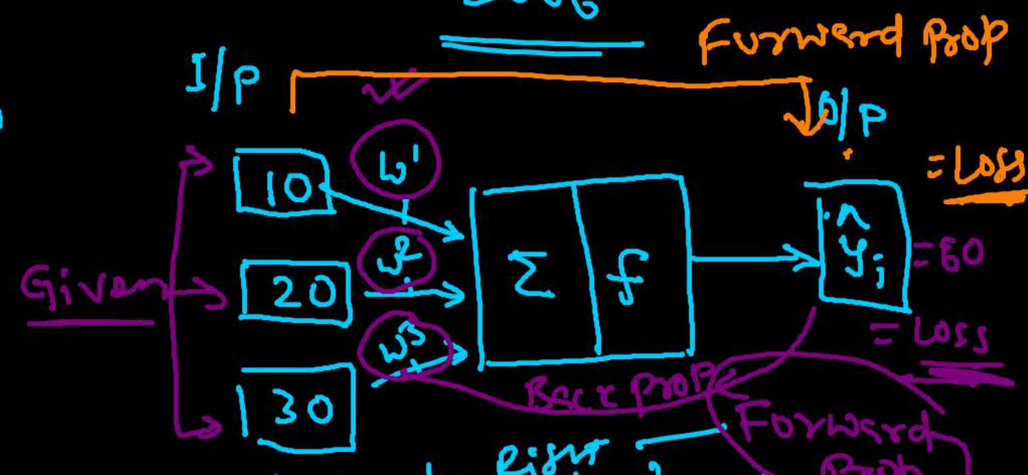
Ind variable

Dep variable

$D = \{x_i, y_i\}$

$x_i \in \mathbb{R}$ (10, 20, 30)

$y_i \in \mathbb{R}$ = 100



Case 1 :- Weight : 1, 1, 1 → I/P → O/P → T / F

cal = 10×1 + 20×1 + 30×1 = 60

f = Linear / Relu

f(60) = 60

y_i = 100

$\hat{y}_i$ = 60

Loss = $y_i - \hat{y}_i$ = 100 − 60 = 40 (Error)

Case 2 :- Change the weight

w_1	w_2	w_3
2	1	2

10×2 + 20×1 + 30×2 = 100 = $\hat{y}_i$

Loss = $y_i - \hat{y}_i$ = 100 − 100 = 0

Perfect

1 Epoch = 1 Backprop + 1 Forward Prop

↳ is nothing but iteration. This is just a fancy word

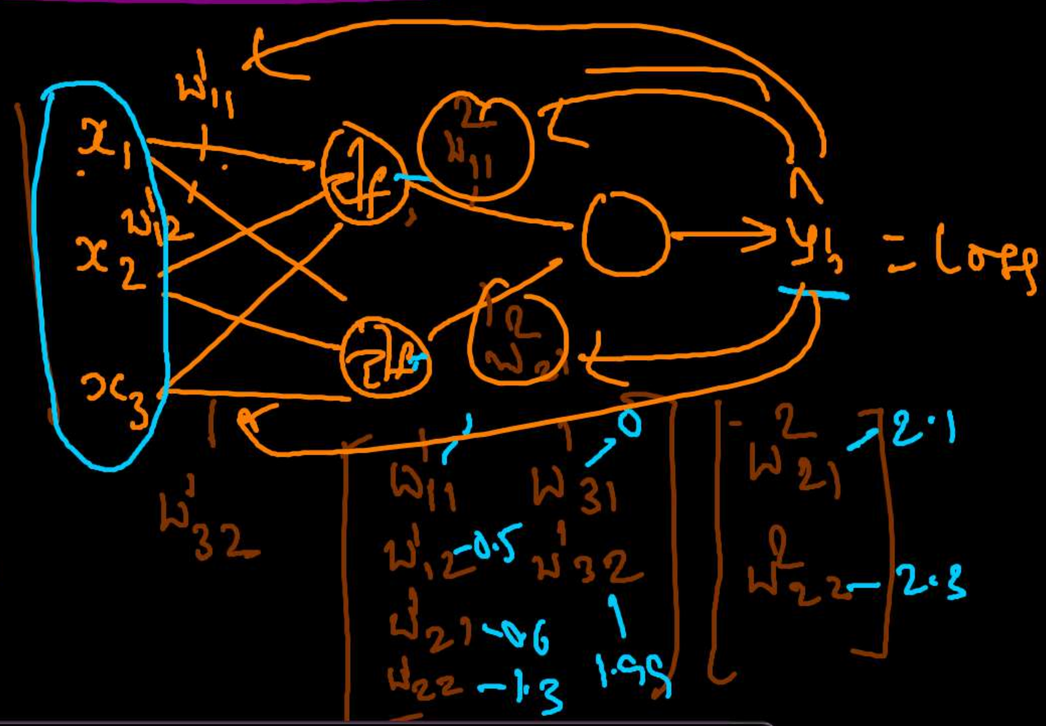
100 epochs = 100 Backprop + 100 Forward prop

epochs = 100

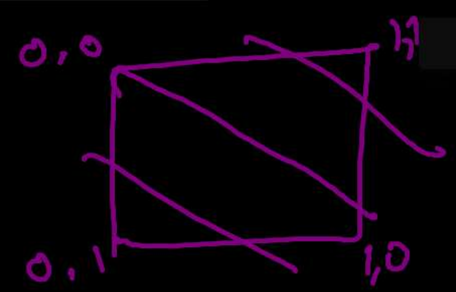
epochs = 10 Loss = Zero

(90) Stop

0 cost



Back Prop



$$D = \{x_i, y_i\}$$

Heinrst
Xavier
System
Rule 1 :- initialize the weights W_{ij}^{k} - Randomly

Rule 2 :- For each x_i in D

(a) pass x_i forward through the network → Forward propagation

E | F

(b) compute Loss value $Loss(y_i, \hat{y}_i)$

(c) compute all the derivatives using chain rule of

diff - Functions

Memoization

(a) update weight for end of the network to the start.

Known as Backpropogation Algorithm

B.Prop

v.v.imp derevative

$$W_{new} = W_{old} - \eta * \frac{\partial L}{\partial W_{old}}$$

: Gradient Descent

Chain Rule

New weight

Old weight

eta / learning rate / parameter

0 to 10

$(weight)_{old} = 50$ -2

$W_n = 50 - 1 * 2 = 48$ ✓

$= 50 - 1 * -2 = 52$ ✓

Rule 3 :- Repeat Step 2 till convergence

↳ old = 50

↳ new = 49.999999, 50.0000001
= 49.9999998

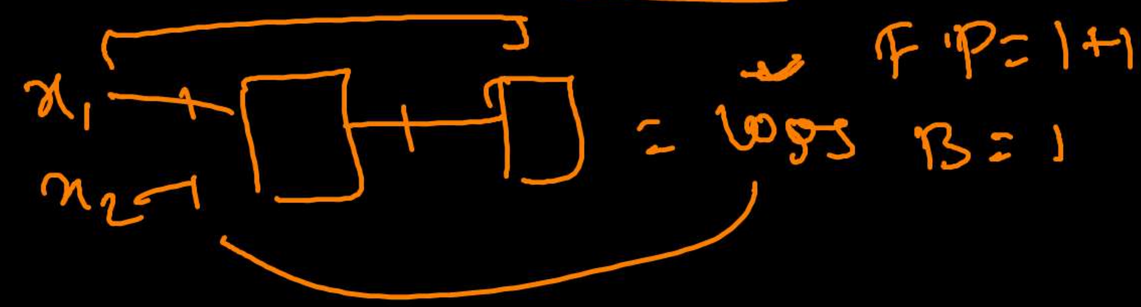
New weight is very similar to the old weight

old ≈ new

$(w_{ij}^k)_{new} \approx (w_{ij}^k)_{old}$ = convergence

f = 11

e = 10



original value = 100

Pred = 60

Loss = (40)

(2) Pnew = 190 = -90

(3) pro = 80 = 20

Epoch = 100

| Pred = 100 = zero.

Pred = 100.001 = 0.00009

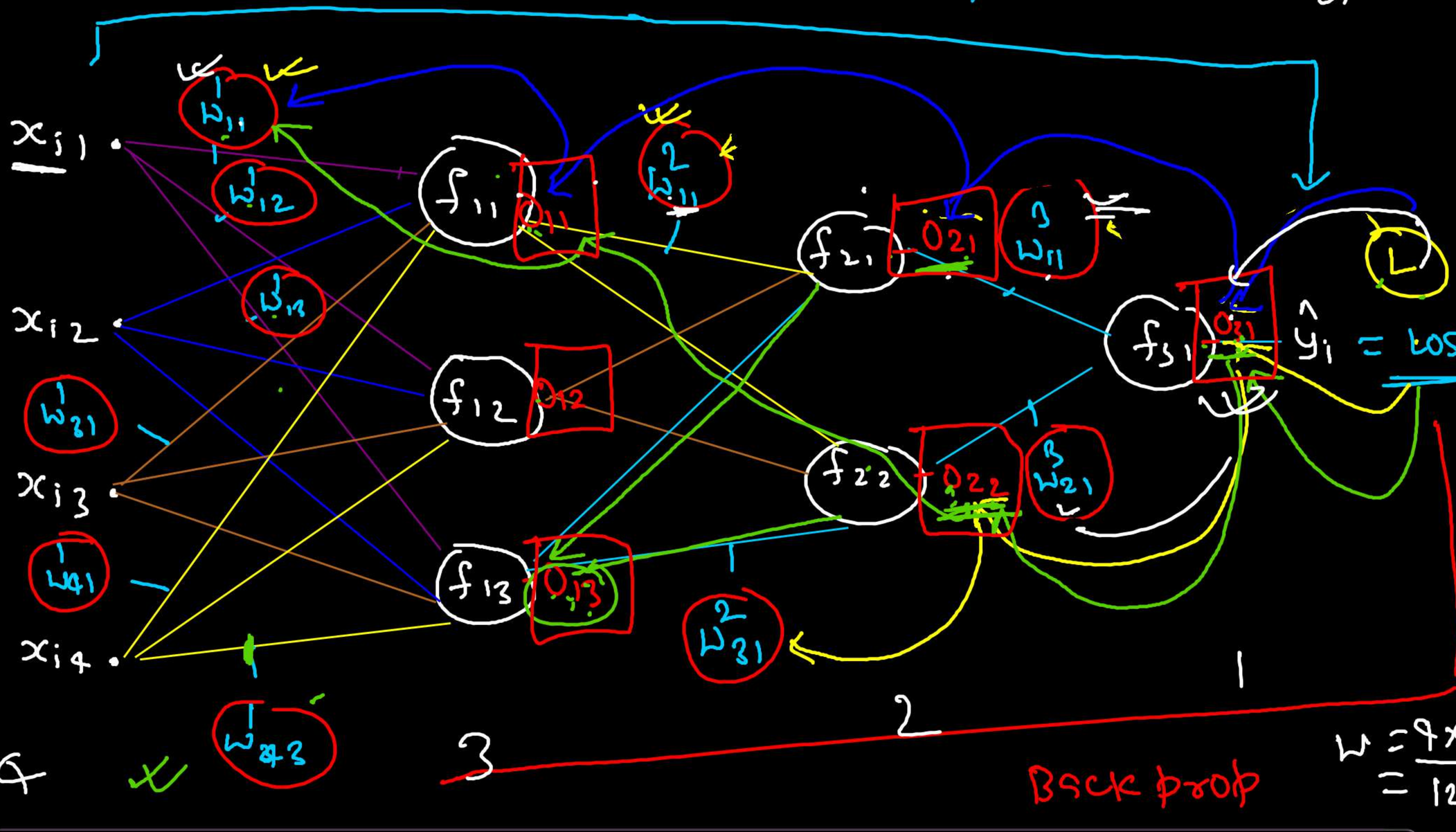
Pred = 100.00009 = 0.00001

convergence

Chain Rule

$\frac{\partial L}{\partial w_{31}} = ?$

Forward Prop.



$w = 4 \times 3 + 3 \times 2 + 2 \times 1$
$= 12 + 6 + 2 = 20$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial v} \cdot \frac{\partial v}{\partial x} = \frac{\partial y}{\partial x}$$

- Chain Rule

$$\frac{\partial L}{\partial w^3_{11}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial w^3_{11}}$$

$$\frac{\partial L}{\partial w^3_{21}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial w^3_{21}}$$

} Chain Rule / Sanity Check / correctness check

$$\frac{\partial L}{\partial w^2_{11}} = \frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial w^2_{11}}$$

$$\frac{\partial L}{\partial w^2_{31}} = \frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial w^2_{31}}$$

} Chain Rule

$$\frac{\partial L}{\partial w^1_{11}} = \boxed{\frac{\partial L}{\partial o_{31}}} * \boxed{\frac{\partial o_{31}}{\partial o_{21}}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w^1_{11}} + \frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w^1_{11}}$$

$$\frac{\partial L}{\partial w^1_{11}} = \boxed{\frac{\partial L}{\partial o_{31}}} * \frac{\partial o_{11}}{\partial w^1_{11}} \left\{ \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} + \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{11}} \right\} \text{- Chain Rule}$$

$$\frac{\partial L}{\partial w^1_{43}} = \boxed{\frac{\partial L}{\partial o_{31}}} + \boxed{\frac{\partial o_{31}}{\partial o_{21}}} * \frac{\partial o_{21}}{\partial o_{13}} * \frac{\partial o_{13}}{\partial w^1_{43}} + \frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{13}} * \frac{\partial o_{13}}{\partial w^1_{43}}$$

$$\frac{\partial L}{\partial w^1_{43}} = \boxed{\frac{\partial L}{\partial o_{31}}} * \frac{\partial o_{13}}{\partial w^1_{43}} \left\{ \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{13}} + \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{13}} \right\} \text{- Chain Rule}$$

In Calculus :

$$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial f(x)} * \frac{\partial f(x)}{\partial x} + \frac{\partial h}{\partial g(x)} * \frac{\partial g(x)}{\partial x}$$

(1) Define loss Function

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \text{regularization}$$

50 - loss function

(2) optimization - Gradient Descent
- SGD
- BGD
- MBGD

Training an MLP : Memoization

Please note, it is not Memorization. it is called "Memoization"

$$\frac{\partial L}{\partial w^3_{11}} = \boxed{\frac{\partial L}{\partial o_{31}}} * \frac{\partial o_{31}}{\partial w^3_{11}}$$

0.1 (5)

$$\frac{\partial L}{\partial w^3_{21}} = \boxed{\frac{\partial L}{\partial o_{31}}} * \frac{\partial o_{31}}{\partial w^3_{211}}$$

"Memoization"

↓

computer science - AI algorithm

↓

Dynamic Programming

Compute once & reuse it

→ The core idea of memoization is this :—

1) if there is any operation that is used many times repeatedly

→ it is a good idea to compute it once & then Store it → reuse purpose

This concept is known as Memoization

Benefit → Speed up.

Deficit → Slightly more memory reqm.

$$w_n = w_0 - \eta \frac{\partial L}{\partial w_0}$$

$$b_n = B_0 - \eta \frac{\partial L}{\partial b_0}$$

42

$3 \times 9 + 9 \times 3 + 3 \times 2 + 2 \times 1 = 32 = 42$

x_1	0	0	0	0
x_2	0	0	0	
x_3	0	0		
	9	3	2	1
	9	+ 3	+ 2 + 1 =	10
				B

Linear Reg

→ Linear classific → Binary num

Binary -

→ Sigmoid num

→ Softmax

$B = \underset{\text{u}}{0 \& 1}$

$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+\frac{1}{e^x}} = \frac{1}{\frac{e^x+1}{e^x}} = \frac{e^x}{e^x+1}$$

$\Rightarrow y(e^x+1) = e^x \Rightarrow y \cdot e^x + y = e^x$

$\log_a a = 1$

$\log_a a^2 = 2$

$y = 0$

$\left(\frac{0}{1-0} = 0\right) \to -\infty$

$\Rightarrow y = e^x - y \cdot e^x = e^x(1-y)$

$\Rightarrow \boxed{\frac{y}{1-y} = e^x}$

$\log_e$

$\log_e\left(\frac{y}{1-y}\right) = \log_e e^x$

$\boxed{\log_e\left(\frac{y}{1-y}\right) = x}$

$y = 1$

$\log\left(\frac{1}{1-1}\right) = \log(\infty) \to \infty$

odd fn

Relu

$x = 5$

$\max(0, x) = \max(0, 5) = 5$

$x = -5$

$\max(0, -5) = 0$

Sigmoid(5) = 0.01

Sigmoid(500) = 0.2

$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w^1_{11}}$$

(1) (2) (3) (6) = (36)

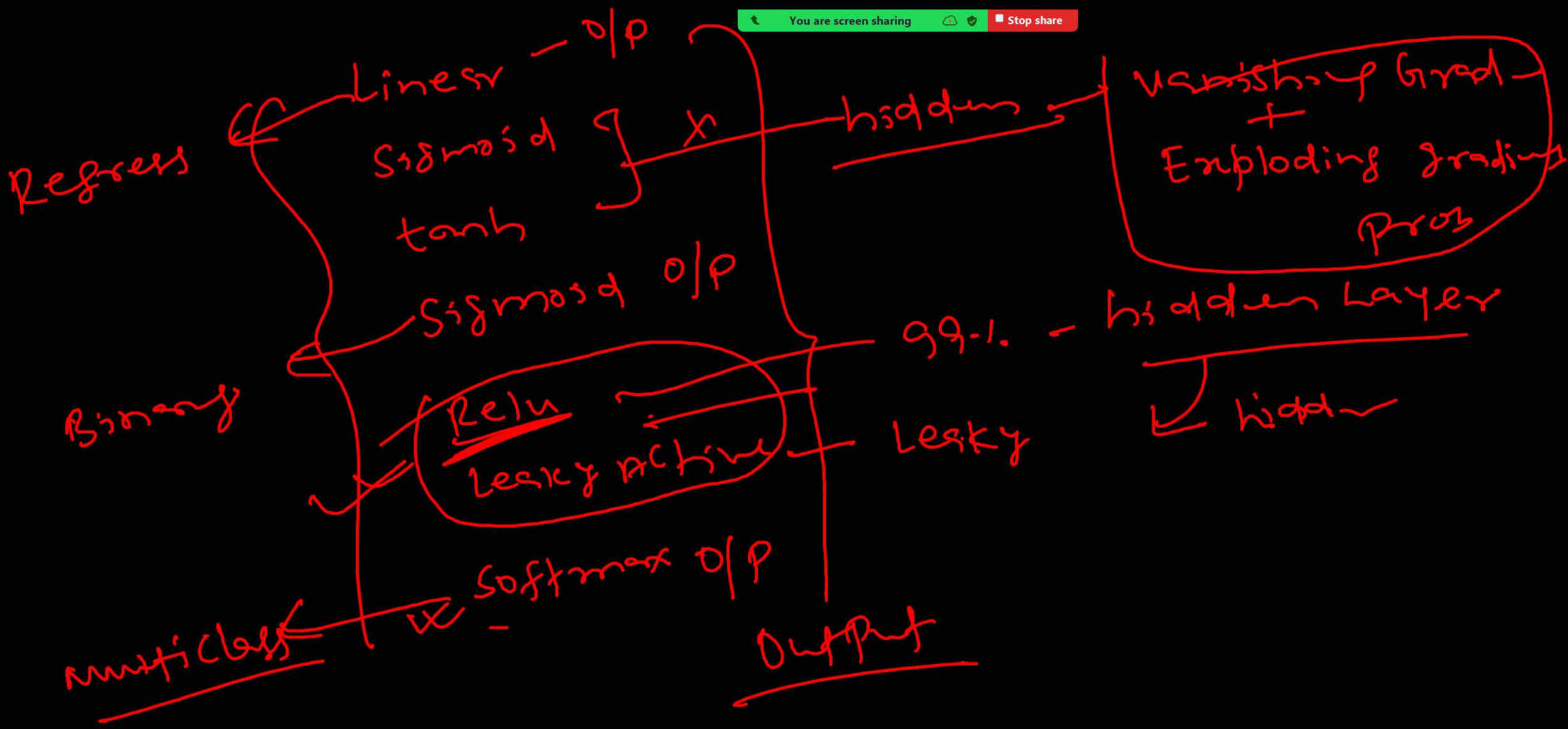
Vanishing

0.3 0.2 0.01 0.001 = 0.0000006

0.01

Sigmoid

zero



Early Stopping

Talking:

You are screen sharing | Stop share

Weight calculation

+

Tomorrow - Batch Normalization

&

+

Next Week

Dropout

+

Optimization

+

hyperparameter tuning

DNN

+ code — Tensorflow + Keras

Pytorch / Torch

+ Problem - Image Process