How to train a Deep MLP?

(1) Pre-Process → Data Normalization (Feature Scaling)

(2) Weight init
- Xavier / Glorot init → Sigmoid | Tanh
- He-init → Relu & LeakyRelu
- Gamma - (Small σ)

(3) Choose Right Activation Function
1. Hidden Layer — Relu or LeakyRelu
2. Output — Regression → linear
   - Classification → Binary — Sigmoid
   - Multi-class — Softmax

late
Selu - 2017
X not
two poin
like Relu

④ Batch Normalization ⇒ especially for deep MLP

⑤ Regularization ← Ridge (L2) ✓
lasso (L1)

⑥ Dropout

⑦ Early Stopping

4 to 7 → Overfitting Problem

⑧ optimizer — GD ← BGD, SGD, MBGD + Adagrad / momentum / NAG / Adadelta / RMSprop / Adam ✓ ***

↳ By default — RMSprop (keras)
→ Suggestion — Adam

(2019, 2024)

⑨ Hyperparameters — Architecture — # Layer ✓
Neuron ✓
dropout

optimizer, Activation, Weights etc.

(10) Loss Function

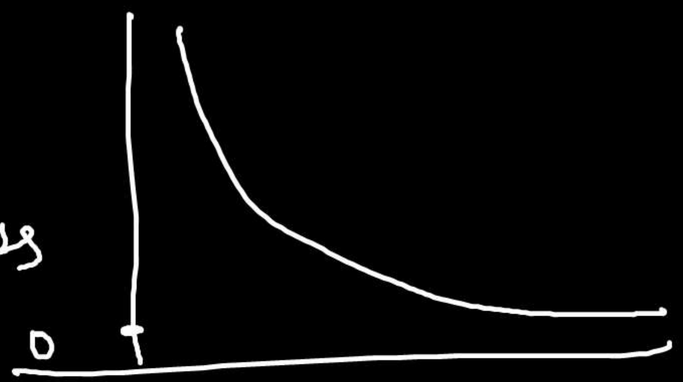
- 2 class classification = log-loss - binary cross-entropy
- K-class ,, = categorical Cross entropy
  or
  Sparse Categorical Cross entropy.
- Multi class ,,

Regression - Square Loss Function (Mean Absolute error)
MSE / RMSE
-

(11) Always monitor your gradient — Loss, acc, val_loss, val_

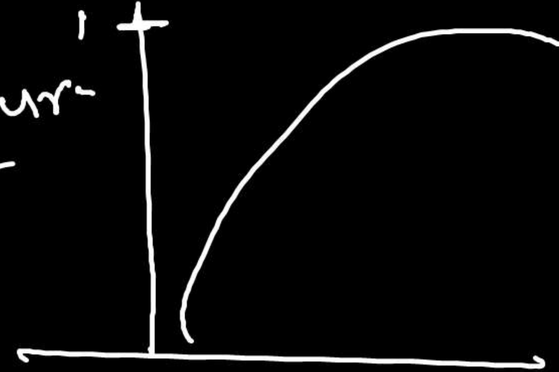
(12) Plots

Loss



Train & Test

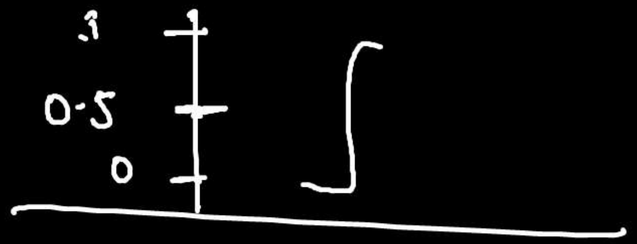
Accuracy



Please NOTE :— in MLP/DNN/ANN — Avoid overfitting Problem by the help of point no - 4 to 8 ✗✗

Binary class ← Yes True Selected 1
No False Not selem 0

Sigmoid



Multi Class – Softmax – Prob = 1

Image

① Cat = 0.01
② Dog = 0.02
③ Cow = 0.03
④ Monkey = 0.04
⑤ Human = 0.9

1 (0.07)

Final : hmm

Relu : max(0, x)

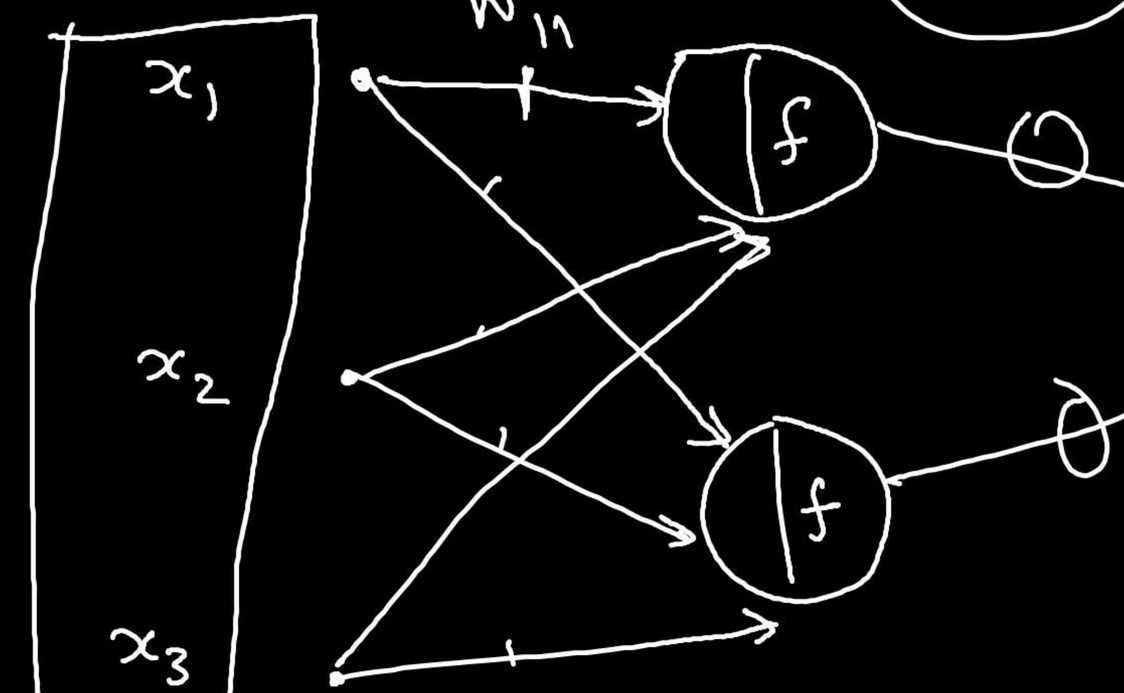
$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial z_2} * \frac{\partial z_2}{\partial z_{11}} * \frac{\partial z_{11}}{\partial w^1_{11}}$$

0.1 | 0.2 | 0.3 | 0.01

$= 0.00006$ — Small

$\hat{y}_1$ = Loss

Normalize



Input

Weight first layer

He-Inst

Hidden layer

Relu — To avoid Vanishing / Exploding Gradient Problem

Leaky Relu

Output

Problem Statement

① Regression — Numeric

② Classif. — binary = Sigmoid
— multi = Linear

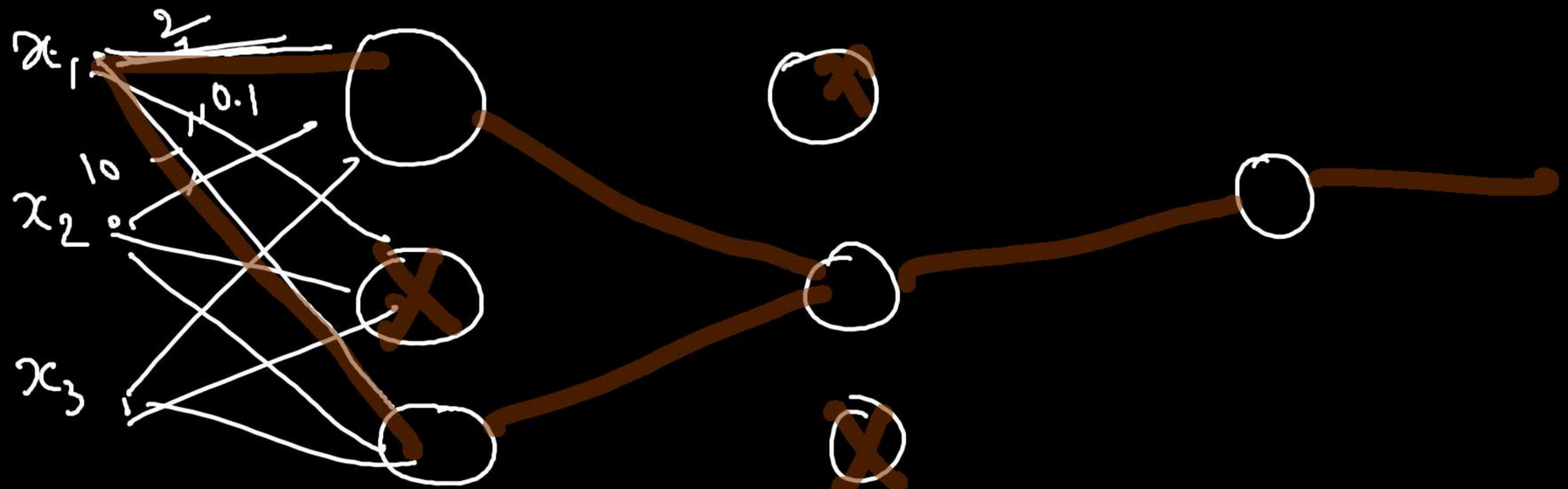
old weight = 50

$new = w_0 - \eta * \frac{\partial L}{\partial w^1_{11}}$

$new = 50 - 0.1 * 0.00006$

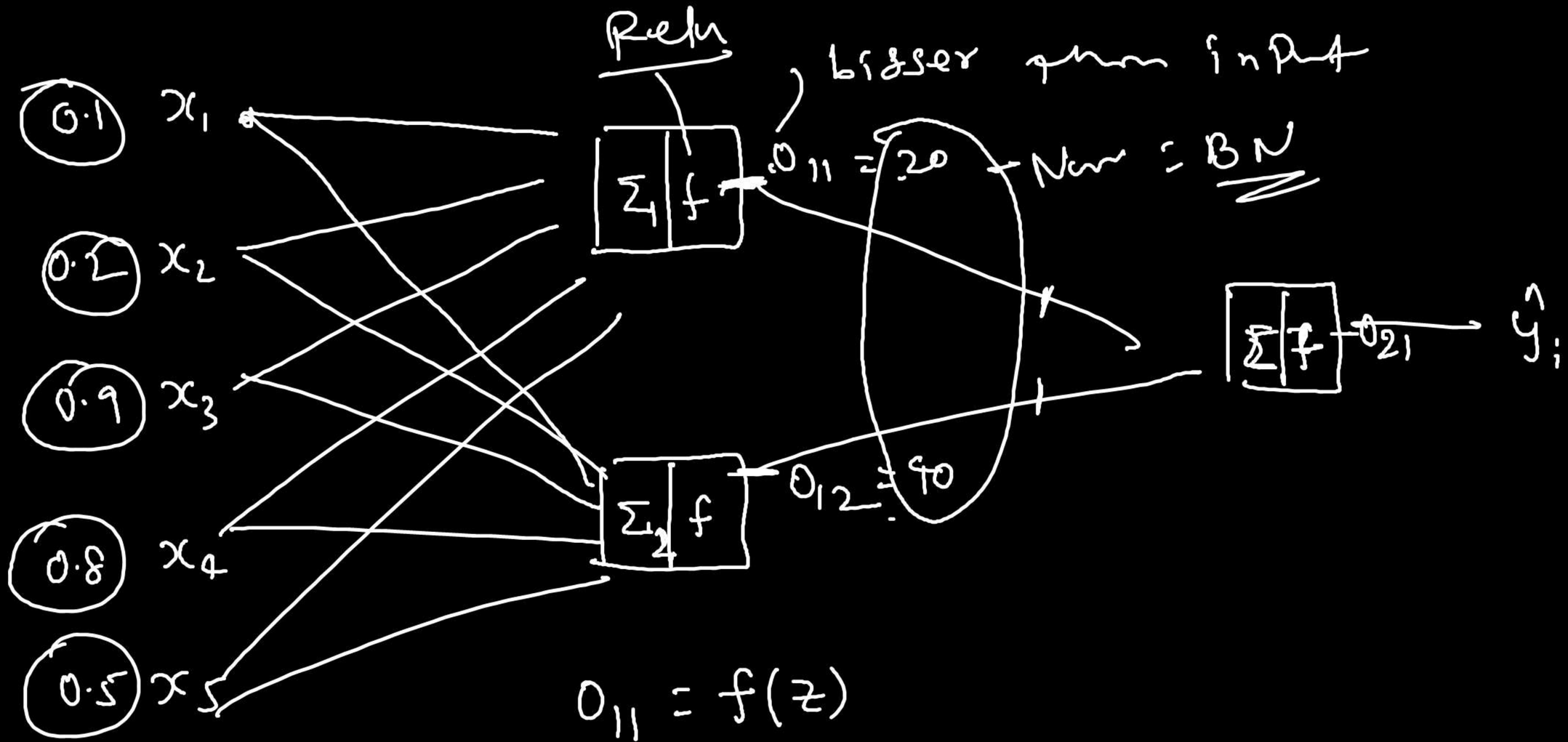
$= 50 - 0.0000006$

$= 49.9999994$



Regularization w_1 w_2 w_3 → Reducing the weight → Closure to zero = Ridge

Lasso = exact or close to zero



$$O_{11} = f(z)$$

$$z = \sum_{i=1}^{n} x_i w_i \quad \Sigma_1 = x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 + x_5 w_5$$

$$\Sigma = 20 \quad O_{11} = f(20) = 20$$

Regularization

- Lasso — L1
- Ridge — L2
- Elastic Net — DL

$$L(\beta) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

$\beta_0 = 5$

bias 0.1

$0.1 = \beta_1 = 100 = 0.1$

$0.01 = \beta_2 = 1 = 0$

$0.02 = \beta_3 = 0.5 = 0$

$x_1 = 100$ → $500 \times 100 = 50000$

$x_2 = 101 = 505$ → $5 = 505$

$x_3 = 99 = 495 = 247.5$

10 1.01 0.495

* Dropout - 2014 → Nitish Srivastava

↓ Toronto University

Prof Geoffrey Hinton - Father of Modern Deep Learning

↳ 2006 - Backpropagation

Extremely Simple & Elegant

↓

Random Forest

DNN - 10 to 50 L.

25 L. 50 L.

CNN - 10 to 30 L.

RNN - 30, 50 L.

Tensorflow + Keras

Meta AI Pytorch - LLM

Transformer — 2017 — Attention All You need to know

BERT

50 hrs

GPT — Chat GPT

LLM | Gen AI

+ MLOps

11:30

MLFlow
Kubernetes
Docker
Git | Github | DVC
Life cycle - Pipeline

Tensors ⟵ Scalars, Vectors, Matrices to higher dimensions

0

0D Tensor

Scalar

↓

Single digit matrix

$\begin{bmatrix} 1 \\ 6 \\ 0 \end{bmatrix}$

1 D Tensor

Vector

1 Row vector

$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$

1 column vector

$\begin{bmatrix} 1 & 1 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

2 D Tensor

Matrix

↓

Two dimension array of one or more Col's & rows

$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}$

3D Tensor

Tensor

[[[ ]]]

n D Tens

n D Tensor

Add & Subtraction matrix

$$\begin{bmatrix} 3 & 1 & 6 \\ 4 & 2 & 1 \\ 5 & 3 & 5 \end{bmatrix} + \begin{bmatrix} 1 & 3 \\ 2 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 7 \\ 6 & 4 & 2 \\ 8 & 4 & 6 \end{bmatrix}$$

multiplication matrix

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}_{3\times 2} * 2 = \begin{bmatrix} 1*2 & 4*2 \\ 2*2 & 5*2 \\ 3*2 & 6*2 \end{bmatrix}_{3\times 2} = \begin{bmatrix} 2 & 8 \\ 4 & 10 \\ 6 & 12 \end{bmatrix}_{3\times 2}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times3} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3\times2} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}_{2\times2}$$

2×3 and 3×2 — inner dimensions match

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times3} * \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}_{2\times3}$$

dot ⇒ $\sum$

↳ Transpose

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} a_1 & c_4 \\ b_2 & e_5 \end{bmatrix}_{3\times3}$$

Remaining - optimization

&

hyperparam