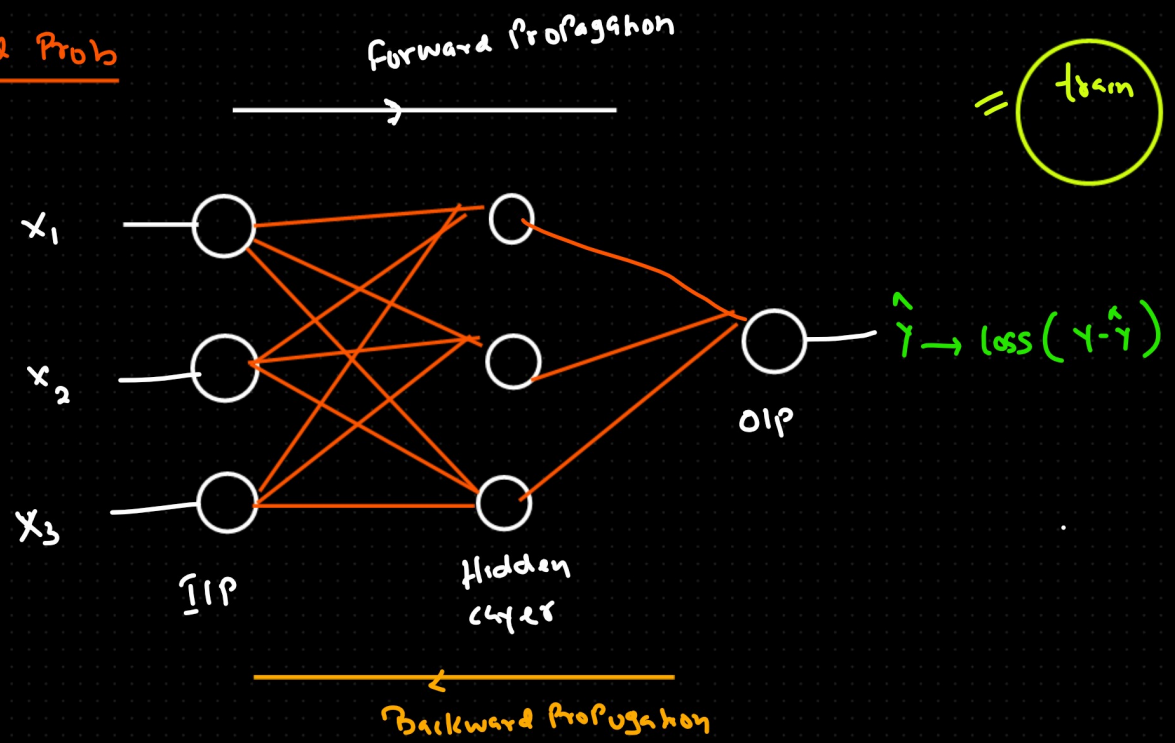
1. Vanishing grad | Exploding grad. ✓
2. Data Scaling → Batch Norm. ✓
3. Drop out ✓
4. Regularization ✓
5. Weight initialization ✓

Vanishing grad Prob

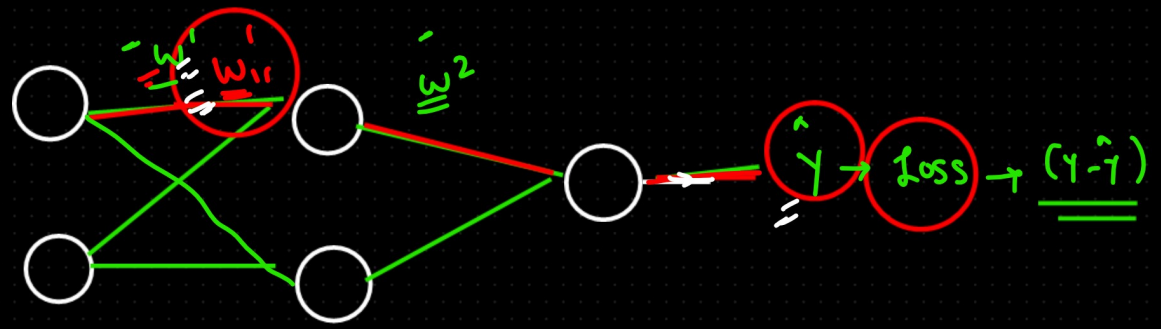


Gradient will be very very small → there won't be any sort of a weight update

Vanishing

Vanishing gradient

Neural Network ⇒ Layers Nodes ⇒ Multiple ⇒ Deep Neural Network



$$w_n = w_o - \eta \frac{\partial L}{\partial w}$$

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial I} \times \frac{\partial I}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}^1}$$

= Calculation (I/P x weights) = O/P ⇒ Back Propagation

[0-1]

(Derivation / Partial Derivation)

Sigmoid → [0-0.25]

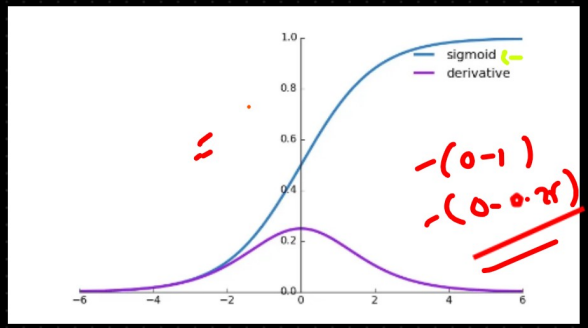
1

Neural Net → Deep Neural Net ⇒ Vanishing grad. ⇐ wrong activation

2

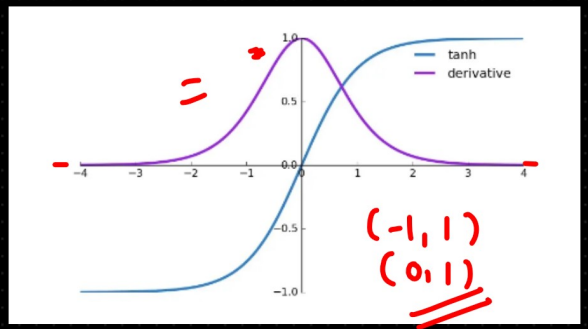
1. sigmoid
2. tanh

there is no update in the weight



-(0-1)
-(0-0.25)

Sigmoid



(-1, 1)
(0, 1)

tanh.

$0.1 \times 0.1 \times 0.1 \times 0.1 = 0.0001$

= Deep Neural Network ⇒ Weight ⇒ Forward Propagation ⇒ (loss)

Sigmoid / tanh

Update the weight

$W_n = W_o - \eta \frac{\partial L}{\partial W_o}$

Chain rule ← (0 – 0.25)

0.01, 0.1, 0.001, 0.2, 0.15

Small × Small × Small ... = very small value

1 − 0.01 × 0.0001

Where = $W_{old} - \eta \frac{\partial L}{\partial W}$

= 1 − 0.00001

= 0.9999

negligible

1 − 0.9999 = 0.000001

= gradient value

which is going to be vanish

Vanishing grad. ⟶ Deep Neural Ner

{ 1. Loss will not change
  2. Weights will not change

Vanishing grad (How you gonna resolve it)

1. Reduce the Complexity of your NN:
   - ↳ Reduce the Nb. of layer
   - ↳ Reduce the No. of Nodes

   (Shallow Neural Ner)

2. Use a diff act. fn

Relu → max(0, z)

(−∞, +∞)

(0, +∞)

{ It will remove negtive value from your data }

Diff ⇒ -ve ⇒ 0

+ve ⇒ 1

Dying relu ← 0 or 1

(chain rule $\frac{\partial L}{\partial w} = (\ldots\ldots\ldots)$ ← 0

Leaky relu

3 Proper weight initialization

1. Xavier
2. Glorot

4 Batch Normalization ←

How to improve a Performance of Neural Network ?

1. Handle or TC vanishing gradient
   - Activation fn
   - Correct weight initialization
2. Overfitting
   - Early stopping
   - Dropout
   - Regularization
   - Reduce complexity (Layer, Node)
   - Increase data
3. Normalization (fast training)
   - Normalization / Std.
   - Batch Normalization
4. Optimizer
   - Momentum
   - Adagrad
   - Adadelta / RMSProp
   - Adam
5. HyperParameter
   - No. of hidden layer
   - Node / Neuron
   - Batch size
   - Learning rate
6. Activation
   - Sigmoid
   - tanh
   - type Relu
   - Softmax
   - Linear / Step

Data Scaling

Age

[1-95]

Salary

[1 - 100000 00]

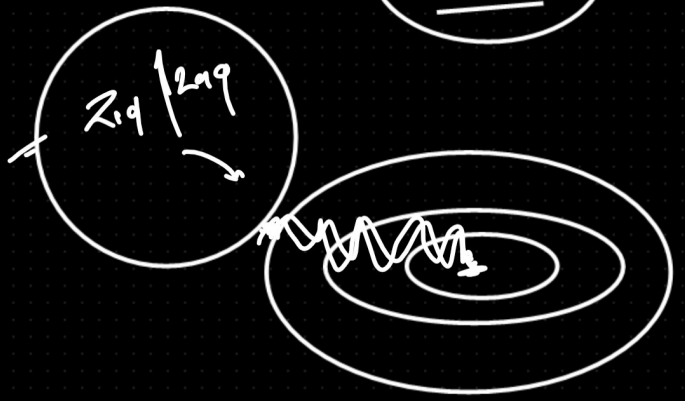
Keep the data in a same scale

= Input x weight + bias

Age x w_1 + Sal x w_2 + bias

50 2 100000

Contour



1. Std

$$\frac{X_i - \mu}{\sigma}$$

[-3, +3] z-table SND

Salary

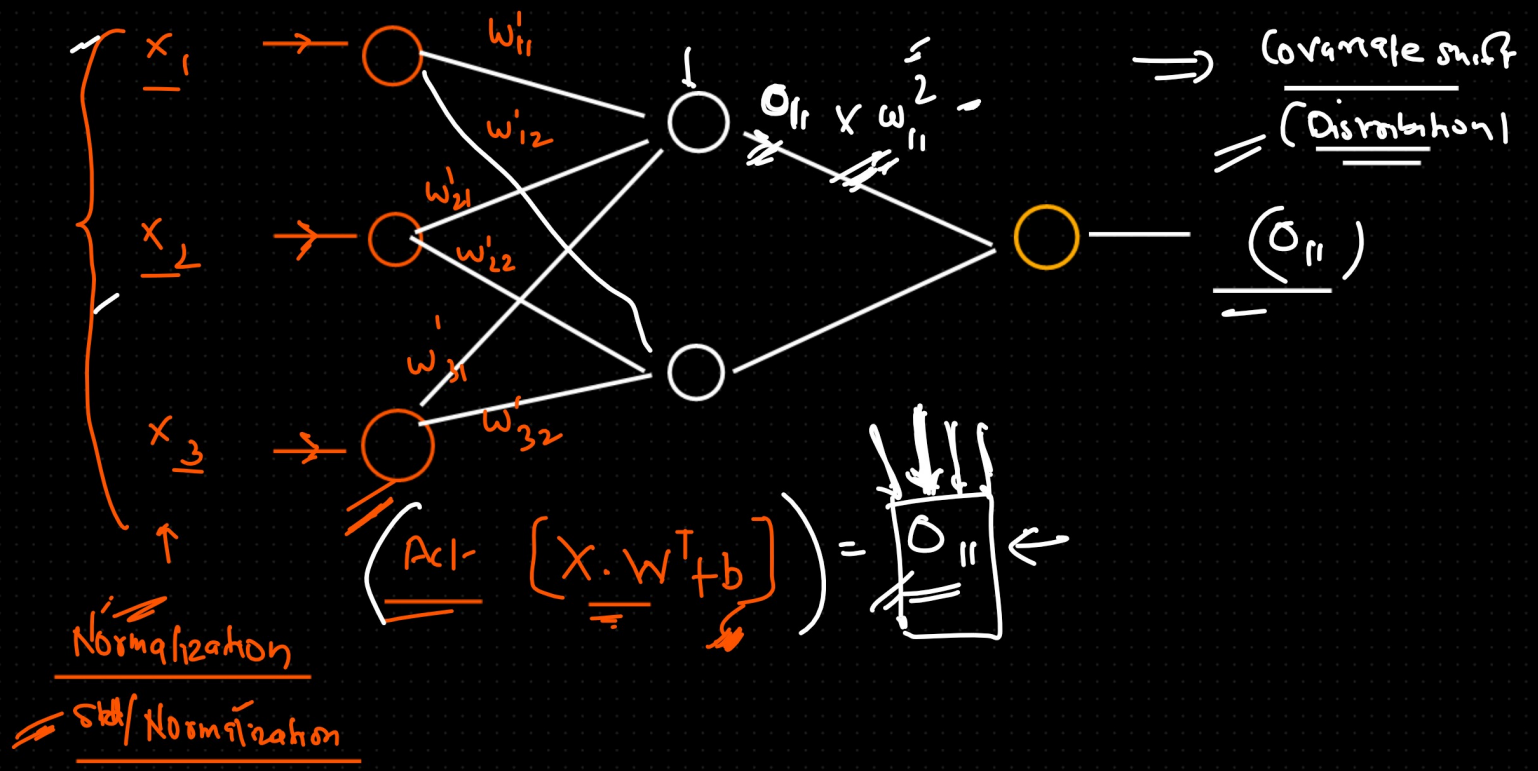
2. Normalization

$$\frac{X_i - X_{min}}{X_{max} - X_{min}}$$

[0-1]

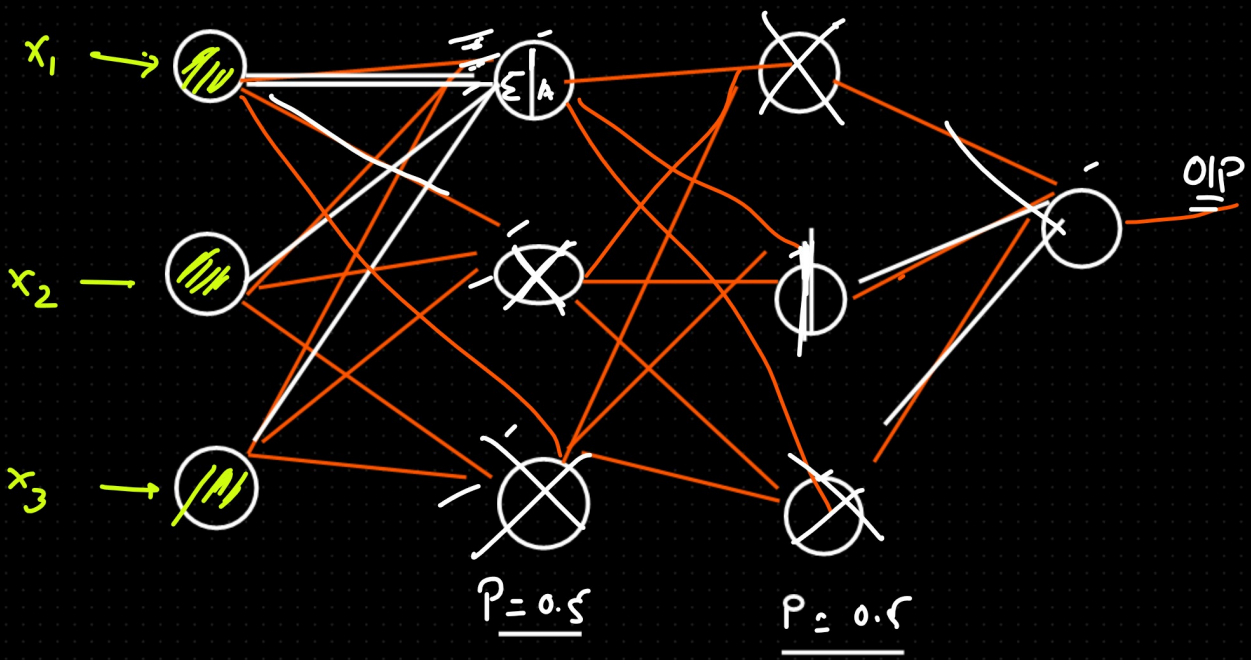
CGPA

(0-10)

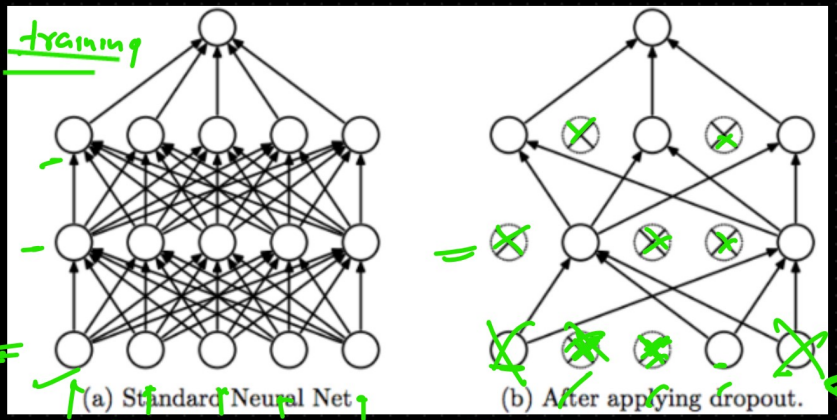

x_1
x_2
x_3
w'_{11}
w'_{12}
w'_{21}
w'_{22}
w'_{31}
w'_{32}
$O_{11} \times w^2_{11}$
Covariate shift
(Distribution)
(O_{11})
Act- $[X \cdot W^T + b]$ = O_{11}
Normalization
Std/Normalization

Drop out-

Training (1 epoch)



5 epoch



Dropout → 0.5 → 50%

→ 0.30 = 30%

→ 0.25 = 25%

→ 0.5

Shutdown

Randomization

ff-

Regularization ⇒

- L_1 → Lasso ← $w \to 0$ ⇒ Sparsity
- L_2 → Ridge → toward to 0 (not completly zero)
- $L_1 + L_2$ → elastic

Overfitting

$$\text{Loss} + \text{Penality}$$

↳ $L_1 / L_2 / L_1 + L_2$

$$L_1 \rightarrow \text{Loss} + \lambda |w|$$

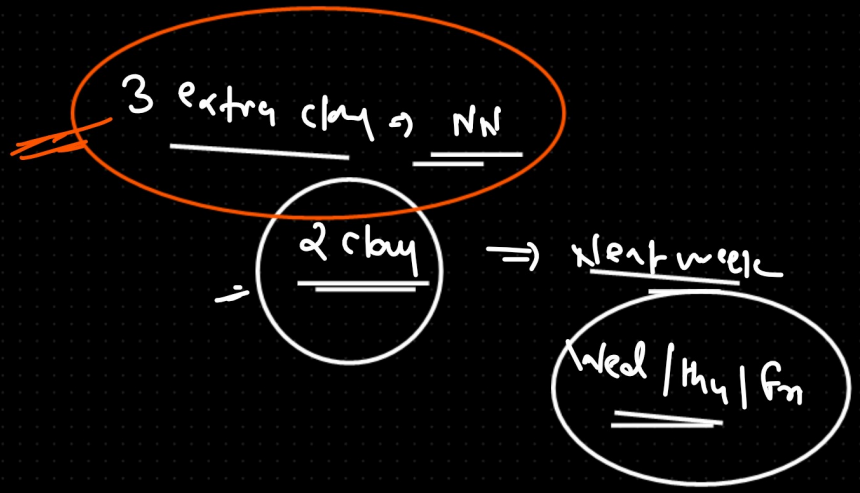
$$L_2 \rightarrow \text{Loss} + \lambda |w|^2$$

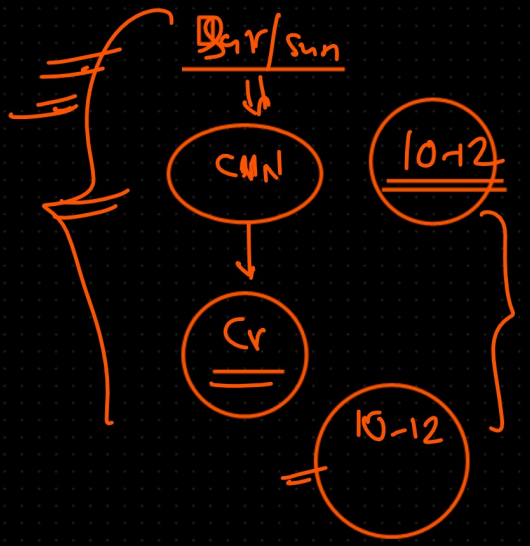
$$L_1 + L_2 \rightarrow \text{Loss} + \lambda(|w1|) + \lambda(w2)$$

} Single value

Cost

$$\text{Loss} + \lambda \sum_{n=1}^{n} |w|^2$$


3 extra classes NN
2 days
Next week
Wed / Thu / Fri


Sat/Sun
CMN
10-12
Cr
10-12