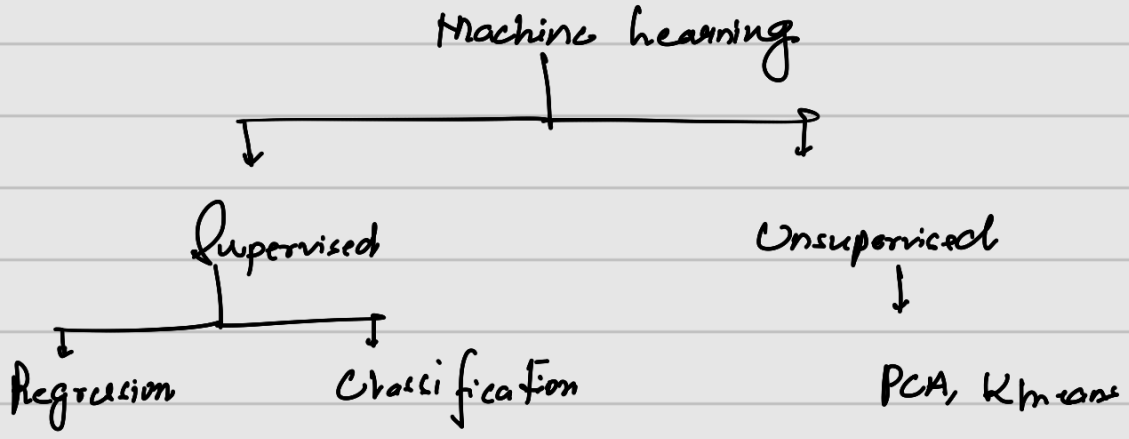
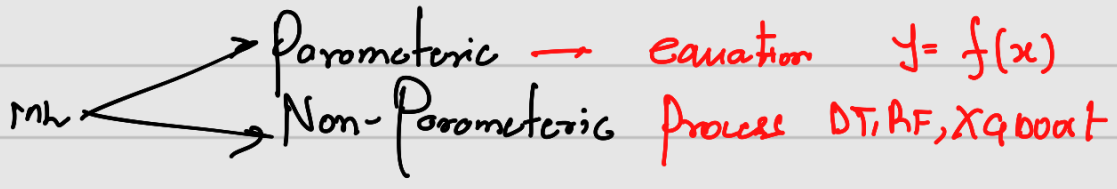
Introduction to Deep learning:



Linear Reg
Log Reg
DT, RF, Boosting



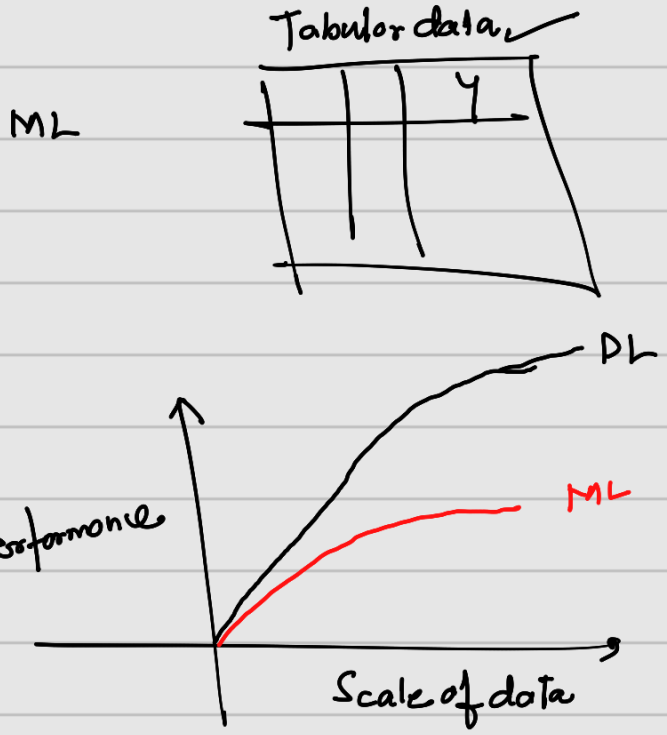
Parametric → equation $y = f(x)$
Non-Parametric process DT, RF, XGBoost

Deep learning → Parametric form.

$y \sim f(x)$

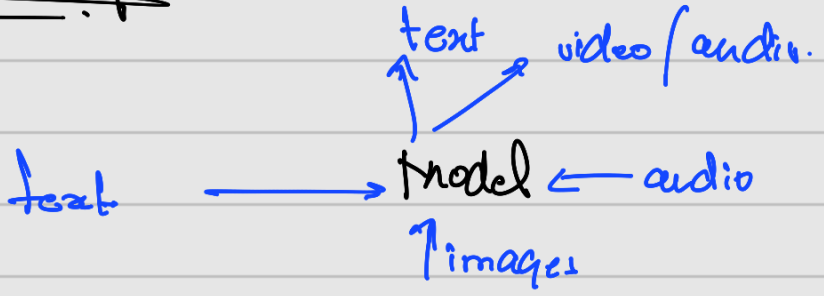
↓

equation



automated feature engineering → Images, videos, text, Audio. graph, etc.

Multimodality



Deep learning a framework wherein we create a parametric model to predict y as a function of x.

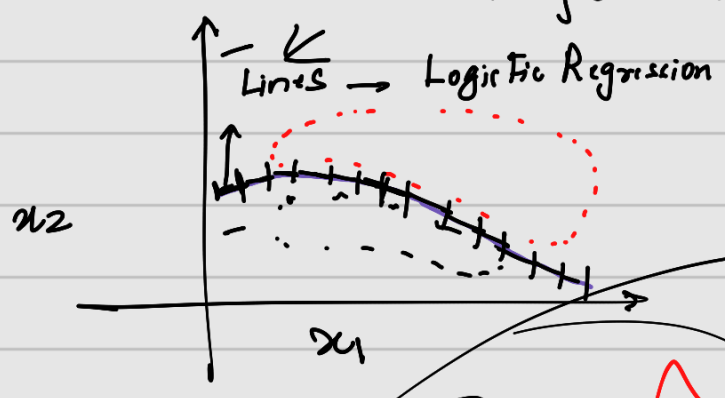
$y \leftarrow (f)(x)$

- Tabular data → Artificial Neural Network
- Image → Convolution neural network
- Audio → Sequence (RNN, LSTM)
- Text
- Graph → Graph neural network

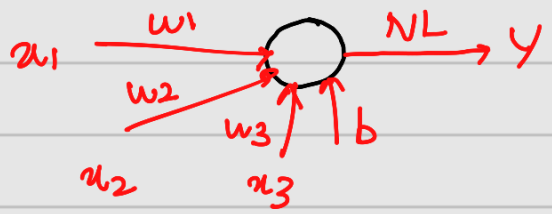
Lecture-2 What is a Neuron?

$y \approx f(x_1, x_2)$

Lines → Logistic Regression



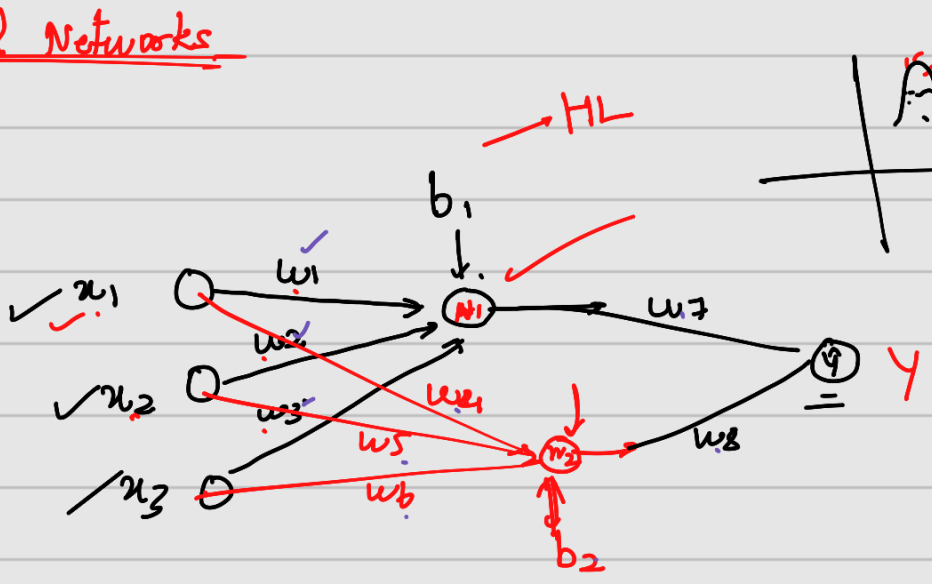
- intercept
- Slope
- Curvature



$$y = NL(w_1x_1 + w_2x_2 + w_3x_3 + b)$$

activation functions

Neural Networks



$$\hat{y} = \text{sigmoid}(w_7N_1 + w_8N_2 + b_3)$$

x_1	x_2	x_3	y

I want my $\hat{y}$ to go as close as possible by modifying the weights and biases.

- Define the architecture.
- weights and biases → randomly initialize.
- $\hat{y}$ ($\hat{y} \Rightarrow y$) → loss
- Reduce the loss by modifying the weights and biases.

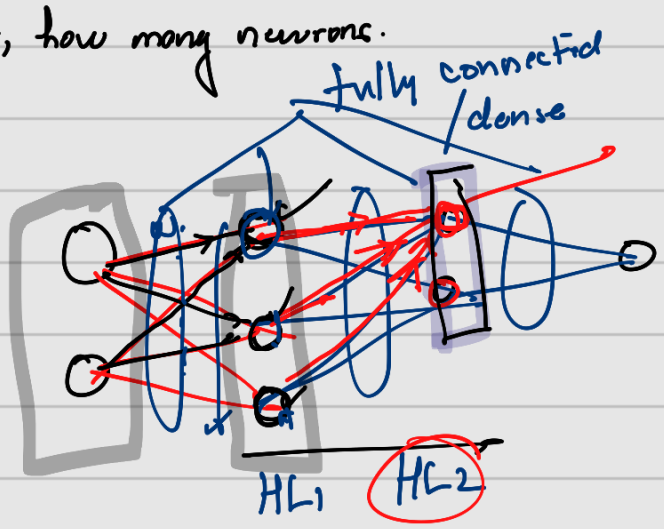
↓

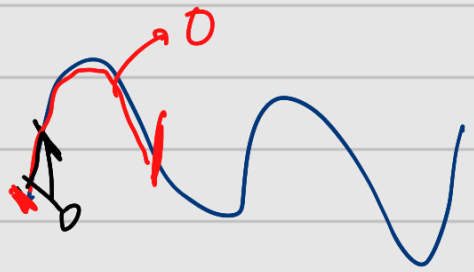
Gradient Descent

How to train Neural networks:

x_1	x_2	y

- Define my neural net architecture.
  How many HL, how many neurons.

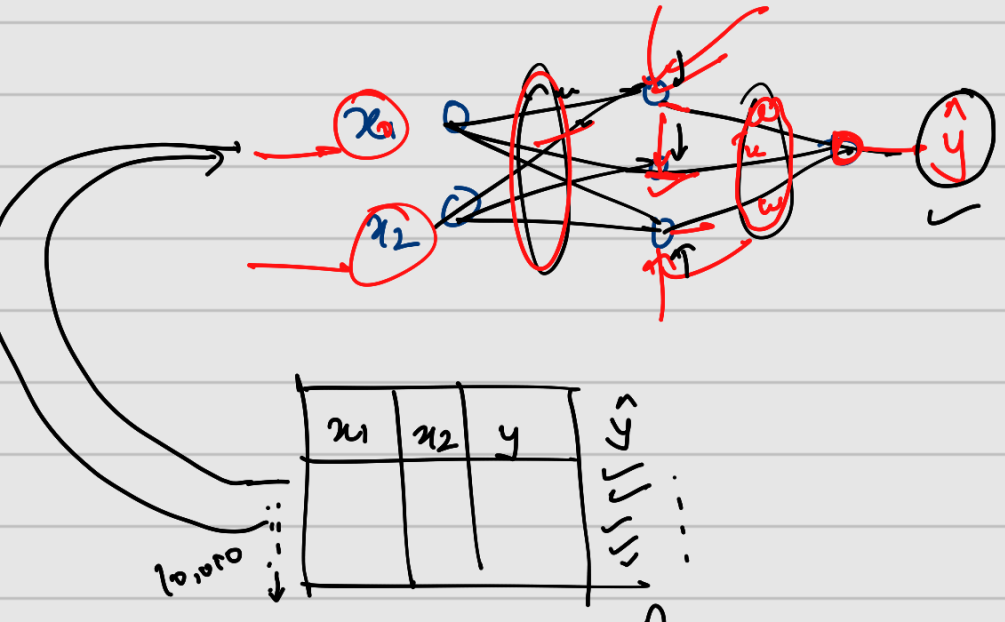




→ add more degree of non-linearity.

→ all weights and biases will be randomly initialized.

→ given these weights, I will multiply the weights to the inputs add the biases



Regression

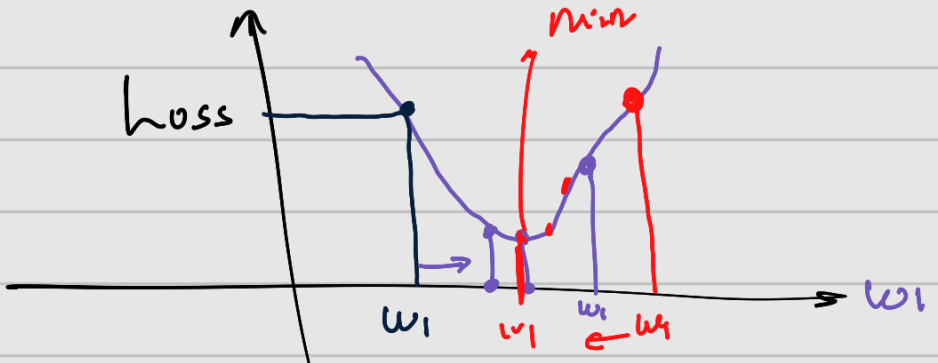
$$\text{loss} = \sum_{i=1}^{n} \frac{1}{n} (y_i - \hat{y}_i)^2$$

Forward Propogation

Based on the loss, we find out better set of weights and biases.

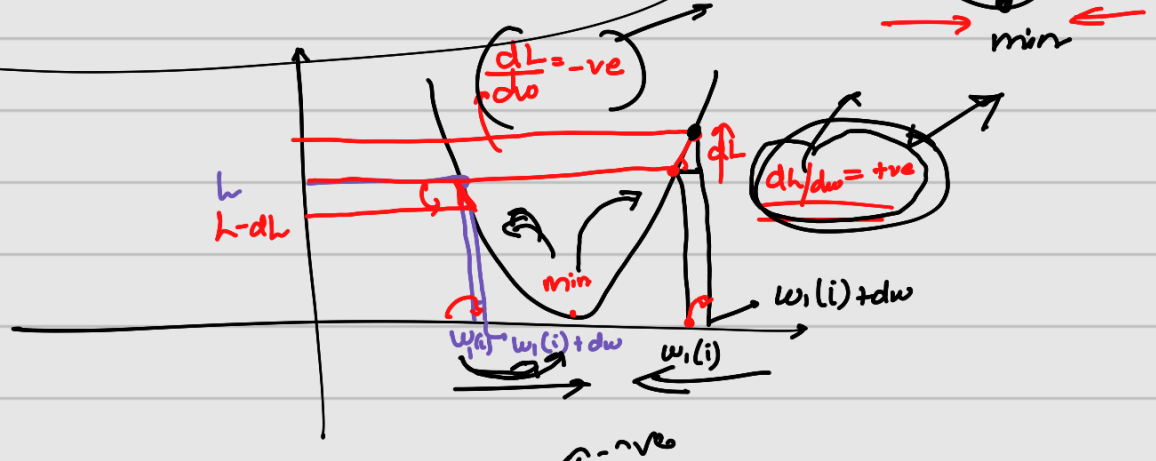
$$loss = \sum_{i=1}^{n} \frac{1}{n} (y_i - \hat{y}_i)^2$$

$$loss = f(w, b)$$



for all weights and biases.

$$w_1(i+1) = w_1(i) - \alpha \frac{dh}{dw(i)}$$



$$w_1(i+1) = w_1(i) - dh/dw(i)$$

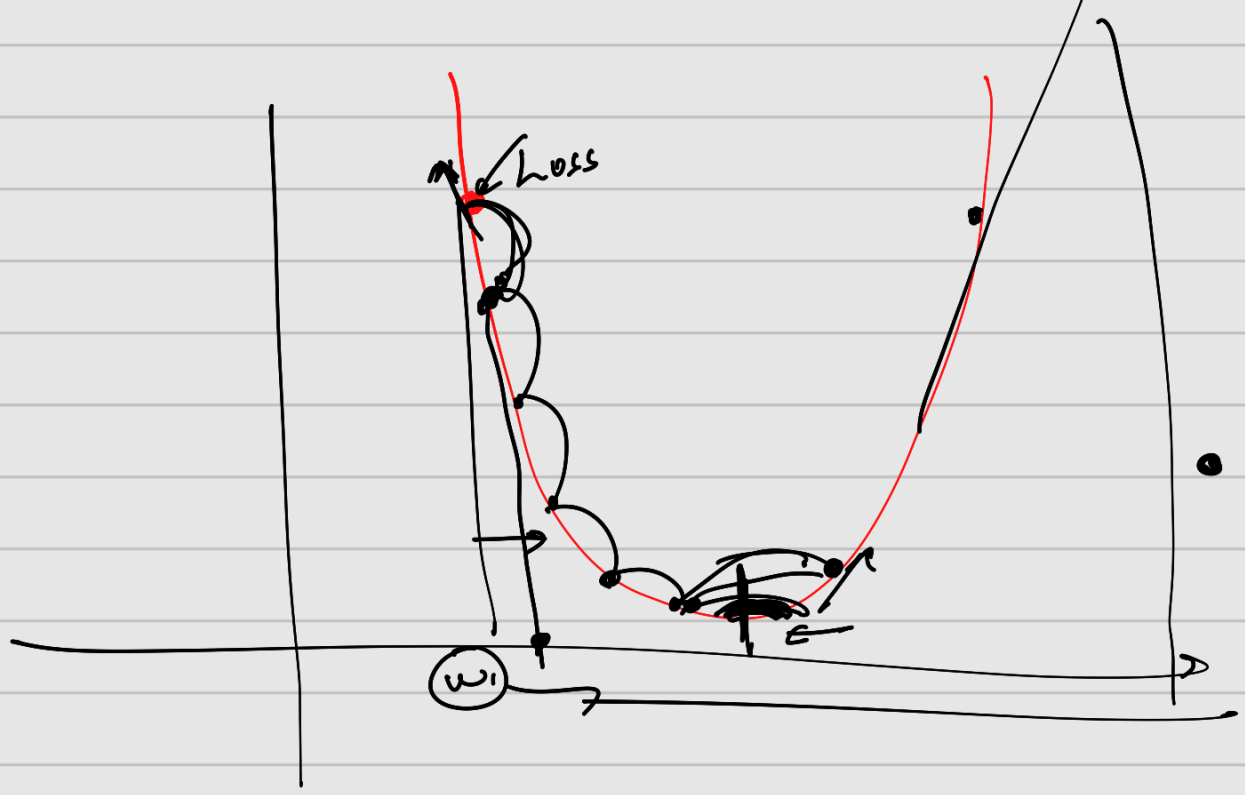
(-nve)

$w_1(i)$ + amount

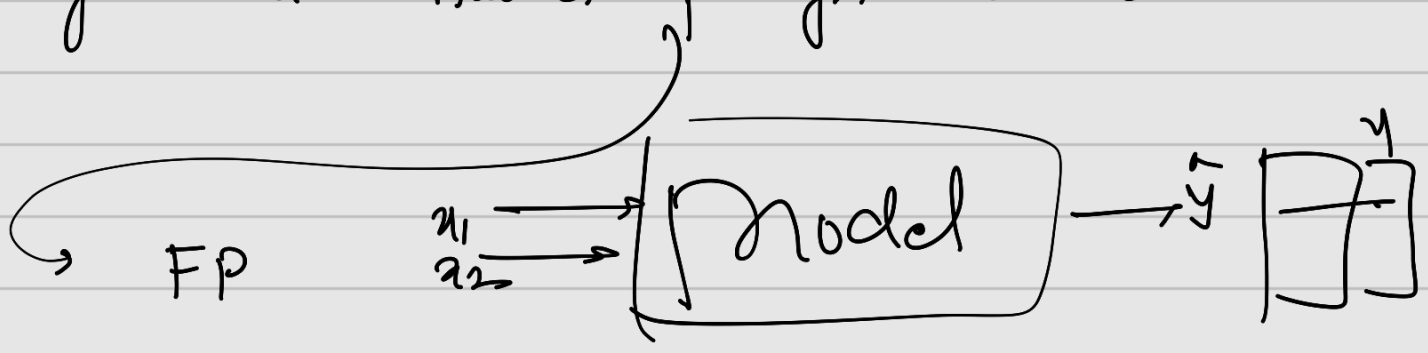
$$w_1(i+1) = w_1(i) - (dh/dw)$$

(+ve)

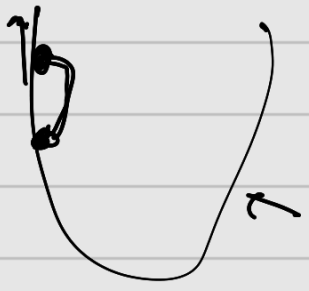
$w_1(i)$ − amount



g will have new set of weights and biases



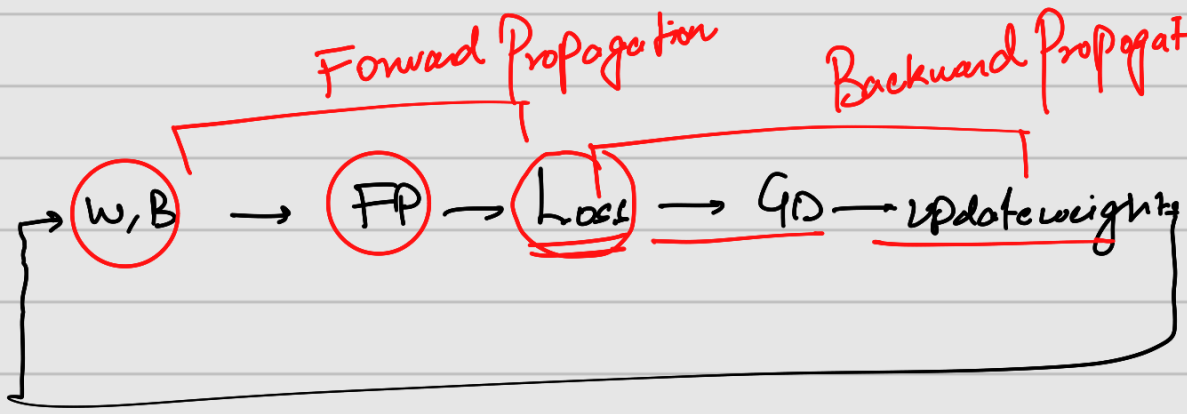
loss



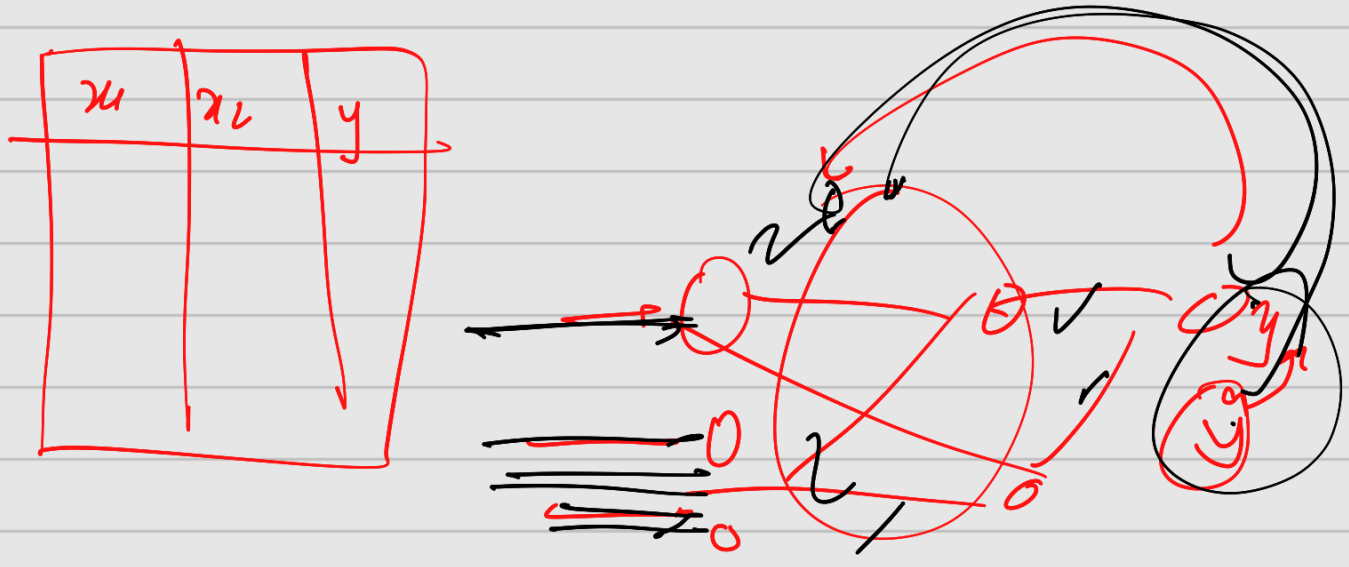
gradient descent

$$w_j(i+1) = w_j(i) - \alpha \frac{\partial h}{\partial w_j(i)}$$

j for all weights

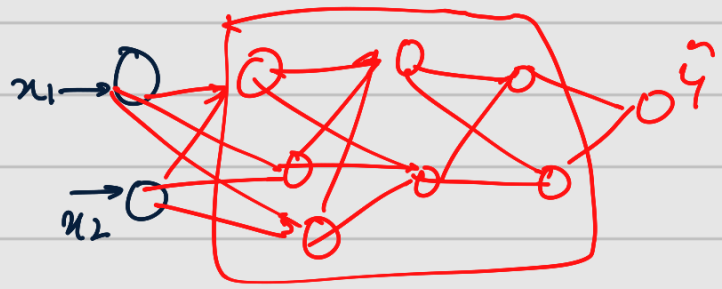

Forward Propagation
Backward Propagation
W, B
FP
Loss
GD
update weights


Loss
w1 ; w2, w3


x1
x2
y

FP, BP → Epoch.

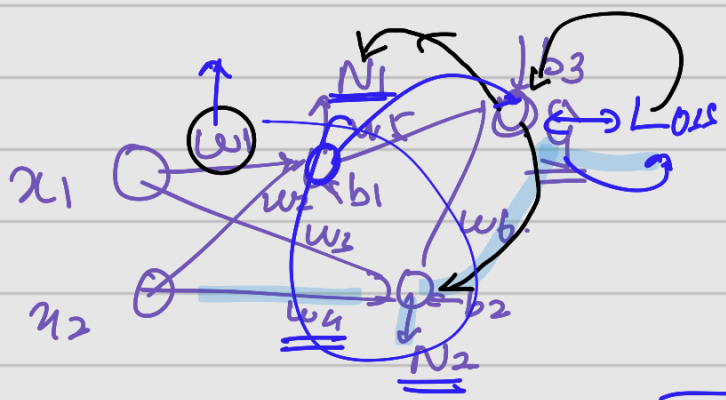
Chain Rule:



$$w_i(t+1) = w_i(t) - \frac{\partial L}{\partial w_i(t)}$$ for all is

$$\left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4} \ldots\right) \left(\frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial b_2} \ldots\right).$$

Regression



$$N_1 = w_1 x_1 + w_2 x_2 + b_1$$

$$N_2 = w_3 x_1 + w_4 x_2 + b_2$$

$$\hat{y} = w_5 N_1 + w_6 N_2 + b_3$$

$$\text{loss} = \sum_{i=1}^{n} \frac{1}{n} (y_i - \hat{y}_i)^2$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial N_2} \times \frac{\partial N_2}{\partial w_4}$$

$$\frac{\partial L}{\partial \hat{y}} = \sum_{j=1}^{n} \frac{1}{n} 2 \times (y_i - \hat{y}_i) \times -1$$

$$\frac{\partial \hat{y}}{\partial N_2} = w_6 \qquad \frac{\partial N_2}{\partial w_4} = x_2$$

$L = f(w, b)$

$$\frac{\partial L}{\partial w_1} = ? \qquad \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial N_1} \times \frac{\partial N_1}{\partial w_1}$$

$m = x^2$

$z = m^2$

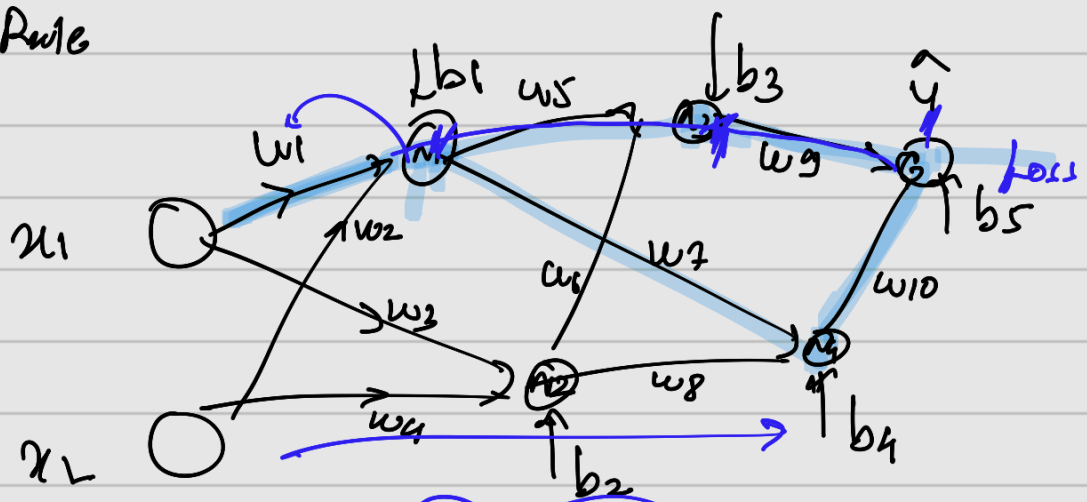
$y = z^3$

$$\frac{dy}{dz} = 2z$$

$$\frac{\partial y}{\partial x} = \frac{dy}{dz} \times \frac{dz}{dm} \times \frac{dm}{dx}$$

$$\frac{\partial y}{\partial x} = 2z \times 2m \times 2x$$

Chain Rule



IHL → 3

2HL → 4

100HL → 102

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial N_3} \times \frac{\partial N_3}{\partial N_1} \times \frac{\partial N_1}{\partial w_1} + \cdots$$

$$\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial N_4} \times \frac{\partial N_4}{\partial N_1} \times \frac{\partial N_1}{\partial w_1}$$

$(0.99)^{365} = 0.\ldots0$

$(1.01)^{365} \to \infty$

Vanishing gradient.

Exploding gradient.

Activations

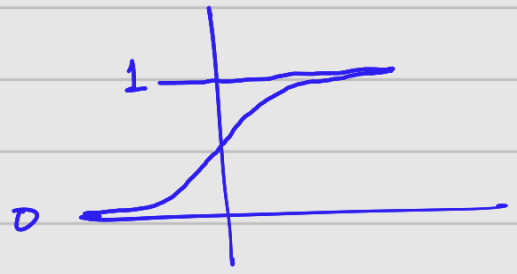
- Sigmoid.
- Relu (Rectified Linear Unit).
- Leaky Relu (Parametric ReLu).
- tanh (hyperbolic tangent).
- Softmax.

Sigmoid activation:

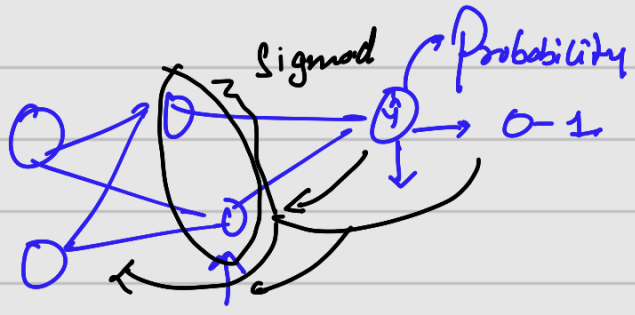
$$Y = B_0 + B_1 x_1 + B_2 x_2 \cdots B_p x_p$$

$-\infty, \infty$

$$\sigma(Y) = \frac{1}{1 + e^{-Y}}$$



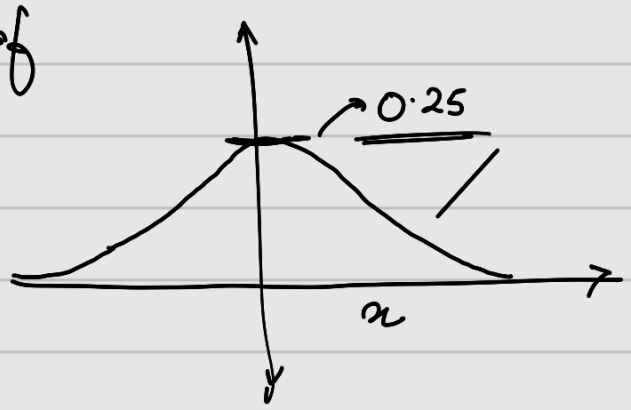
Binary Classification



$$w_{i+1} = w_i - \alpha \partial J / \partial w$$

$$\sigma'(x) = \sigma(x) \times (1 - \sigma(x))$$

Derivative of Sigmoid



d. x..x ....

ReLu

$Relu(x)$ $\quad x>0 \quad x$

$x \leq 0 \quad 0$

$Relu'(x) = \quad x>0 \quad 1$

$x \leq 0 \quad 0$

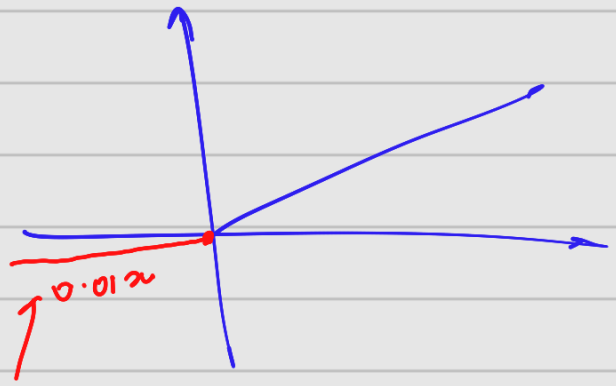
Parametric Relu.

$Relu(x) \quad x \quad x>0$

$0 \quad x=0$

$\alpha x \quad x<0$

$0 < \alpha < 1 \rightarrow$ learnt during the training



Leaky Relu

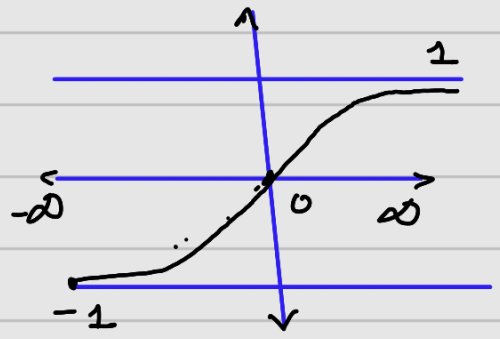
$x > 0 \rightarrow x$

$x = 0 \rightarrow 0$

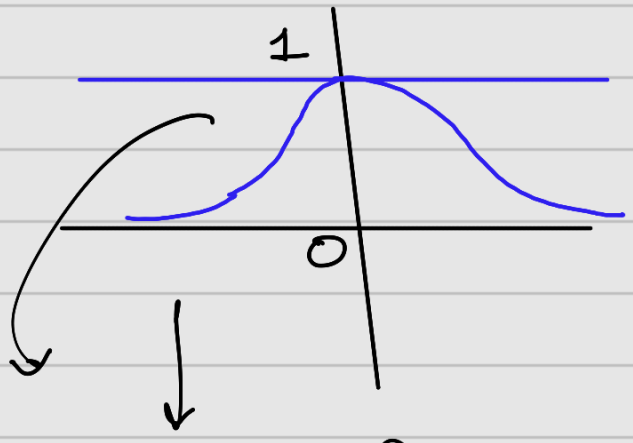
$x < 0 \quad 0.01x$

Hyperbolic Tangent (tanh).

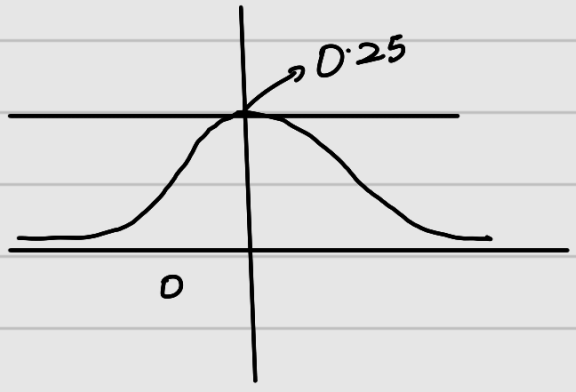
$$\tanh(x) = \left[\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right] =$$



$$\tanh'(x) = 1 - \tanh^2(x)$$

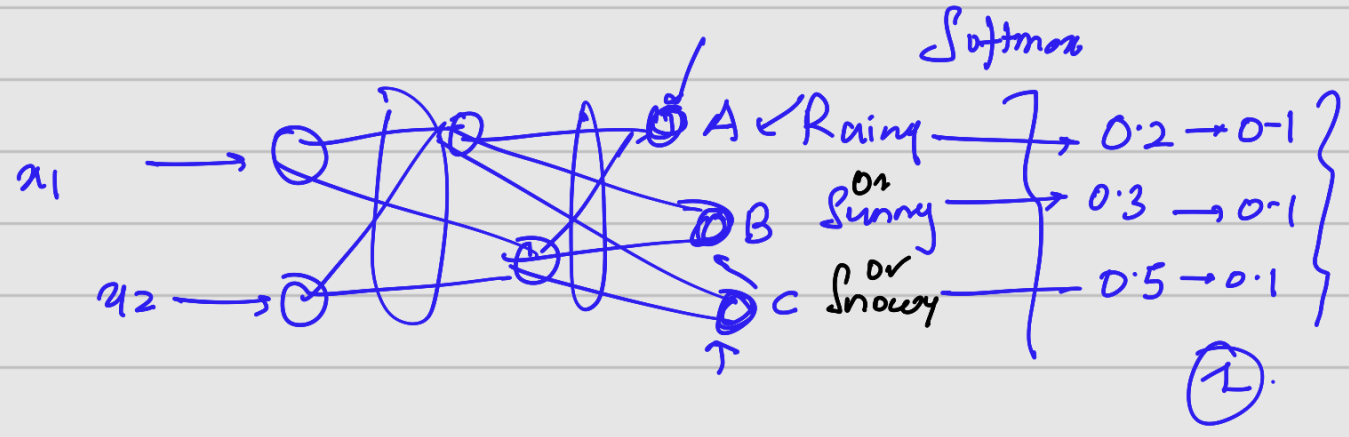


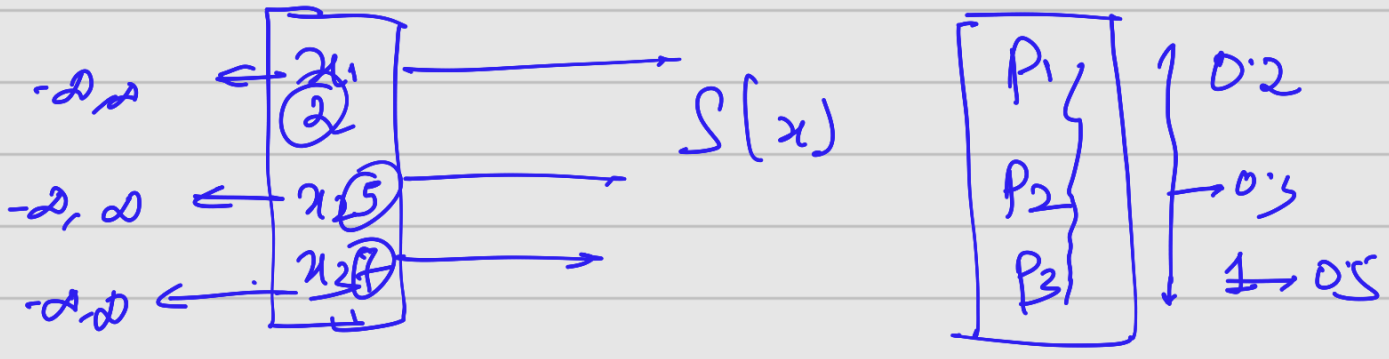
Vanishing gradient problem ✗



Softmax :

Sample / row





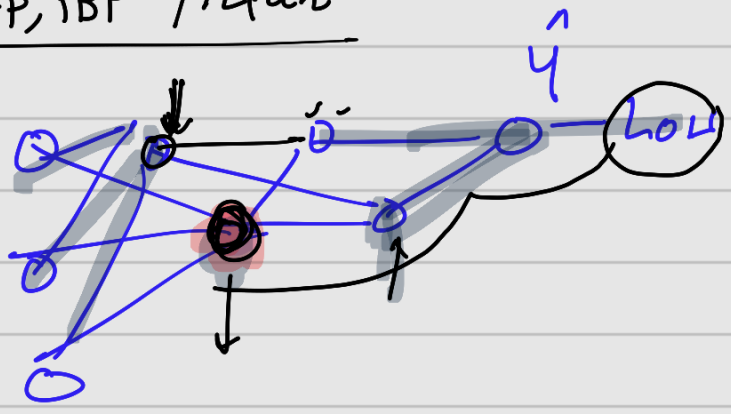
$$P_3 = \frac{e^{x_3}}{e^{x_1}+e^{x_2}+e^{x_3}} \qquad P_1 = \frac{e^{x_1}}{e^{x_1}+e^{x_2}+e^{x_3}} \qquad P_2 = \frac{e^{x_2}}{e^{x_1}+e^{x_2}+e^{x_3}}$$

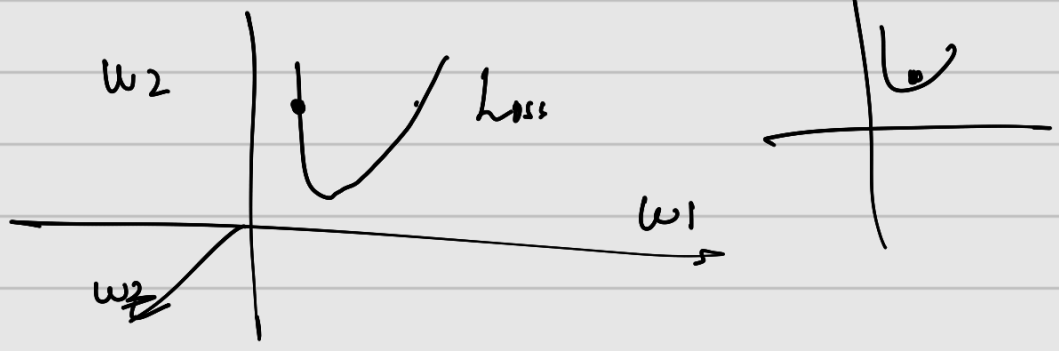
- Dropout
- Regularization
- optimizers
- weight initialization
- Callbacks
- Batch Normalization

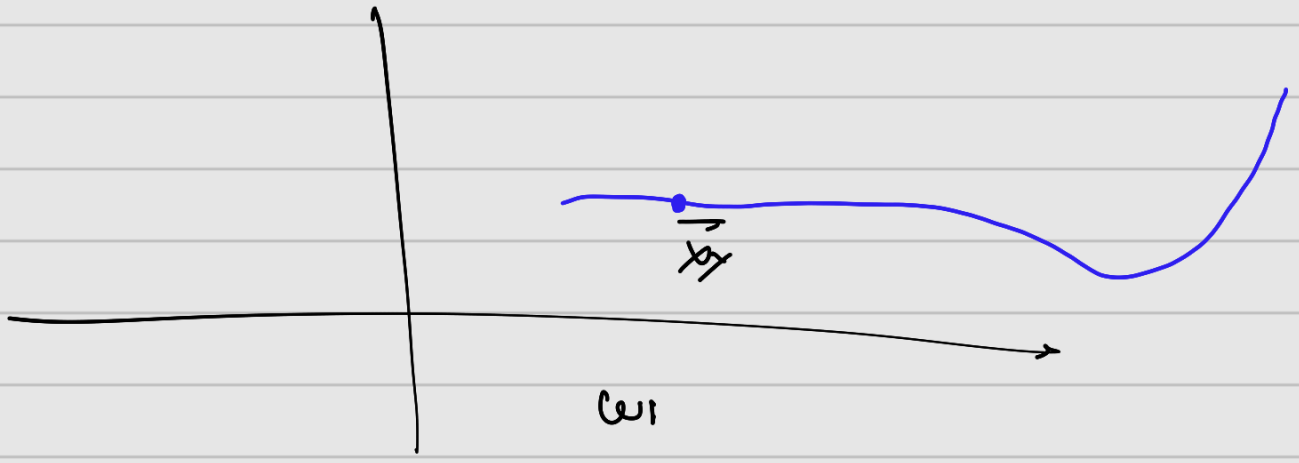
→ Hyperparameter Tuning.

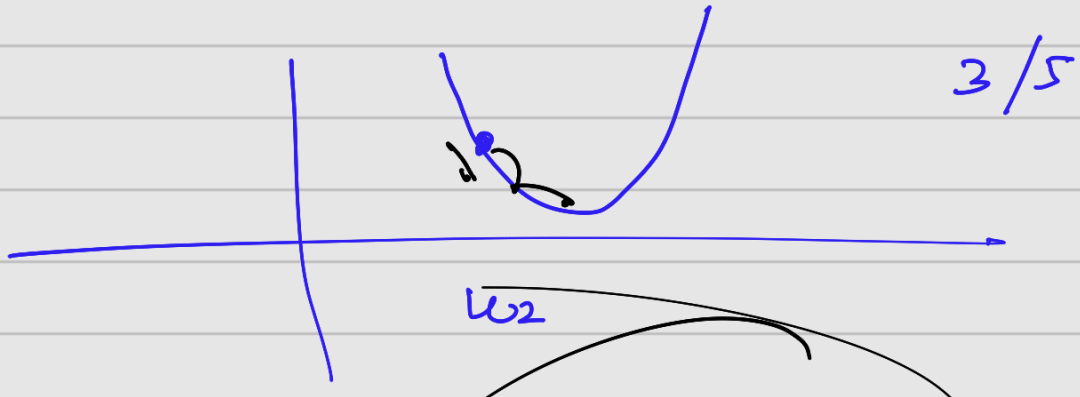
Dropout:

1FP, 1BP / 1epoch





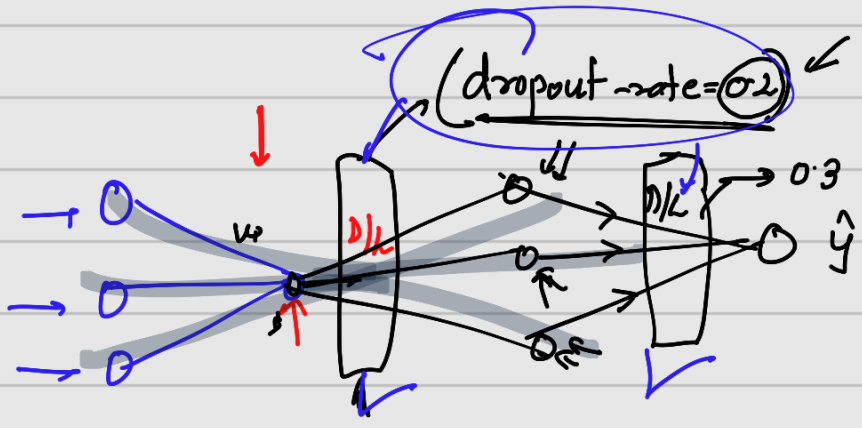




3/5

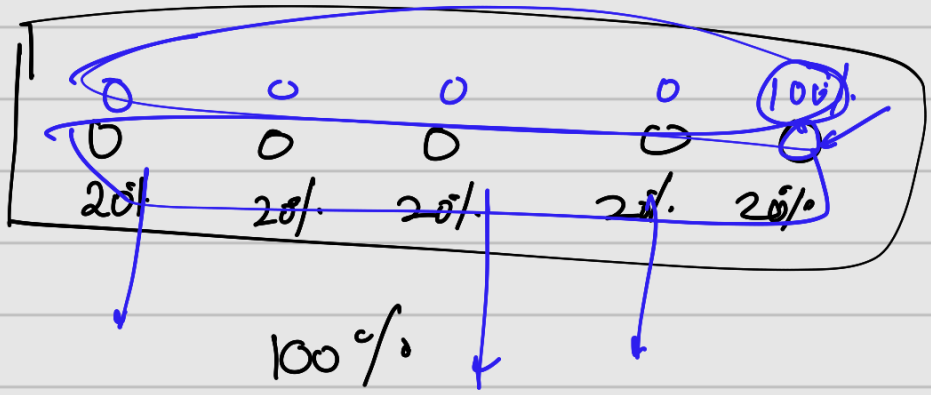
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

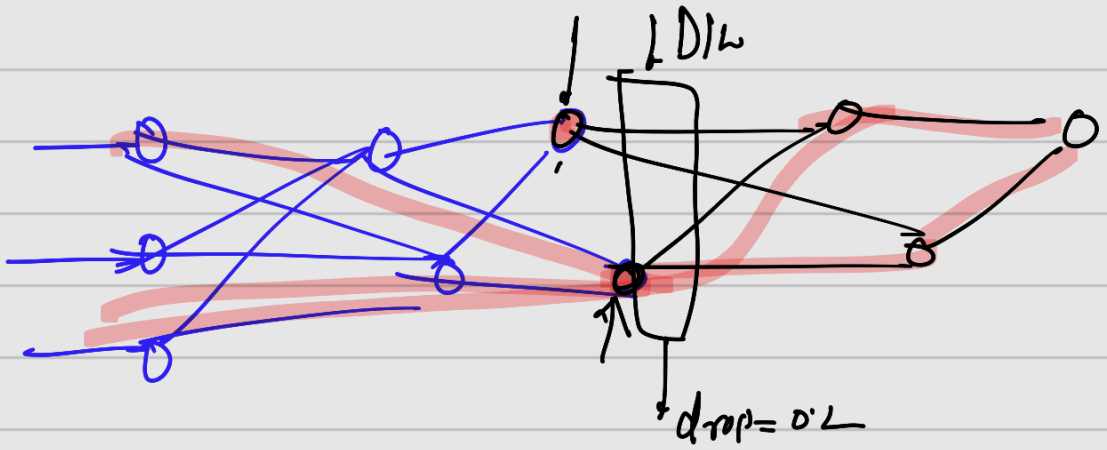
IFP, IBP



H Kep 0.8

T Drop. 0.2


20%
20%
20%
20%
20%
100%


LDIL
drop=0.2