Agenda (NN):

1/2/3
- Why DL / What is DL, What is ANN.
- What is a Neuron, Why ANN / DL
- What is MLP
- Forward / Backward Propagation.
- Training steps of a ANN.

4/5/6
- implement NN in code. Tensorflow / Keras
- Dropout, Regularization, Batch Normalization, losses, Callbacks. etc., optimizers

7/8/9
- Code implementation, hyperparameter tuning, optimization in NN.
- Explainability / Autoencoding (Denoising).

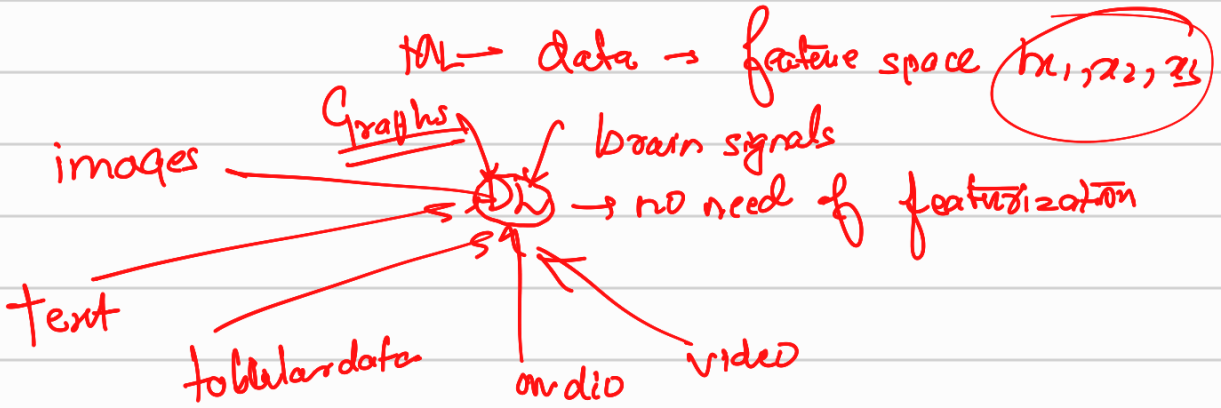
What is Deep learning, Why DL, vs ML.

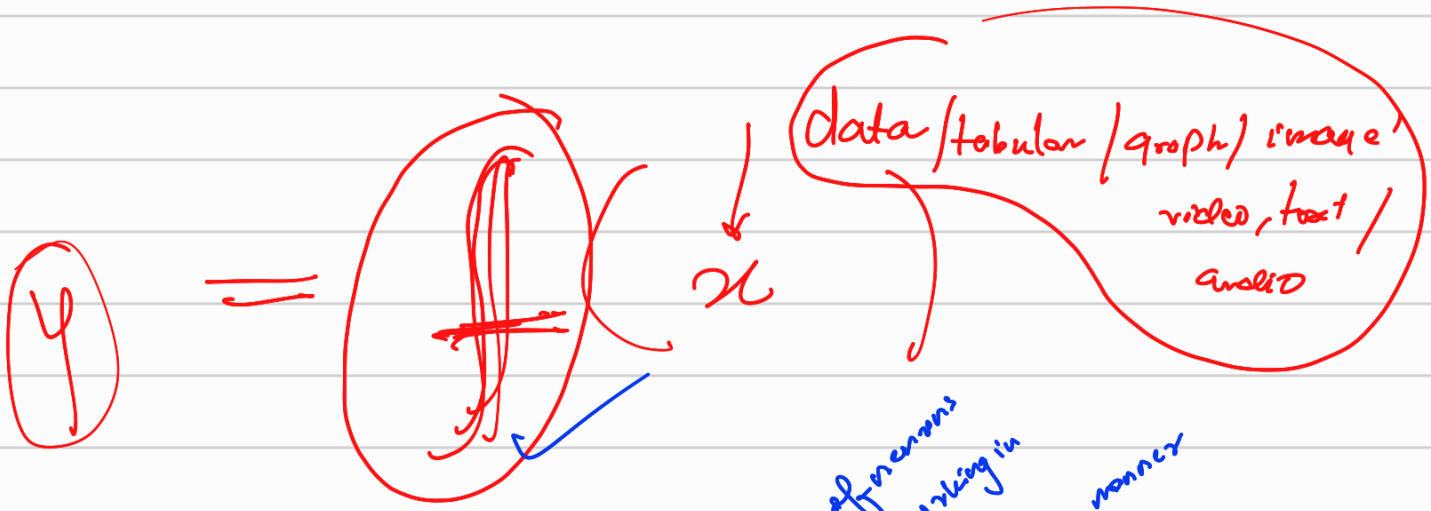
DL

DL

Deep learning is basically a field of study where in we use something known as neurons for training the data.

Automated feature engineering.



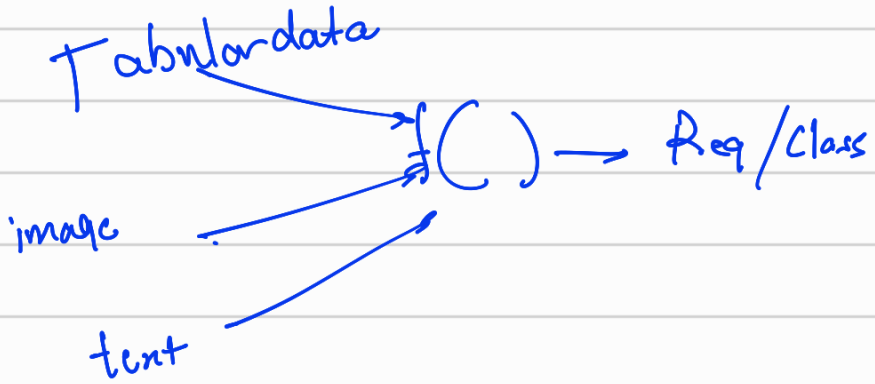


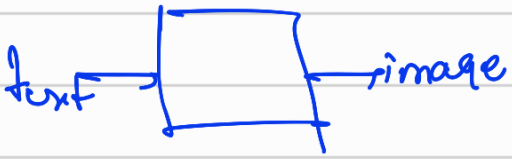
DL techniques—

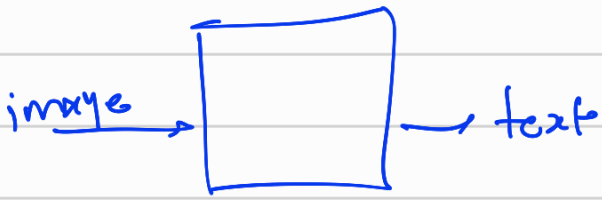
Deep Learning is basically a parametric approach to learn your data (equation based)

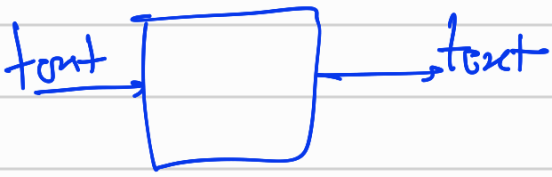
- Artificial Neural Networks- NN is designed to etract information from tabular data
- Convolution Neural Networks- NN is designed to extract information from image data
- Recurrent Neural Networks (LSTM/GRU?Transformers)- NN is designed to extract information from sequence data (like text, audio, videos)
- Graph Neural Network for Graphs (molecule)

Multi Modality:

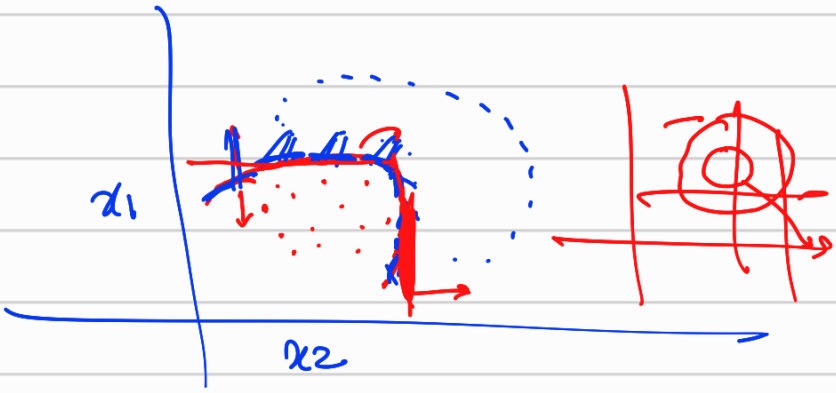




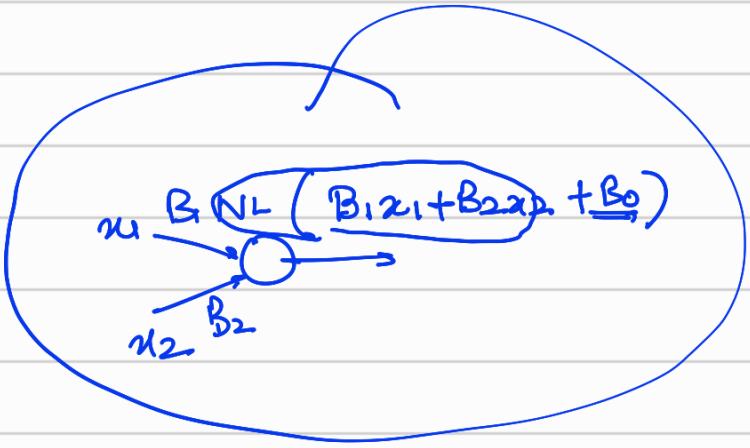


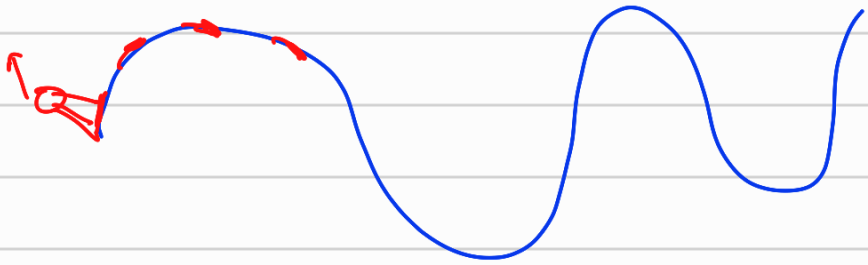


what is Neural network:



Can i fit a decision boundary along the red curve which can be thought of multiple connected curves, and each curve has the independence to move up or down and has the liberty to change o the amount of curvature





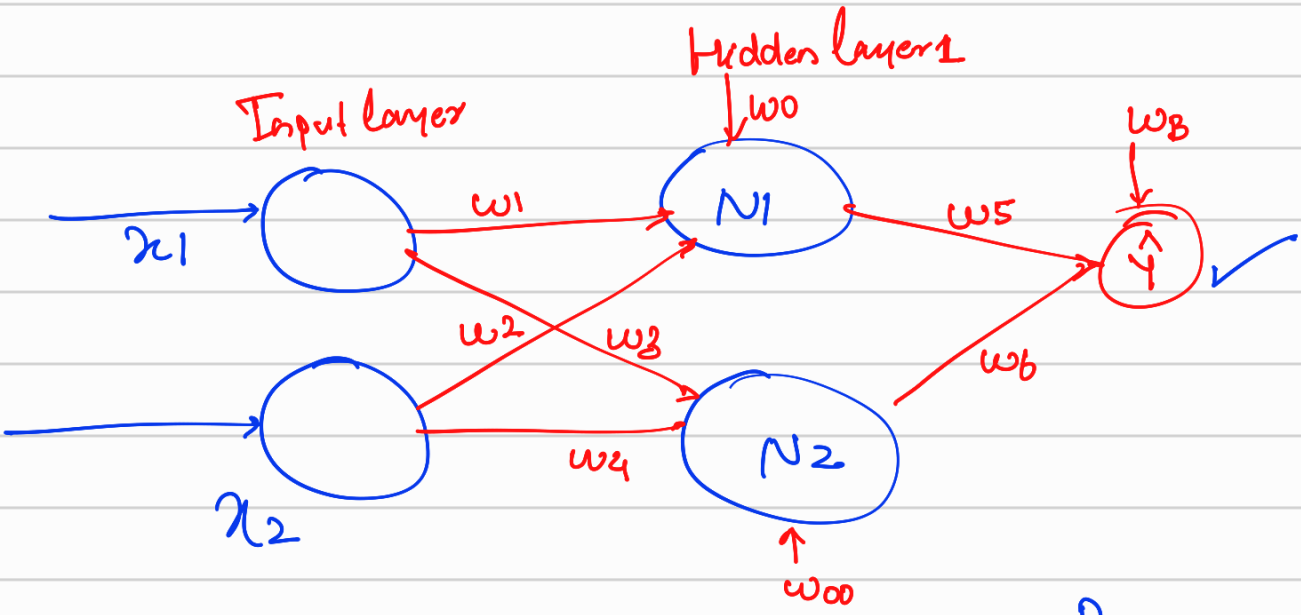
22:55 pm Break

15 – 20 mins Proving why non-linearity is important

① Why the non-linearity is absolutely vital.

① Gradient descent → underlying algo for training the NN.

Input layer



Biases

$$N_1 = \tanh(w_1 x_1 + w_2 x_2) + w_0$$

$$N_2 = \tanh(w_3 x_1 + w_4 x_2) + w_{00}$$

Biases

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

$$\hat{y} = w_5 (w_1 x_1 + w_2 x_2 + w_0) + w_6 (w_3 x_1 + w_4 x_2 + w_{00}) + w_B$$

$$\hat{y} = w_5 w_1 x_1 + w_5 w_2 x_2 + w_5 w_0 + w_6 w_3 x_1 + w_6 w_4 x_2 + w_6 w_{00} + w_B$$

$$\hat{y} = x_1 (w_5 w_1 + w_6 w_3) + x_2 (w_5 w_2 + w_6 w_4) + w_B + w_6 w_{00} + w_5 w_0$$

$$\boxed{\hat{y} = x_1 A + x_2 \times B + C}$$

How do we train such a network of neurons:

1. Initialise the architecure (define the number of hidden layers, neurons etc.)
2. Randomly initialise the weights
3. Given the weights calculate the loss..-- → Forward Propogation
4. Given the loss, update the weights- Gradient Descent Algorithm +Chain Rule → BP (Backward Propogation)
5. using the updated weights go to step 3 and keep on doing it till you stabilise your training loss to the min possible (min Val loss)

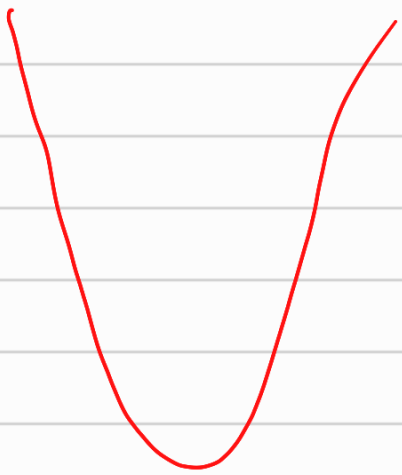
$$loss = \sum_{i=1}^{n} \frac{1}{n}(y_i - \hat{y}_i)^2$$

$$loss = f(???)$$

(biases, weights), label, input(x)

minimise the Loss

Gradient Descent Algorithm and Chain Rule:



x_1

x_2

$\boxed{x_1 \times x_2}$

$y\ (x_1 \times x_2)$

100 neurons

$\log x_1$ $\log x_2$ 0

1 neuron

$x_1 \times x_2 = \boxed{\log x_1 + \log x_2}$