Neural Network and Fuzzy Logic
CSE 451

Neural Network and Fuzzy Logic Laboratory
CSE 452

1st July, 2025
First Class.

→ Building the architecture to make learn the machine ⟶ Neural network.

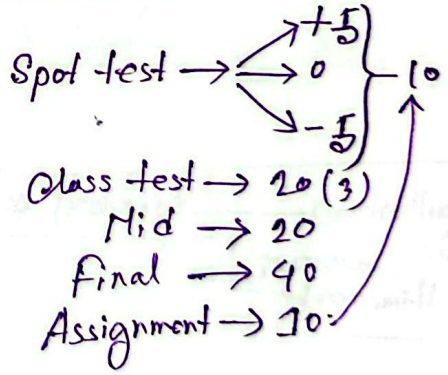
Neural ⟶ Multiple Neuron (Biological system)
↓ makes
Neural Network

Fuzzy (Represents uncertainty) on (Partial truth/ Partial false)
⟶ value lies between 0 and 1
(Maybe I would attend the class)

Problem with binary logic:

Can't deal with logic between Yes/No.

Marks distribution:



Outline:

→ Human brain — how it works?
→ Perceptron — how it learns?
→ Linear regression
→ Logistic regression.
→ Gradient descent
→ forward propagation
→ Backward propagation

→ Activation function
→ Performance matrices, (Accuracy, Precision Recall.)
→ Advanced NN (CNN, RNN, LSTM)
→ Recent (eg, like transformer, bert, CLIP)

13.07.25 – 2

Biological neuron:

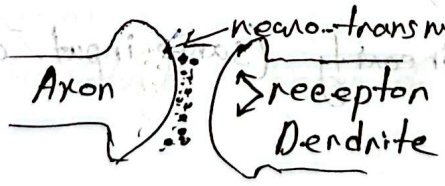
* Machine doesn't have massive parallelism unlike human. (multitasking)

* Humans have:
  - → distributed representation and computation.
  - → learning ability
  - → generalization ability
  - → adaptibility.
  - → inherent contexual info. processing.
  - → fault tolerance.
  - → low energy consumption.

* Biological Neuron is a special biological cell, essence of life, has information processing ability. (10^{11})

one neuron is connected to other 10^3–10^4 ones.

neuron — transmits impulse through neuro-transmitter.



* learning — controlling neuro-transmitter

Synapses — transmit signal to next neuron.

Dendrite —

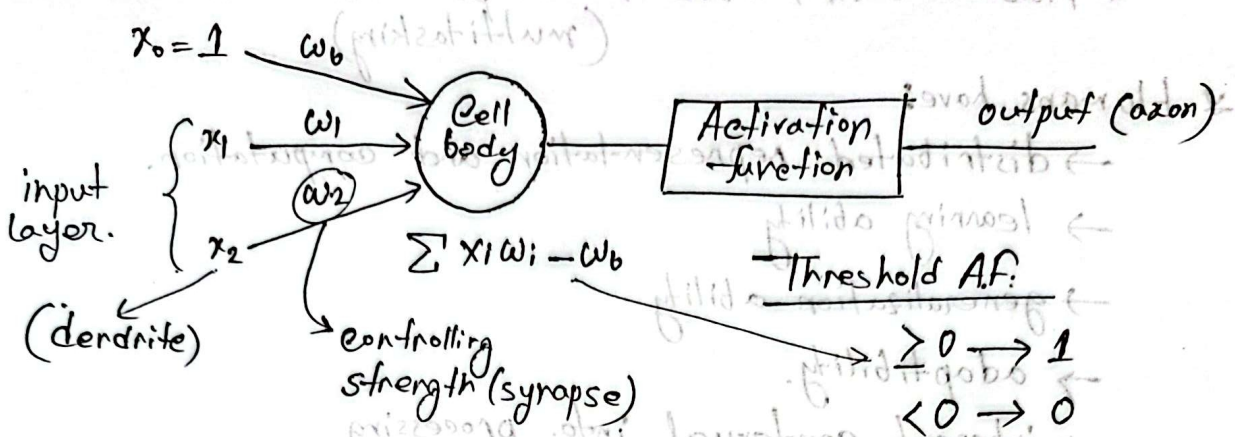
Soma —

Axon —

Synapse —

McCulloch Pitt (Artificial neuron)

Neuron model.



Learning rate:

Activation function:

- Gaussian A.F.
- Sigmoid A.F.
- Piecce-wise A.F.
- ReLU A.F

★ Role of the BIAS in Neural Networks:

→ bias value allows to shift the A.F to the left or to the right,

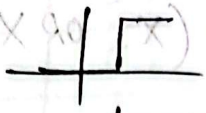
→ introduces non-linearity (same input can causes diff. output)

$y = mx + c$ — helps in shifting

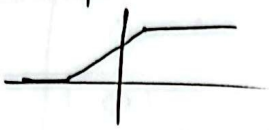
$f = \omega_i x_i - \omega_b$

Diff. type of Activation function:

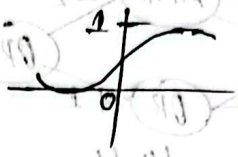
i) Threshold A.f.



ii) Piecewise A.f.

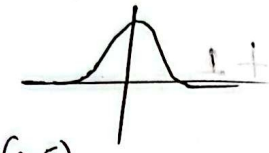


iii) Sigmoid A.f., $y = \frac{1}{1+e^{-x}}$,

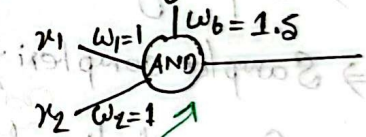


$\hookrightarrow [0, 1]$

iv) Gaussian A.f.



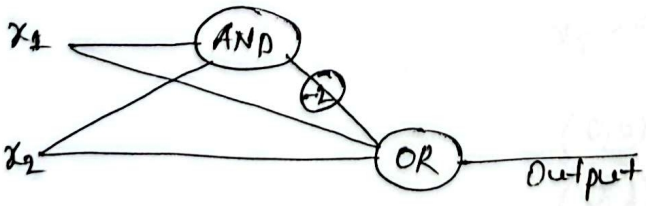
x_1	x_2	OR (bias (0.5))	AND (1.5)	NOR	NAND	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	0	1



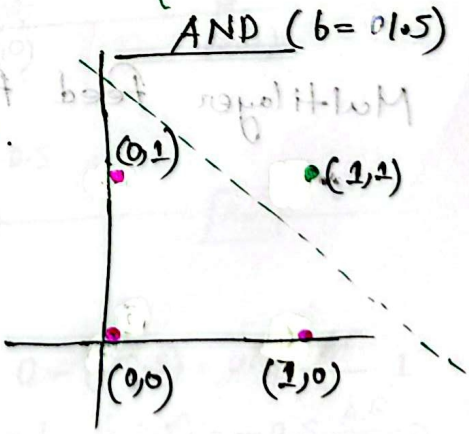
XOR, XNOR (single line can't separate data points)

★ What's the solution?

Sol1: Can be solved in higher dimension.



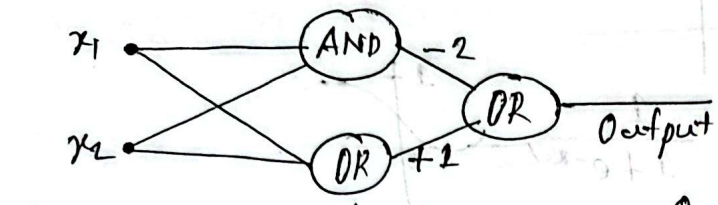
X-OR



(0,0) → 0 − 1.5 = −1.5 (0)
(1,0) → 1 − 1.5 = −0.5 (0)
(0,1) → 1 − 1.5 = −0.5 (0)
(1,1) → 2 − 1.5 = 0.5 (1)

Sol2:

$$XOR = (X_1 \text{ AND } X_2)' \text{ AND } (X_1 \text{ OR } X_2)$$



Input layer | Hidden layer | Output layer.

$$(2\times2)+2 + (2\times1)+1$$
$$= 6+3$$
$$= 9$$

Biological neuron to ANN

Learning issue:

The ability to learn.

Fundamental issue with learning:

⇒ Capacity

⇒ Sample complexity

⇒ Computational complexity.

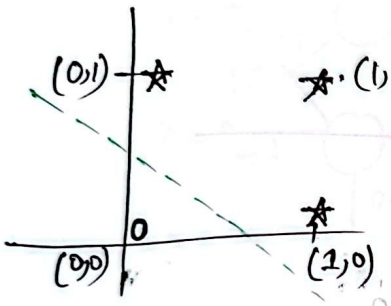
Multilayer feed forward Network:

X_1	x_2	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

1 → ★
0 → 0

1	0
0	1

OR



Activation function

Threshold function:

$\geq 0 \rightarrow 1$

$< 0 \rightarrow 0$

0	0
1	0
0	1
1	1

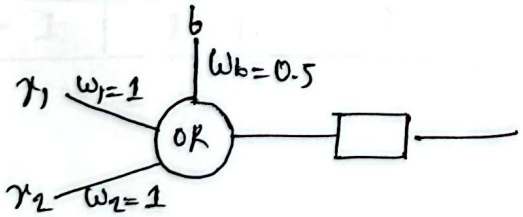
$b = 0.5$

$(0,0) \longrightarrow 0 - 0.5 = -0.5\ (0)$

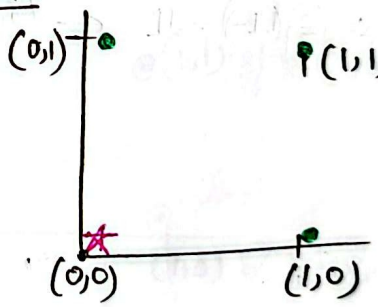
$(1,0) \longrightarrow 1 - 0.5 = +0.5\ (1)$

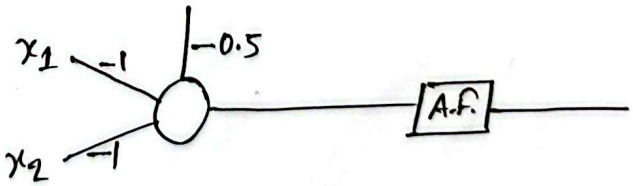
$(0,1) \longrightarrow 1 - 0.5 = +0.5\ (1)$

$(1,1) \longrightarrow 2 - 0.5 = +1.5\ (1)$



NOR





$(0,0) \longrightarrow 0 - (-0.5) = 0.5 \xrightarrow{A.f.} 1$

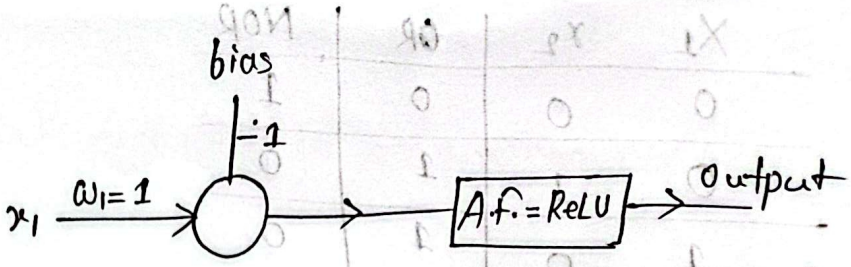
$(0,1) \longrightarrow -1 - (-0.5) = -0.5 \xrightarrow{A.f.} 0$

$(1,0) \longrightarrow -1 - (-0.5) = -0.5 \xrightarrow{A.f.} 0$

$(1,1) \longrightarrow -2 - (-0.5) = -1.5 \xrightarrow{A.f.} 0$

NOT gate:

X_1	Output
0	1
1	0



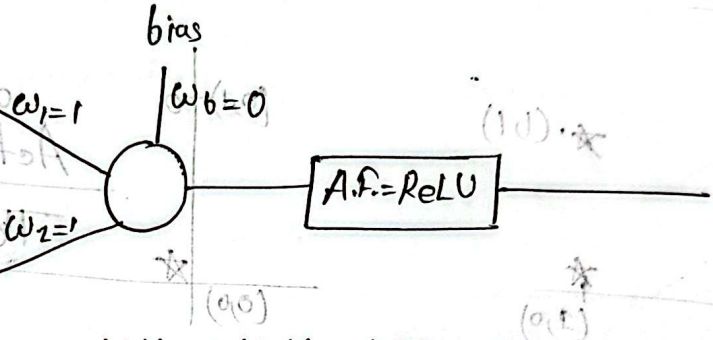
$0-(-1) = 1$

$1-(-1) = 0$

$ReLU(h) = \max(h, 0)$

OR:

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	1



$\omega_1 x_1 + \omega_2 x_2 + \omega_b = 0$

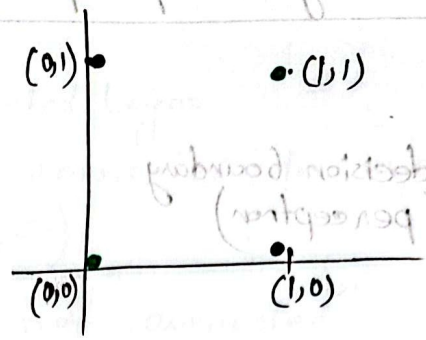
Case I: 0 $\quad \max(0,0) = 0$ OFF

Case II: 1 $\quad \max(0,1) = 1$ ON

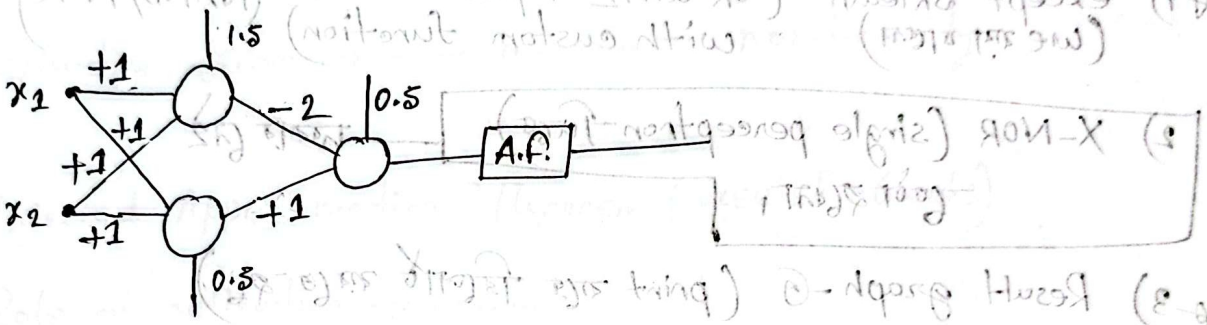
Case III: 1 $\quad \max(0,1) = 1$ ON

Case IV: 2 $\quad \max(0,2) = 2$ ON

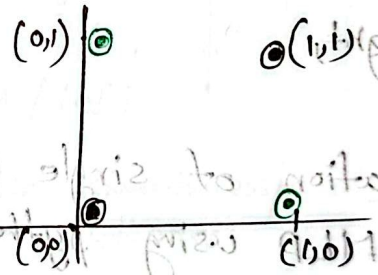
x_1	x_2	X-OR (output)
0	0	0
0	1	1
1	0	1
1	1	0



Observe:



x_1	x_2	X-NOR (output)
0	0	1
0	1	0
1	0	0
1	1	1



Medium — Implementing the perceptron.

- naukri.com
- code360
- (visualizing decision boundary perceptron)

- huggingface
- kaggle
- github.

Labwork:

*1) except sklearn (use করা যাবেনা) (OR-GATE implementation with custom function) (NAND/NOR)

2) X-NOR (single perceptron দিয়ে) — চেষ্টা হবেনা। — দরকার নেই

*3) Result graph-এ (print করে রিপোর্ট করতে হবে).

*4) X-NOR (multi-layer perceptron) (using sklearn)

bias, weight,

Implementation of single layer perceptron and MLP using python (OR)

Introduction
Function to implement
Result (graph)

MLP — Forward propagation.

⇒ Dense layer / fully-connected layer
(when neuron is connected to every other neuron in the next layer).

⇒ Sparse layer — when not connected

* Activation function — introduces non-linearity.

* Universal Approximation Theorem (GeeksforGeeks)

→ Role of activation function.

→ Limitation.

Video = Why neural network can learn any function?

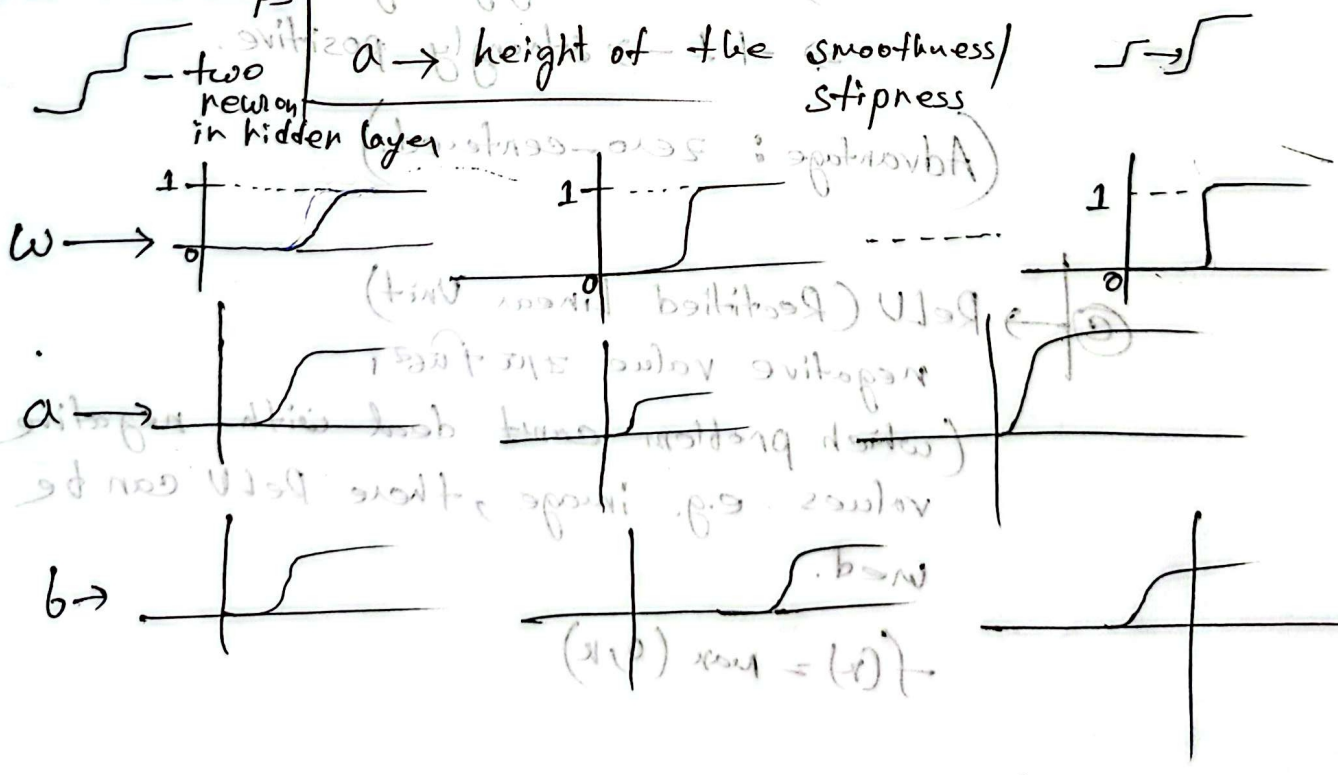
adding more neurons → can add more steps

- two neuron in hidden layer

w → smoth/stip

b → shift at right / left

a → height of the smoothness/stipness



Types of Activation function:

decides wheather a neuron will be activated or not

(i) → Binary activation function.

↳ Binary step function (Threshold function)

Limitation: can't be used in multi-class classification.

(ii) → Linear activation function

input = output
$f(x) = x$

} can't be used in every hidden layer, cause makes no change.

(iii) → Non-linear activation function

(a) → Sigmoid / Logistic activation function $f(x) = \frac{1}{1+e^{-x}}$

(returns value in probabilistic form) can be used in multiclass classification

(b) → tanh.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

generates value between −1 to +1

⇒ −1 → strongly negative

⇒ +1 → strongly positive.

(Advantage: zero-centered.)

(c) → ReLU (Rectified Linear Unit)

negative value রাখে না।

(which problem can't deal with negative values e.g. image, there ReLU can be used.

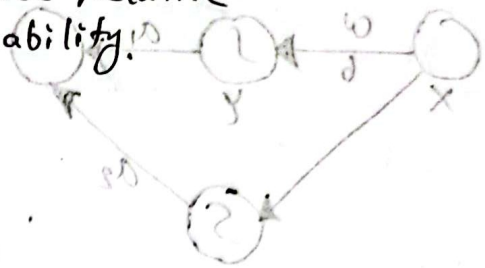
$$f(x) = \max(0, x)$$

Sigmoid ——— extension ——— Softmax function.

Sigmoid → softmax → so that it can provide relative probability.

(probability)

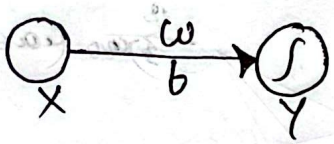
used in last layer



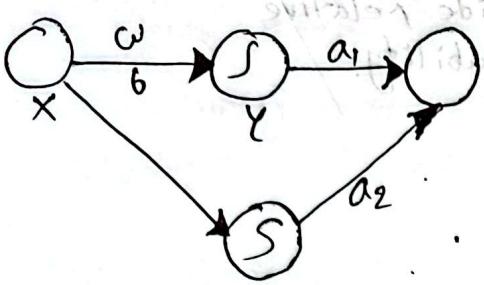
$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum \exp(z_i)}$$

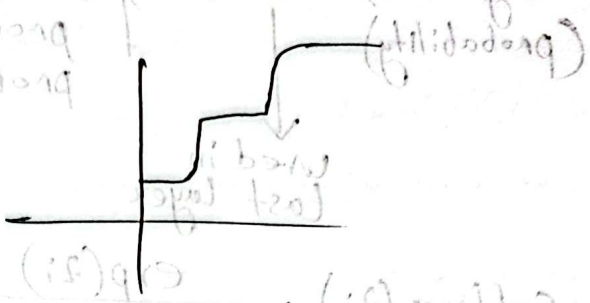
Output —— sigmoid / softmax

hidden layer —— ReLU / tanh.



$$S(x) = \frac{1}{1+e^{-x}}$$





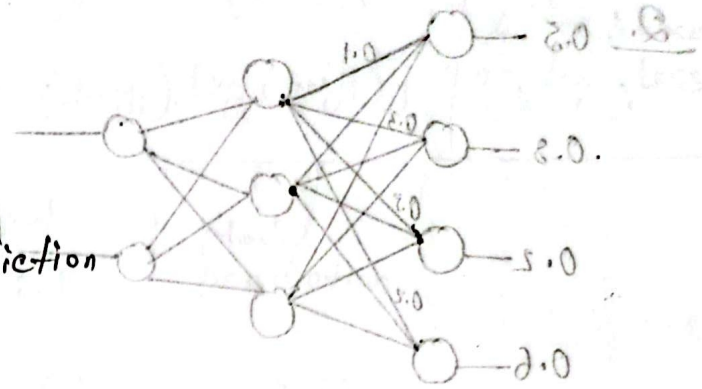
Universal Approximation Theorem:

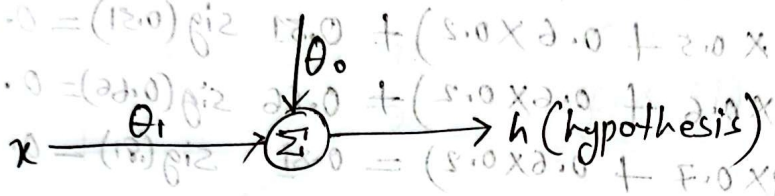
— foundational result in the theory of neural network.

Linear Regression
Logistic Regression

Cost Function

ML problems.
- Regression Problem / Prediction
- Classification Problem.





$h = \theta_0 + \theta_1 x$ $\quad \theta_0, \theta_1$ = parameter.

single example error — loss
total error — cost

Mean Absolute error — MAE
Mean Squared error — MSE

$$\frac{1}{m} \Sigma | h_i(x) - y |$$

$$\text{Cost function (MSE)} = \frac{1}{2m} \Sigma \left(h_\theta(x_i) - y_i \right)^2$$

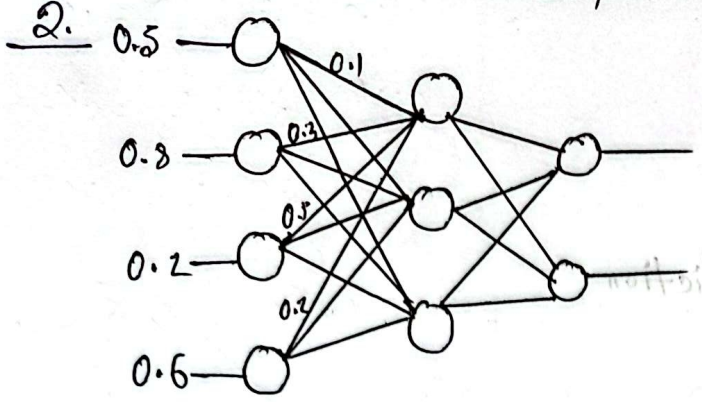
$$MAE = \frac{1}{m} \Sigma | h_\theta(x_i) - y_i |$$

Size	actual price	predicted price
1200	20,000	25
1500	15	10
1800	30	35

$$MSE = \frac{5^2 + 5^2 + 5^2}{2 \times 3}$$

$$= MAE = \frac{5+5+5}{3}$$

2.



Input to hidden:

w_1: [0.1, 0.3, 0.5, 0.2]

w_2: [0.2, 0.4, 0.6, 0.2]

w_3: [0.3, 0.5, 0.7, 0.2]

Hidden to output:

w_4: [0.4, 0.1, 0.3]

w_5: [0.5, 0.2, 0.4]

$(0.5\times0.1 + 0.8\times0.3 + 0.2\times0.5 + 0.6\times0.2) = 0.51$ $sig(0.51) = 0.624$

$(0.5\times0.2 + 0.8\times0.4 + 0.2\times0.6 + 0.6\times0.2) = 0.66$ $sig(0.66) = 0.659$

$(0.5\times0.3 + 0.8\times0.5 + 0.2\times0.7 + 0.6\times0.2) = 0.81$ $sig(.81) = 0.692$

$(0.4\times0.624) + (0.1\times0.659) + (0.3\times0.692) = 0.5231$ $sig(.5231) = 0.63$

$(0.5\times0.624) + (0.2\times0.659) + (0.4\times0.692) = 0.7206$ $sig(.7206) = 0.67$

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

Loss function: Binary Cross-Entropy loss: (For binary classification)

Also known as log-loss

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

Question:

Hours Studied	Hours Sleep	Previous score	Output Label	Model prediction
5	7	80	pass (1)	0.85
3	6	45	fail (0)	0.12
8	8	95	pass (1)	0.92

Cost value = ?

(natural log — ln)

For first student: →

$$-\left[1 \times \log(0.85) + (1-1) \times \log(0.15)\right] = 0.1625$$

For second student: →

$$-\left[0 \times \log(0.12) + (1-0) \times \log(0.88)\right] = 0.1278$$

For third student: →

$$-\left[1 \times \log(0.92) + (1-1) \times \log(0.08)\right] = 0.0834$$

$$\text{Average cost} = \frac{0.1625 + 0.1278 + 0.0834}{3}$$

$$= 0.1246$$

$Y_1 = 0$, $\hat{Y}_1 = 0.15$

$Y_2 = 1$, $\hat{Y}_2 = 0.90$

$Y_3 = 1$, $\hat{Y}_3 = 0.92$

Loss function used:
Binary cross entropy.

$$L_1 = -[0 \times \log(0.15) + (1-0)\log(1-0.15)] = 0.162$$

$$L_2 = -[1 \times \log(0.90) + (1-1)\log(1-0.90)] = 0.916$$

$$L_3 = -[1 \times \log(0.92) + (1-1)\log(1-0.92)] = 0.089$$

$$L_{avg} = \frac{L_1 + L_2 + L_3}{3} = \frac{1.162}{3} = 0.387$$

Multiclass classification:

→ Loss function ⟶ Categorical Cross Entropy. (standard choice)

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\,\log(\hat{y}_{ij})$$

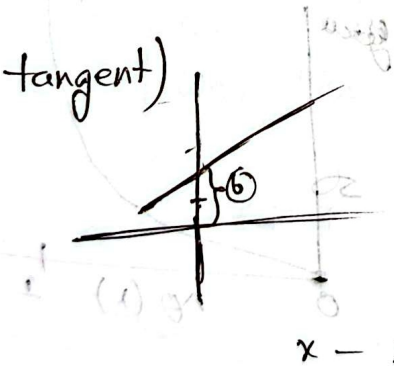
y_{ij} = true label
$\hat{y}_{ij}$ = predicted label
} for class j

C = number of classes.
N = number of samples.

---

i) Sigmoid
ii) ReLU
iii) Threshold
iv) softmax
v) tanh (hyperbolic tangent)

★ Softmax ⟶ combination of multiple sigmoids,
→ used in last layer,
→ returns the probability of each class.



Threshold

≥ 0, —(1)
< 0 —(-1)

Σwixi + b

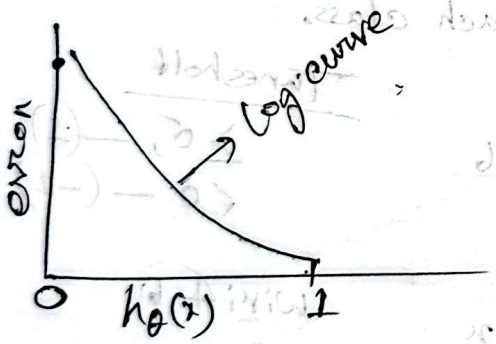
f(x) = x

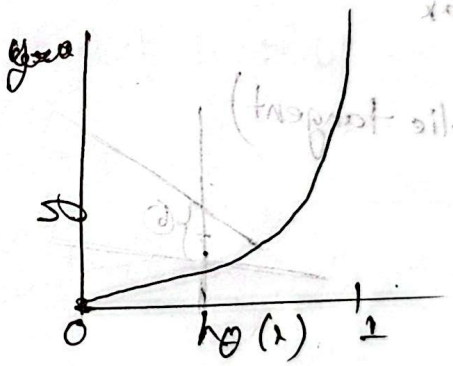
$h_\theta(x) = \theta_0 + \theta_1 x$ $\quad h \rightarrow$ hypothesis.

Logiste Regression — Classidication.

Cost function (Regression)	Cost function (Classification)
→ MSE → MAE	

$$\cos(h_\theta(x)) = \begin{cases} -\log(h_\theta(x)), & \text{if } y=1 \\ -\log(1-h_\theta(x)), & \text{if } y=0 \end{cases}$$





$(1-h_\theta(x))$ বাম হলে

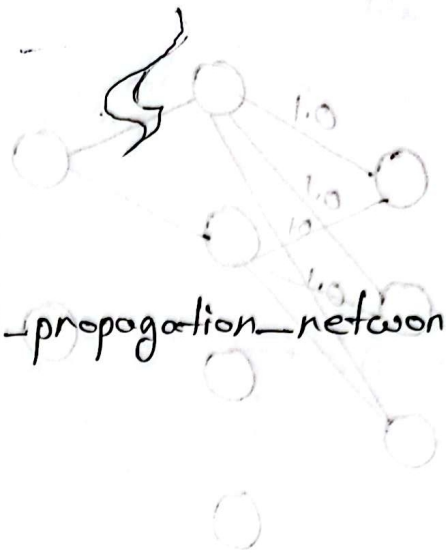
★ Binary-cross entropy: (যখন output column 0/1 হয়)
Log-loss.

$$-y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

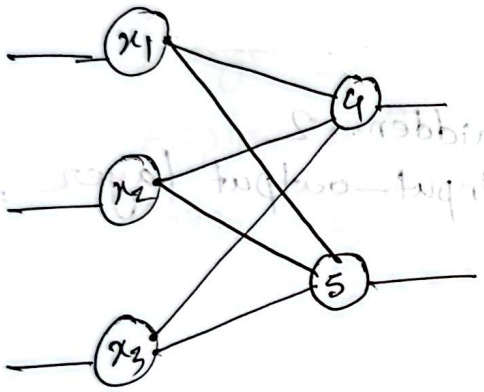
$$= -\frac{1}{N}\left[\sum_{i=1}^{N} y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right]$$

→ No of training example.

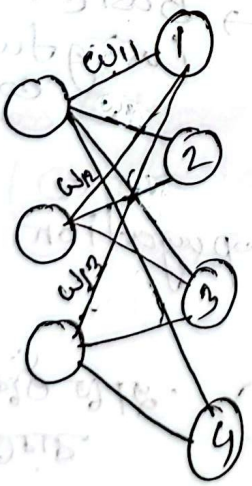
Heat	Weight	OR	AND	XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

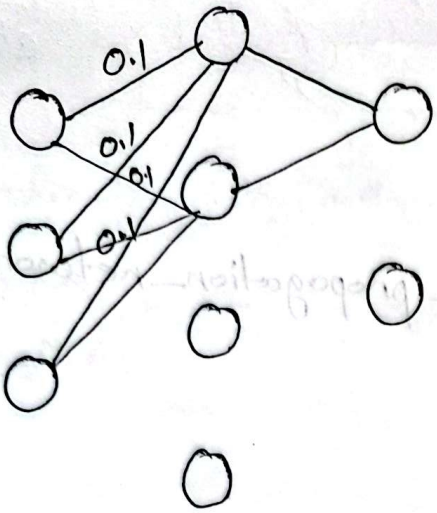


Fonaard propagation: (datacamp.com) forward_propagation_networks.



$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} w_{14} & w_{15} \\ w_{24} & w_{25} \\ w_{34} & w_{35} \end{bmatrix}$$





{hidden — 2
input — output layer,

Lab-task (30.

* 4-5 layer (feed-forward)
Big size neural network.

image.net
inception.net.

PyTorch — ~~Google~~ Facebook
Tensorflow - Google

(1) → PyTorch
→ Tensorflow
→ Basic.
(using function and construction)

Introduction to forward propagation in neural network.

(2) - হাতে কষে calculation বানাতে হবে।

Multiclass classification:

categorical → encoding → numerical value

encoding → One hot encoding
encoding → Label encoding.

One hot encoding:

Input	O	M	A
Orange	[1	0	0]
Mango	[0	1	0]
Apple	[0	0	1]
Mango	[0	1	0]

Output → Discretization

[.2 0.6 0.6] → [0 1 0] (Orange, Mango, Apple)

it's mango.

O → .2
M → .6
A → .6

$$Loss = -\sum_{j=1}^{K} y_j \log(1-\hat{y}_j)$$

$$Cost = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{K} y_{ij}(1-\hat{y}_{ij})$$

m = no. of training example

Sparse - ~~Cross~~ Categorical cross entropy → (ওয়ান হট label encoding এর মতো)

Scenario:
1000 class
(the 1 — 1
999 — 0)
Huge no. of zeros
(sparse values)

Optimization algorithm:

Gradient descent never guarantee to reach global minima.

It may / may not.

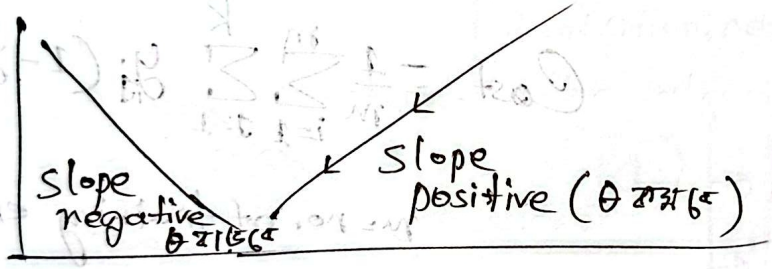
$\frac{dJ}{d\theta}$ — J: cost function, θ: parameter.

η (learning rate) decides step size.

$\left(\theta_{new} \; \theta_{old} - \eta \frac{dJ}{d\theta_{old}}\right)$ — equation of gradient descent.

→ Parameters should be updated simultaneously.

→



Learning rate:

too small — needs much time in training.

too large — zig-zag (over-shoot)

* Gradient descent assures to make reach to the local minima.

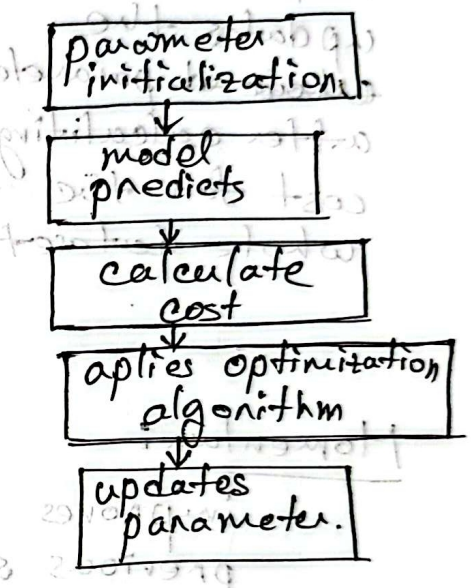
* Solution → update big
→ update small. } for learning rate and slope.

Variants of Gradient descent:

i) Mini-batch gradient deseent:

ii) Stochastic gradient descent:

iii) Momentum.



Stochastic Gradient Descent:

When dataset is large, then computationally expensive.

Gradient Descent

Too many loops required updates the ~~whole~~ ~~of~~ parameter after calculating cost for the whole dataset.

Mini-batch Gradient

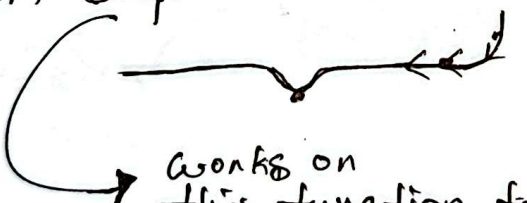
→ Batch-size is decided on trial & error process.

Stochastic Gradient Descent

→ Updation happens per one training example (also computationally expensive)

Momentum:

improves SGD by adding info. of the previous steps of the algorithm to the next step.



RMSProp:

Used when dataset is sparse. (Data ~~of~~ ~~of~~)

Adam:

mostly ~~used~~ used (adaptive learning rate)

Adaptive Momentum Estimation

→ learning rate is changed based on parameter value (historical gradient)

- Complex optimization algorithm
- Genetic optimization algorithm.
- ~~ant~~ colony optimization algorithm

} inspired by nature.
(Genetic optimization algorithm)

Swarm intelligence

Till row + Back-propation → Mid

→ one loop

Performance matrix:

Performance metrices:

Regression:

- MSPE
- MSAE
- R square
- Adjusted R square

Unsupervised models.

Classification problem:

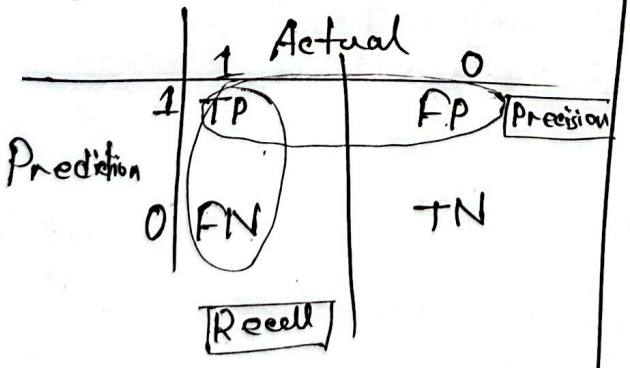
$$\Rightarrow \text{Accuracy} = \frac{\text{no. of correct prediction}}{\text{Total no. of prediction (Test set)}}$$

When to use accuracy?

target variable classes in data is approximately balanced.

Confusion matrix (classification task)

4 parts:



Actual → ↑ → prediction

TP = True positive

FP = False positive.

FN = False negative

TN = True negative

Multiclass classifier:

~~Accuracy~~ Confusion.

Precision:

Positive portion of actually right correct

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

FP ↑ Precision ↓

Recall / Sensitivity:

$$\text{Recall} = \frac{TP}{TP + FN}$$

False negative of cost is more compared to False positive

FN (cost) > FP (cost)

It changes based on scenario.

When to we Recall and Precision?

F1-score

combination of Recall and Precision.

F_1

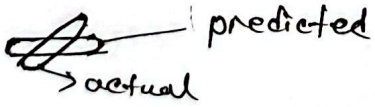
(when classes are equally important)

$R^2 = 1 - \frac{MSE(Model)}{MSE(Baseline)}$

→ tolerable error.

→ known as coefficient of determination.

IOU — Intersection over union



SOTA
HOTA
} → For tracking