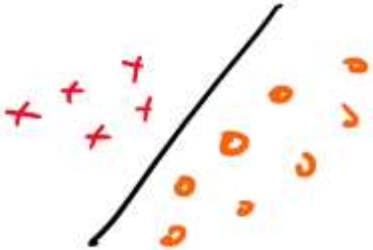
B_Innovative Part-2

Agenda

1) Understanding Similarities b/w Logistic regression and perceptron.
2) what is Multi-layered perceptron.
3) Understanding mathematical Notations.
4) Training a Single Neuron Model.
5) Training M.L.P.
6) Memoization

Logistic Regression and perceptron.



A Logistic regression w,

Given any point $x_i \longrightarrow \hat{y}_i$

$\hat{y}_i$ is predicted value of y_i.

So, In General $\hat{y}_i = \text{Sigmoid}(w^T x_i + b)$

A dataset $D = \{x_i, y_i\}$ where $x_i \in R^d$, $w \in R^d$, $b \in R$:)

we train L.R $\longrightarrow$ to find w and b.

$\therefore \quad \hat{y}_i = \text{Sigmoid}(w^T x_i + b).$ } Logistic Regression.

$$\Rightarrow \hat{y}_i = \text{Sigmoid}\left(\sum_{i=1}^{d} w_i x_{ij} + b\right)$$

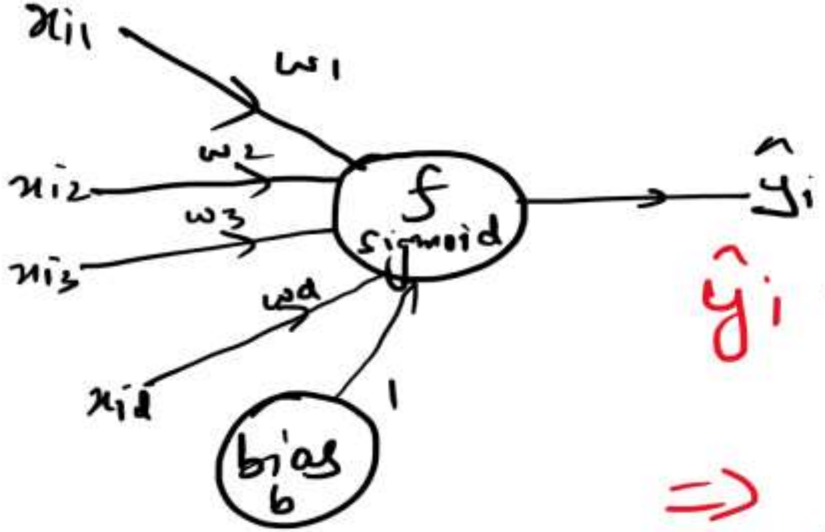
Let assume a point x_i , $x_j \in R^d$

So we can represent x_i as a vector.

$x_i = [x_{i1}, x_{i2}, x_{i3} \text{ - - - } x_{id}]$

weights $w = [w_1, w_2, w_3 \text{ - - - - } w_d]$

Output of Neuron $= f\left(\sum_{i=1}^{d} w_j x_{ij}\right)$



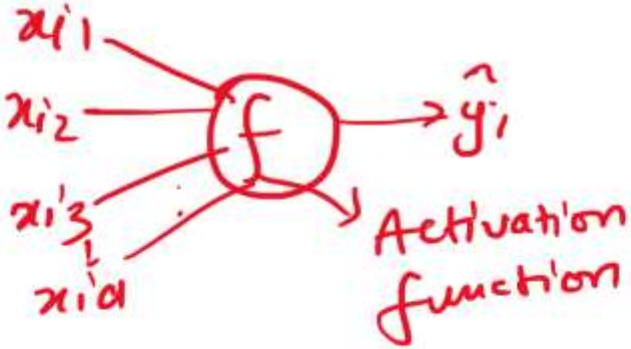
$$\hat{y}_i = f(w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} - - - w_d x_{id})$$

$$\Rightarrow \hat{y}_i = f(w^T x_i + b)$$

Training a Neural Network.

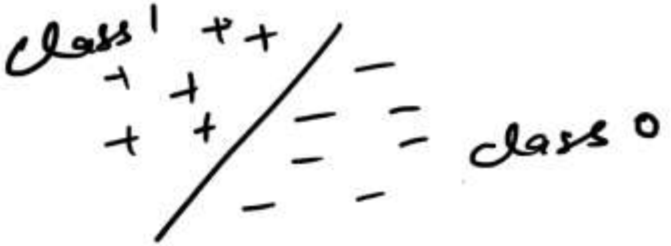
→ It states when we have to find weights on edges | vertices.

Let's talk about perceptron.



$$f(x) = \begin{cases} 1 & \text{if } w^T x_i + b > 0 \\ 0 & \text{otherwise.} \end{cases}$$

A perceptron is also called a Linear classifier.



A perceptron will always tries to find out the best line which separates class 1 from class 0.

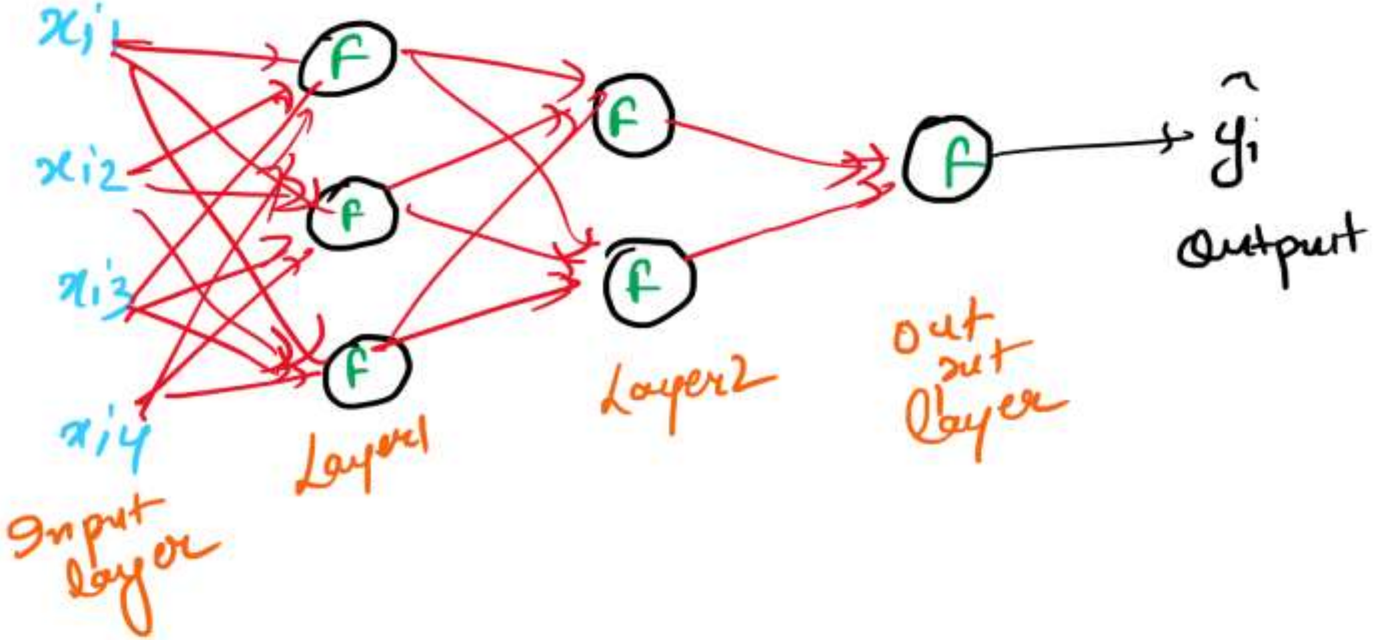
In Logistic Regression → Squashing (sigmoid) function

In perceptron → No Squashing function

Simple Neuron model, only difference is activation function.

Multi layered perceptron.

↳ Bunch of Connected Neurons.



Multi layered perceptron or Neural Network.

Why should we care about Multi layered perceptron?

→ Since perceptron has been feature of Biological Neural Network, In Neuroscience → Humans, rats, monkeys, ants, found collection of Neurons interconnected in smartest way.

How do you Connect it and Make it a Network?

→ Lot of mathematical arguments happened.

Let's take Regression problem, $D = \{x_i, y_i\}$

Determine $y_i = f(x_i)$, $y_i \in \mathbb{R}$.

Let's assume, 

$\mathcal{D} = \{x_i, y_i\}$

$x_i \in \mathbb{R}^1$ } (one
$y_i \in \mathbb{R}$ } dimensional)

Objective → To find f in , $y_i = f(x_i)$

Case-1 Let say $f(x_i) = y_i$

$\therefore x_i = y_i$

$f(x) = 2\sin(x^2) + \text{sqrt}(5x)$.

↳ Composition of function.

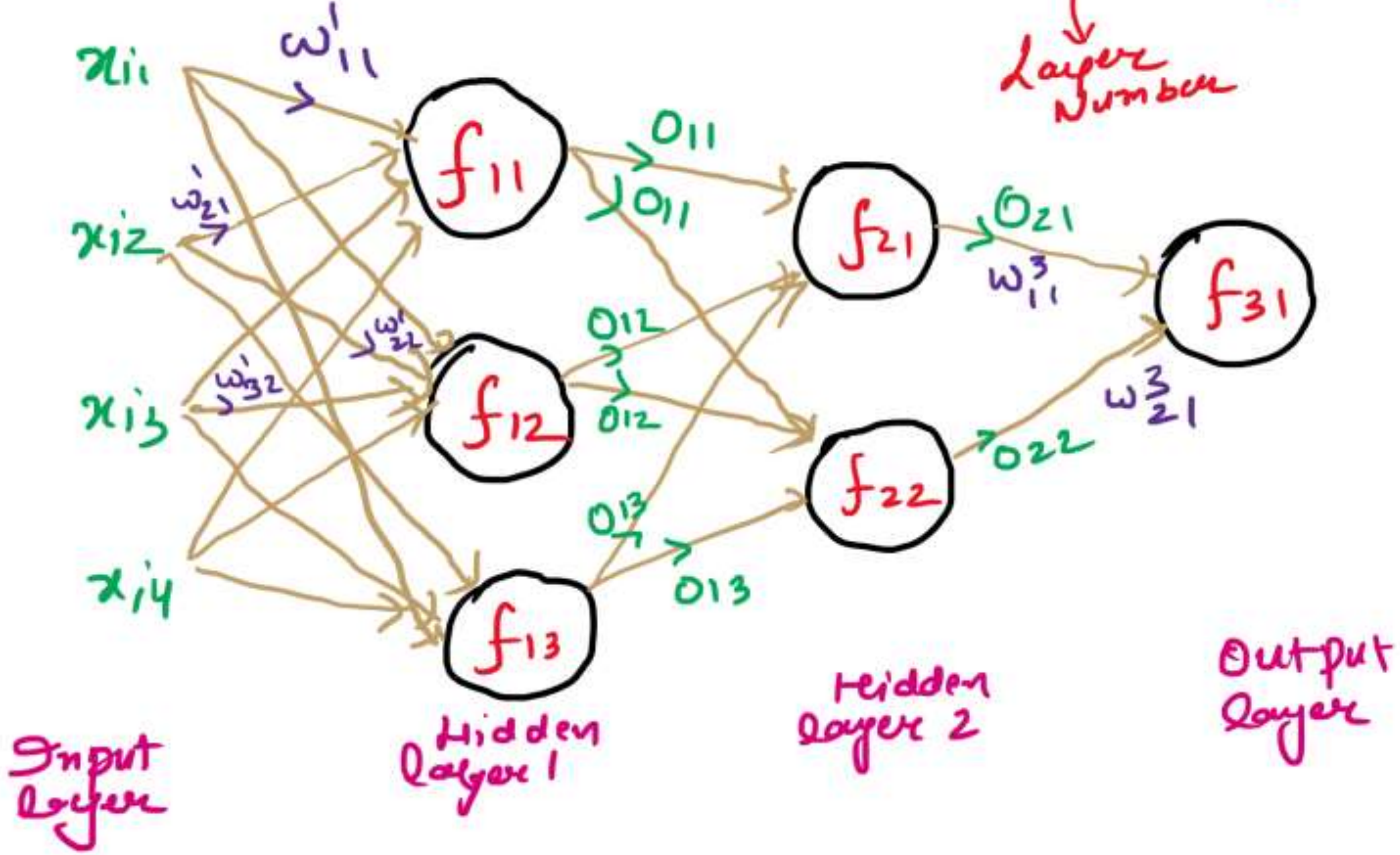
} Basically Complex functions gives enormous power to regress the problem.

Composition of function.

- → $fog(x)$ or $gof(x)$
- → MLP is a Graphical way of representing simple function composition.
- → It results in powerful model.
- → we can also overfit easily.

Notations:

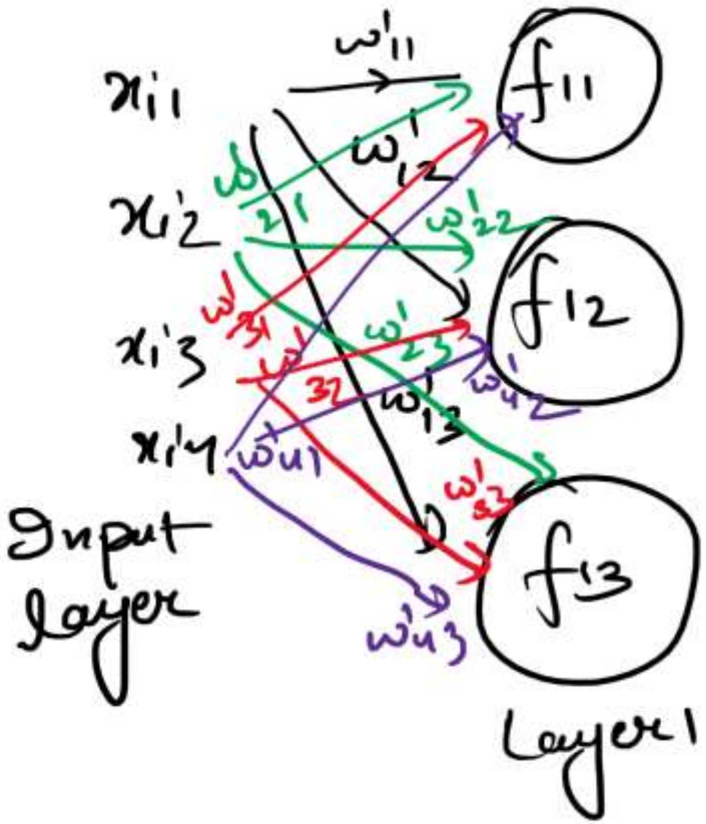
f_{ij} → function Number (j)
→ Layer Number (i)



f_{11} → Layer 1, function 1
f_{22} → Layer 2, function 2

O_{11} → output of layer 1 from function 1
O_{12} → output of layer 1 from function 2
O_{13} → output of layer 1 from function 3.

w^k_{ij} k → what is next layer number.
i → from which neuron
j → to which Neuron.



How many weights.

4 inputs and 3 units (Neurons)

$4 \times 3 \Rightarrow 12$ weights.

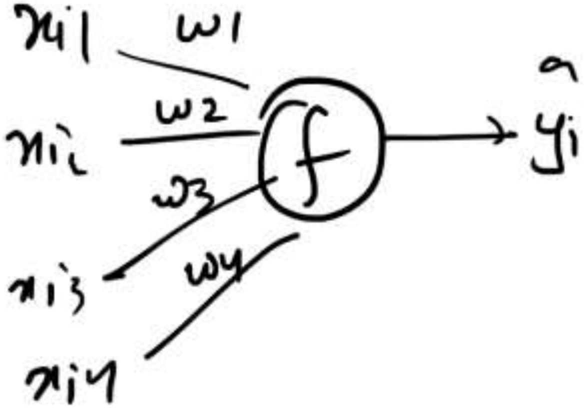
$$
\begin{array}{c}
x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4}
\end{array}
\begin{bmatrix}
w'_{11} & w'_{12} & w'_{13} \\
w'_{21} & w'_{22} & w'_{23} \\
w'_{31} & w'_{32} & w'_{33} \\
w'_{41} & w'_{42} & w'_{43}
\end{bmatrix}_{\underline{4 \times 3}}
$$

Step by Step process of Training a Single Neuron Model.

→ Training refers to find the best edge weights in a network model using training data.

→ perceptron and Logistic regression → Single Neuron model for classification.

→ for Linear Regression → Single Neuron model for regression.



Linear regression

$$\hat{y}_i = \sum_{j=1}^{d} w_j x_{ij} \qquad x_i \in R^d$$

$$\hat{y}_i = w^T x_i \qquad y_i \in R$$

Activation function in Linear regression is

Linear function $f(z) = z$ (called Identity function)

In Linear regression

$$\min_{w_i} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + reg.$$

where $\hat{y}_i = w^T x_i$

$n \rightarrow$ No. of points in training data.

$$\Rightarrow \min_{w_i} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \|w\|_2^2$$

becomes a optimization problem

Defining a loss function.

$$\sum_{i=1}^{n} \mathcal{L}_i \Rightarrow \mathcal{L} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \text{reg.}$$

Let's take on one training point.

$$\mathcal{L}_i = (y_i - \hat{y}_i)^2$$

$$\therefore \hat{y}_i = w^T x_i$$

Computing loss function on top of y_i

$$\mathcal{L}(y_i, \hat{y}_i)$$

(2) write the optimization problem.

$$\min_{w_i} \sum_{i=1}^{n} \left(y_i - \underbrace{w^T x_i}_{\hat{y}_i}\right)^2$$

$$\hat{y}_i = f(w^T x_i)$$

↳ linear function.

$$\therefore \quad \min_{w_i} \sum_{i=1}^{n} \left(y_i - f(w^T x_i)\right)^2 + \text{reg.}$$

for perceptron, Activation function
$\downarrow$
Threshold function.

$\therefore$ updated weights.

$$w^* = \arg\min_{w} \sum_{i=1}^{n} (y_i - f(w^T x_i))^2 + \text{reg.}$$

(3) Solve the optimization problem.

a) Initialization of weights

↳ random initialization. w_i's

b) Computing the derivative of $\mathcal{L}$ w.r.t w.

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_4} \end{bmatrix} \quad \because x_i \in R^4.$$

c) $w_{new} = w_{old} - \eta \left(\nabla_w L \right)_{w_{old}}$

updated weight · old weight · learning rate.

$$(w_i)_{new} = (w_i)_{old} - \eta \left(\frac{\partial L}{\partial w_i} \right)_{(w_i)_{old}}.$$

For iteration 1 to k

Computing Gradient Descent

$\nabla_w L \rightarrow x_i$'s and y_i's

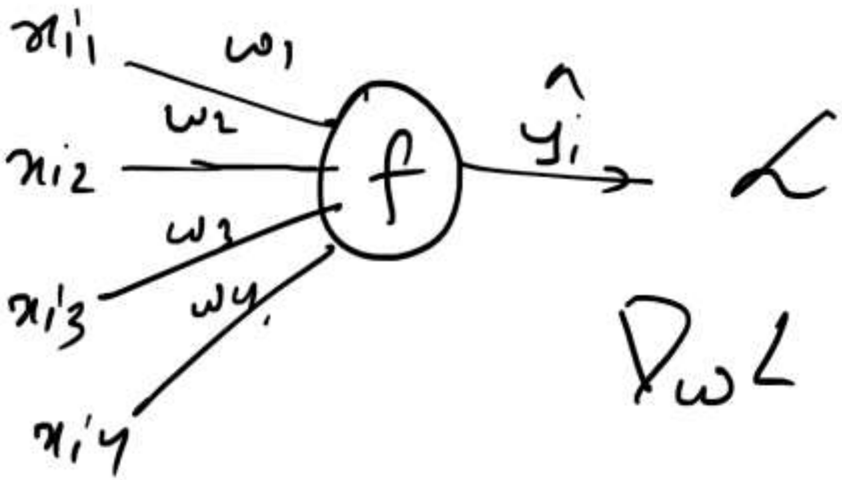
Computing Stochastic Gradient Descent

$\nabla_w L \sim$ one point

Computing mini batch Stochastic Gradient descent

$\{x_i, y_i\}$

How to Compute these derivatives



Squared loss in regression problem.

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_4} \end{bmatrix}^T$$

Applying Chain Rule.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial w_1}$$

Always Check Loss function and the weight (say w_1), what comes in between and start differentiating from.

$$w_1 \xrightarrow[\text{data}]{f} L$$

$$\mathcal{L} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \text{regularizer.}$$

$$\Rightarrow \sum_{i=1}^{n} (y_i - f(\omega^T x_i))^2$$

$$\Rightarrow \sum_{i=1}^{n} (y_i - f)^2$$

$$\frac{\partial L}{\partial f} = (-) \sum_{i=1}^{n} \cdot 2 \left(y_i - f(\omega^T x_i)\right)$$

$$\frac{\partial f}{\partial \omega_1} = x_i$$

$$\Rightarrow \frac{\partial L}{\partial w_1} = -2 \cdot x_i (y_i - \hat{y}_i)$$

$$\Rightarrow \frac{\partial L}{\partial w_1} = \sum_{i=1}^{n} (-2) x_i (y_i - \hat{y}_i)$$