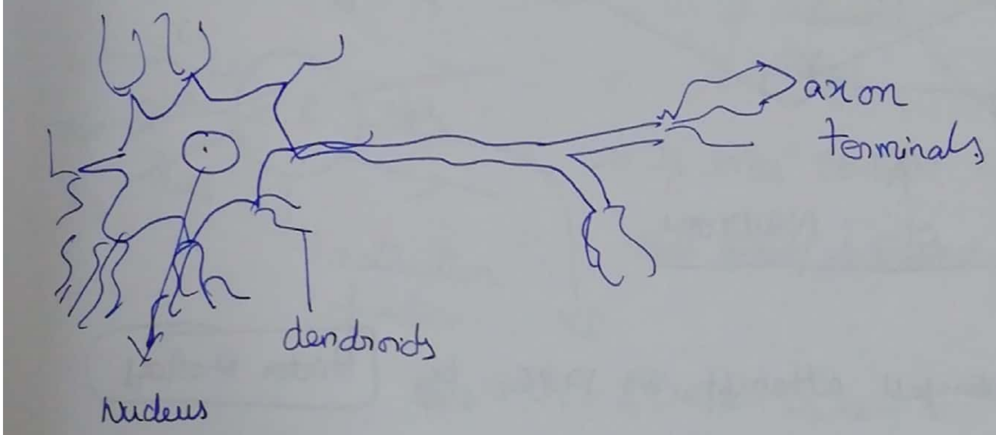
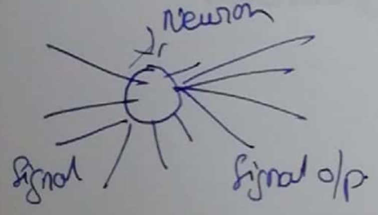
History of Neural Networks

47.1 History of Neural network & Deep learning

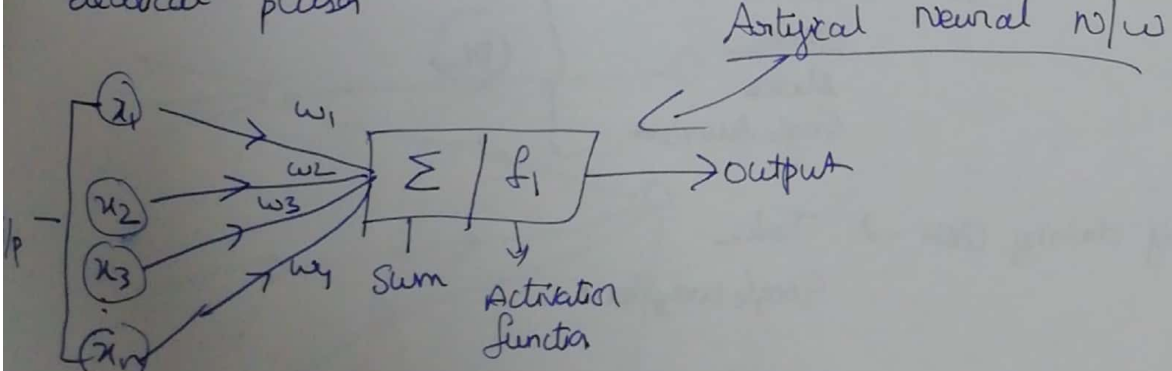
→ Perceptron → 1957 [Rosenblatt]
↳ A simple model
↳ Similar to Logistic Regression





→ Most simple.

Electrical pulses comes in and some process done and sends out the electrical pulses

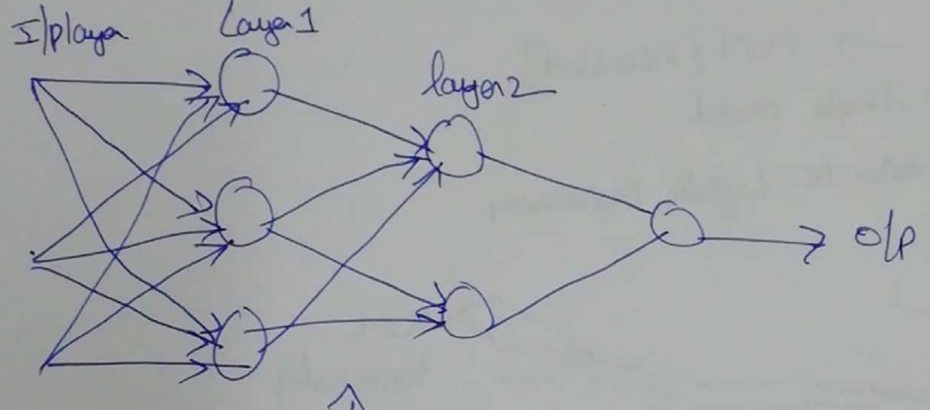


$$f(w_1x_1 + w_2x_2 + \cdots w_nx_n)$$

→ loosely Inspired from biology

In biology → Brain

→ There will be multiple neurons not only one



network of neurons

→ There is successfull attempt in 1986 by Hinton & others

→ Back propagation Algo → Chain rule of differentiation

→ ~~As depth~~

In 2012

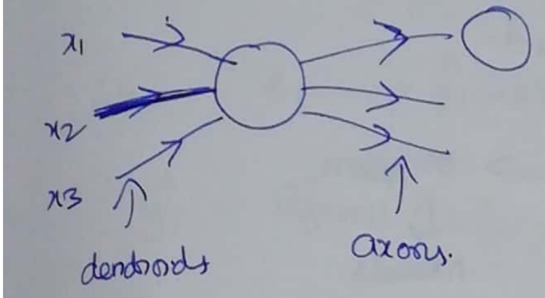
↳ image net

→ Voice assistant → Siri
Cortana
Alexa
Google Assistant } DL

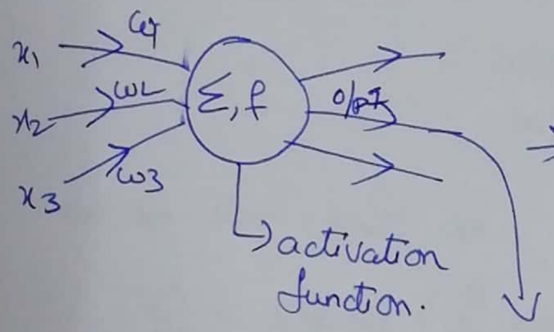
Self driving cars → Tesla
Google waymo

Skin cancer → AI

Simplified View of Neuron



→ In dendroid is thicker, ~~Re~~ more weight is associated with it.



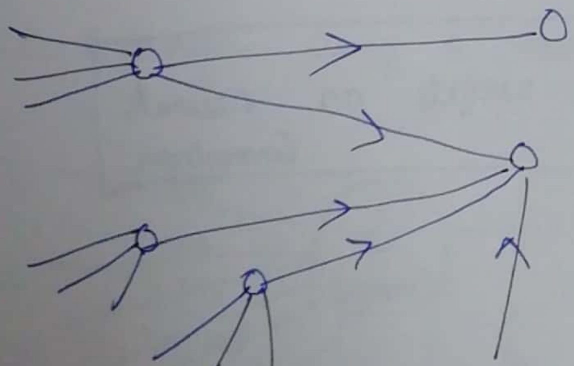
{ activated
fired.

→ If more Electrical signal in the i/p. the cell fired / Activated.

$$o/p_1 \rightarrow O_1 = f(w_1x_1 + w_2x_2 + --- + w_nx_n)$$

$$O = \underbrace{f}_{\text{activation function}}\Big(\sum_{i=1}^{n} \underbrace{w_i}_{\text{weights}} \underbrace{x_i}_{\text{i/p's}}\Big)$$

47.3 Growth of biological Neural network



→ Front part of brain Cerebral Content
↓
front part of brain.

learning → Connecting neurons with Edges

more - Connection → more Calories required.

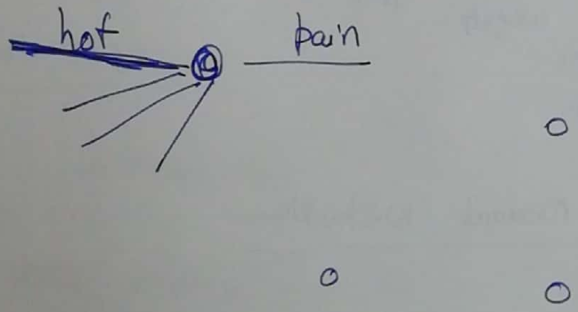
Birth → 6 year → 14 years → 80 years

6 year ↓ Consumes a lot of data ↓ Visual, audio, family, sensory

80 years ↓ neurals degenerate

→ Japanes kid VS ~~Id~~ Indian kid

→ Connectia are formed based on data.



$$\boxed{\text{all biological learning} \xrightarrow{\circledast} \text{weights on neural Connections}}$$

ANN

47.4 ÷ Diagramatic Reporesentation : Logistic Regression & Perceptron.

LR: $x_i \longrightarrow \hat{y}_i \rightarrow$ Predicted Value of (y_i)

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

$D = \{x_i, y_i\}$ Train LR → we will find ~~the~~ vector(w) & B

$w \in \mathbb{R}^d$

$b \in \mathbb{R}$ (Real number)

we can also write as

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

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

$$w = [w_1, w_2, w_3 \cdots w_d]$$

Part $\sum_{j=1}^{d} (w_j x_{ij})$

O/p of neuron (O) is $f\left(\sum_{j=1}^{d} w_j x_{ij}\right)$

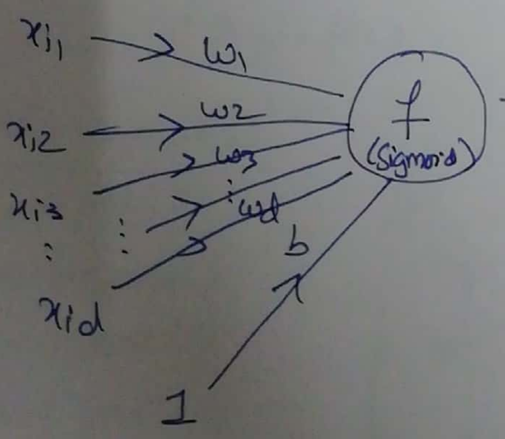
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$
2×2 2×1

$$\begin{bmatrix} 1\times2 + 2\times3 \\ 3\times2 + 4\times3 \end{bmatrix} = \begin{bmatrix} 8 \\ 18 \end{bmatrix}$$
2×1

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * [2, 3]$$
2×1 1×2

$$\begin{bmatrix} 1\times2 & 2\times3 \\ 3\times2 & 4\times3 \end{bmatrix} = \begin{bmatrix} 2 & 6 \\ 6 & 12 \end{bmatrix}$$
2×2

Logistic Regression using a neuron.



$$\hat{y}_i = f(w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} + b)$$

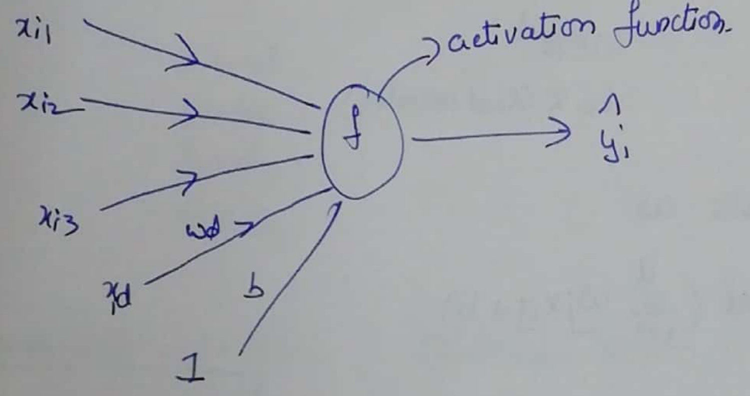
$$\hat{y}_i = f(w^T x_i + b)$$

If f is sigmoid This is how we will represent LR in the form of neuron.

→ Training a neural network (NN) ⇒ Computing the weights on Edges/Vertices

→ Represented LR in the form of neuron using a activation function called Sigmoid

Perceptron (1957)

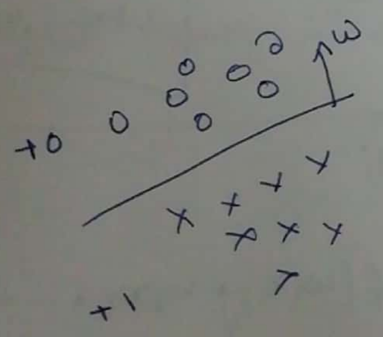


$$f(x) = \begin{cases} 1 & \text{if } w^T x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

If the activation function fires then $f(x) = 1$

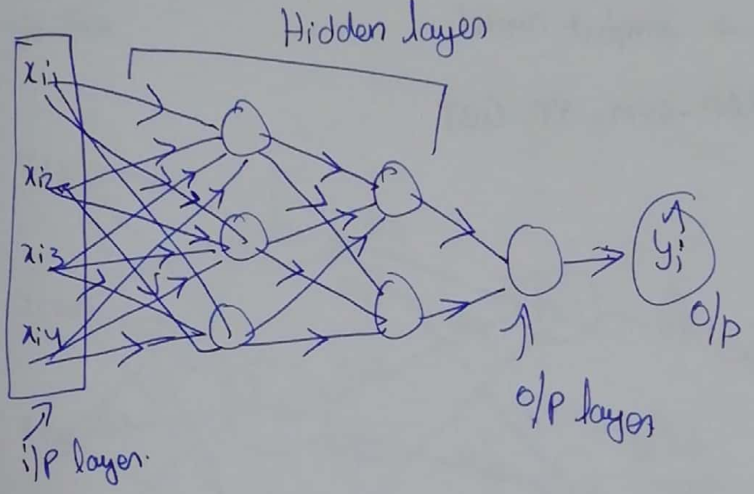
If the activation function doesn't fire $f(x) = 0$

Perceptron is also a lr. classifier.



47.5 → multi-layer Perceptron

Perceptron → Siggle neuron
↳ Logestic Regression



⟵ neural network ⟶

⟵ multi layer perceptron ⟶

(Q) Why should we care about MLP?

(a) biological Inspiration

(b) Mathematical

regression $\{x_i, y_i\} = D$
↳ $y_i = f(x)$ $x_i \in \mathbb{R}^1$
$y_i \in \mathbb{R}$

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

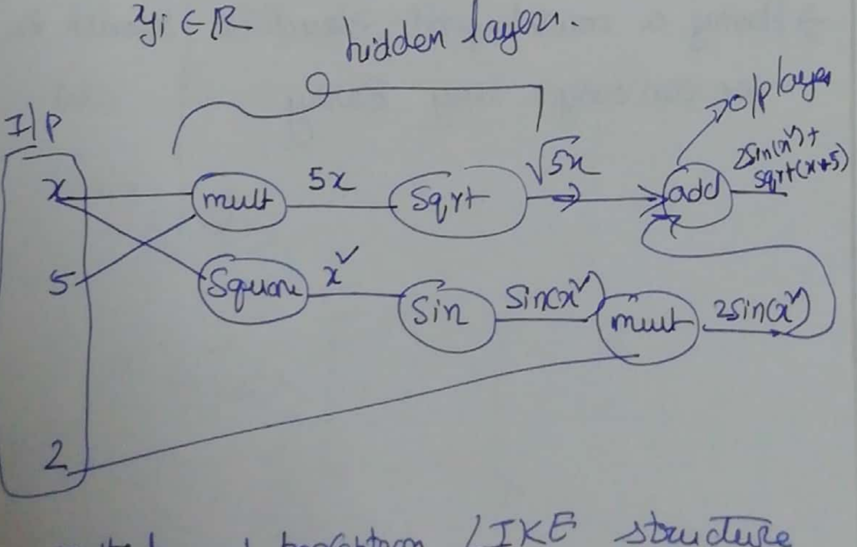
f_1 → add ( )

f_2 → square ( )

f_3 → sqrt ( )

f_4 → sin ( )

f_5 → mul ( )



multi layered perceptron LIKE structure

→By using multi layered structure we can come up with Complex multilayered functions

→ multilayered structure → Enormous power to the model.

- linear model → Simplest model.
- non_linear → RBF-SVM, RF, GBT

Function Composition

$f \cdot g(x) = g \cdot f(x)$

$F(x) = 2\sin x^2 + \sqrt{5x}$

Let's pick $2 * \sin x^2$, $\sqrt{5x}$

$f_5(2, f_4(f_2(x)))$ $f_3(f_5(x,5))$

$F(x) = f_1(f_5(2, f_4(f_2(x))), f_3(f_5(x,5))) \to (MLP)$

$f(g(x)) = f \cdot g(x)$

$g(f(x)) = g \cdot f(x)$

In MLP : graphical way of representing $f \circ g$, $g \circ f$

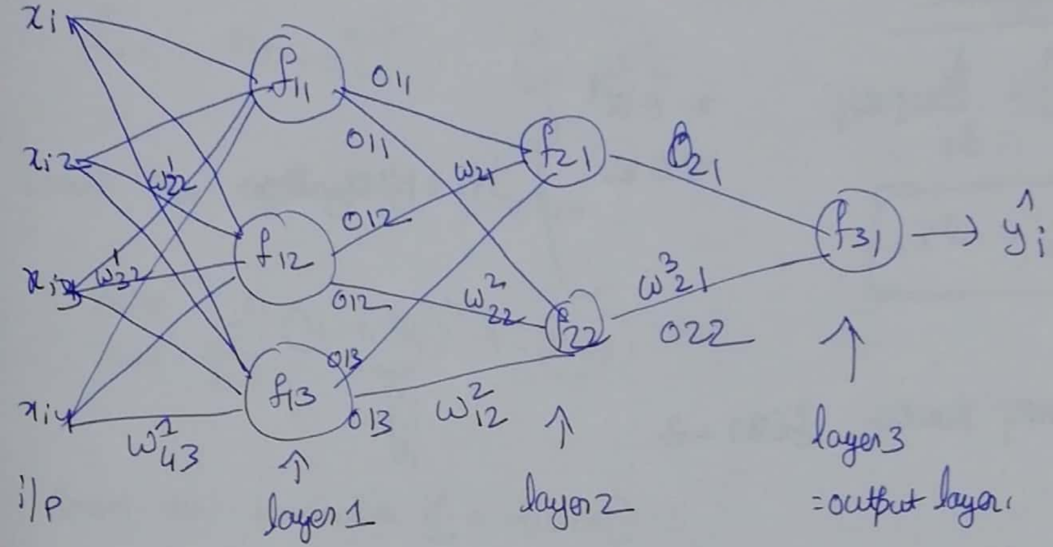
→ Having a multilayered structure result in powerfull models we can overfit very Easily

47.6 Notation

$D = \{x_i, y_i\}$; $x_i \in \mathbb{R}^4$; $y \in R$ (Regression)

x_{ij}

point feature



f_{ij} — layer, index.

O_{ij} — i^{th} layer, neuron

w^k_{ij} — k - next layer. i: from, j: To

$$W^1 = \begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \\ w^1_{41} & w^1_{42} & w^1_{43} \end{bmatrix}_{4\times 3}$$

$$W^2_{3\times 2} = \begin{bmatrix} & \\ & \end{bmatrix}_{3\times 2} \qquad w^3_{2\times 1} = \begin{bmatrix} \\ \\ \end{bmatrix}_{2\times 1}$$

47.7 Training a Single-neuron model

$\downarrow$ find the best Edge-weights using D_{Tr}

Perceptron & LR → Single neuron models for classification

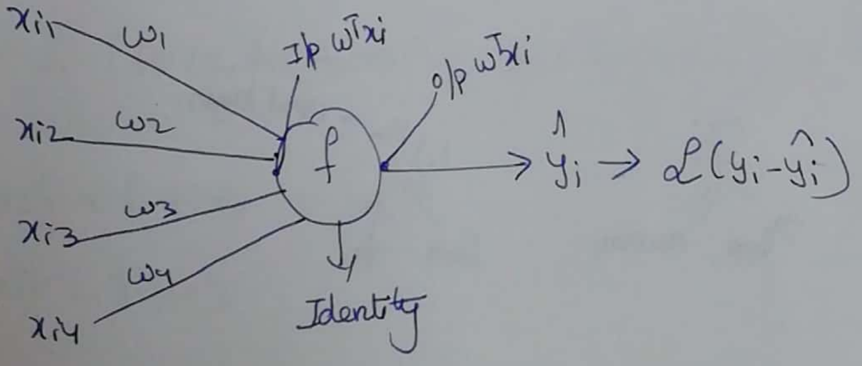
Easy → Lr. Reg → " " for regression

Lr. Regressia

$$\hat{y}_i = \sum_{j=1}^{d} w_j x_{ij} \qquad \left.\begin{array}{l} x_i \in \mathbb{R}^d \\ y_i \in \mathbb{R} \end{array}\right\} \text{lr. optimization.}$$

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

Identity function $f(z) = z$



Lr-reg

$$\hookrightarrow \left\{ \min_{w_i} \sum_{i=1}^{\overset{\#\text{pts in tr data}}{n}} (y_i - \hat{y}_i)^2 + reg \right.$$

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

$D = \{x_i, y_i\}_{i=1}^{n}$

$x_i \in \mathbb{R}^4$, $y_i \in \mathbb{R}$

① Define loss-function

$$\mathcal{L} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \boxed{reg} \rightarrow \text{keep away for now}$$

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

② write the optimation problem.

$$\min_{w_i} \sum_{i=1}^{n} (y - \underbrace{w^T x_i}_{\hat{y}_i})^2 + \boxed{reg} \Rightarrow \times$$

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

↳ Lr.reg f is Identity (I)

↳ Log.reg f is sigmoid & logistic.

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

↳ I : lr. reg

↳ Sigmoi : log reg

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(w^T x_i))^2 + reg$$

③ Solve the optimization problem

ⓐ Initialization of wi's → random

ⓑ $\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial w_3} \\ \frac{\partial L}{\partial w_4} \end{bmatrix}$ $\quad w \in \mathbb{R}^4$, $x_i \in \mathbb{R}^4$

→ vector represent

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

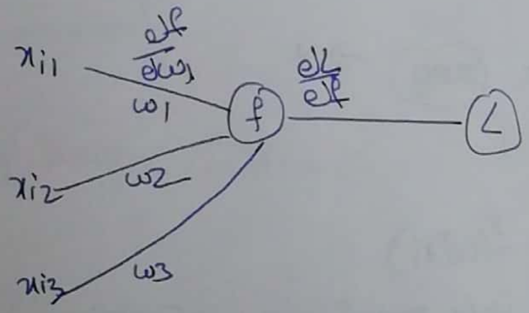
$(w_i)_{new} = (w_i)_{old} - \eta \left[\frac{\partial L}{\partial w_i} \right]_{(w_i)}$

for iter = 1 to k

{

$\nabla_w L = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4} \right]^T$

$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial w_1}$ → chain rule of diff


x_{i1}
$\frac{\partial f}{\partial w_1}$
w_1
f
$\frac{\partial L}{\partial f}$
L
x_{i2}
w_2
x_{i3}
w_3

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

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

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

$\frac{\partial f}{\partial w_1} = \frac{\partial f(w^T x_i)}{\partial w_1} = x_i$

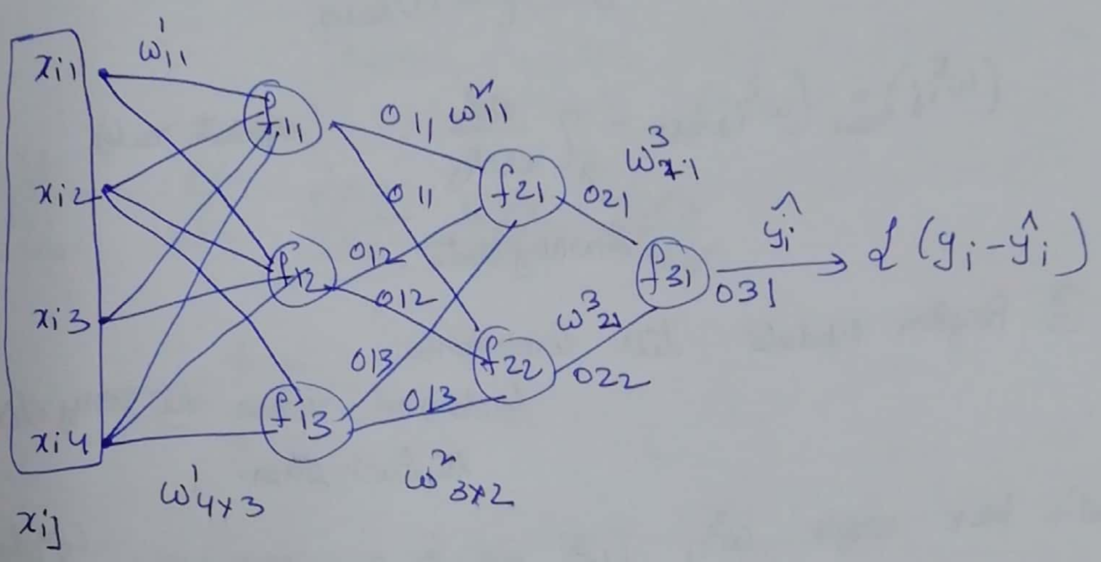
$\frac{\partial L_i}{\partial w_1} = -2 (y_i - \hat{y}_i)^{\checkmark} x_i$

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

47.8 Training a MLP : Chain Rule

$D = \{x_i, y_i\}$

$x_i \in \mathbb{R}^4$
$y_i \in \mathbb{R}$ } regression problem



~~step 1~~ $D = \{x_i, y_i\}$

determin $w^1_{4\times3}$, $w^2_{3\times2}$ $w^3_{2\times1}$ $= 20$

12 6 2

① $L = \sum_{i=1}^{n} \underbrace{(y_i - \hat{y}_i)^2}_{sq.loss} + reg.$

$\sum_{i,j,k} (w^k_{ij})^2 \to L2$

$\sum_{i,j,k} |w^k_{ij}| \to L1$

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

$L = \sum_{i=1}^{n} L_i + reg$

optimization : $\min_{w^1, w^2, w^3}$ the sq-loss L < sq.loss, reg

$$\boxed{\min_{w^k_{i,j}} L}$$

② SGD & GD

$$\frac{\partial L}{\partial w^k_{ij}} \quad \text{(Generic)}$$

ⓐ Initialize w^k_{ij} randomly } → lots of techaque

ⓑ Suppose we have a weight $(w^k_{ij})_{new}$

$$(w^k_{ij})_{new} = (w^k_{ij})_{old} - \underset{\nearrow \text{ learning rate}}{\eta} \frac{\partial L}{\partial w^k_{ij}} \qquad \text{(update rule)}$$

ⓒ Perform update till Convergence
↳ old & new value are very close to Each other.

Let's pick weight w^3_{11} (w^3_{11} impacts $\underset{\Uparrow \hat{y}_i}{O_{31}}$ which impact $\overset{\text{Loss functn}}{\hat{y}_i}$)

(look before page)

$$\left.\begin{aligned} \frac{\partial L}{\partial w^3_{11}} &= \underbrace{\frac{\partial L}{\partial O_{31}}}_{1} \cdot \underbrace{\frac{\partial O_{31}}{\partial w^3_{11}}}_{2} \quad \leftarrow \text{chain rule} \\ \frac{\partial L}{w^3_{21}} &= \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w^3_{21}} \end{aligned}\right\} w^3$$

$$\left.\begin{aligned} \frac{\partial L}{w^2_{11}} &= \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial w^2_{11}} \\ \frac{\partial L}{w^2_{21}} &= \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial w^2_{21}} \\ \frac{\partial L}{w^3_{31}} &= \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial w^2_{31}} \end{aligned}\right\} w^2$$

$\cancel{\frac{\partial L}{\partial w_{11}^{1}}} =$

$$\begin{tikzcd}
 & f(x) \arrow[dr] & \\
x \arrow[ur] \arrow[dr] & & h(f(x), g(x)) \\
 & g(x) \arrow[ur] &
\end{tikzcd}$$

$$\frac{\partial h}{\partial x} = \boxed{\frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial x}} \underset{\text{sum.}}{\boxed{+}} \boxed{\frac{\partial h}{\partial g} + \frac{\partial g}{\partial x}} \rightarrow \text{chain rule}$$

$$\cancel{\frac{\partial L}{\partial w_{11}^{1}}} = \cancel{\frac{\partial L}{\partial O_{31}}} \; \cancel{\frac{\partial O_{31}}{\partial O_{21}}} \; \cancel{\frac{\partial O_{21}}{\partial O_{11}}} \; \cancel{\frac{\partial O_{11}}{\partial w_{11}}}$$

$$\begin{tikzcd}
 & f \arrow[dr, dash] & & \\
x \arrow[ur, dash] \arrow[dr, dash] & & h \arrow[r, dash] & k \\
 & g \arrow[ur, dash] & &
\end{tikzcd}$$

$$\frac{\partial k}{\partial x} = \frac{\partial k}{\partial h} \cdot \frac{\partial h}{\partial x} \longrightarrow \boxed{\frac{\partial h}{\partial x} = \frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial x} + \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x}}$$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial O_{31}} \cdot \boxed{\frac{\partial O_{31}}{\partial w'_{11}}}$$

$$\frac{\partial O_{31}}{\partial w'_{11}} = \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial w'_{11}} \oplus$$

$$\frac{\partial O_{31}}{\partial O_{22}} \cdot \frac{\partial O_{22}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial w'_{11}}$$

47.9 Training an MLP : Memorization

In Computer Science
↳ Algorithms → Dynamic Programming

Memoization

If there is any operation that is used many times repeatedly

{ It's a good idea to compute it once → save it → reused it

⇓ Speed up

⇓ Take some memory

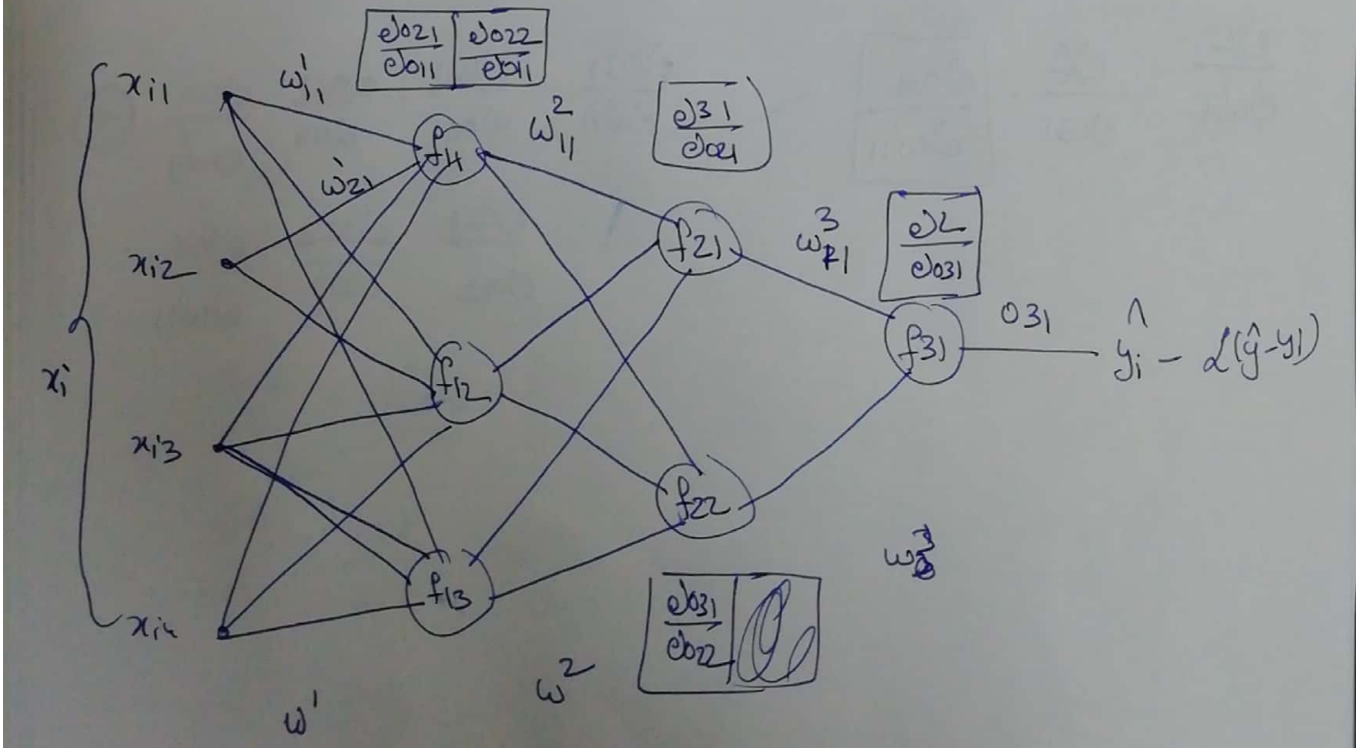
(Math) Chain rule + (Computer-science trick) memoization

$$\underbrace{\text{Chain rule} + \text{memoization}}$$

↓

Back Propagation

47.10 Back Propagation



$\frac{\partial O_{21}}{\partial O_{11}}$
$\frac{\partial O_{22}}{\partial O_{11}}$
x_{i1}
w^1_{11}
w^1_{21}
f_{11}
w^2_{11}
$\frac{\partial O_{31}}{\partial O_{21}}$
x_{i2}
f_{21}
w^3_{21}
$\frac{\partial L}{\partial O_{31}}$
x_i
f_{12}
f_{31}
O_{31}
$\hat{y}_i - L(\hat{y}-y_i)$
x_{i3}
f_{22}
w^3_{2}
f_{13}
$\frac{\partial O_{31}}{\partial O_{22}}$
x_{i4}
w^2
w^1

$D = \{x_i, y_i\}$

1) Initalize $w^k_{i,j}$'s

2) for Each x_i in D:

a) Pass x_i forward through the n/w → Forward Bropagation.

b) Compute the loss $\mathcal{L}(\hat{y}_i, y_i)$

c) Compute all the derivative using chain rule & memoization

d) update weights from End of The n/w to The start

<u>for Ex:</u>

Let's take w^3_{11}

$$\frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w^3_{11}}$$

$$(w^3_{11})_{new} = (w^3_{11})_{old} - \eta\left(\frac{\partial L}{\partial w^3_{11}}\right)$$

Baacword propagation.

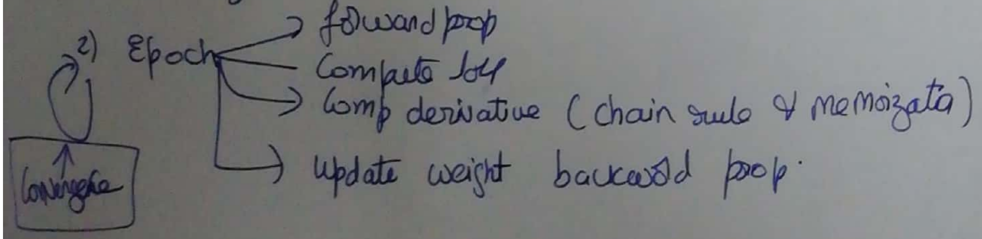
3) Repeat step2 till Convergence.

Epoch: Input all the points in the dataset once it is called an epoch
If we 5 times then It is called 5 Epoch

In real time we will run multiple Epoch's

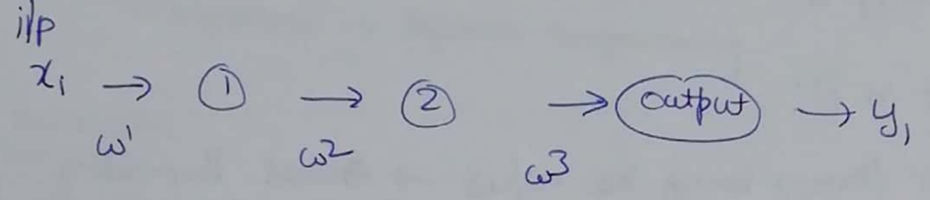
Back propation

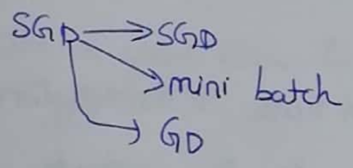
1) Initalize



2) Epoch → forward prop → Compute loss → Comp derivative (chain rule & memozata) → update weight backward prop.

V.V. Imp → Back propagation will only work if our activation functions are differentiable





{ Keeping all the data points in RAM & Computing "∂" using D is Extremly time Consuming

mini-batch based back propagation is widely used technique

① 10K points in D

↳ mini batch = 100 → $\boxed{64, 128, 256, 32}$ → RAM

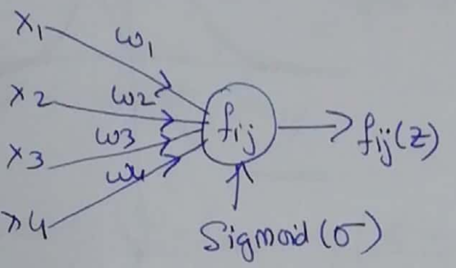
Epoch : $\frac{10,000}{100} = 100$

② For Each batch of size 100

Forward prop, L, $\frac{\partial L}{\partial w_{ij}^k}$, update, back prop

47.10 Activation functions

In 1980 & 90's only two activation function sigmoid & tanh.



$z = \sum w_i x_i = w^T x$

$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$

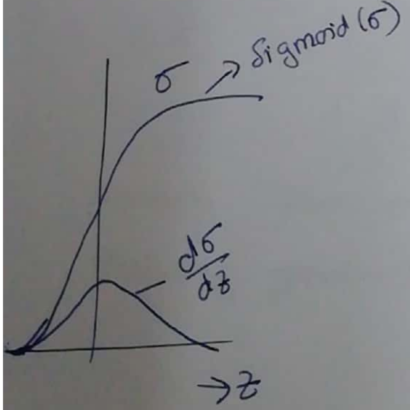
$\sigma(z) = \frac{e^z}{1+e^z}$ & $\frac{1}{1+e^{-z}}$ (This is we will use in LR)

Activation function

- should be differentiable
- should be Easy & fast.

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

$\frac{\partial \sigma}{\partial z} = \sigma(z)\,(1-\sigma(z))$



$\leq \frac{d}{dz} \leq 1$

lign b/w 0 & 1 (σ)

$\boxed{\frac{da}{dz} = a(1-a)}$

$a = (\underbrace{1+e^{-z}}_{g(z)})^{-1}$ $\quad \underbrace{\quad}_{a(g)}$

$\frac{da}{dz} = \frac{d\,a(g)}{g} \cdot \frac{d\,g(z)}{dz}$

$= \frac{d}{dg} g^{-1} \cdot \frac{d}{dz}(1+e^{-z})$

$= -1\, g^{-2} \cdot \frac{d}{dz} 1 + \frac{d}{dz} e^{\overset{h=-z}{-z}}$

$= \frac{-1}{g^2} \cdot \underset{0}{\downarrow} + \frac{d}{dh} e^h \cdot \frac{d}{dz}(-z)$

$e^h \cdot -1$

$-e^h$

$= \frac{-1}{g^2} \quad \boxed{-(e^{-z})}$

$\cdot\,(0 + -e^{-z}$

$= e^{-z}/g^2 = \frac{e^{-z}}{(1+e^{-z})^2}$

$\frac{da}{dz} = \frac{1+e^{-z}-1}{(1+e^{-z})^2}$

$= \frac{1+e^{-z}}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2}$

$= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2}$

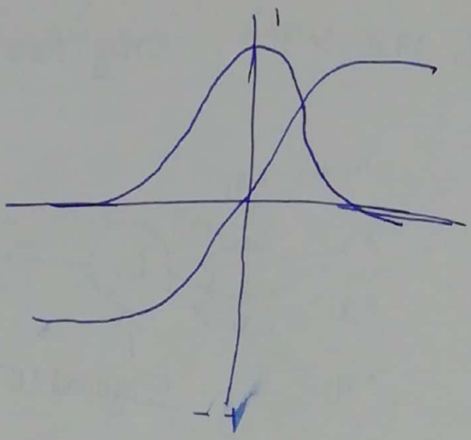
$= a - a^2$

$\boxed{= a(1-a)}$

Tanh function

$$a = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$\frac{d\tanh}{dz} = 1 - \tanh^{v}(z)$$



→ tanh lies to -1 to 1

→ $0 \leq \frac{\tanh}{dz} \leq 1$

47.12 **Vanishing Gradient Descent**

→ { 80's, 90's, 00's } → nn faced a problem called

→ Typically ppl used Sigmoid

To update a weight $\boxed{w^1_{11}}$

$$(w^1_{11})_{new} = (w^1_{11})_{old} - \eta\left(\frac{\partial L}{\partial w_{11}}\right)$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial O_{31}}\left[\frac{\partial O_{31}}{\partial O_{21}}\cdot\frac{\partial O_{21}}{\partial O_{11}}\cdot\frac{\partial O_{11}}{\partial w^1_{11}} + \frac{\partial O_{31}}{\partial O_{22}}\cdot\frac{\partial O_{22}}{\partial O_{11}}\cdot\frac{\partial O_{11}}{\partial w^1_{11}}\right]$$

$$\frac{\partial O_{31}}{\partial O_{21}} = \frac{\partial f_{31}}{\partial O_{21}}$$

$$0 < \frac{\partial \overset{\sigma}{\boxed{f_{31}}}}{\partial O_{2}} < 1$$

Vanishing grad. → V.V. small.

Exploding grad → VV large.

47.13 Bias - Variance Tradeoff.

① # layers ↑ => more weights/params

⇓

higher chance of overfitting

⇓

high Variance.

② 1-layer = Log.Reg + +/1 → higher chance of underfit

⇓

high bias.

→ Typically There is a higher chance of overfitting

↳ we can restrict by adding a regularization parameter

$$\mathcal{L} = \sum_{i=1}^{n} \text{loss}(y_i, \hat{y}_i) + \sum_{i,j,k} (w_{ij}^{k})^2$$

we can Either L1 & L2 regularization.

$$\mathcal{L} = \sum_{i=1}^{n} \text{loss}_i + \cancel{\lambda} \text{ reg on weight}$$

larger λ ⇒ lesser overfit

L1 reg ⇒ sparsity => some $w^k_{ij} = 0$

⇓

MLP sparse

→ here 'λ' is a hyper parameter

→ # of layers ↑ => var ↑

} hyper parameters

47.14 Decision surface: Play Ground

playground.tensorflow.org/#activation=tanh&batchsize=10 - - - ——— -