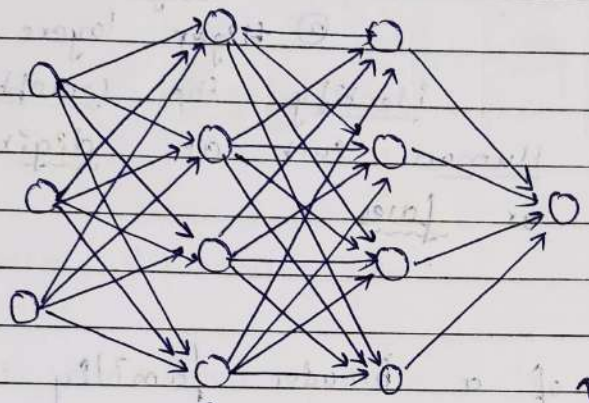
Deep learning :-

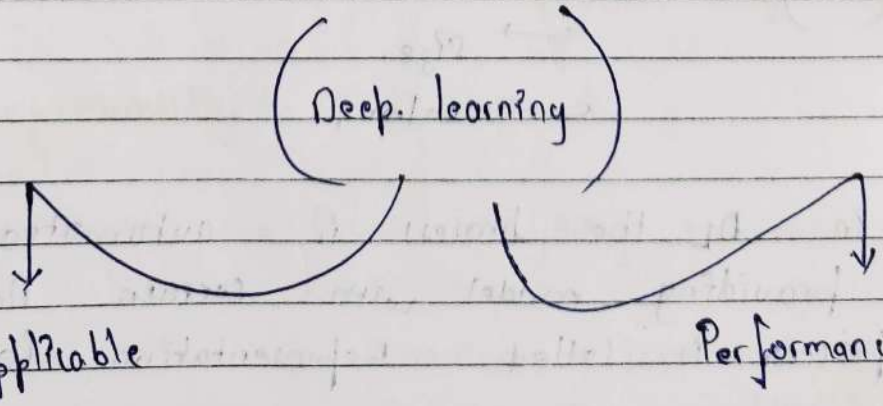
→ DL algo. attempts to draw similar conclusions as humans could by continually analyzing data with a given logical structure. called neural network.

eg.



→ Different types of neural network. :-

1. ANN
2. CNN → ex (works best on image data)
3. RNN → " " " " speech / text
4. GAN → generate thing. (text, images)



PAGE NO.:
DATE: / /

- DP ⟶ (Algo uses multilayer).
  → Progressively extract higher level features

eg. (Image processing) ⇒ ① lower layers may identify edges / shapes.
② Higer layers may identify the concepts relevant to Human such as digit or letters or faces.

- (DL) is part of a broader family of (ML) methods based on artificial neural network with representation learning.

- Representation learning :-
  → ex :- for Dog / cat classification in ML you have to create features manually.

(ML) → Features.
↳ size.
↳ color.

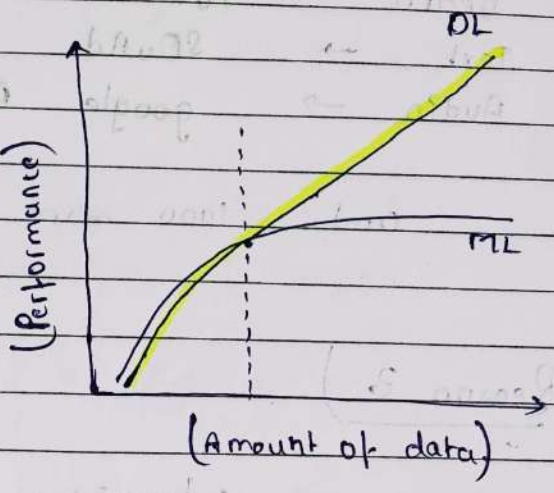
But.
In DL the process is automated by providing model with Data.
this process is called Representation learning.

Deep Learning V/s Machine Learning.

PAGE NO.:
DATE: / /

① Data Dependency :-

DL need more data as compaire to machine learing.



- DL is Data Hungry

② Hardware Dependency :-

{ DL me hamari jyada complex matrix multiplication ka rana hota hai }

→ which needs GPU to run (Quickly)

↑
(PowerFull)

③ Training Time :-

- In DL Training time is high

④ Features selection :-

ML → manual feature creation.

DL → Automatically done by model.

⑤ Interpretability :-

ML → we know everything happening in each and every step.

DL → Proces of extraction of feature cannot be explained to anyone It is like a black box.

Data for Deep learning → (Reason 1)

PAGE NO.:
DATE: / /

① Image → microsoft coco
② Video → Youtube 8M (6.1 million videos)
③ Text → SQuAd (Wikipedia 1,50,000)
④ Audio → google Audio set (20,00,000 sound clips)

and 1000 more data sets available. !!

• (Reason 2)

(IT) → transistor → 2 years (2)
→ cost → half

(DL) → lot of data → matrix operations → CPU → low speed → (2010)
→ NVIDIA → CUDA → GPU

Parallel processing

GPU → reduces 10-15 x time than CPU

→ Hardware

• FPGA → FAST, low power reprogramable, customizable
↳ expensive
eg. Xilinx

• ASIC
→ TPU → Tensor Processing unit (google) ↓ google colab GPUs option.
→ Edge TPU → Drone, watches, smart glass

→ NPU → Neural Processing units

PAGE NO.:
DATE: / /

- Reason 3 → {frameworks / libraries}.

sk learn → ML

for Deep Learning → ① Pytorch (facebook)
② Tensorflow (Google)

- Tensor flow. (2015)

→ Powerfull → Difficult

(keras) (Tensorflow)

(keras + Tensorflow)

(user)

↳ excessable to everyone.

- more industry driven

- Pytorch

AI researchers loved Pt.

→

- research Driven.

→ for conversion of code in two libraries.

Pytorch ⟷ Tensorflow

⇒ Drop drow GUI [ ] export → Pytorch
→ tensorflow

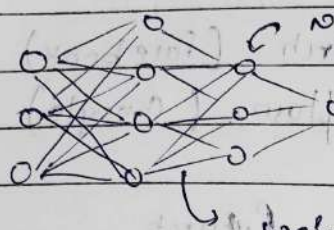
↓

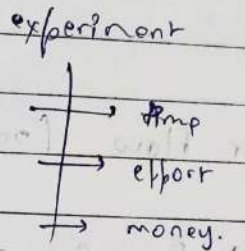
AutoML or (Custom vision AI) or (create ML)

PAGE NO. :
DATE : / /

- (Reason 4) → Deep learning Architectures.

→ Creating different architectures for best results





→ use existing architecture.

{transfer learning}

- Image classification → ResNET
- Text classification → BERT
- Image segmentation → UNeT
- Imag dram → Pix 2 Pix
- object detection → Yolo
- Speech generation → Wave NET

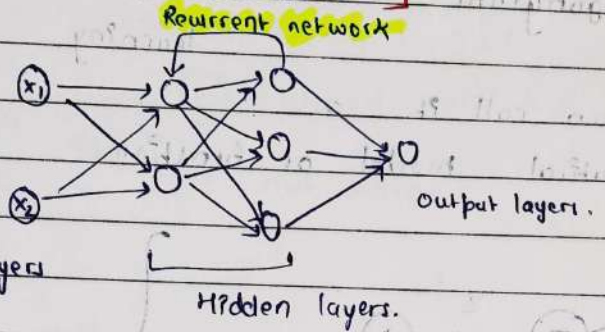
PAGE NO. :
DATE : / /

- Types of Neural Network.

1.) [Multi layer Perceptron.] { supervised Problems.}

2.) [Convolutional Neural Network.] { ~~Image~~ object detection etc }

3.) [Recurrent Neural Network]



eg LSTM

4.) [Auto encoders.]

5.) [Generative Adverserial Networks]

- History! (OL)

→ Applications.

(1) Game-playing agents. { Deep reanforcement learning }
(2) Virtual Assistants.
(3) Image colorization
(4) Adding audio to mute videos
(5) Image caption generation.
(6) Text translation.
(7) Pixel Restoration
(8) object detection

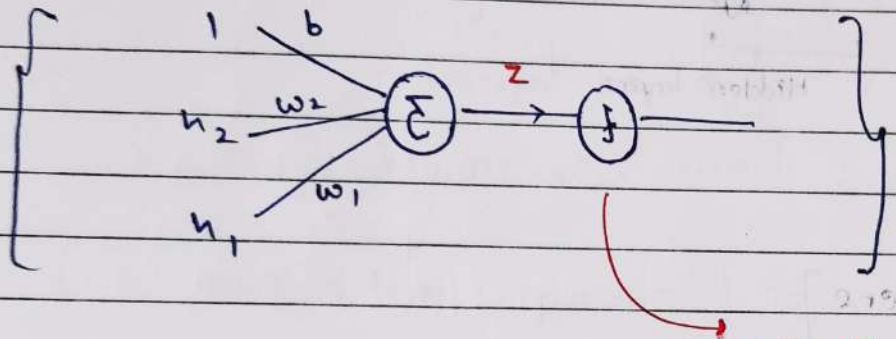
PAGE NO. :
DATE : / /

- **Deep dreaming.**

* Perceptron ??

Algorithm → supervised machine learning

well,
we can also call it a mathematical model or function



w_1, w_2 → weights

b → bais

Activation function { convert z into a perticular range }

$$z = w_1 x_1 + w_2 x_2 + b$$

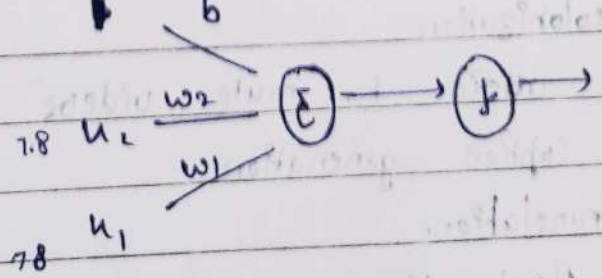
eg :-

iq	CgPA	Placement
78	7.8	1
69	5.1	0
:	:	:

By me mathematical model of perceptrons

x_1 → iq

x_2 → cgpa



The whole purpose of training process is to find the value of w_1, w_2, b in training

→ let say By training we get $w_1 = 1$ $w_2 = 2$

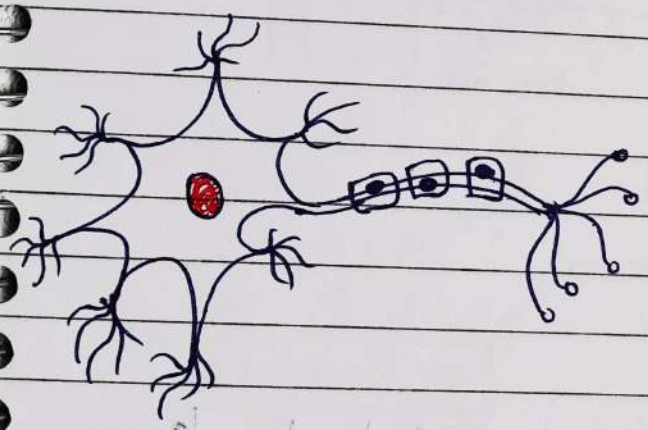
and student with

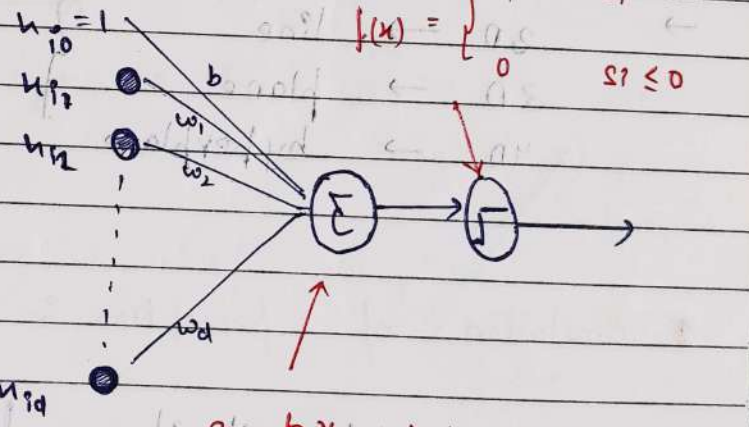
IQ	cg PA	Placem	$b = 3$
100	5.1	?	

So. $z = 100 \times 1 + 5.1 \times 2 + 3 \times 1$

$z \geq 0$ → (1) placed

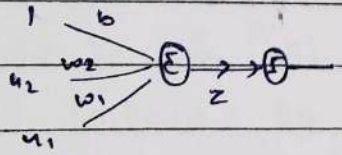
$z \leq 0$ → 0 not placed.



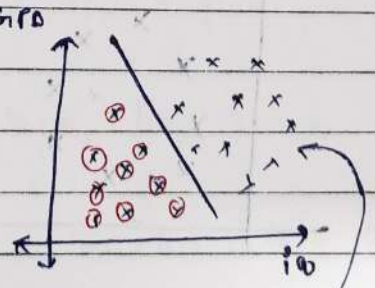


Neuron Perceptron

- Geometric intuition



$z = w_1 x_1 + w_2 x_2 + b$



$y = f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$

$AX + BY + C \geq 0$ region

PAGE NO. :
DATE : / /

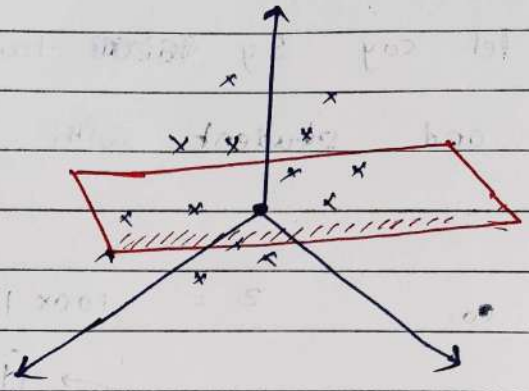
let say I have 3 inputs.

iq | CGPA | 12th marks | Placed

$f(z) = w_1x_1 + w_2x_2 + w_3x_3 + b \geq 0$

$Ax + By + Cz + b \geq 0$

↑ Plane.



3D

↳ How Perceptron acts.

- It is a **Binary classifier.**

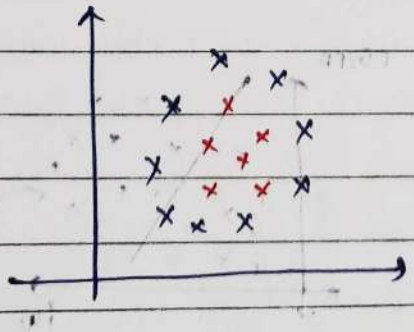
→

2D → line.
3D → plane
4D → hyperplane

* limitation of perceptron :-

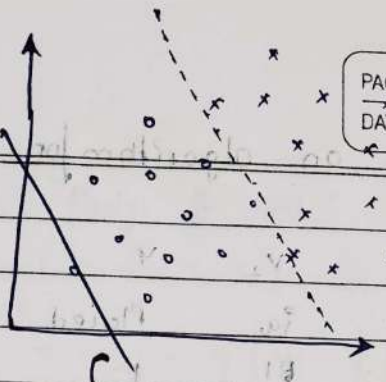
- It can only classify linear or sort of linear cases.
  It fail in non-linearly separable cases.

eg.

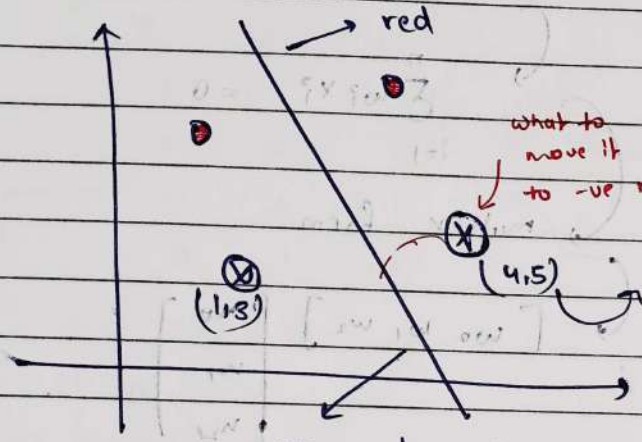


∴ (Perceptron fails Here)

Perceptron trick



Apply transformation to converge the random line to original



$2x + 3y + 5 = 0$

Trick

2	3	5
4	5	1

$-2x \quad -2y + 4 = 0$

(4,5,1) Put (1)

(1)

This line will bring (4,5) in non red region

or

(to move a point to ~~(-)~~ +ve region

2	3	5
1	3	1

(2)

$3x + 6y + 6 = 0$

- If a negative pt is in positive region. DO (1)
- If a +ve pt in (-)ve region. DO (2)

Note :- In ML do not apply these big changes instantly. Instead do that.

n = learning rate = 0.01

So

2	3	5
0.01	0.03	0.01

Same for (-)ve

$2.01x + 3.03y + 5.01 = 0$

PAGE NO. :
DATE : / /

- Let's Build an algorithm for this :-

x_0	x_1	x_2	y
	CgPA	ia	Placed
1	7.5	81	1
1	8.9	109	1
1	:	:	0
1	:	:	1

$AX + BY + C = 0$

$w_0 + w_1 x_1 + w_2 x_2 = 0$

$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

$\sum_{i=1}^{n} w_i x_i = 0$

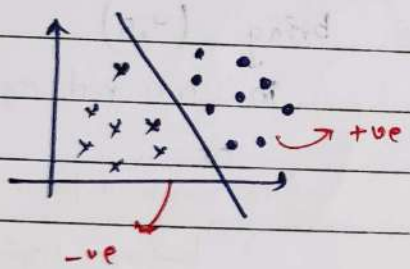
matrix form

$\begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$

i/p

$\sum w_i x_i \geq 0 \quad -(1)$

$< 0 \quad -(0)$



epoch → 1000, $\eta = 0.01$

for i in range (epochs) :

randomly select a row

if $x_i \in (N)$ and $\sum w_i x_i \geq 0$:

$w_{new} = w_{old} - \eta x_i$

if $x_i \in (P)$ and $\sum w_i x_i < 0$:

$w_{new} = w_{old} + \eta x_i$

PAGE NO. :
DATE : / /

Simplifying algo.

for i in Range (0, 1000) :

Pick random student. ;

$$w_{new} = w_{old} + \eta (y_i - \hat{y}_i) x_i$$

Intuition

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$	
1	1	0	→ $w_{new} = w_{old}$
0	0	0	
1	0	1	→ $w_1 = w_0 + \eta x_i$
0	1	-1	→ $w_n = w_0 - \eta x_i$

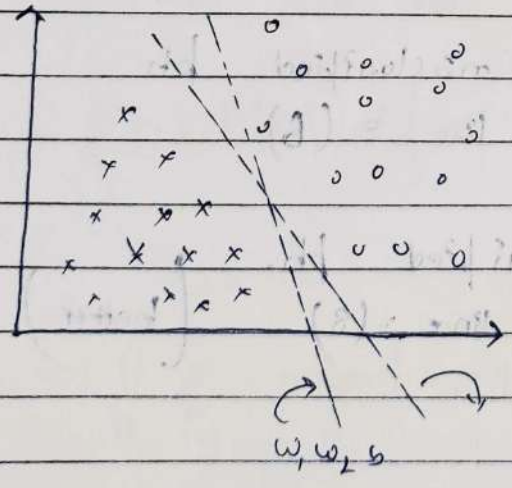
y_i → actual result

$\hat{y}_i$ → model prediction

fve pt in -ve region

→ Practical done in (DL folder)

✱ Problems with Perceptron



- result by ml trick can't be quantify. (kitna acha result aaya hai??
- running code u can get any of these lines. → can be different with 2-3 times.

Loss function (Perceptron) :-

PAGE NO.:
DATE: / /

→ the method to tell which solution is better then the other.

- ML loss functions

LR → mse $(y_i - \hat{y}_i)^2$

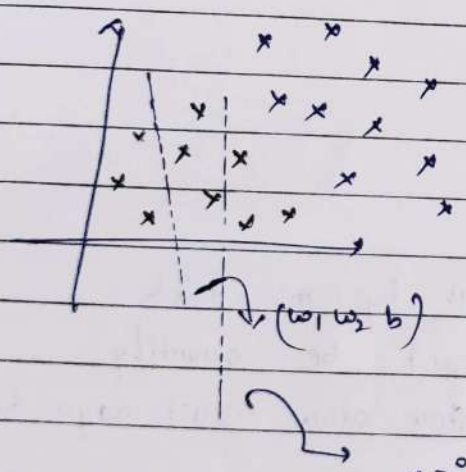
Logi° Reg. → Log loss

SVM → Hinge loss

→ lets find a loss function for perceptron trick.

① $f(w_1, w_2, b)$ → number → error.

misclassified pts. (number)

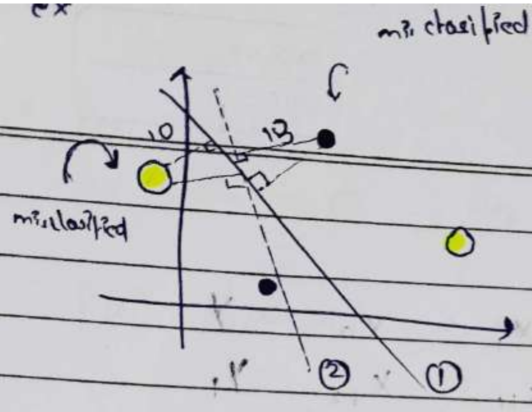


(w_1, w_2, b) → no° of misclassified pts for this line = (6)

no° of misclassified pts. for this line = (3) (better)

② loss function = sum of ⊥ dist° of misclassified pt from the line.

Prev. funi° we took every misclassified pt. of the same weight.



① LF = 10 + 13

= 23

② So same for other as well man compair.

* sklearn (formula) for Perceptron loss function :-

↳ on page of (Stochastic Gradient Decent)

↳ (optimization)

$$L(w_1, w_2, b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

↳ for Regularization

let take this for now

$$L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$$

$$f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$$

$$L = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i)) \quad \text{— ①}$$

∴ (n = no. of rows in data.)

∴ we need to find w_1, w_2, b for which value of (L) in eqⁿ ① is minimum

PAGE NO.:
DATE: / /

Explanation of loss function:-

$$L = \frac{1}{n} \sum_{i=1}^{n} \max\left(0, -y_i f(x_i)\right)$$

x_1	x_2	y
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2
⋮	⋮	⋮
x_{i1}	x_{i2}	y_i
⋮	⋮	⋮
x_n		

n rows

where

$$f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$$

now

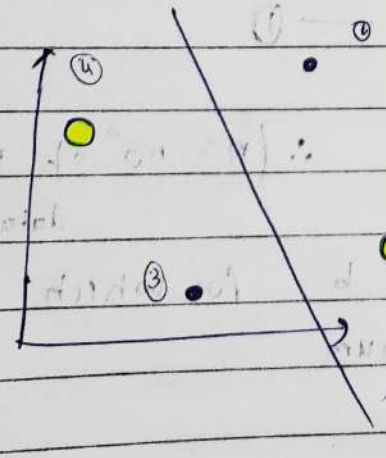
$$\max\left(0, -y_i f(x_i)\right)$$

$$\begin{cases} -y_i f(x_i) & \text{if } -y_i f(x_i) \geq 0 \\ 0 & \text{if } -y_i f(x_i) < 0 \end{cases}$$

↳ lets do for only two pts

$$L = \frac{1}{2} \left[\max\left(0, -y_1 f(x_1)\right) + \max\left(0, -y_2 f(x_2)\right)\right]$$

→ Geometric intuition.



CGPA	iq	placed
7	8	+1
6	8	-1
4	2	1
1	1	-1

PAGE NO. :
DATE : / /

- for pt (2)
  - $y_i = 1$ $\quad \max(0, -y_i f(x_i))$

$f(x_i) = (w_1 x_1 + w_2 x_2 + b) \geq 0$

y_i = +ve

f_x = +ve

$-y_i f(x_i) < 0$

So $\max(0, -y_i f(x_i)) = 0$

↳ -ve.

- for pt (3)
  - $y_i = -1$ $\quad \max(0, -y_i f(x_i))$

$f(x_i) = w_1 x_1 + w_2 x_2 + b.$

$f(x_i) < 0 \rightarrow$ Due to pt in -ve region

$y_i < 0$ = -ve

$f(x_i) < 0$ = -ve

So $\max(0, -y_i f(x_i)) = 0$

- for pt (4)

$y_i = +1$

$f(x_i) = w_1 x_1 + w_2 x_2 + b < 0 \rightarrow$ in +ve side of line

$\max(0, -y_i f(x_i))$

$= -y_i \cdot f(x_i)$

$y_i > 0$ = +ve

$f(x_i)$ = +ve

- for pt (1).

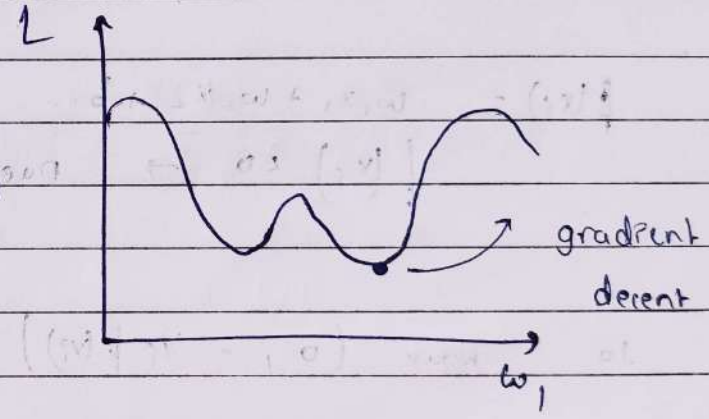
$\max(0, -y_i f(x_i)) = -y_i f(x_i)$

PAGE NO. :
DATE : / /

To find values for which values of w_1, w_2, b so that loss function becomes minimum.

$$L = \underset{w_1, w_2, b}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \max\left(0, -y_i f(x_i)\right)$$

→ Gradient Descent :-



start a loop.

for i in epochs (100/1000) :

$$w_1 = w_1 + \eta \frac{\partial L}{\partial w_1}$$

$$w_2 = w_2 + \eta \frac{\partial L}{\partial w_2}$$

$$b = b + \eta \frac{\partial L}{\partial b}$$

$\eta = 0.01$

loop ends.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f(x_i)} * \frac{\partial f(x_i)}{\partial w_1}$$

PAGE NO. :
DATE : / /

$$\left\{ L = \frac{1}{n} \sum \max\left(0, -y_i f(x_i)\right) \right\}$$

$$\frac{\partial L}{\partial f(x_i)} = \begin{cases} 0 & -y_i f(x_i) < 0 \\ -y_i & -y_i f(x_i) \geq 0 \end{cases}$$

$$\frac{\partial f(x_i)}{\partial w_1} = x_{i1}$$

(1) $$\frac{\partial L}{\partial w_1} = \begin{cases} 0 & -y_i f(x_i) < 0 \\ -y_i x_{i1} & -y_i f(x_i) > 0 \end{cases}$$

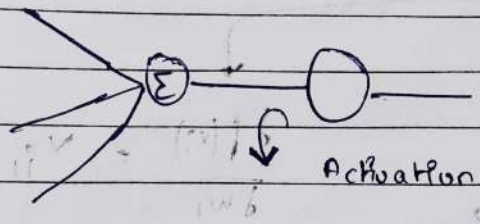
(2) $$\frac{\partial L}{\partial w_1} = \begin{cases} 0 & -y_i f(x_i) < 0 \\ -y_i x_{i2} & -y_i f(x_i) > 0 \end{cases}$$

(3) $$\frac{\partial L}{\partial b} = \begin{cases} 0 & -y_i f(x_i) < 0 \\ -y_i & -y_i f(x_i) > 0 \end{cases}$$

Let's convert it to code.

PAGE NO. :
DATE : / /

• More loss function



it is a mathematical model which is very flexible

→ flexibility => we can change activation function (A.F)

(step) ⟶ (sigmoid func°)

u(t)

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

• if Activation function = $\sigma(z) = \frac{1}{1+e^{-z}}$

then

loss function = binary cross entropy.

$$L = -y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)$$

↓

so, 0 → out put will probability

(Schocke gradient decent)

so we can say.

Perceptron = logestic Regression with A.f

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

DATE : / /

- If I am working on multiclass classification problem.

the. A.f = Soft max

Loss funct = Categorical cross entropy.

$$L = \sum_{i=1}^{n} y_i \log(\hat{y}_i) \quad \text{— ①}$$

Soft max ⇒ $$f = \frac{e^{z_i}}{\sum_{i=1}^{k} e^{z_i}} \quad \text{— ②}$$

let say we have 3 classes.

$$f = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

So Perceptron = Softmax regression

for ① & ②

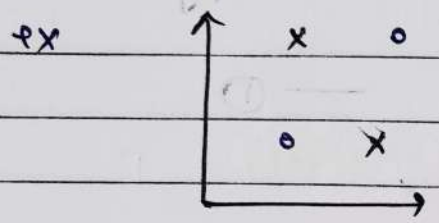
Summary

Loss function	Activation	Output
• Hinge loss	Step	perceptron - Binary classification (-1, 1)
• log-loss (binary cross entropy)	~~Sigmoid~~ Softmax	logistic regression. (0-1) - Binary C.
• Categorical cross entropy	Soft max	Softmax regression → multiclass classifi°

PAGE NO. :
DATE : / /

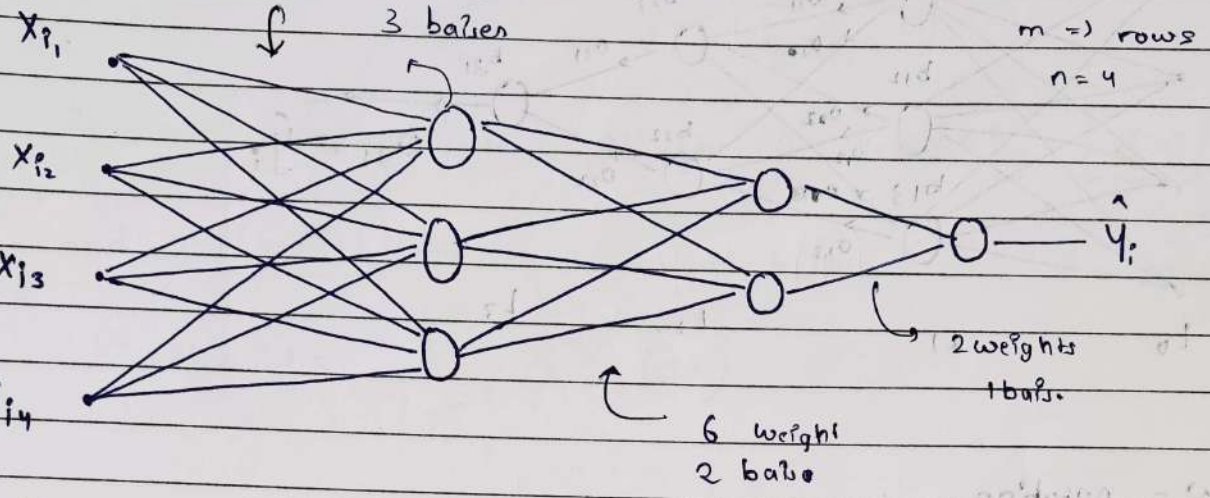
- Problem with perceptron :-

→ It is of No use with non-linear data.
(cannot classify XOR function).

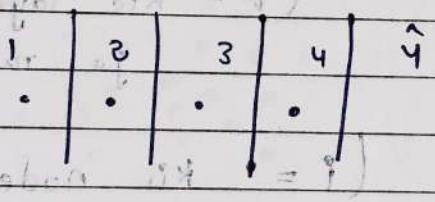


thes. type of functions cannot be classified with perceptron

Multi - Layer - Perceptron (MLP)



Input layer (L_0) | hidden layer (L_1) | hidden layer (L_2) | output layer (L_3)

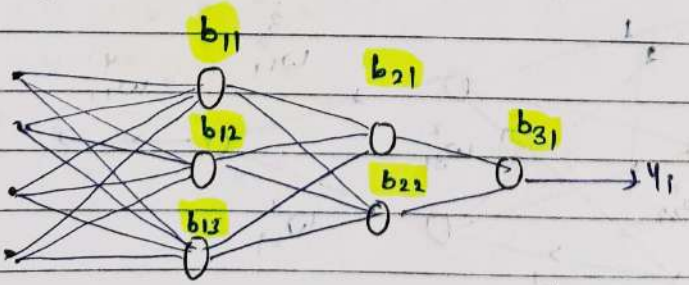


So In this above neural network structure there are

$= (12+3) + (6+2) + (2+1)$

(26 trainable - parameters.)

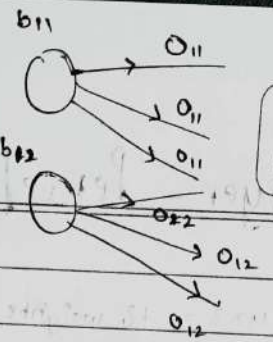
- (w, b) - Notation.

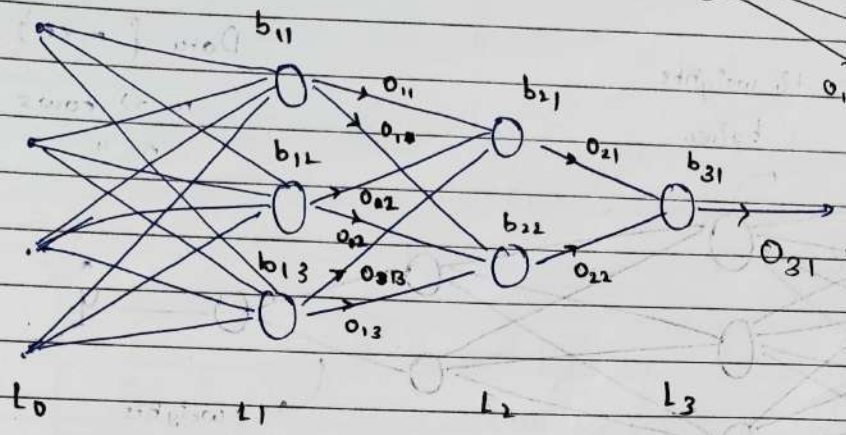


b_{ij}

i = layer

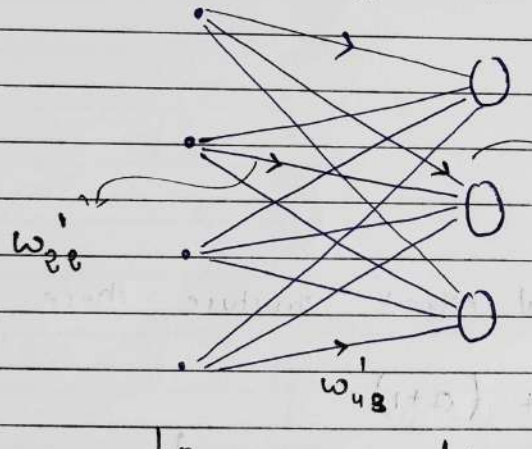
j = node n°





PAGE NO. :
DATE : / /

ω - notation.



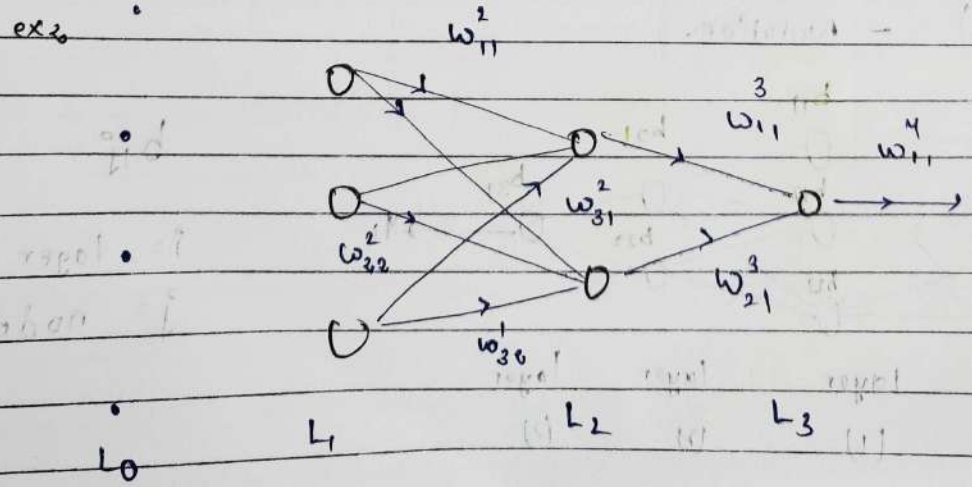
w^K_{ij}

(K = Kis layer me ja raha hai)

(i = Kis node se aa raha hai)

(j = Kis node me ja raha hai)

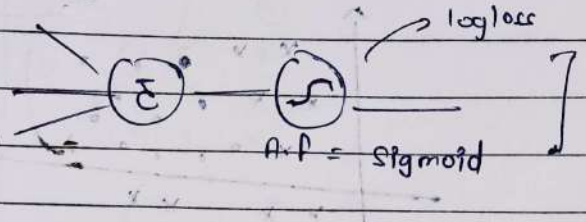
ex:-



MLP (intiution)

PAGE NO.:
DATE: / /

↳ classification



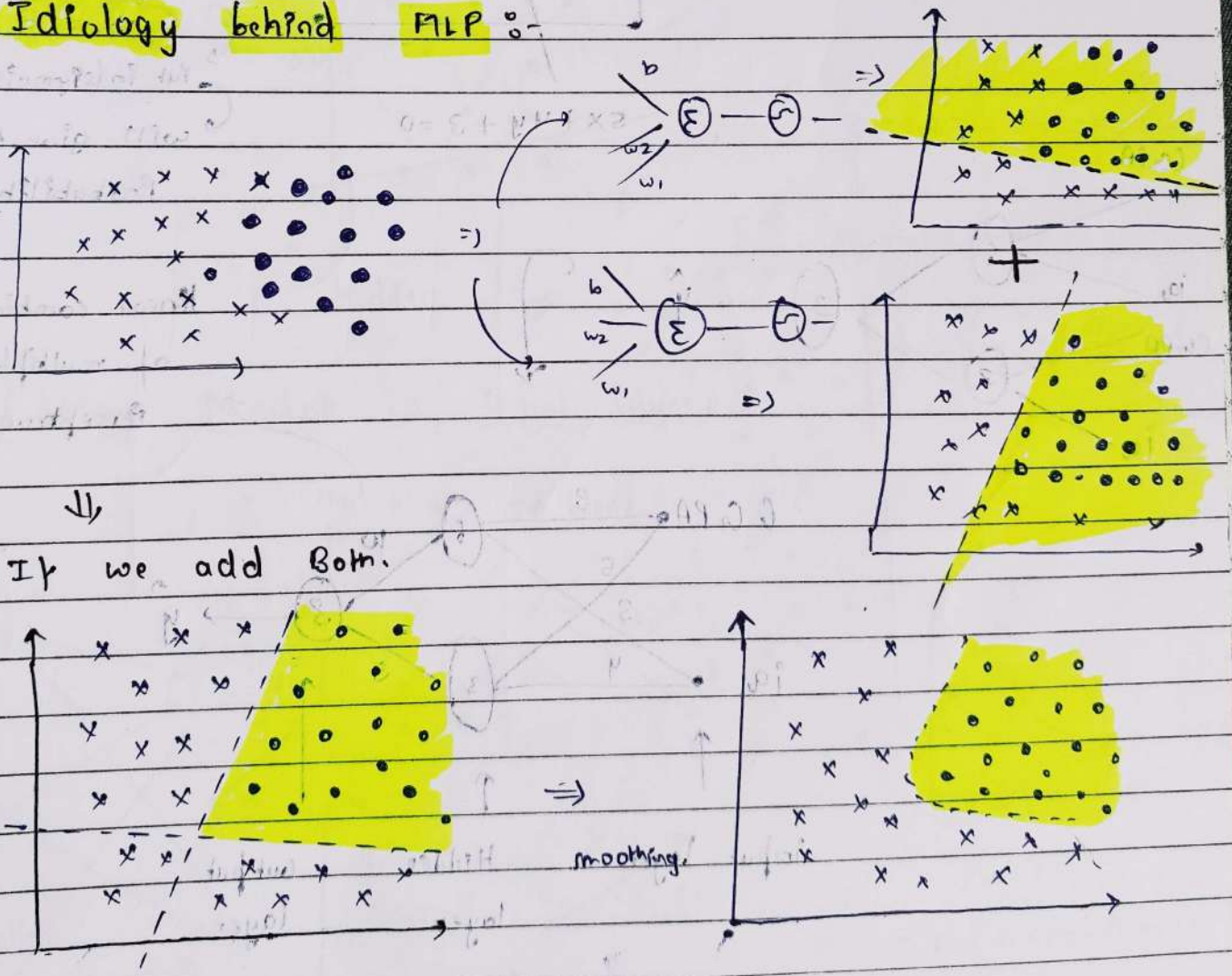
for MLP

for this perceptron we will get 2 probability. $P(x)$ ↓ of event to happen

and $1 - P(x)$ → event not to happen.

$$\text{Sigmoid} = \frac{1}{1-e^{-z}} \Rightarrow (0 \to 1)$$

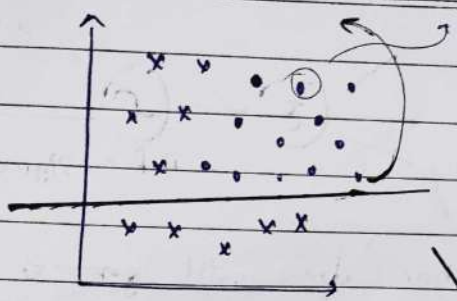
* **Idiology behind MLP :-**

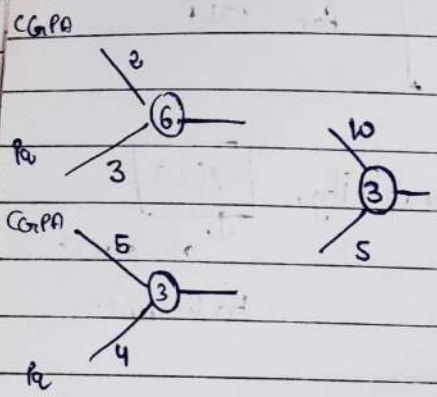


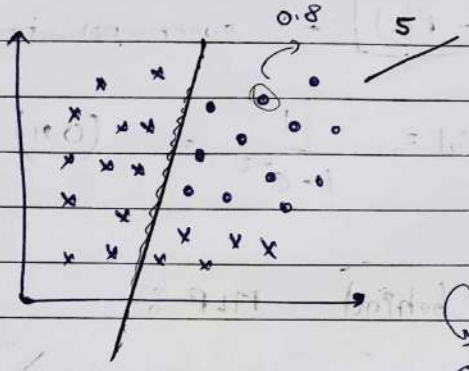
If we add Both.

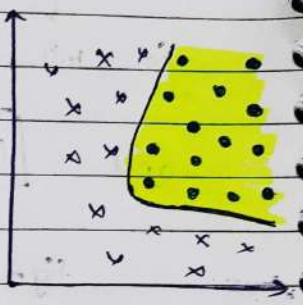
PAGE NO.:
DATE: / /

$2x + 3y + 6 = 0$







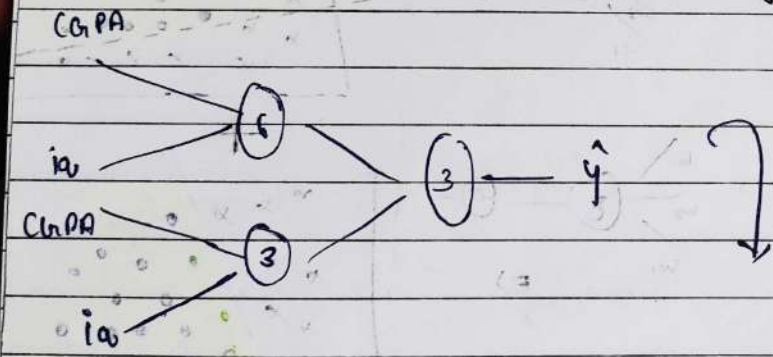


Bais = 3

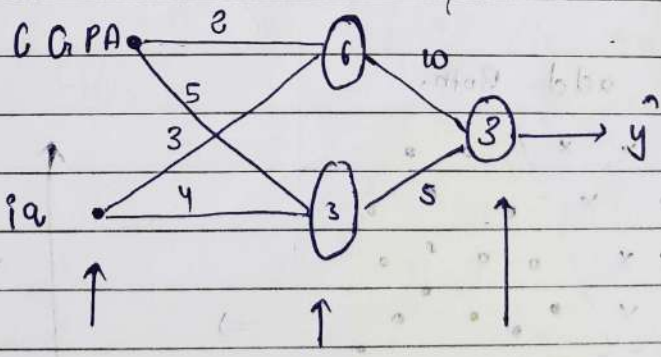
$z = 0.6 \times 10 + 0.8 \times 5 + 3$

Put in (sigmoid) will give the Probability

$5x + 4y + 3 = 0$



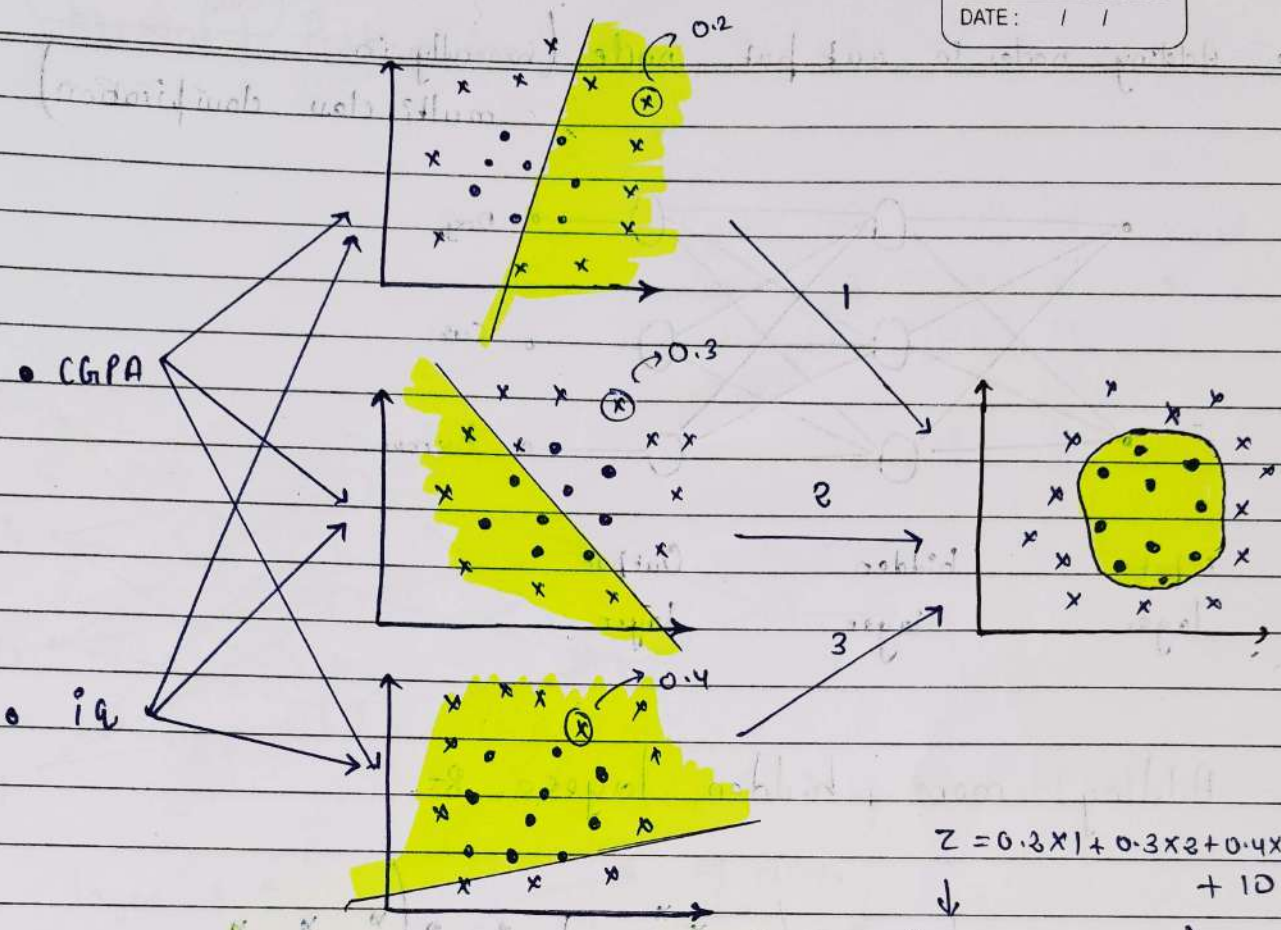
linear combination of multiple Perceptron



input layer Hidden layer output layer

out put of all nodes will probabilities.

- Adding Nodes in hidden layers.



$z = 0.3 \times 1 + 0.3 \times 2 + 0.4 \times 3 + 10$

↓

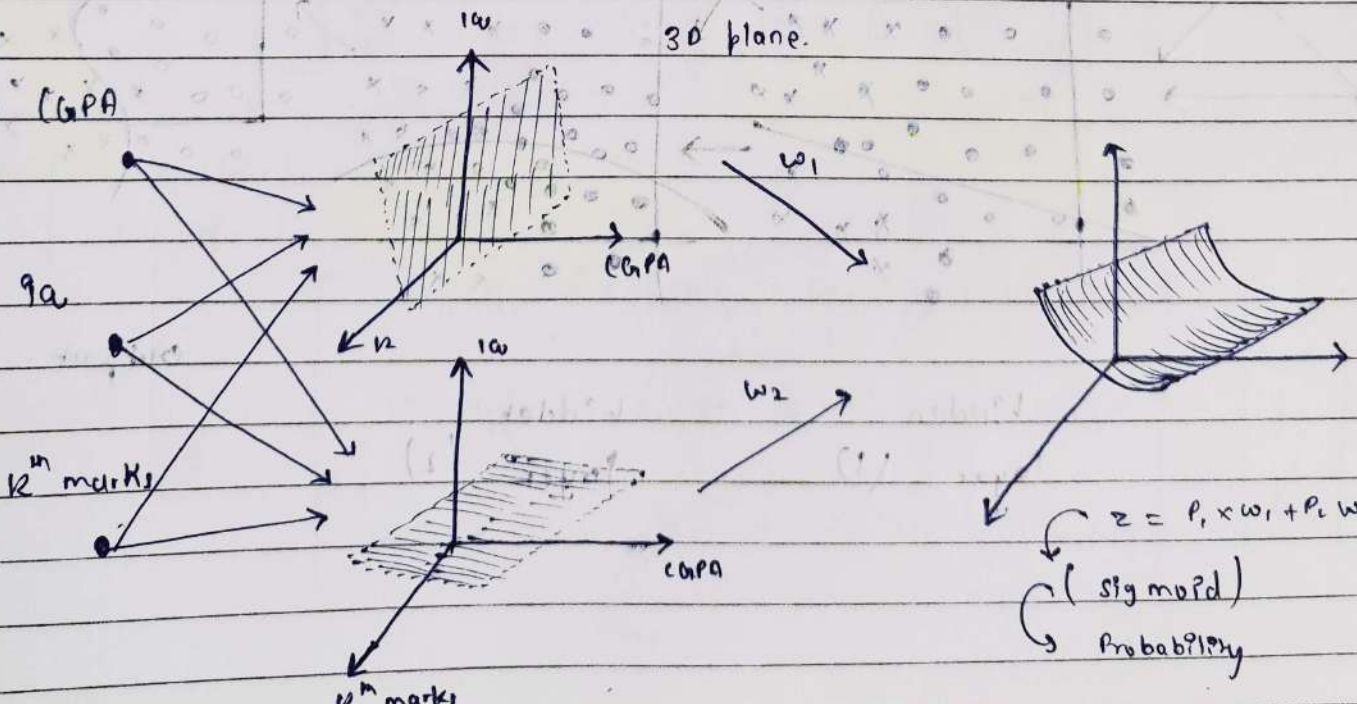
(Sigmoid function)

↓

Probability b/w (0 → 1)

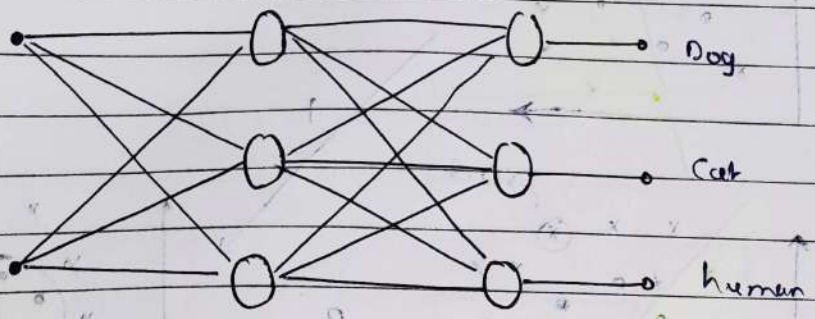
hidden layer

- Adding Nodes in Input layers :-



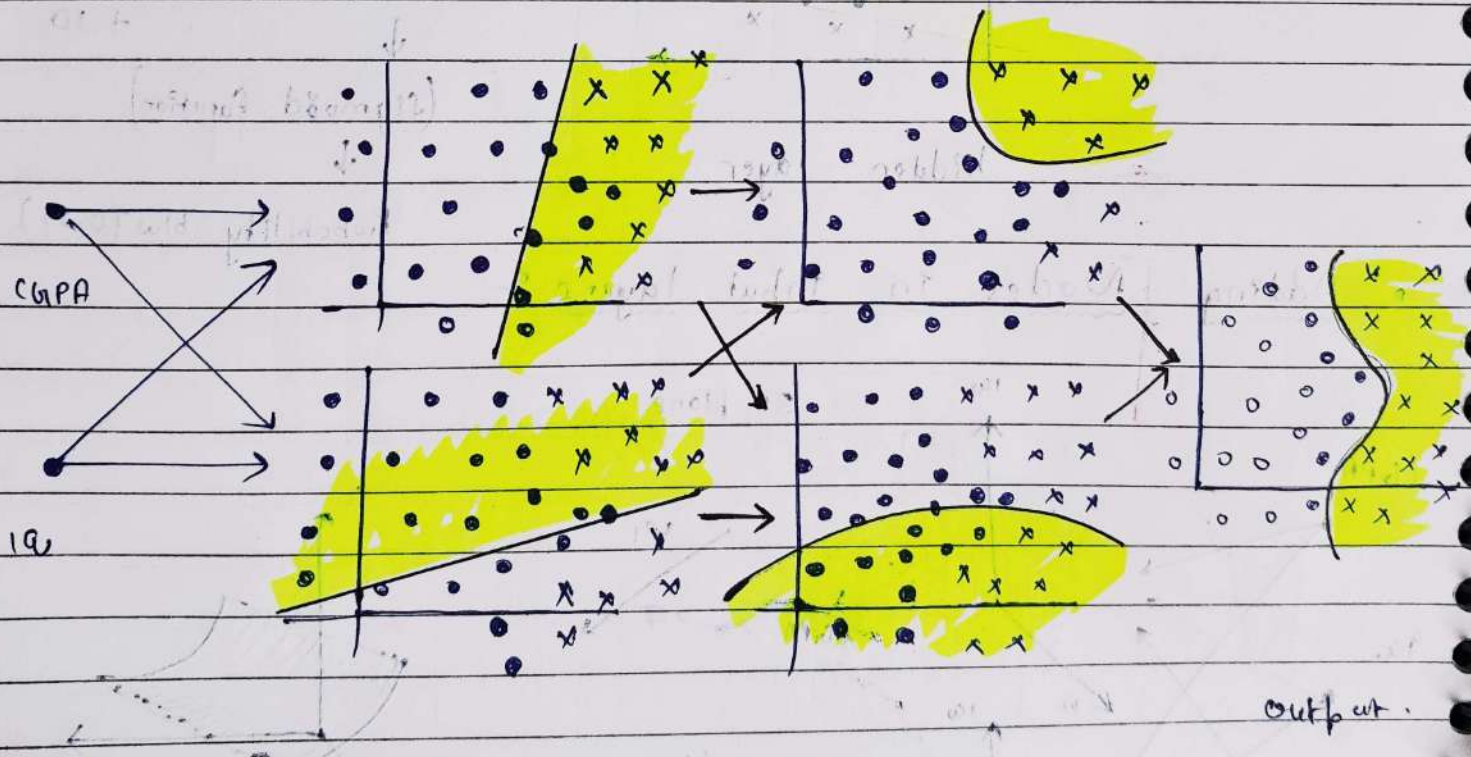
PAGE NO.:
DATE: / /

- Adding nodes in output node. (Generally in multi-class classification)



Input layer | Hidden layer | Output layer

- Adding more hidden layers. &-

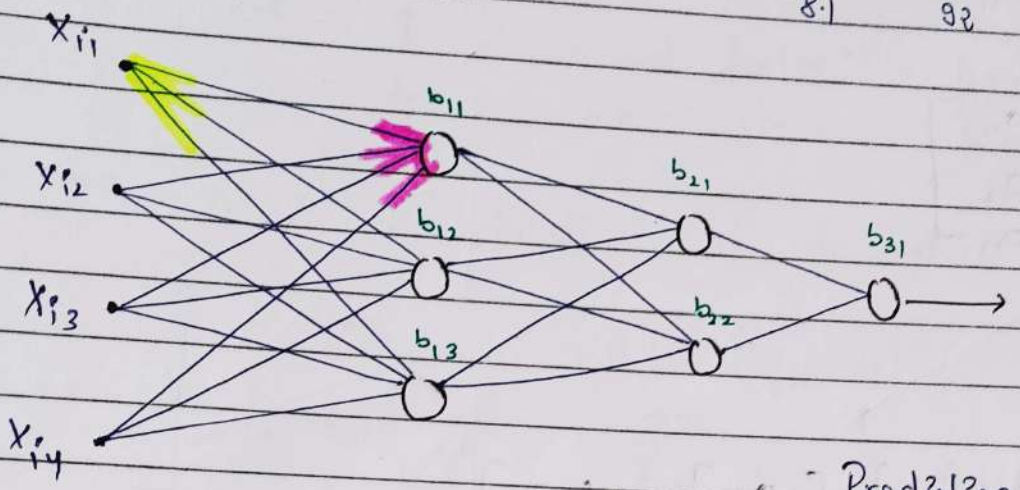


Hidden layer (1) | Hidden layer (2)

Forward Propagation :-

PAGE NO.:
DATE: / /

CGPA	IQ	10^{th}	12^{th}	placed
7.2	78	69	81	0
8.1	92	75	76	1



Prediction $\Rightarrow \sigma(w^T x + b) \Rightarrow$

layer 1 #

[all exiting x_{i1}] $\rightarrow$ T

$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{44} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

4×1 $\quad 3 \times 1$

[all entering b_{11} node]

$4 \times 3^T \rightarrow 3 \times 4$

$$\begin{bmatrix} w'_{11} x_{i1} + w'_{21} x_{i2} + w'_{31} x_{i3} + w'_{41} x_{i4} \\ w'_{12} x_{i1} + w'_{22} x_{i2} + w'_{32} x_{i3} + w'_{42} x_{i4} \\ w'_{13} x_{i1} + w'_{23} x_{12} + w'_{33} x_{i3} + w'_{43} x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

PAGE NO. :
DATE : / /

$$\sigma\left(\begin{bmatrix} w'_{11}x_{i1} + w'_{21}x_{i2} + w'_{31}x_{i3} + w'_{41}x_{i4} + b_{11} \\ w'_{12}x_{i1} + w'_{22}x_{i2} + w'_{32}x_{i3} + w'_{42}x_{i4} + b_{12} \\ w'_{13}x_{i1} + w'_{23}x_{i2} + w'_{33}x_{i3} + w'_{43}x_{i4} + b_{13} \end{bmatrix}\right)$$

$$= \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix}$$

layer (2) #

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix}^T \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

$$\sigma\left(\begin{bmatrix} w^2_{11}o_{11} + w^2_{21}o_{12} + w^2_{31}o_{13} + b_{21} \\ w^2_{12}o_{11} + w^2_{22}o_{12} + w^2_{32}o_{13} + b_{22} \end{bmatrix}\right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

layer (3) #

$$\begin{bmatrix} w^3_{11} \\ w^3_{21} \end{bmatrix}^T \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix}$$

$$\sigma\left(\begin{bmatrix} w^2_{11}o_{21} + w^2_{21}o_{22} + b_{31} \end{bmatrix}\right) = \hat{y}_i$$

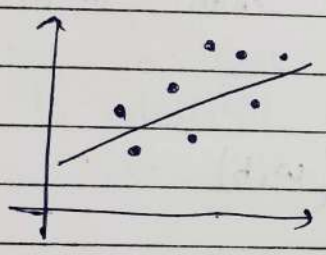
Loss functions :-

PAGE NO. :
DATE : / /

→ loss function is a method of evaluating How well your algorithm is modelling your dataset.

If L.f => ↑ model_perfor° = poor
if L.f = ↓ model_perfor = great.

→ It is a function of parameters.



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

$\hat{y}_i$: $mx_i + b$

$L(m, b)$: By changing (m, b) to make $L(m, b)$ min

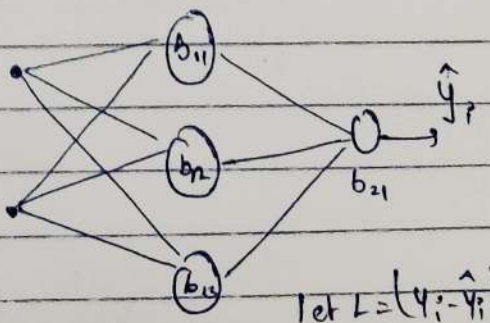
→ Why is loss function important?

[You can't improve what you can't measure]
→ Peter Drucker

→ L.f is the eye on model (ML/dL)
tells you where to go to correct your performance.

→ loss function in Deep learning?

Cgpa	ia	Package (LPA)
7.1	85	3.2
8.5	91	4.5



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

PAGE NO. :
DATE : / /

Starting with random (w, b)

→ we get a output $\hat{y}_i$

but it in $(L \cdot f)$

let = $L \cdot f = (y_i - \hat{y}_i)^2$ = gives a no°

according to this adjust the value of (w, b)

→ with the help of (gradient decent)

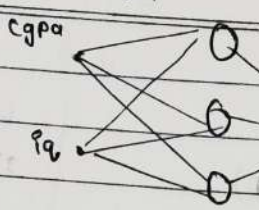
And the method will be repeated with other row with multiple epoches.

And at end we will get $\min(w, b)$

Loss function in DL

- object detection
  - → focal loss
- Regression
  - mse
  - MAE
  - Huber loss
- classification
  - binary cross entropy
  - Categorical cross entro
- Auto encoder
  - KL divergence
- embedding
  - Tripled loss
- GAN
  - discriminator loss
  - MinMax gan loss

Cost function (v/s) Loss function



cgpa	iq	Package (y_i)	Prediction ($\hat{y}_i$)
6.3	100	6.3	6.1
7.1	91	4.1	4
8.5	83	3.5	3.7
9.2	102	7.2	7

loss func° → Single training eg

$$L = (y_i - \hat{y}_i)^2$$
$$= (6.3 - 6.1)^2$$

Cost function

→ Is calculated for the whole Batch.

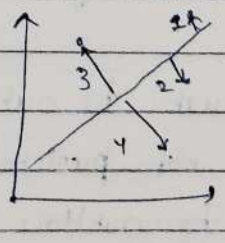
$$L.F = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$$

$$L.F = \frac{1}{4}\left[(6.1-6.3)^2 + (4.1-4)^2 + (3.5-3.7)^2 + (7.2-7)^2\right] = C.F$$

→ Types of Loss functions :-

(1) mean-squared-error (mse) :-

$$L.F = (y_i - \hat{y}_i)^2$$ → why square



let's say point

$y_i - \hat{y}_i$ = ① = 1, L.F = 1

② = 2 L.F = 4

③ = 3 L.F = 9

④ = 4 L.F = 16

as distance b/w pt and line matters

∴ (futher pt will be consider greater loss)

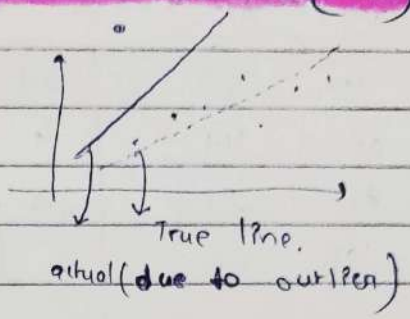
PAGE NO. :
DATE : / /

Advantages

1) easy to interpret
2) Differentiable (GD)
3) 1 local minima

Disadvantages.

1) error unit (squared) → diff
2) Robust to outliers (Not)



Note

→ for applying MSE in DL

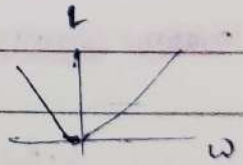
→ last node activation function should be ["linear"]

$$\text{Cost func}^{n} = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$$

(2) **Mean - absolute - error** :-

$$L.f = |y_i - \hat{y}_i|$$

$$C.f = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$



Advantage

It Robust to outliers (do not punishes extra to outliers

Disadvantage

It Not Differentiable
↳ GD ↘ differentiation

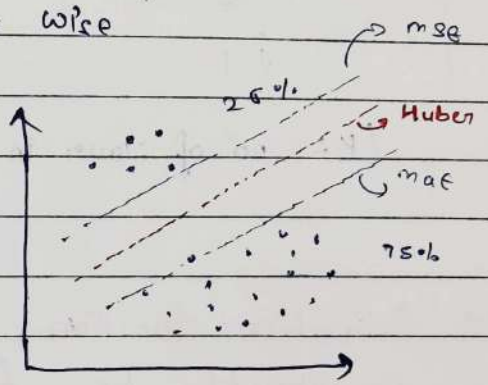
Conclusion ⇒ more outliers ⇒ MSE

less outliers = MAE

(3) **Huber loss** :-

$$L = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & \text{other wise} \end{cases}$$

mse ↑ (first branch) ; huber ↑ (for $|y-\hat{y}| \le \delta$) ; mae ↑ (second branch)



Note works best when there are very much outliers

(4) **Binary cross entropy** :- (logloss)

→ classification

→ Two classes.

Cgpa	Pa	Placement
.	.	1
		0
		1

$$\text{Loss\_function} = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

Note

$y \rightarrow$ actual

$\hat{y} \rightarrow$ predicted

Activation function can only be (sigmoid)

$$\text{cost\_function} = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

PAGE NO. :
DATE : / /

(5) Categorical Cross Entropy : [used in softmax Regression]

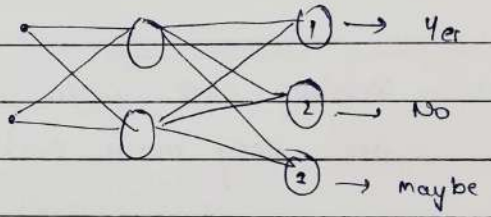
→ {Multidata classification}

$$\left\{ L = -\sum_{j=1}^{k} y_j \log(\hat{y}_j) \right\}$$

(k = no° of classes in data)

one-Hot encoding

cgpa	iq	placed	Yes	No	maybe
8	80	Yes	1	0	0
6	60	No	0	1	0
7	70	maybe	0	0	1



→ Activation function should be (Soft max)

⇒ Sparse categorical Cross Entropy.

cgpa	iq	placed	Encoding
7	70	Yes	1
8	80	No	2
6	60	maybe	3

→ no one Hot encoding is done

→ Fast than CCE

⇒ CCE & SCCE are same

But SCCE is Faster when you have many output categories.

$$\left\{ L.F = -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \right\} \quad \left\{ C = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} \log(\hat{y}_{ij}) \right\}$$

PAGE NO. :
DATE : / /

Summary

→ Reg → mse
↳ outliers — mae

→ Classification → binary → bce
↳ multi → ccs
↳ sce

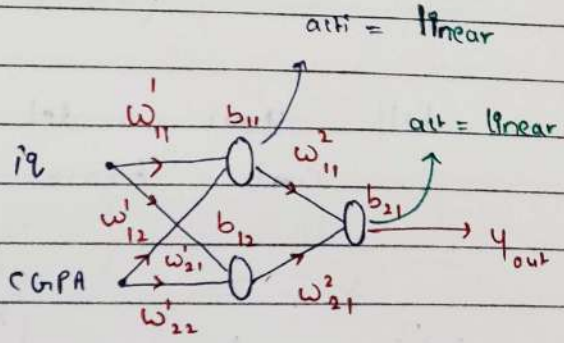
Back propogation :-

PAGE NO :
DATE : / /

- Algo that helps to find the correct value of weights and Baises.
- forward propogation
- Gradient decent.

→

iq	CGPA	LPA
80	8	3
60	9	5
70	5	8
120	7	11



→ Steps.

(1) Init w, b
↳ You can give random values.
↳ or you can initialize with $w \to 1, b \to 0$

(2) Select a point (row) → students.

(3) Predict (lpa) → forward prop [Dot product]

	iq	CGPA	LPA
① →	80	8	3

→ let say prediction comes to be (18 lpa)

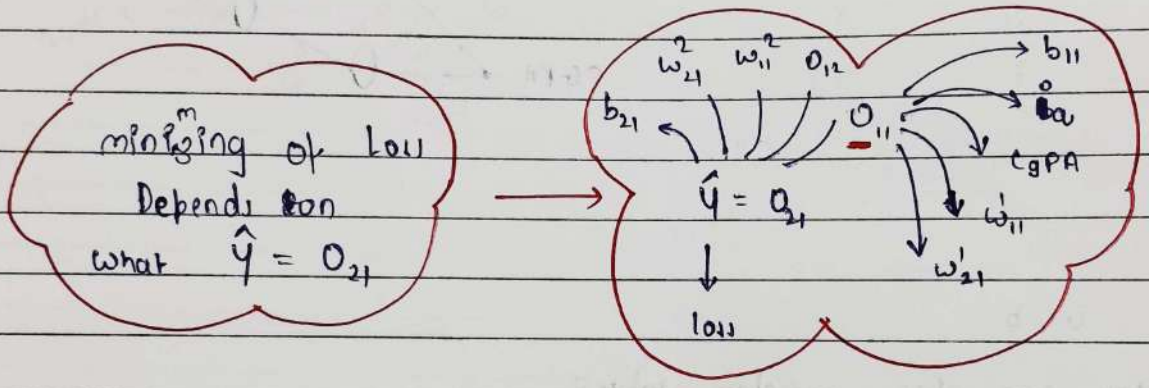
{ Bic weights / Baises are not correct. }

(3). Choose a loss function

for linear $L = (y - \hat{y})^2$ (mse)

$L = (3-18)^2 = 225$ ← error

tell your model to adjust the weights and biases to reduce it.



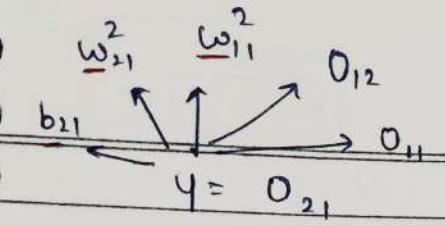
complex backward dependencies

→ to going backward and changing all parameters is called Back propagation

(4). Update weights | Biases :-

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

$$B_{new} = B_{old} - \eta \frac{\partial L}{\partial B_{old}}$$



PAGE NO.:
DATE: / /

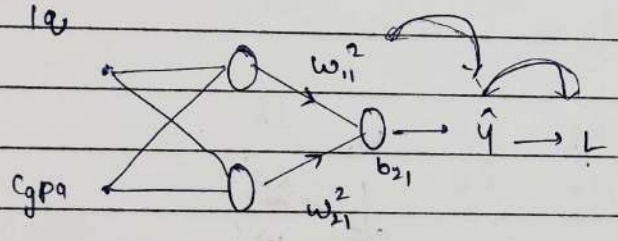
$$w_{11\,new}^2 = w_{11\,old}^2 - \eta \frac{\partial L}{\partial w_{11}^2}$$ needs to be calculated

$$w_{21\,new}^2 = w_{21\,old}^2 - \eta \frac{\partial L}{\partial w_{21}^2} \qquad b_{21\,new} = b_{21\,old} - \eta \frac{\partial L}{\partial b_{21}}$$

Target

$$\frac{\partial L}{\partial w_{11}^2}, \frac{\partial L}{\partial w_{21}^2}, \frac{\partial L}{\partial b_{21}} \;\Big|\; \frac{\partial L}{\partial w_{11}^1}, \frac{\partial L}{\partial w_{21}^1}, \frac{\partial L}{\partial b_{11}} \;\Big|\; \frac{\partial L}{\partial w_{12}^1}, \frac{\partial L}{\partial w_{22}^1}, \partial$$

ie



$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{w_{11}^2}$$

$\therefore$ (Chain rule)

$$\rightarrow \left\{ \frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} (y - \hat{y})^2 = -2(y - \hat{y}) \right\}$$

$$\left\{ \frac{\partial \hat{y}}{\partial w_{11}^2} = \frac{\partial}{\partial w_{11}^2} (O_{11} w_{11}^2 + O_{12} w_{21}^2 + b_{21}) \right\}$$

$$= O_{11}$$

So

$$\left\{ \frac{\partial L}{\partial w_{11}^2} = -2(y - \hat{y}) O_{11} \right\}$$

Similarly.

$$\left\{ \frac{\partial L}{\partial w_{12}^2} = -2(y - \hat{y}) O_{12} \right\}$$

deber° deber°

$L \longrightarrow \hat{y} \longrightarrow 5$

PAGE NO. :
DATE : / /

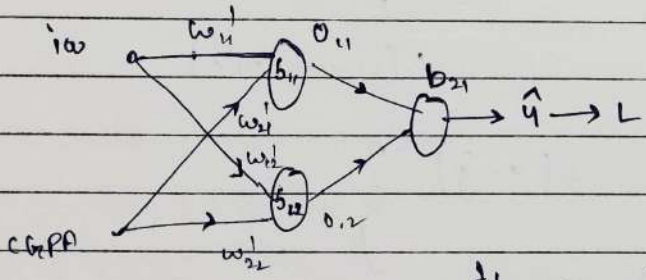
$$\frac{dL}{d\hat{y}} = \frac{d}{d\hat{y}}(y-\hat{y})^2 = -2(y-\hat{y})$$

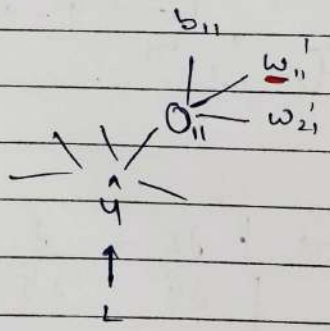
$$\frac{d\hat{y}}{db_{21}} = \frac{d}{db_{21}}\left[o_{11}w_{11}^2 + o_{12}w_{21}^2 + b_{21}\right] = 1$$

$$\boxed{\frac{dL}{db_{21}} = -2(y-\hat{y})}$$

→ we have calculated all the 3 derivatives in first block.

(2) $\left(\frac{dL}{dw_{11}'} + \frac{dL}{dw_{21}'} + \frac{dL}{db_{11}}\right)$





$$\frac{dL}{dw_{11}'} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{do_{11}} \cdot \frac{do_{11}}{dw_{11}'}$$

$$\frac{dL}{d\hat{y}} = -2(y-\hat{y})$$

$$\frac{d\hat{y}}{do_{11}} = \frac{d}{o_{11}}\left(o_{11}w_{11}^2 + o_{12}w_{21}^2 + b_{21}\right)$$

$$= w_{11}^2$$

$$\frac{do_{11}}{dw_{11}'} = \frac{d}{dw_{11}'}\left(w_{11}'(iw) + w_{21}'(CGPA) + b_{11}\right) = (iw)$$

PAGE NO.:
DATE: / /

$$\left(\frac{\partial L}{\partial w'_{11}} = -2(y-\hat{y}) \cdot w_{11}^2 \cdot (iq)\right)$$

$$\left(\frac{\partial L}{\partial w'_{21}} = -2(y-\hat{y}) \cdot w_{11}^2 \cdot (CGPA)\right)$$

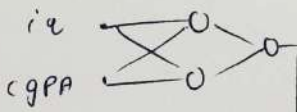
$$\left(\frac{\partial L}{\partial b_{11}} = -2(y-\hat{y}) \cdot w_{11}^2 \cdot (1)\right)$$

(3) $$\left(\frac{\partial L}{\partial w'_{12}}, \frac{\partial L}{\partial w'_{22}}, \frac{\partial L}{\partial b_{12}}\right)$$

$$\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial w'_{12}} = -2(y-\hat{y}) \cdot w_{21}^2 \cdot (iq)$$

$$\frac{\partial L}{\partial w'_{22}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial w'_{22}} = -2(y-\hat{y}) \cdot w_{21}^2 \cdot (GPA)$$

$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial b_{12}} = -2(y-\hat{y}) \cdot w_{21}^2 \cdot (1)$$



(9 parameter)

loop → (100 times) epoches.

for i in Range (4) :

(w → 1) (b → 0)

(a) 1st student → forward prop → predict (LPA)

(b) loss calculate (mse) →

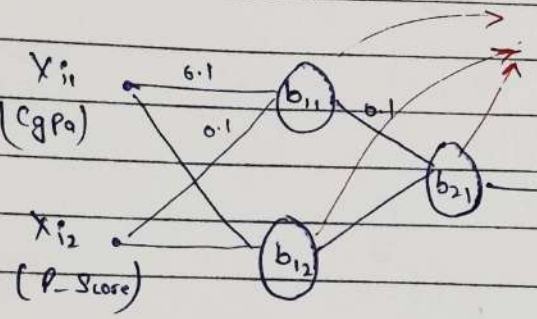
(c) adjust all weights and bais.

$$(w/b)_{new} = (w/b)_{old} - \eta \frac{dL}{dw_{old}}$$

(or)

→ Calculate avg loss for the epoch (9 times)

Back Propagation - classification

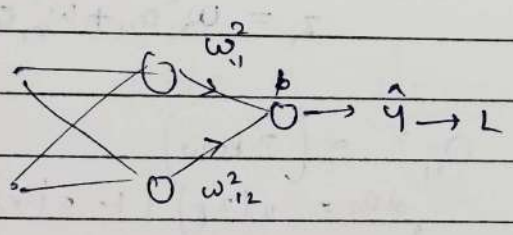


All three node have activation func° - (sigmoid)

$$\left\{ \text{Loss} = -y \log(\hat{y}) - (1-y) \log(1-\hat{y}) \right\}$$

↳ (Binary cross entropy)

CgPa	P_Score	Placed.
8	8	0
9	7	1
6	4	0
3	6	0
2	1	0



$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{11}^2} = -(y-\hat{y}) O_{11}$$

$$\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_{21}^2} = -(y-\hat{y}) O_{12}$$

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial b_{21}} = -(y-\hat{y})$$

$$\frac{\partial z}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left[-y \log(\hat{y}) + (1-y) \log(1-\hat{y}) \right]$$

$$= -\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})} = \frac{-y + y\hat{y} + \hat{y} - y\hat{y}}{\hat{y}(1-\hat{y})} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})}$$

$\hat{y} = \sigma(z)$

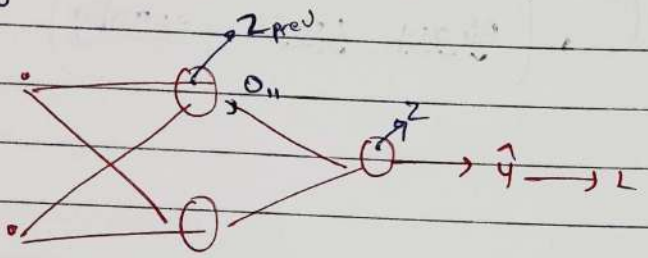
PAGE NO.:
DATE: / /

$$\frac{d\hat{y}}{dz} = \frac{d}{dz}\left(\sigma(z)\right) = \sigma(z)\left[1-\sigma(z)\right] = \hat{y}(1-\hat{y})$$

$$\rightarrow \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz} = -(y-\hat{y})$$

$$z = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$$

→ now



$$\frac{\partial L}{\partial w_{11}^1}, \frac{\partial L}{\partial w_{21}^1}, \frac{\partial L}{\partial b_{11}}$$

$(L \rightarrow \hat{y} \rightarrow z_p \rightarrow o_{11} \rightarrow z_{prev} \rightarrow w_{11}^1)$

$z_p = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$

$o_{11} = \sigma(z_{prev})$

$do_{11} = \sigma(z_p)\left[1-\sigma(z_p)\right]$

$\sigma(z_p)$

$$\frac{dL}{dw_{11}^1} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_p} \cdot \frac{dz_p}{do_{11}} \cdot \frac{do_{11}}{dz_{prev}} \cdot \frac{dz_{prev}}{dw_{11}^1}$$

$-(y-\hat{y})$ $\quad w_{11}^2$ $\quad o_{11}(1-o_{11})$ $\quad x_{11}$

$$\frac{dL}{dw_{11}^1} = -(y-\hat{y})\, w_{11}^2\, o_{11}(1-o_{11})\, x_{11}$$

$$\frac{dL}{dw_{21}^1} = -(y-\hat{y})\, w_{11}^2\, o_{11}(1-o_{11})\, x_{12}$$

$$\frac{dL}{db_{11}} = -(y-\hat{y})\, w_{11}^2\, o_{11}(1-o_{11})$$

$\frac{\partial L}{\partial w'_{11}}, \frac{\partial L}{\partial w'_{12}}, \frac{\partial L}{\partial b_{12}}$

PAGE NO.:
DATE: / /

$$\frac{\partial z_{1P}}{\partial o_{12}} = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$$

$$\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_{1P}} \times \frac{\partial z_{1P}}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial z_{1P}} \times \frac{\partial z_{1P}}{\partial w'_{12}}$$

$\frac{\partial L}{\partial \hat{y}} \rightarrow -(y - \hat{y})$

$\frac{\partial z_{1P}}{\partial o_{12}} \rightarrow w_{21}^2$

$\frac{\partial o_{12}}{\partial z_{1P}} \rightarrow o_{12}(1 - o_{12})$

$\frac{\partial z_{1P}}{\partial w'_{12}} \rightarrow x_{i1}$

$L \rightarrow \hat{y} \rightarrow z_{1P} \rightarrow o_{12} \rightarrow z_P \rightarrow w'_{12}$

$$\frac{\partial L}{\partial w'_{12}} = -(y - \hat{y})\, w_{21}^2\, o_{12}(1 - o_{12})\, x_{i1}$$

$$\frac{\partial L}{\partial w'_{22}} = -(y - \hat{y})\, w_{21}^2\, o_{12}(1 - o_{12})\, x_{i2}$$

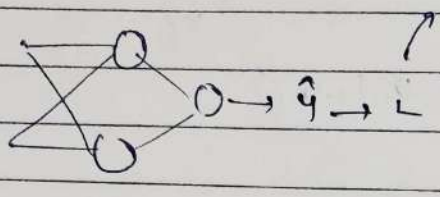
$$\frac{\partial L}{\partial b_{12}} = -(y - \hat{y})\, w_{21}^2\, o_{12}(1 - o_{12})$$

→ code is done as well.

PAGE NO. :
DATE : / /

→ The Why. (Back propogation).

→ **Loss function** is a function of **all trainable parameters**



$L = (y-\hat{y})^2$ → mathematical function

↓ cont.

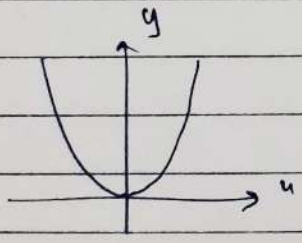
$L(?) \longrightarrow L(\hat{y})$

$\hat{y} = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$

$\hat{y} = w_{11}^2 [w_{11}^1 x_{11} + w_{21}^1 x_{12} + b_{11}] + w_{21}^2 [w_{12}^1 x_{11} + w_{22}^1 x_{12} + b_{12}] + b_{12}$

$\hat{y}$ → depends on all trainable parameters

→ Concept of minima :-



$y = u^2$ $\quad \frac{dy}{dx} = 2u = 0$

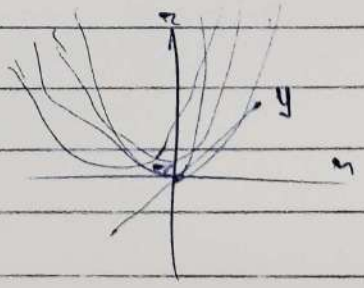
$\boxed{u = 0}$

$y \to \min \to \frac{dy}{du} = 0 \to (u = 0)$

---

let $z = u^2 + y^2$

$(u, y) \to z_{min}$



$\frac{\partial z}{\partial u} = 0 \qquad 2u = 0 \Rightarrow (u = 0)$

$\frac{\partial z}{\partial y} = 0 \qquad 2y = 0 = (y = 0)$

now

L → function of 9 parameter

9 dimensional → minima.

$$\left\{ \frac{dL}{\partial w_{11}} \cdots \frac{\partial L}{b_{12}} = 0 \right\}$$

- Back Propagation Assumption.

$$w_{new} = w_{old} - \eta \frac{dL}{dw}$$

small learning rate make the transition smooth.

ML1 Memorization.

PAGE NO. :
DATE : / /

A technique used to primarily speed up computer program by storing the results of expensive function calls and returning the cached result when the same input occurs again.

eg: Fibonacci Series

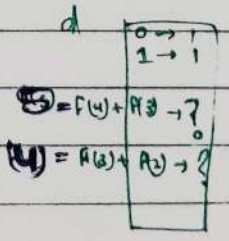
→
```
def Fib (n) :
    if n==1 or n==0 :
        return 1
    else:
        return fib(n-1) + Fib (n-2)
```

But this code is very inefficient, Time to calc Big Fabonacci no° will reach ∞

Instead

→
```
def fib (n,d) :
    if n in d :
        return d[n]

    else :
        d[n] = fib (n-1,d) + fib (n-2,d)
        return d[n]
```



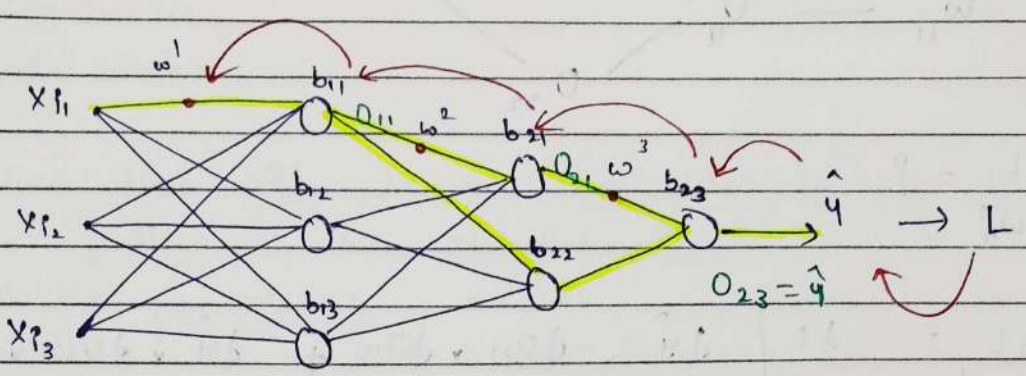
→ d = {0:1, 1:1}

Fib (5, d) — call

PAGE NO. :
DATE : / /

• Small topic of Backpropagation

→ How complecated it can be w^1 in 3 layer in B.P.



chain rule.

$(L \rightarrow \hat{y} \rightarrow O_{21} \rightarrow w_{11}^2)$

↓

All the weights of second layer can be calculated by this,

$$\frac{\partial L}{\partial w_{11}^3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{11}^3}$$

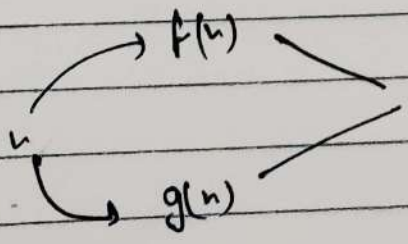
$$\left\{ \frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial w_{11}^2} \right\}$$

→ for 1 st layer.

$$L \rightarrow \hat{y} \rightarrow O_{21} \rightarrow O_{11} \rightarrow w_{11}^1$$

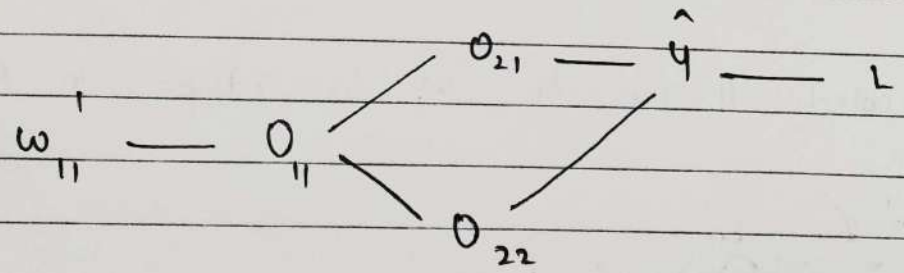
Additional complexity is there due to two path.

→ concept



$$\frac{\partial h}{\partial h} = \frac{\partial h}{\partial f(h)} \cdot \frac{\partial f(h)}{\partial h} + \frac{\partial h}{\partial g(h)} \cdot \frac{\partial g(h)}{\partial h}$$

The path is.



$$\frac{dL}{dw_{11}^{1}} = ?$$

$$\frac{dL}{dw_{11}^{1}} = \frac{dL}{d\hat{y}}\left[\frac{d\hat{y}}{dO_{21}} \times \frac{dO_{21}}{dO_{11}} \times \frac{dO_{11}}{dw_{11}^{1}} + \frac{d\hat{y}}{dO_{22}} \times \frac{dO_{22}}{dO_{11}} \times \frac{dO_{11}}{dw_{11}^{1}}\right]$$

→ expression complicated and complicated.
So, the derivative which are already calculated are stored so that it need not to be calculated again.

Back prop → (chain diff. (rule) + memoization

↑
maths

Keras.

PAGE NO. :
DATE : / /

Types of Gradient Descents.

→ Batch
→ stochastic
→ mini Batch.

→ These three differ in how much data we use to compute the gradient of the objective function
→ Depending on the amount of data, we make a trade-off b/w the accuracy of the parameter update and time it takes to perform an update.

Batch G.D (Vanilla)

```
for i in range(nb_epochs):
    params_grad = evaluate_gradient(loss_func, data, params)
    params = params - learning_rate * params_grad
```

- entire dataset → update.

Stochastic G.D

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for example in data:
        param_grad = evaluate_gradient(loss_func, example, params)
        params = params - learningrate * paramgrad
```

no° of epochs = no° of update.

PAGE NO. :
DATE : / /

note. → In stocastic G.D The freq of weights update is very much Higher.

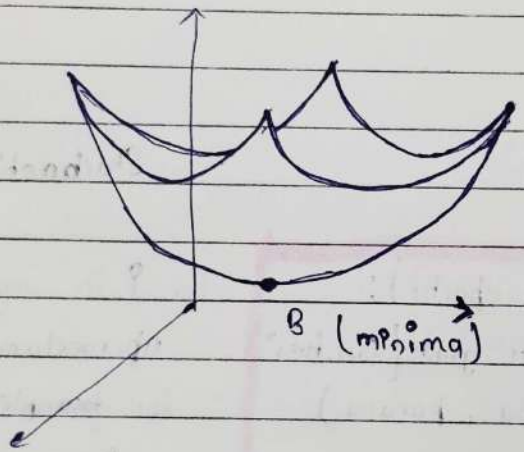
If Batch_size = n , Batch size = 1

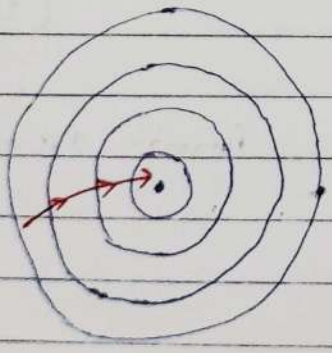
↓ ↓

Batch G.D Stocastic G.D

(faster) (converge faster)

output





Batch G.D

Stocastic G.D.

- This behaviour helps algo to move out of local minima.
- Do donot converges exactly.

Best of Both worlds.

PAGE NO. :
DATE : / /

Mini Batch Gradient Descent :-

- Batch Size => (1 ⟷ n)
  (in between)

(n = no of rows)

BGD ⟷ SGD

320 rows

in every epoche
10 batches

batches (32)

10 updates

fast :-

Batch > mini-Batch > Stocastic

convergence

Batch < mini-Batch > Stocastic

• Vanishing Gradient Problem in ANN. (with code)

Thing to remember

1.) $0.1 \times 0.1 \times 0.1 \times 0.1 \Rightarrow 0.0001$

{ no° smaller than 1 multiplied by each other given : even smaller no° more }

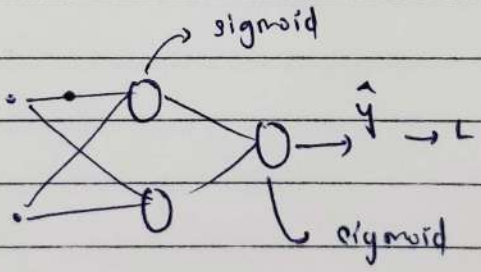
2.) VGP }

only found in Deep N.N. (many layers)

3 for VGP to occur A.Func° should be { sigmoid / tanh }

→ intuition of VGP in Deep ANN.

→ Back prop.



$$w_n = w_o - \eta \frac{dL}{dw}$$

let's say we need to find.

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}}$$

$0 < (\cdot) < 1 \quad 0 < (\cdot) < 1$

let $w_o = 1$

L.R

$w_n = 1 - 0.01 \times 0.0001$

$w_n = 0.99999$.

did not change

PAGE NO.:
DATE: / /

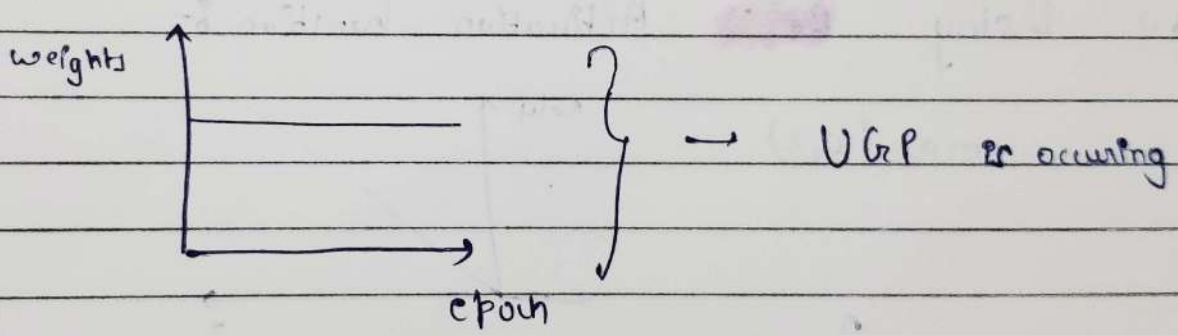
* *

- So in very deep N.N the initial layer weights will not change very much and then the model will not converge

- **How to recognize ?**

If loss after every epoch → no change → VGP

If graph of weights (W_k) epoch.

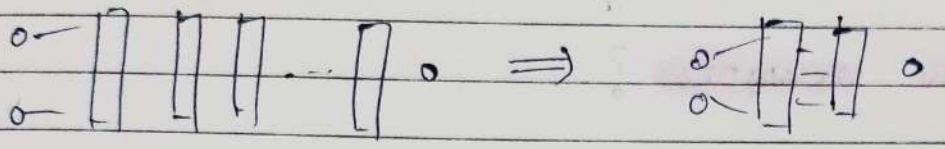


- weight is not changing after every epoch

PAGE NO. :
DATE : / /

→ How to handle vanishing Gradient problem.

1.) Reduce model complexity
(if data do not consist of non linear complexity)

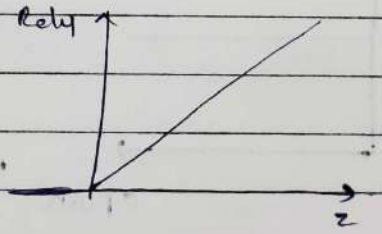


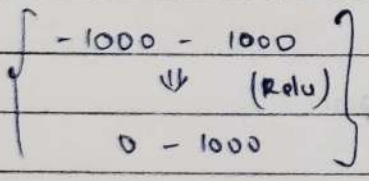
$\frac{\partial L}{\partial w}$ = bigger

(not applicable)

2.) Using ReLU Activation function :-

max (0, z)





→ so derivative in Relu is (0, 1)

→ 1 * 1 * 1 ... = 1 { Vanishing doesn't occur. }

→ if "0" then use leaky relu

PAGE NO. :
DATE : / /

3.) Proper weights init → (Glorot)
→ (Xavier)

(4) Batch normalization => (layer)

(5) Residual network.

PAGE NO. :
DATE : / /

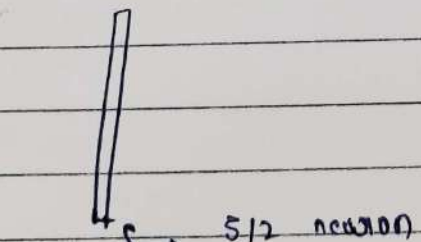
• How to Improve a neural network.

1.) fine tuning of NN Hyperparameters.

- hidden layers
- neurals per layer
- learning rate
- optimizer
- Batch size.
- Activation fun°
- epochs.

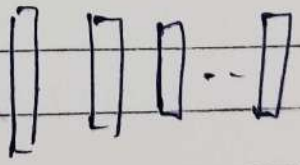
2.) - Vanishing / Exploding gradient
- Not enough data
- Slow training
- overfitting

• no° of Hidden layers



512 neuron

(single) layer



{ Deep network is better than single layer }

PAGE NO.:
DATE: / /

- How much layers ⇒ { until you get to overfitting }
  - Hit / trail of layers.

- 
  → Deep network.
  - First few layer recognize lines.
  - after that few layer reggnize pattern like eyes / nose / ears.
  - The after that layers recognizes full faces..

  → thats why multilaye network is better.

- **no of neurons.**

  → must be sufficient per layer. (not less)

- **Batch size.**

  - 2 approches.

  - **Small Batch size** { (8 to 32) genralize better results. } - slow But give results.

  - **large Batch size** { (approx 8000) } → fast.

To make it give good results, one thing you can do is →

DATE : / /

→ warming up the learning rate.

- initially take small values of (L.R)
- after every epoch increase (L.R)

→ This process will give good result in large batch size.

↳ {Try this technique first}.

- no of epochs.

→ Take as big as you want.

→ But apply early stopping

↓

Keras →

* * [• when results are not getting better or changing then keras stops the traing → applying early stopping]

Keras ⟶ (call back)

PAGE NO.:
DATE: / /

Problem with Neural network.

Vanishing / exploding gradients

- weights init
- Activation function
- Batch normalization (for exploding
- Gradient clipping

Over fitting

- l_1 and l_2 reg
- Dropouts.

Not enough data

- Transfer learning
- unsupervised pretraining

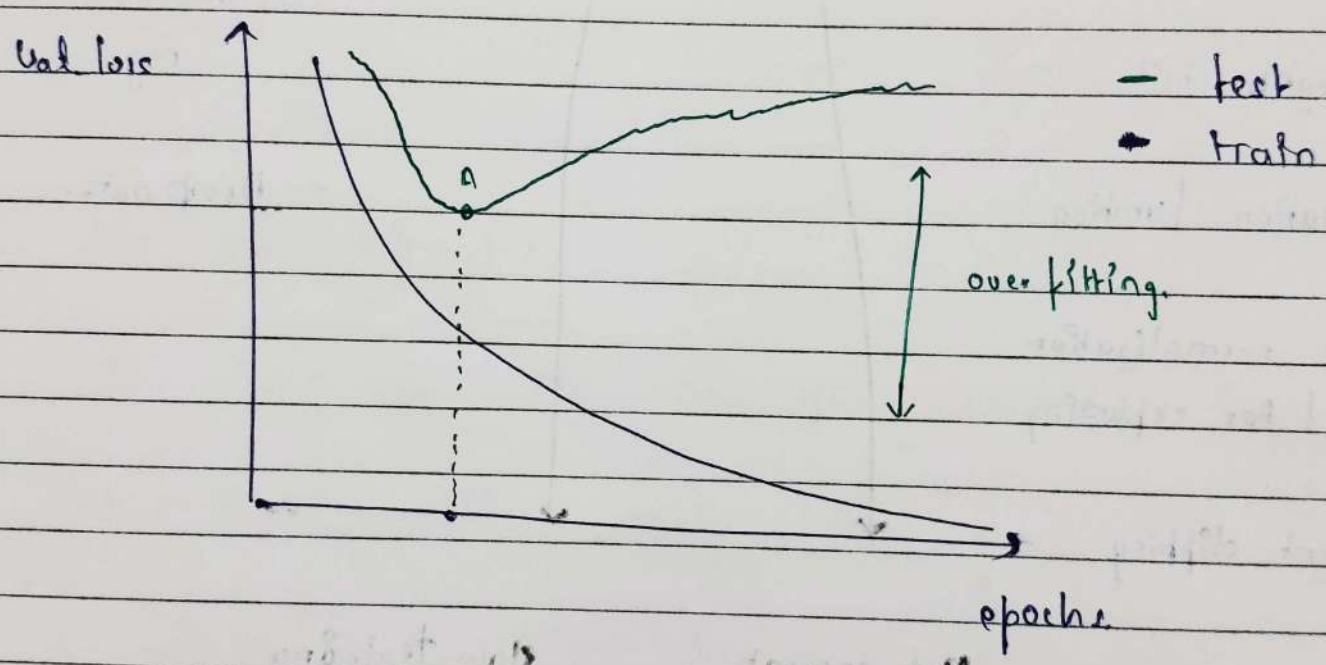
Slow training

- Optimizers
- learning rate schedular

PAGE NO. :
DATE : / /

* Early Stopping :-

- By this concept you can stop the training where overfitting is started.
- By this concept you can first take As big no° as you want



- after this point overfitting starts increasing.
- So the epoch where "A" is occuring the training stops.

⇒ Add in code.

Add in code.

PAGE NO.:
DATE: / /

- callback = EarlyStopping(
    monitor = "val_loss",
    min_delta = 0.0001,
    patience = 20,
    verbose = 1
    mode = "auto"
    baseline = None
    restore_best_weights = False
)

↳ Patience → no° of epochs the algo have to weight after overfitting starts.

Then add.

- model.fit(X_train, y_train, validation_data = (Xtest, y_test), epoch = 3500, call back = callback)

PAGE NO.:
DATE: / /

Improving Neural Network Performance.

1. Overfitting
   - Dropout layers
   - Regularization L_1 & L_2
   - Early stopping ✓
2. Normalization
   - Normalizing inputs ✓
   - Batch Normalizing
   - Normalizing Activation
3. Vanishing Gradients.
   - Activation function
   - weight initialization
4. Gradient checking and clipping
5. Optimizers.

PAGE NO. :
DATE : / /

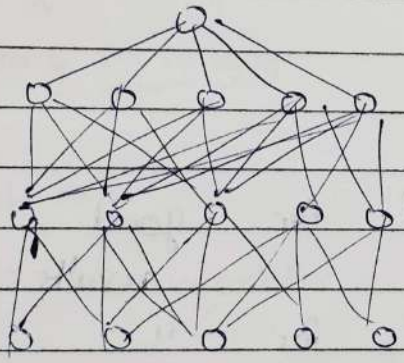
- Drop out layer :-

- N.N is prone to overfit
- To reduce that we have few techniques.
  - Add more data
  - Reduce complexity
  - Early stopping
  - Regularization → L1, L2
  - Dropout layers.

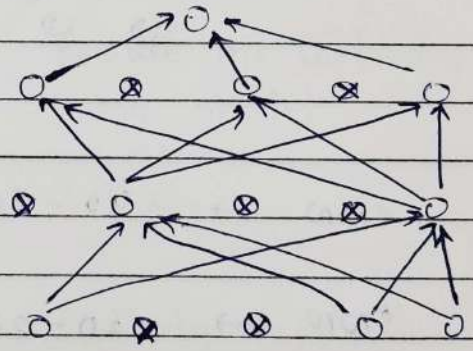
- Dropout layers =>

(P = drop out ratio) Tells How many nodes will be turned off

P = 0.2 => 20%



(N.N)



some of nodes are droped

- in every epoch we randomly remove some nodes in the layers.
- that means in every epoch we are changing architecture of neural network which helps in reducing overfitting

{ P = 0.2 , P = 0.5 , P = 0.75 }

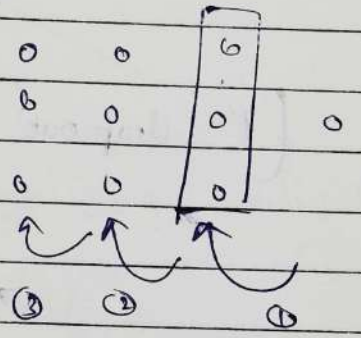
overfitting < ~ underfitting.

PAGE NO. :
DATE : / /

Practical Tips.

1.) if overfitting then ↑P, if underfitting then ↓P

2.) Try applying drop out to the last layer first then go feather to initial layers.



3.) CNN → 40 - 50% ⇒ P for good results

RNN → 20 - 30% ⇒ P for "

"

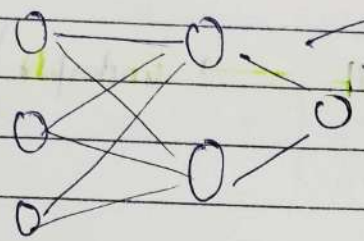
- Draw back.

1. Convergence → delay → Training gets slow
2. loss function value changes

PAGE NO. :
DATE : / /

• Regularization (L_1 & L_2)

- when to use. → when model is overfitting



(ways to solve overfitting)

Adding more data

- Add rows
- Data augumentation

Reduce the complexity of model

- Dropout
- Early stopping.
- Regularization → L_1, L_2, L_1+L_2

→ In ANN → (w | b)

↳ Loss (min)

$$C = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i) + (\text{Penalty term})$$

$$C = L + \frac{\lambda}{2n}\sum_{i=1}^{k} \|w_i\|^2$$

C → Cost, L → Loss

$\frac{\lambda}{2n}\sum_{i=1}^{k} \|w_i\|^2$ → Penalty Term

for $(w_1 \rightarrow w_{10})$ (λ = Regularization factor)

PAGE NO.:
DATE: / /

$$C = L + \frac{\lambda}{2n}\left[w_1^2 + w_2^2 \dots w_{10}^2\right]$$

d = Hyperparameter

→ As $\lambda \uparrow \Rightarrow$ (overfit ⟶ underfit)

$$\Rightarrow \quad C = L + \frac{\lambda}{2n}\sum_{i=1}^{K}|(w_i)|^2 \longrightarrow L_2 \text{ (mostly used)}$$

$$C = L + \frac{\lambda}{2n}\sum_{i=1}^{n}||w_i|| \longrightarrow L_1$$

→ most accurate representation :

$$C = \sum_{i=1}^{m} L(y_i, \hat{y}_i) + \sum_{l=1}^{L}\sum_{i=1}^{n}\sum_{j=1}^{l}||w_{ij}^{l}||^2$$

→ mathematical intuition

$$w_{\eta} = w_0 - \eta\frac{dL}{w_0}$$

L = loss function.

Add Penalty to apply regularization

$$L' = L + \frac{\lambda}{2n}\sum|w_i|^2$$

↓

n=1 for single node

PAGE NO. :
DATE : / /

$$\frac{\partial L'}{\partial w_0} = \frac{\partial L}{\partial w_0} + \frac{\lambda}{2}(2w_0)$$

$$= \frac{\partial L}{\partial w_0} + \lambda w_0$$

$$w_n = w_0 - \eta\left(\frac{\partial L}{\partial w_0} + \lambda w_0\right)$$

$$\boxed{w_n = (1 - \eta\lambda) w_0 - \eta \frac{\partial L}{\partial w_0}}$$

$(1 - \eta\lambda)$ → (weight decay)

$l_2 \rightarrow$ weight decay.

PAGE NO.:
DATE: / /

• Activation function (Brief)

$$a = g(w_1 x_1 + w_2 x_2 \dots + b)$$

g = activation function → sigmoid, relu, tanh.

- If $g(z) = z$ is linear the neuron perform linear regression or classification.
- In $g(z)$ = taken non linear to make non-linear regression or classification

→ Ideal activation func° :-

1) should be non-linear eg: $\left\{\sigma(z) = \frac{1}{1+e^{-z}}\right\}$

2) It should be differentiable.
exception ⇒ relu not differentiab at "0"

3) computationally in expensive.
↳ derivative → simple
→ easy fast.

4) Zero centered → zero centred.
↳ normalized.
eg: tan h
{ Proven that model converges fast when normalized data is given in input

→ if (values) of ~~activation function~~ ⇒ ka $\begin{cases} \text{mean} = 0 \\ \text{?} \end{cases}$ then data is normalized.

PAGE NO.:
DATE: / /

**

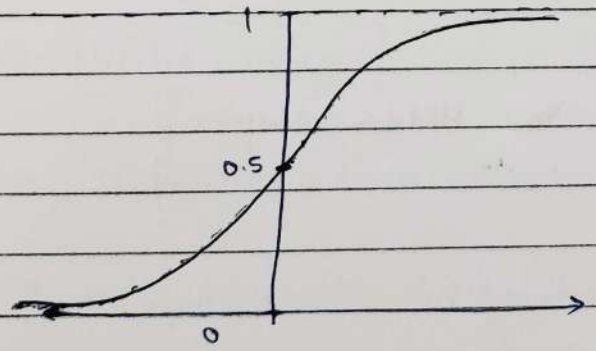
5.) Non - saturating :- (considered) * (good)

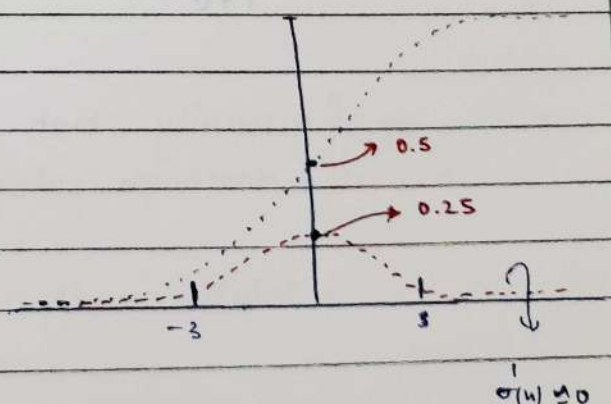
→ saturating function = which bound their output values in a Range.

Saturating	non-saturating
sigmoid → (0,1)	relu = max(0, x)
tanh → (-1,1)	

Problem ⇒ vanishing gradient decents.

(1) Sigmoid Activation function :-





$$\sigma = \frac{1}{1 + e^{-u}}$$

→ Advantages

(1) [0,1] → probability → output layer → Binary - class°

(2) Non - linear → (non-linear data) → good option

(3) Differentiable → Back prop → $\frac{\partial L}{\partial w}$ ✓

PAGE NO.:
DATE: / /

→ Disadvantages.

1) Saturating func^n → input $(-\infty, \infty)$ → output $[0,1]$

↳ vanishing decent problem

2) Non Zero centered

→ output not normalized. either → +ve / -ve

which leads to (Training slow)

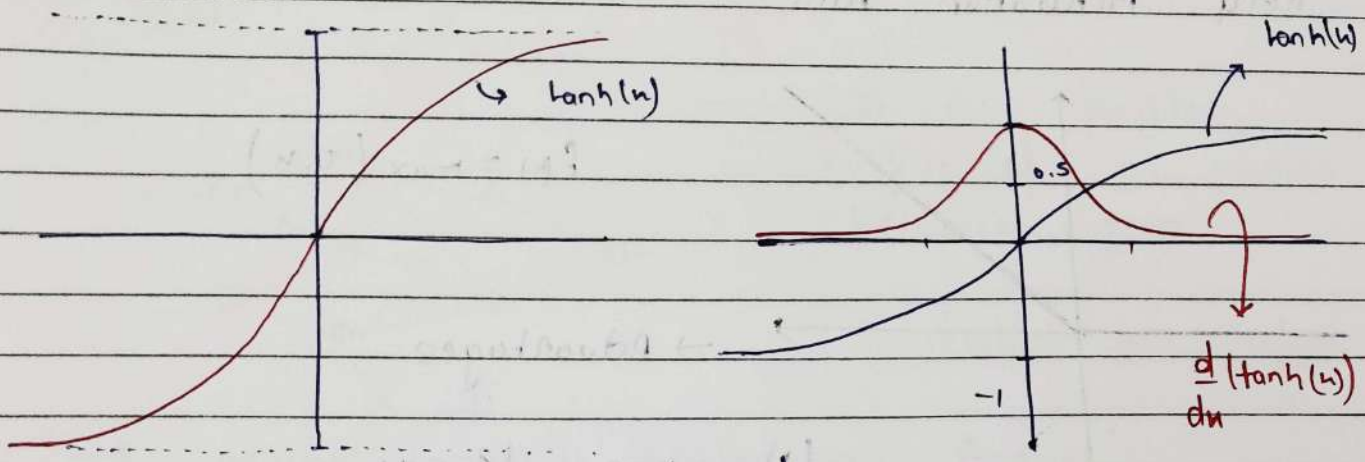
(convergence problem)

3) $\sigma(u) = \frac{1}{1+e^{-u}}$ → (computationally expensive)

{generally not used in Hidden layers now a days.}

PAGE NO.:
DATE: / /

(2) Tanh Acti° function.



$$f(u) = \tanh(u) = \left(\frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}\right) \qquad f'(u) = \left(1 - \tan h^{2}(u)\right)$$

→ **Advantage.**

(1) non linear
(2) differentiable.
(3) zero centred. → training faster than sigmoid.

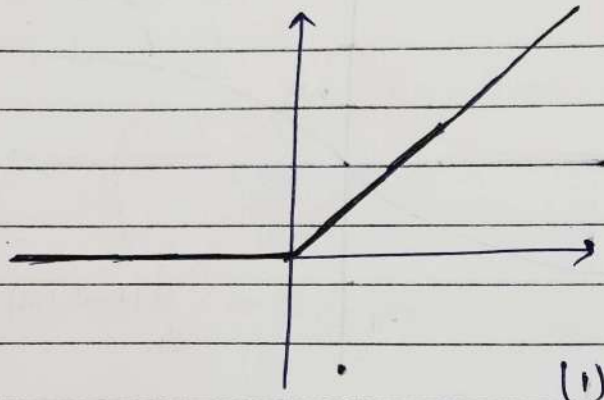
→ **Disadvantages.**

→ saturating function → vanishing gradient problem.

→ computationally expensive → slow training

PAGE NO. :
DATE : / /

- Relu Activation fun^{n}:



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

→ Advantages

(1) non - linear

(2) Not saturated in +ve region

(3) Computationally in-expensive.

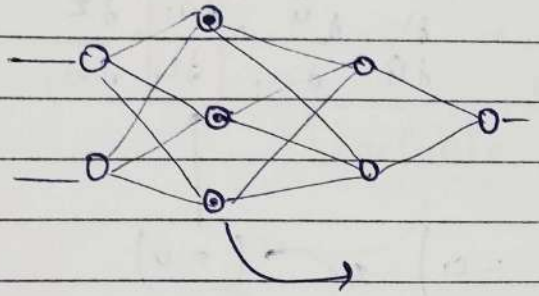
(4) Converge's → faster → sigmoid
→ tanh.

→ Disadvantages

- (1) non differentiable at "0" ⇒ (But in code we take) ⇒ $f'(x) = \begin{cases} 0 & , x < 0 \\ 1 & , x \geq 0 \end{cases}$

(2) Non zero centered →
But this problem is solved
By Batch normalization

Dying Relu Problem



→ In this problem. output of a neuron becomes zero irrespective of input

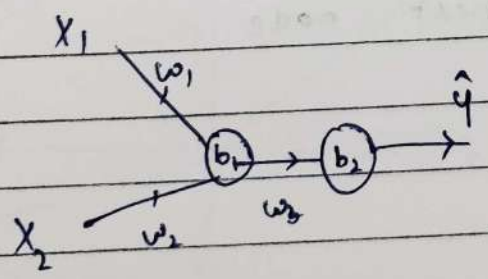
→ neuron's output become zero forever

→ If It becomes case with

$$\left\{ \begin{array}{l} \text{no of neuron} \\ \text{output} = 0 \end{array} \right\} > 50\%$$

Then patterns will not be recognized.

→ Let's see why ?



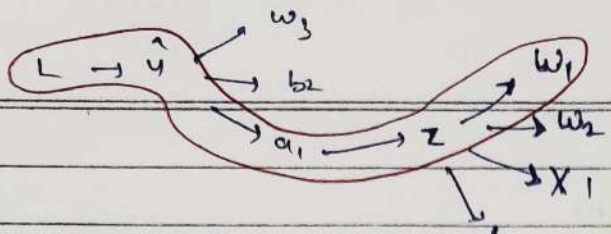
$$a_1 = \max(0, z_1) =$$

if, $z_1 = w_1 x_1 + w_2 x_2 + b_1 < 0$

then $(a_1 = 0) = \frac{da_1}{dz_1} = 1$

PAGE NO. :
DATE : / /

now



$$\left[w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} \right]$$

$$\left[w_2 = w_2 - \eta \frac{\partial L}{\partial w_2} \right]$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_1} \cdot \boxed{\frac{\partial a_1}{\partial z}} \cdot \frac{\partial z}{\partial w_1}$$

$$\left(= 0 \right)$$

so $\left(\frac{\partial L}{\partial w_1} = 0 \right)$

similarly $\left(\frac{\partial L}{\partial w_2} = 0 \right)$

→ so weights will not get updated.

→ reasons why

$$z = w_1 X_1 + w_2 X_2 + b < 0$$

(1) learning rate is very High (η)

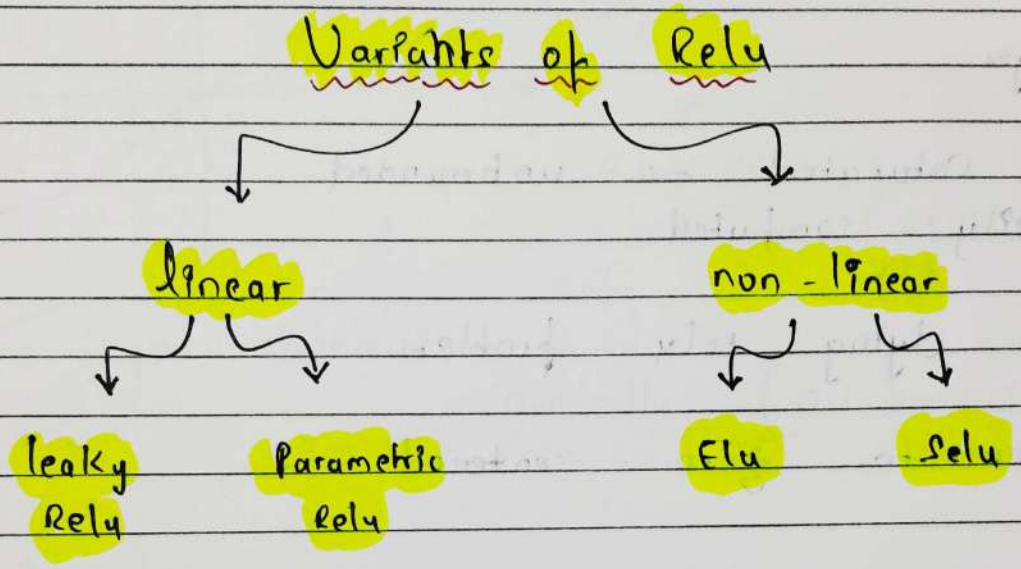
then $z < 0$ in next node can be

(2) High $(-)$ve bais.

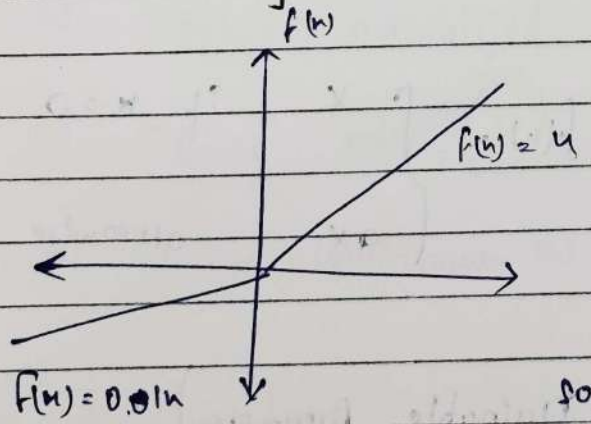
PAGE NO. :
DATE : / /

- Solution of Dying Relu.

→ Set low learning rate
→ bais → +ve value → (0.01)
→ Don't we relu → use variants.



① Leaky Relu



$$f(z) = \max((0.01)z, z)$$

$$f(z) = \begin{cases} z & z \geq 0 \\ \frac{z}{100} & z < 0 \end{cases}$$

$$f'(z) = \begin{cases} 1 & z \geq 0 \\ 0.01 & z < 0 \end{cases}$$

does not become zero.

so $w_n = w_0 - \eta\left(\frac{dL}{dw_0}\right)$

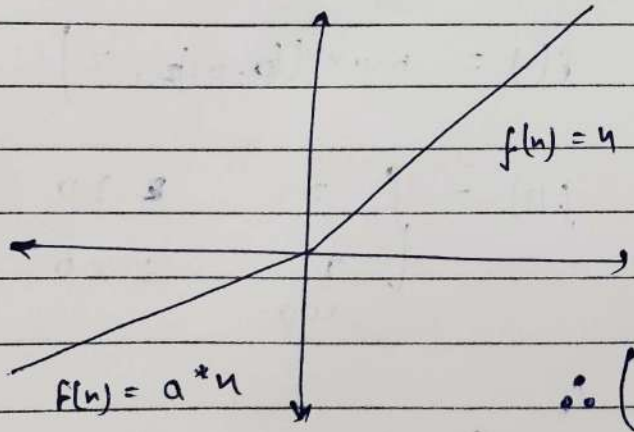
PAGE NO.:
DATE: / /

will not get zero
will always get small changes.
, so Dying relu will not occur.

- Adguantages

→ non Saturated. → unbounded
→ easily computed
→ No dying relu problem
→ close to zero centered.

(2) Parametric Relu



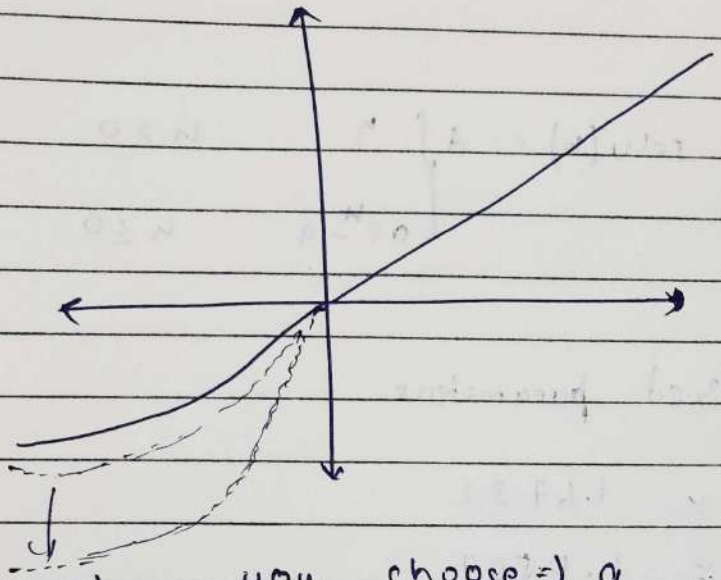
$$f(u) = \begin{cases} x & \text{if } u > 0 \\ ax & \text{otherwise} \end{cases}$$

∴ (a → trainable Parameter)

Best value gets while training

PAGE NO. :
DATE : / /

Elu - Exponential linear Unit.



$$ELU(x) = \begin{cases} x & x > 0 \\ a(e^x - 1) & x \le 0 \end{cases}$$

$$ELU(x) = \begin{cases} 1 & x > 0 \\ ELU(x) + a & x \le 0 \end{cases}$$

(a↑) you choose ⇒ a generally (0.1 to 0.3)

- Advantages

(1) close to zero centered → converges faster.

(2) Better generalized

(3) Dying relu (X)

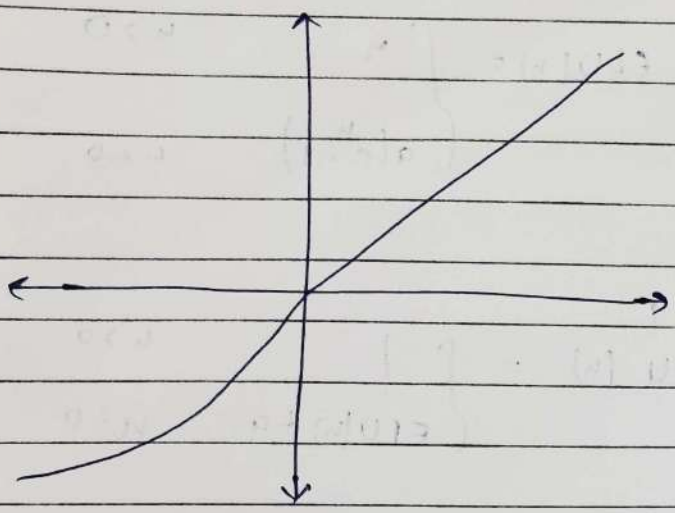
(4) always continuous as well as differentiable.

- disadvantage

(1) computationly expensive.

PAGE NO. :
DATE : / /

(4) Selu - Scaled Exponential Linear Unit



$$selu(x) = d \begin{cases} x & x > 0 \\ ae^x - a & x \leq 0 \end{cases}$$

Fixed parameters

$a \simeq 1.6732$

$d \simeq 1.0507$

$$selu'(x) = d \begin{cases} 1 & x > 0 \\ ae^x & x < 0 \end{cases}$$

Advantage.

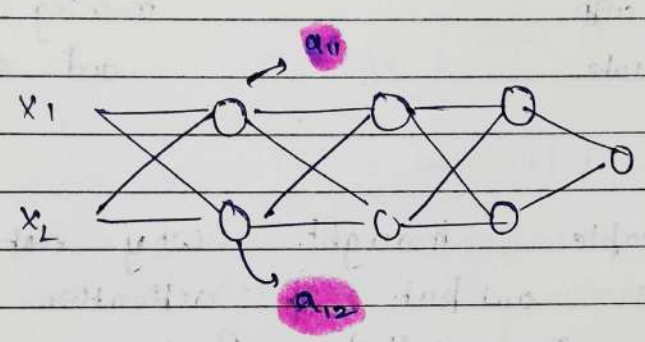
- self normalizing → activation output → normalized (mean = 0)
- converges fast

PAGE NO. :
DATE : / /

- Batch normalization.

BN makes training of (DNN) faster and stable

- It consists of normalizing activation vectors from hidden layers using the mean and variance of the current
- This ~~activation~~ normalization is done right before (or right after) the non linear function



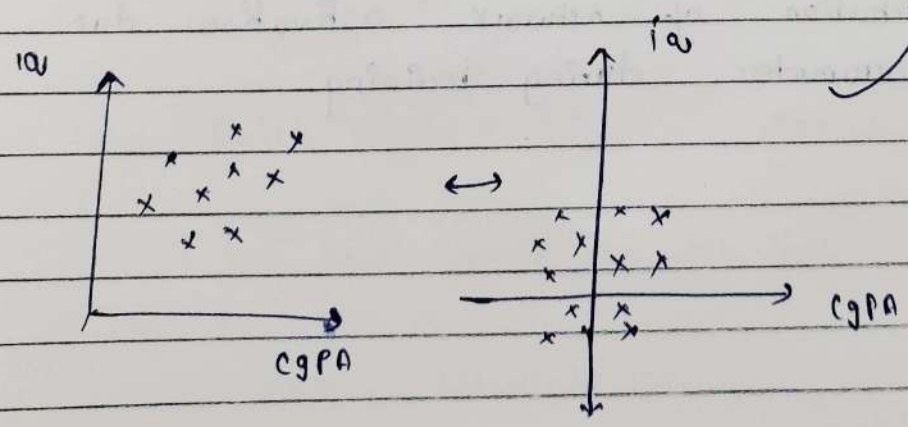
{ activation outputs are also normalized.

mean = 0
Sd. = 1

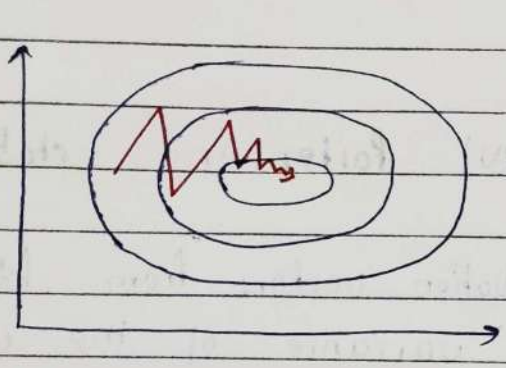
- Why Nom Batch ?

CgPa	Pa	Placed.
7	70	1
8	80	0
9	90	1

$\mu = 0$
$\sigma = 1$

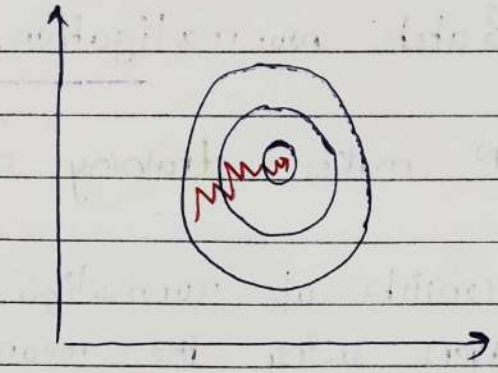


(Before Normalization) (after normalization)



(Before normalization)

↳ cannot work with high learning rate



(with normalization)

↳ training is faster and stable.

→ Then people thought why not to normalize output of activation funits which then is called Batch normalization.

another reason

- Internal covariate shift

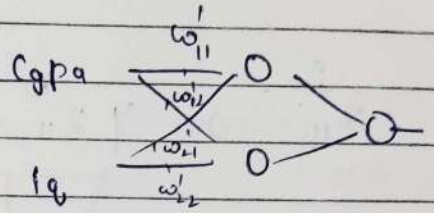
→ we define internal covariate shift as the change in the distribution of network activations due to network parameter during training

PAGE NO.:
DATE: / /

- The How ?
  - Mini- Batch Gradient Desent.
  - layer by layer applied.

Cgpa	ia	Placed
8.9	100	1
6.2	89	0
7.7	91	0
9.1	76	1
⋮	⋮	⋮

normalize $\begin{pmatrix} \mu = 0 \\ \sigma = 1 \end{pmatrix}$



$$z_{11} = w_1(\text{cgpa}) + w_2(\text{ia}) + b$$

→ step by step calculation.

activation func°

(1) $z_{11} \longrightarrow (z_{11}) \xrightarrow{\text{normalized}} g(z_{11}^N) \longrightarrow a_{11}$

other way

(2) $z_{11} \longrightarrow g(z_{11}) = a_{11} \longrightarrow (a_{11})^{\text{normalized}}$

$$\therefore \left(\frac{z_{11} - \mu}{\sigma} = z_{11}^N \right)$$

→ take batch size = (4) assume.

- for each node we will get 4 (z_{11}) value corresponding to each of row

Then calculate $(m = 4)$ → Batch size

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} z_{11}^i \qquad \sigma_B = \sqrt{\frac{1}{m} \sum_{i}^{m} (z_{11}^i - \mu_B)}$$

PAGE NO. :
DATE : / /

then

Calculate $\hat{z}_{ij} = \frac{z_{ij} - \mu_B}{\sigma_B + \epsilon}$ → error term

(normalized)

- step (2)

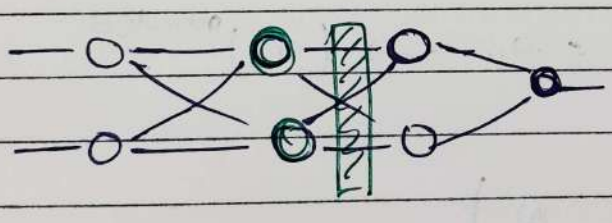
$\hat{z}_{ij} \Rightarrow y \hat{z}_{ij}^{N} + \beta$

learnable parameters

keras initial values $(y = 1)$ $(\beta = 0)$

$z_{ij} \longrightarrow z_{ij}^{N} \longrightarrow z_{ij}^{BN} \longrightarrow g(z_{ij}^{BN}) = a_{ij}$

→ now in BNN During traing.



$[y = y - \eta \frac{dL}{dy}]$

gamma will also be updated during training by G.D

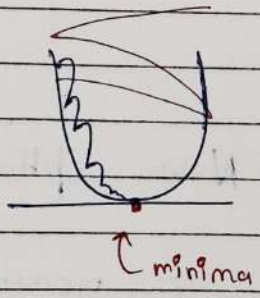
PAGE NO. :
DATE : / /

- Optimizers

- Batch G.D
- Stochastic G.D
- Mini batch G.D

→ challenges with G.D

(1) learning rate → $w_1 = w_0 - \eta \frac{dL}{dw_0}$

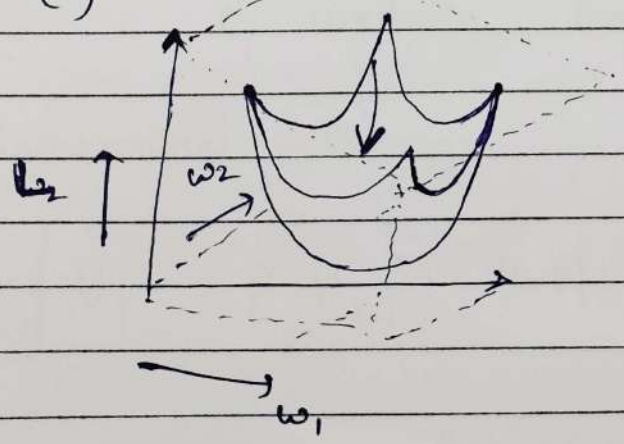


{ It is crucial to decide right learning rate }

- Solution.

- learning rate scheduling or → problem (pre-defined)

(2) (η) is same for both directions ie



3D →
9 weights and Bias.
So then
(10 - Dimensional)

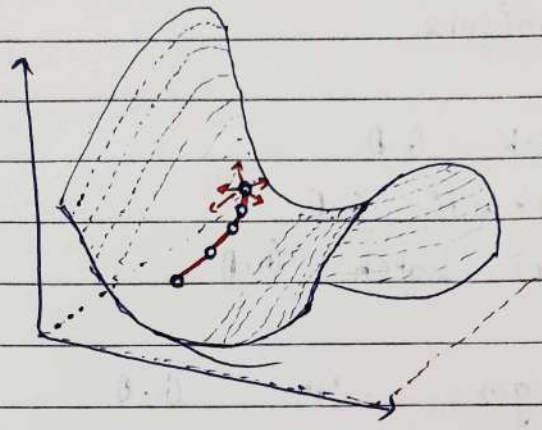
(3) local minima (Problem)

PAGE NO. :
DATE : / /

5) Saddle point

$$\frac{\partial L}{\partial w} = 0$$

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

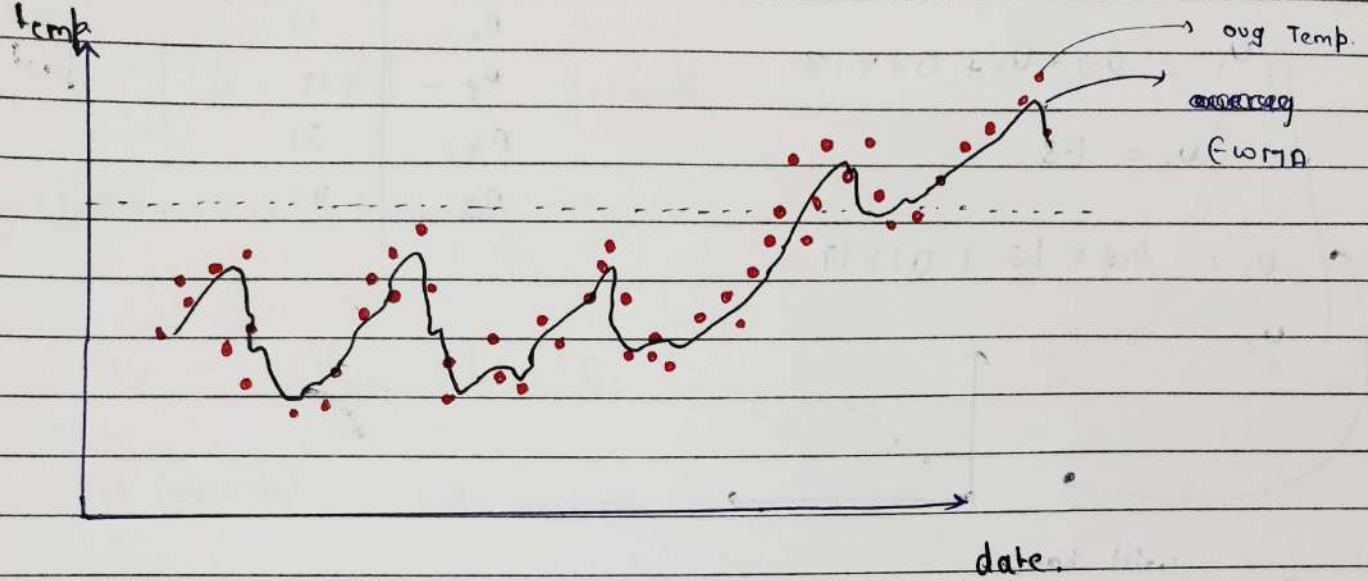


- New optimizers.

1) momentum
2) Ada grad
3) NAG
4) RMS Prop
5) Adam (Popular)

PAGE NO.:
DATE: / /

- Exponentially weighted moving average :- (EWMA)



→ used where ?
- time series.
- financial forcasting
- signal processing
- Deep learning.

→ EWMA is used in time series pattern recognation.

- Mathematical Average.

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

$\beta \rightarrow$ constant.

$(V_0 = \theta_0)$ generally $(\beta = 0.9)$

index	temp(θ)
D_1	25
D_2	13
D_3	17
D_4	31
D_5	43

PAGE NO. :
DATE : / /

$\beta = 0.9$

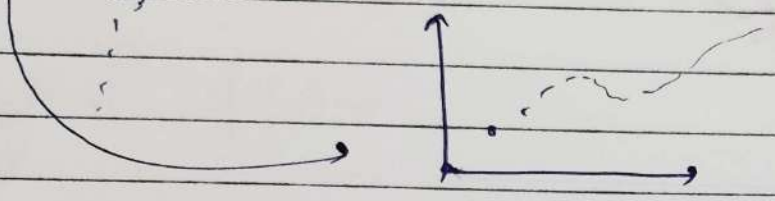
$v_1 = 0.9 \times v_0 + 0.1 \times 13$

$v_1 = 1.3$

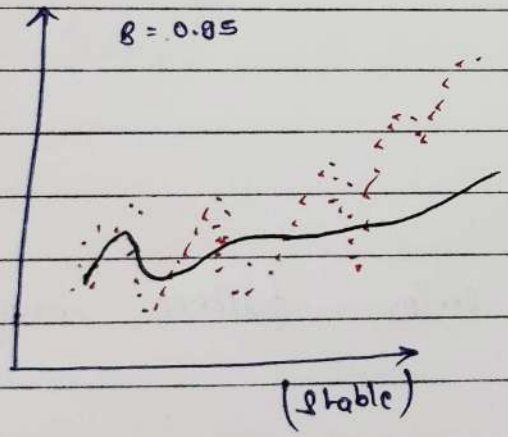
$v_2 = 0.9 \times 1.3 + 0.1 \times 17$

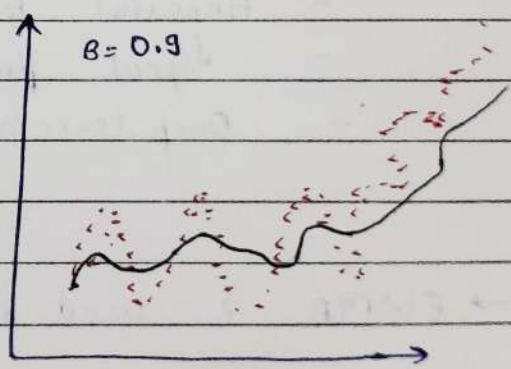
v_3

	Temp.(θ)
D_1	26
D_2	13
D_3	17
D_4	31
D_5	43

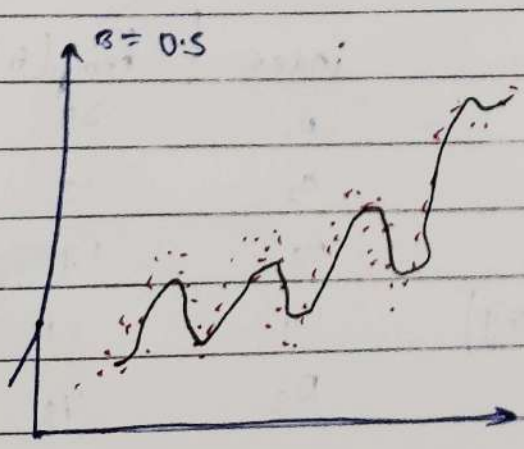


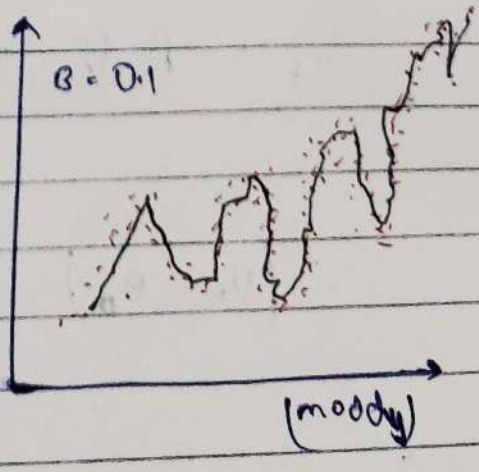
will be formed by these values.





Same data





PAGE NO. :
DATE : / /

→ As (B) is increased value of (EWMA) dependends on more previous data.

$(\beta = 0.9)$ → generally.

- Mathematical intuition →

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

let $(V_0 = 0)$ not necessary

$$V_1 = (1-\beta)\theta_1$$

$$V_2 = \beta V_1 + (1-\beta)\theta_2$$

$$= \beta(1-\beta)\theta_1 + (1-\beta)\theta_2$$

$$V_3 = \beta_2 V_2 + (1-\beta)\theta_3$$

$$= \beta^2(1-\beta)\theta_1 + \beta(1-\beta)\theta_2 + (1-\beta)\theta_3$$

$$V_4 = \beta V_3 + (1-\beta)\theta_4$$

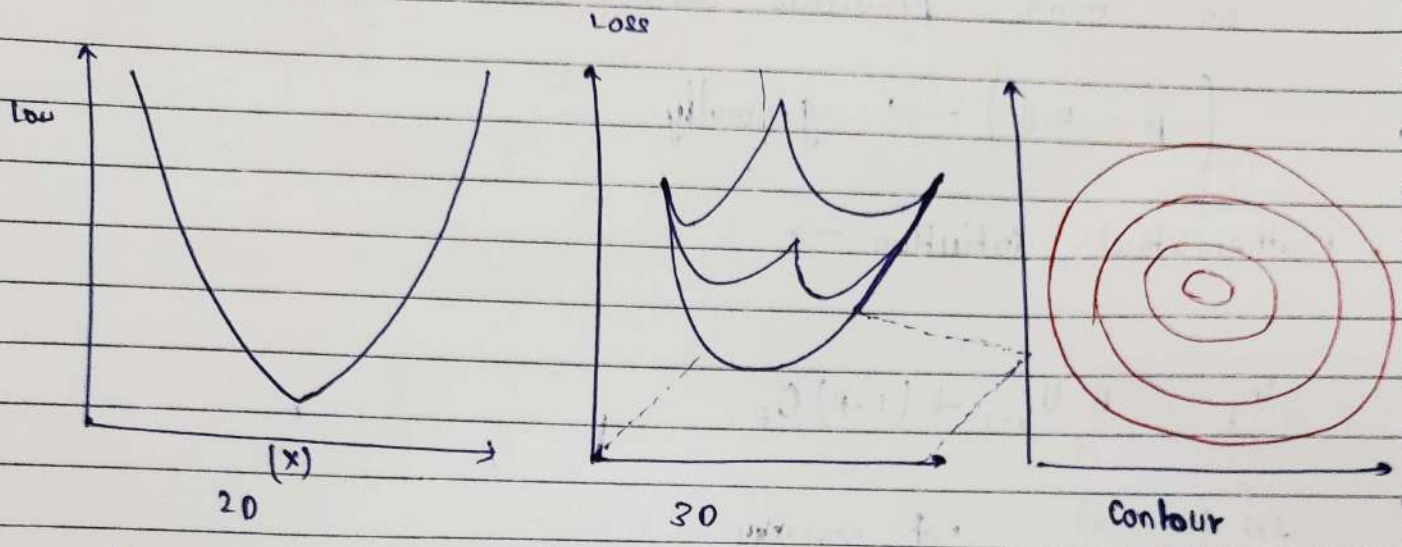
$$V_4 = (1-\beta)(\boxed{\beta^3\theta_1} + \beta^2\theta_2 + \beta\theta_3 + \theta_4)$$

$(\beta^3 < \beta^2 < \beta)$

- smaller values are multiplied by older values.

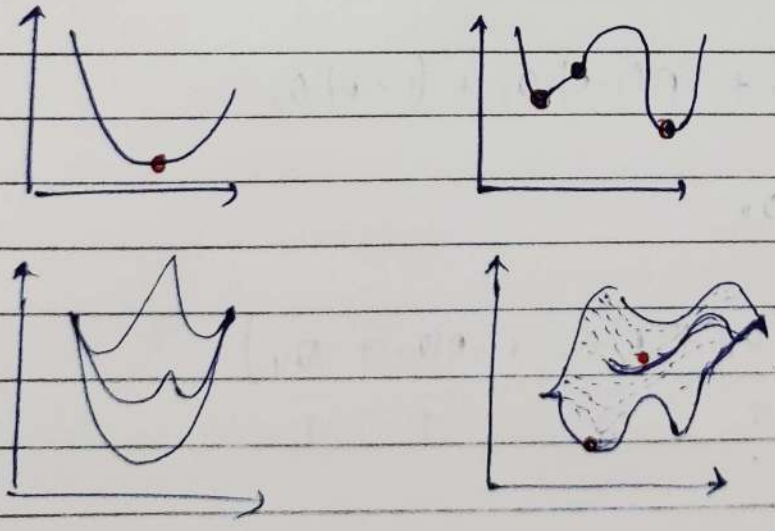
PAGE NO. :
DATE : / /

- SGD with momentum.



[momentum] [Visualization of loss]

- convex (Vs) non convex



SGD cannot navigate all these three.

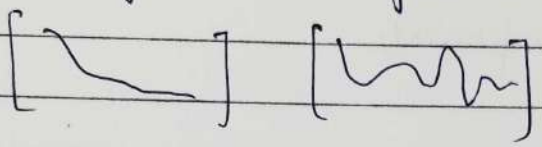
PAGE NO.:
DATE: / /

Momentum optimization → why?

- momentum is good at navigating.

non - convex operation.

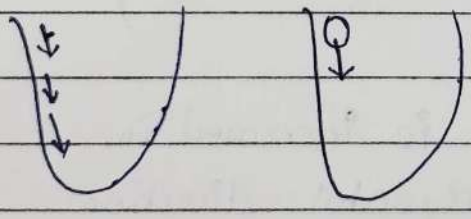
high curvature | consistant grad | Noisy (local minima) grad



→ momentum helps in solving all these three issues.

→ (Intuition)

If previous gradients as saying you to go in a direction then you will increase speed in that direction



mu → momentum unit mass

(v → speed)

→ (Mathematics)

→

$$w_{t+1} = w_t - \eta \Delta w_t$$

$$w_{new} = w_0 - \eta \frac{\partial L}{\partial w}$$

Same

PAGE NO. :
DATE : / /

$$w_{t+1} = w_t - \eta \Delta w_t$$

in momentum.

$$w_{t+1} = w_t - v_t$$

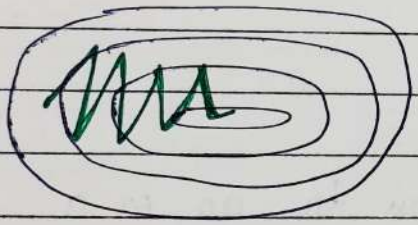
$$w_{t+1} = w_t - v_t$$

$$\therefore \quad v_t = \boxed{\beta \cdot v_{t-1}} + \eta(\nabla w_t)$$

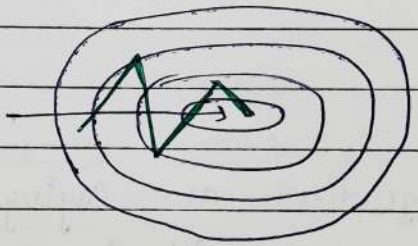
momentum component.

$\beta = 0.9$ [generally.]

(difference.)



(SGD)



(SGD with momentum)

(speed is increased in horizontal direction for faster convergence)

- Effect of (β)

$$w_{t+1} = w_t - v_t$$

$$v_t = \beta \cdot v_{t-1} + \eta \nabla w_t$$

PAGE NO. :
DATE : / /

- Advantage of momentum.

→ SGD with momentum converges faster than normal SGD
→ has the ability to come out of local minima

- disadvantages

→ due to having momentum it ~~surpasses~~ passes the Global minima, oscillates and then converges to minima (Global)

PAGE NO. :
DATE : / /

- NAG – Nesterov Accelerated Gradient

- some disadvantages in momentum were tried to solve in NAG.
(momentum problem)

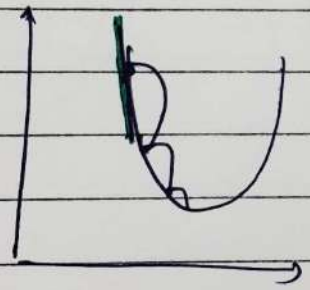
→ (Momentum.)

$$w_{t+1} = w_t - v_t$$

where

$$\left[v_t = \beta \cdot v_{t-1} + \eta \nabla w_t \right]$$

$$w_{t+1} = w_t - (\beta v_{t-1} + \eta \nabla w_t)$$



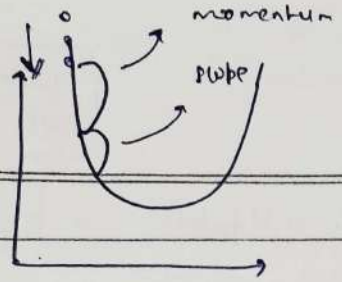
Jumps will be decided by these two components.

(1) past velocity
(2) Gradient of that pt

→ (NAG)

In this we first calculate momentum then take first Jump
then at that point we calculate slope then take second Jump accor° to the slope.

NAG



look ahead

PAGE NO. :
DATE : / /

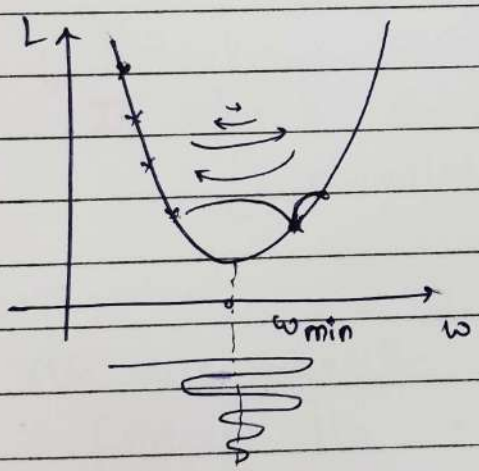
$$w_{la} = w_t - \beta v_{t-1} \longrightarrow \text{first update due to previous velocity}$$

$$v_t = \beta v_{t-1} + \eta \nabla w_{la}$$

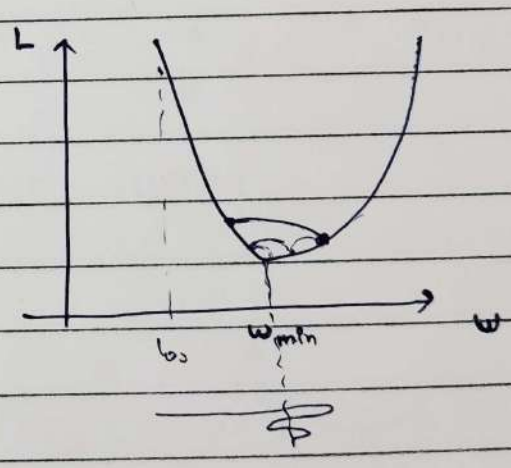
$$\Rightarrow v_t = (w_t - w_{la}) + \eta \nabla w_{la}$$

$$w_{t+1} = w_t - v_b$$

→ Geometric Intuition :-



(momentum)



(NAG)

- disadvantage.

– can stuck in local minima

PAGE NO. :
DATE : / /

Keras code

```
tf.keras.optimizers.SGD(
    learning_rate = 0.01, momentum = 0.0, nesterov = False
    , name = "SGD", ** kwargs.
)
```

SGD	momentum	NAG
	momentum = 0.9	momentum = 0.9
	nesterov = False	nesterov = True.

PAGE NO.
DATE : / /

AdaGrad - Adaptive Gradient.

→ learning rate is not fixed
∴ (adapted as per situation)

→ where Ada grad performs good.

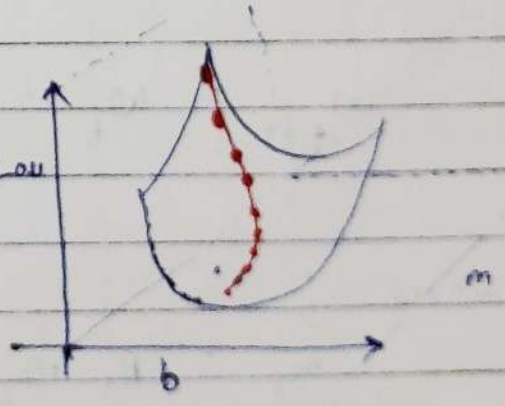
(1) features are (sparsh) → mostly ('0')
then [adagrad] is usefull

eg. Data set =>

iq	cgpa	from iit	pakage
:	:	0	
:	:	:	
:	:	0	→ 0.11 lpa
		0	

In sparsh Dataset
[elongated bowl problem occurs]

→ dis advantage :-
[misses the minima]



PAGE NO.:
DATE: / /

Adam → Adaptive moment Estimation

↓

(most popular) → ANN, KNN, RNN

- SGD | BGD | MB GD
- momentum
- NAG

→ (momentum) [works on]

- Ada grad.
- RMS prop.

→ (learning rate decay) [works on]

Adam → (momentum + learning rate decay)

- Mathematical formulation :-

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{V_t + \varepsilon}} * m_t$$

Where

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla w_t \rightarrow \text{momentum}$$

$$V_t = \beta_2 V_{t-1} + (1-\beta_2) (\nabla w_t)^2 \rightarrow \text{Adagrad}$$

Bais correction

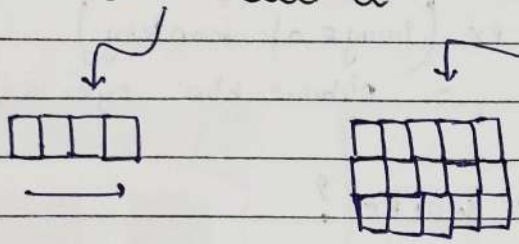
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{V}_t = \frac{V_t}{1-\beta_2^t}$$

$\beta_1 = 0.9$, $\beta_2 = 0.999$ → changable

PAGE NO. :
DATE : / /

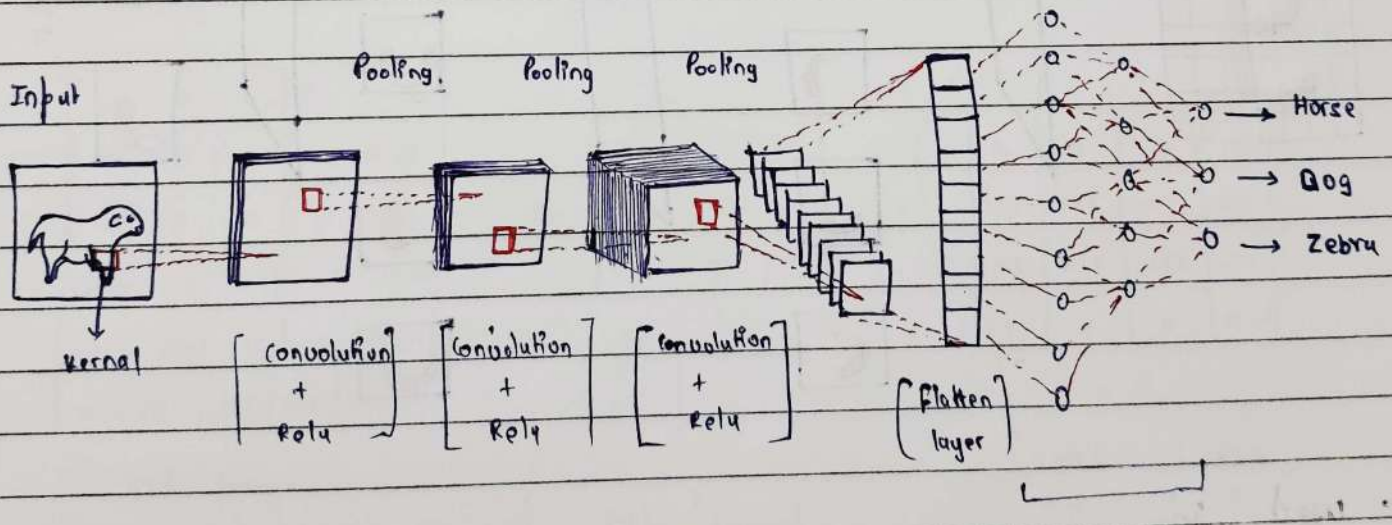
Convolutional neural network.

- CNN are a special kind of neural network for processing data that has a known grid-like topology like time series data (1D) or images (2D)



ANN ⟶ metrix multiplication

CNN ⟶ Convolutional operation.



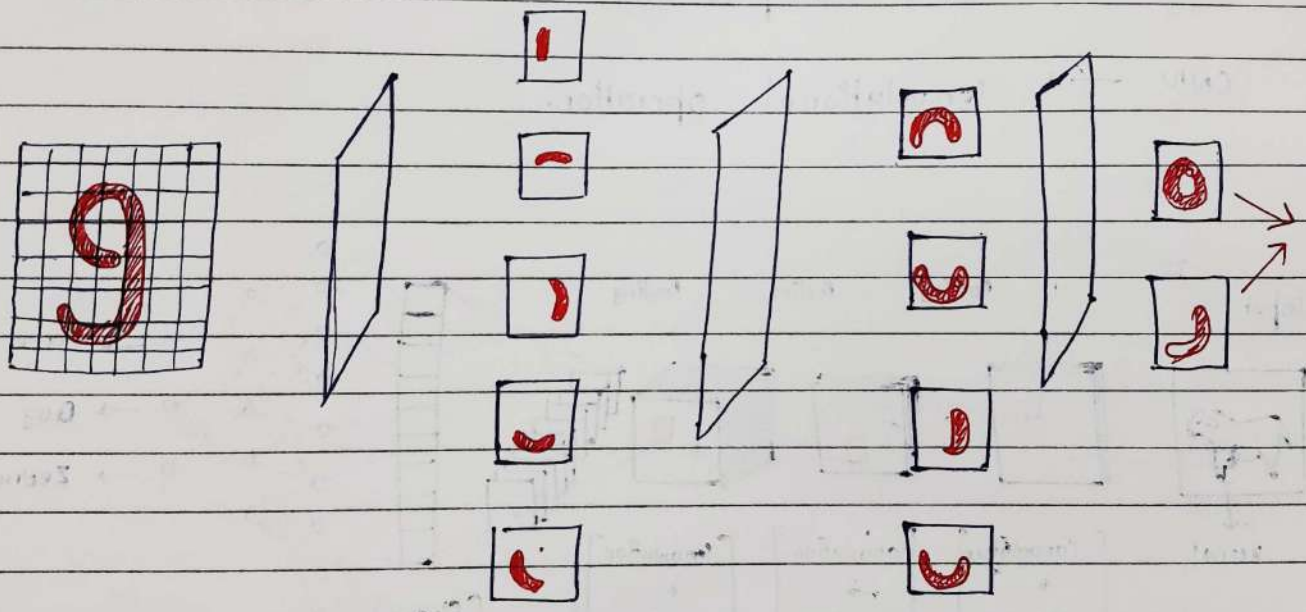
- Components
  - Convolution
  - Pooling
  - FC layer - ANN

PAGE NO.:
DATE: / /

- why not ANN ?

1.) High computational cost
2.) Over fitting
3.) loss of imp info like spatial arrangement of pixels.
ex (Image of monkey)
- Distance blw eyes and nose

- How convolution neural network works ?



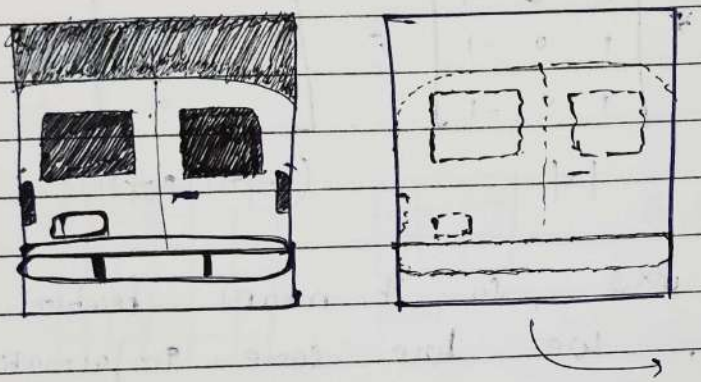
- used in
(1) Image classification (8)
(2) image localization
(3) object detection
(4) face detection and recognition
(5) Image segmentation
(6) Image resolution
(7) Image colourization.

PAGE NO. :
DATE : / /

- Convolutional operation

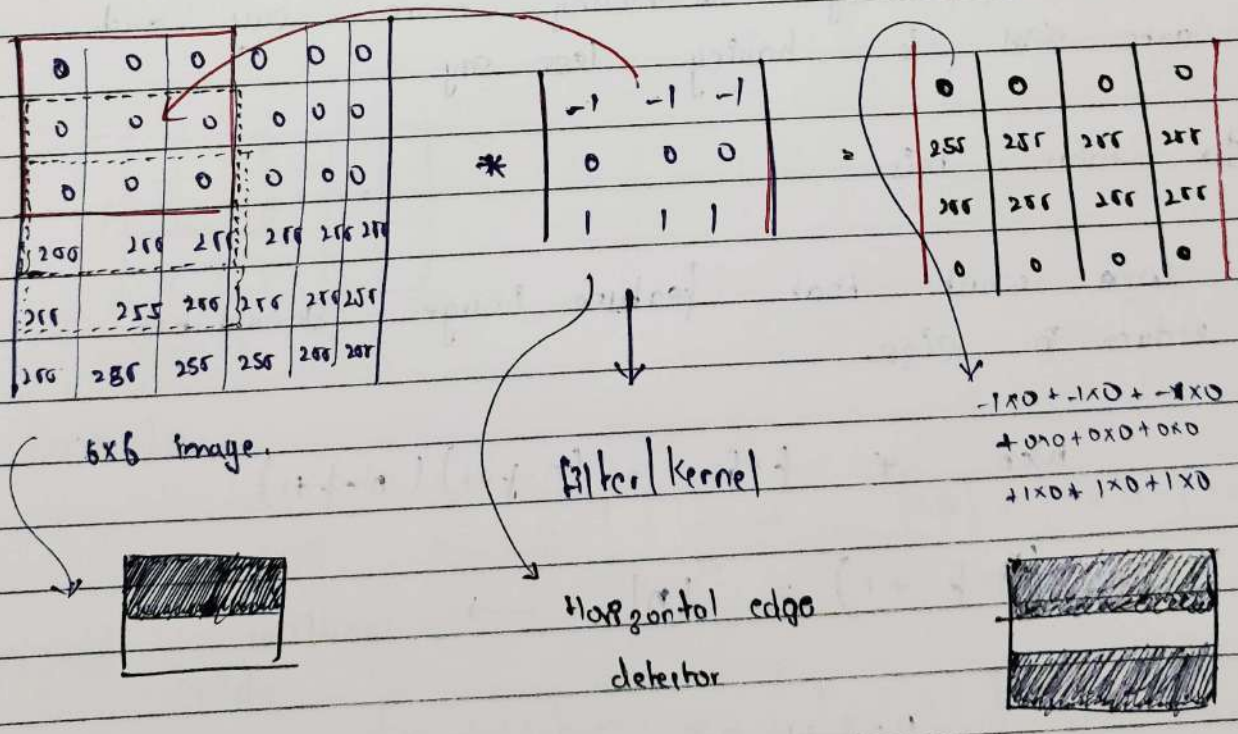
Two Types of Images → Black & White
→ RGB

- Edge detection (Convolution operation)



Vertical and Horizontal edges detection

- How algorithm Detects edges.



PAGE NO.:
DATE: / /

- Padding

- Problem with convolution :

$$\underset{n\times n}{\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 5 & 3 & 8 & 4 \\ 3 & 3 & 2 & 8 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix}} * \underset{f\times f}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{(n-f+1)(n-f+1)}{\begin{bmatrix} \quad \end{bmatrix}}$$

(1) Feature map is of 3×3, so it means feature map is reduced, we lose some information.

(2) when convolution is taking place, the inner value of images is having more say and outer is having less say.

→ to solve this

We want that feature image should not reduce in size.

$$n\times n \quad * \quad f\times f \quad = (n-f+1)(n-f+1)$$

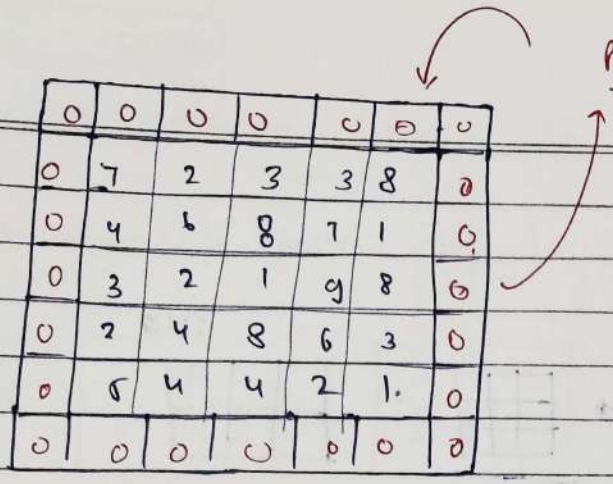
$$(n-f+1) = (n) \longrightarrow \text{condition}$$

$$n-3+1 = 5$$

$$(n=7)$$

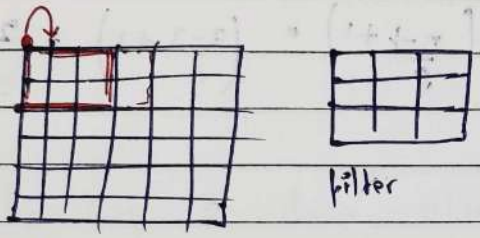
we need to convert input image into (7×7) image.

PAGE NO.:
DATE: / /

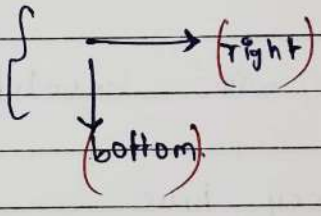


if all values are "0" in padding then called zero padding.

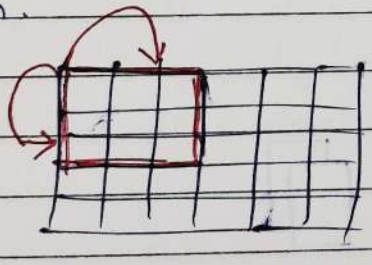
- stride :-



stride = no of block moved by the filter towards right & bottom



→ if stride = (2,2) then.



formula.

$$n \times n \quad * \quad f \times f \longrightarrow (n-f+1) \times (n-f+1)$$

applied stride

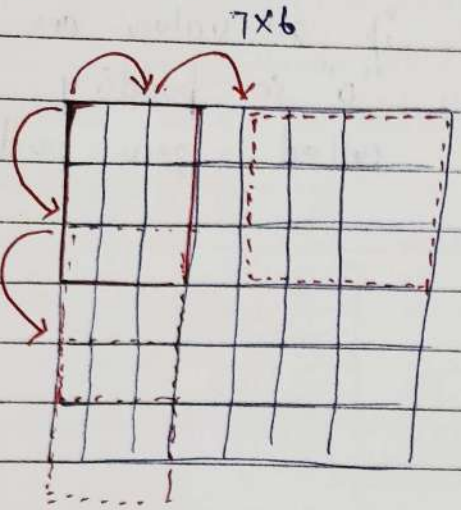
$$(n-f+1) \longrightarrow \left[\frac{n-f}{s}+1\right]$$

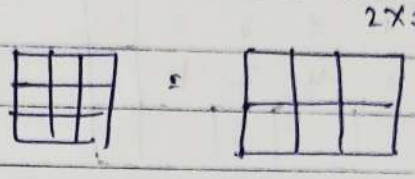
applied padding.

also $$(n-f+1] \longrightarrow \left[\frac{n-f+2p}{s}+1\right]$$

PAGE NO. :
DATE : 4 / .

· Special case :-





$\left(\frac{n-f+1}{s}\right) = \left(\frac{6-3}{2}+1\right) = 1.5+1$

$= 1+1 = 2$

$\left(\frac{n-f+1}{s}\right) = \left(\frac{7-3}{2}+1\right) = 2+1$

$= 3$

Pooling :-

→ Problem with convolution

(1) memory issue.

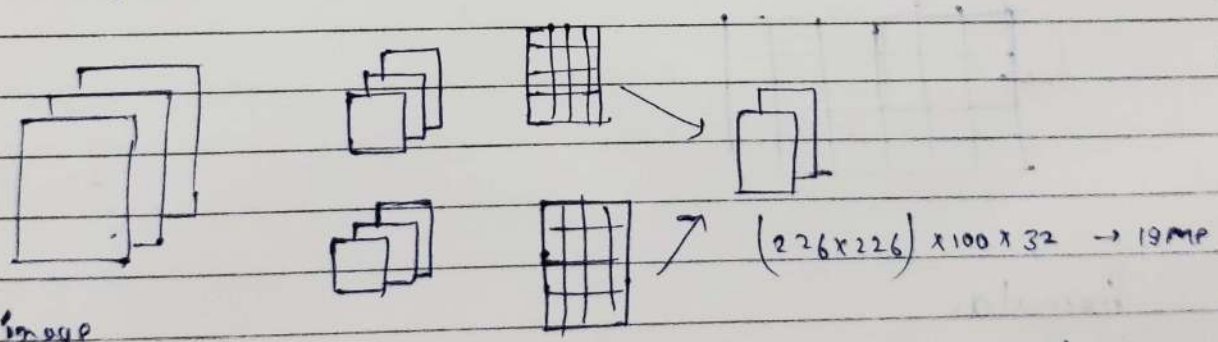


Image (228 × 228 × 3)

filter (100)

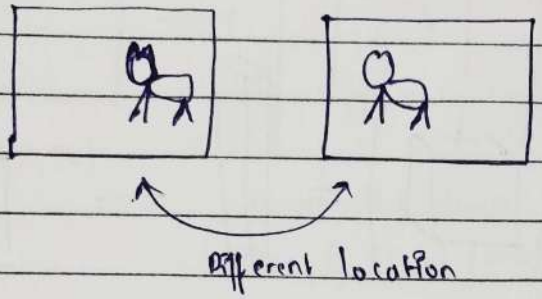
feature map

and whole batch will be 15 GB

· So we have to reduce the size of feature map

→ Translation variance. - (problem with convolution)

→ Convolution is location dependent.



Pooling
↓
Down scale your feature map
↓
which makes it independent to location

→ Pooling is done after convolution.

↳ Types (1) max pooling
(2) min "
(3) Avg "
(4) L_2 Pooling
(5) global Pooling

→ Calculation image size after pooling

$(H \times W) \rightarrow (k \times k) \longrightarrow \left(\frac{H-k+2P}{s}+1\right) \times \left(\frac{W-k+2P}{s}+1\right)$

Image — Pooling

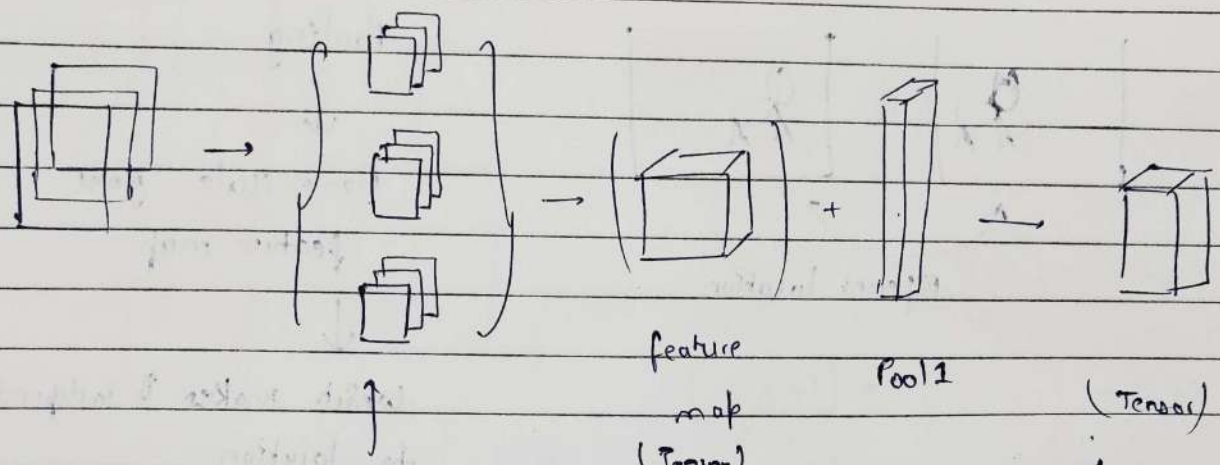
$k = 2$ generally

s = stride

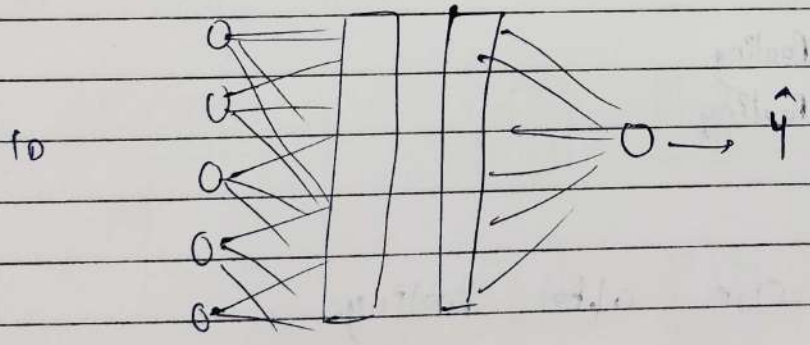
Padding $= 0 = P$

CNN Architecture

1.> Convolution 2.> Padding | stride 3.> Pooling.



Then tensor is flattened

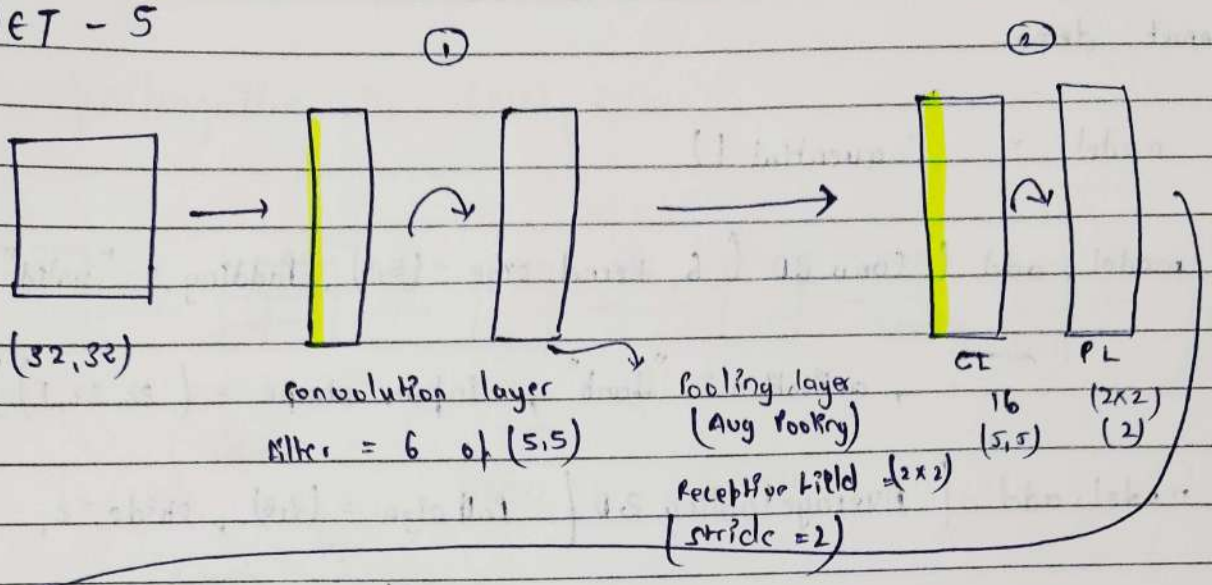


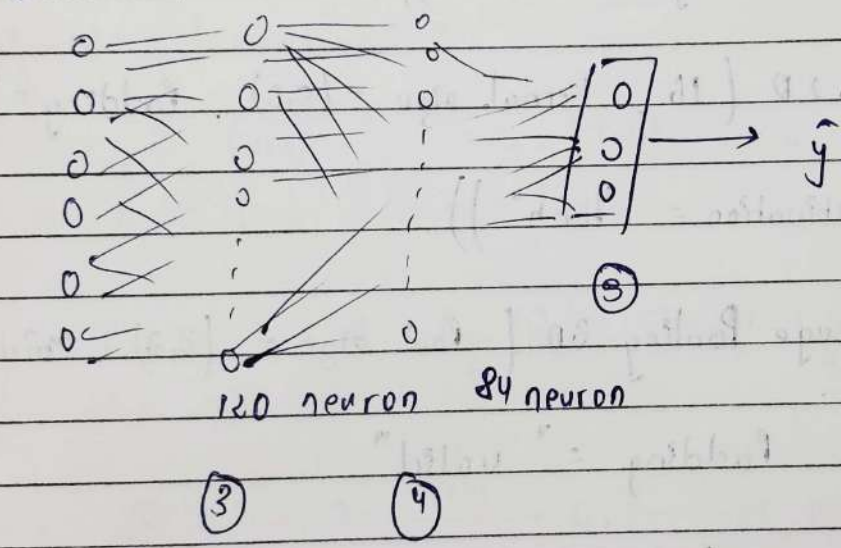
(CNN architectures.)

(1) LeNET →
(2) Alex NET
(3) Google NET
(4) VggNET
(5) ResNTE
(6) Inception

PAGE NO.:
DATE: / /

• LeNET - 5





• size of Image variation

$(32,32) \longrightarrow [(28,28,6) \rightarrow (14,14,6)] \longrightarrow [(10,10,16) \rightarrow (5,5,16)]$

Flatten $5 \times 5 \times 16 = 400 \longrightarrow 120 \longrightarrow 84 \longrightarrow$

features neuron neurons

lenet demo.

→ model = Sequential()

model.add(Conv2D(6, kernel_size=(5,5), padding="valid", activation="tanh", input_shape=(32,32,1)))

model.add(AveragePooling2D(pool_size=(2,2), stride=2, padding="valid"))

model.add(Conv2D(16, kernal_size=(5,5), padding="valid", activation="tanh"))

model.add(AveragePooling2D(pool_size=(2,2), stride=2, padding="valid"

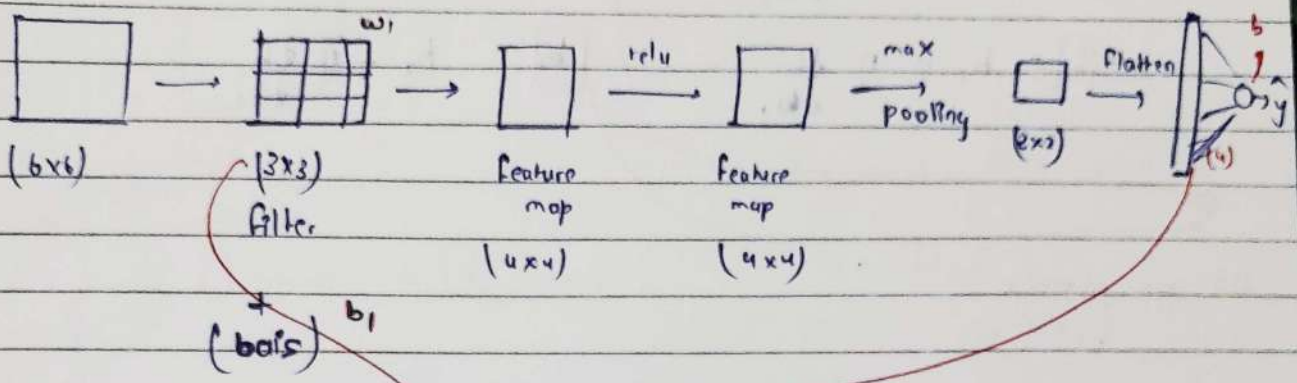
model.add(Flatten())

model.add(Dense(120, activation="tanh"))
model.add(Dense(84, activation="tanh"))
model.add(Dense(10, activation="softmax"))

In CNN trainable parameters don't depend on Input Image

PAGE NO :
DATE : / /

* Backpropagation in CNN :-

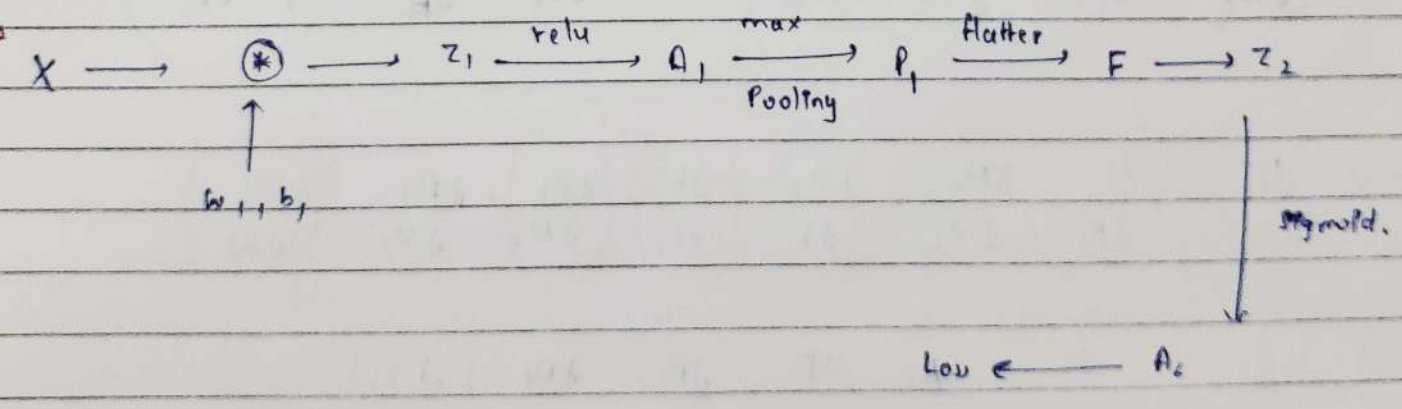


- Trainable parameters

$w_1 = (3,3)$ $w_2 = (1,4)$ = 15, trainable parameters

$b_1 = (1,1)$ $b_2 = (1,1)$

$$Loss = -y_i \log(\hat{y}_i) - (1-y_i) \log(1-\hat{y}_i)$$



→ Forword Propagation

$Z_1 = conv(X, w_1) + b_1$ $F = flatten(P_1)$

$A_1 = relu(z_1)$ $z_2 = w_2 F + b_2$

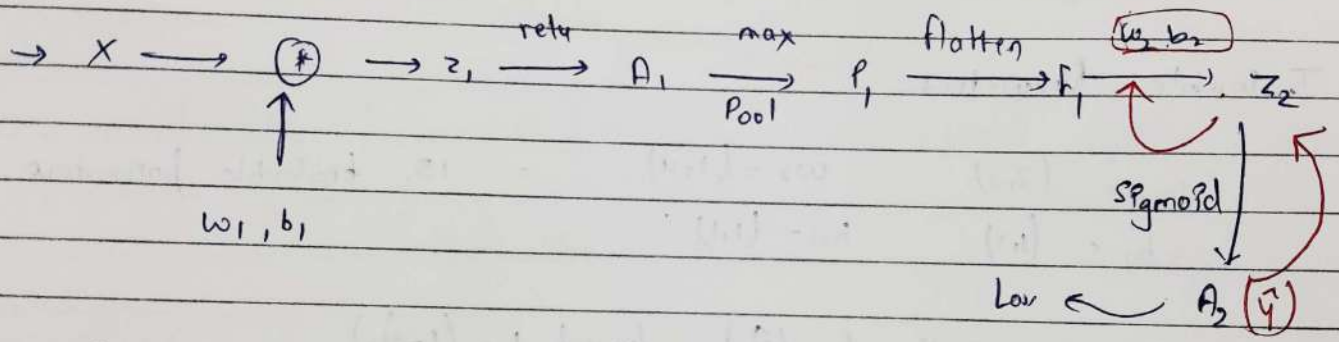
$P_1 = max\,pool(A_1)$ $A_2 = \sigma(z_2)$

→ Gradient Descent :-

$$w_1 = w_1 - \eta \frac{dL}{dw_1} \qquad w_2 = w_2 - \eta \frac{dL}{dw_2}$$

$$b_1 = b_1 - \eta \frac{dL}{db_1} \qquad b_2 = b_2 - \eta \frac{dL}{db_2}$$

All are matrix



$$\left(\frac{dL}{dw_2} = \frac{dL}{dA_2} \times \frac{dA_2}{dz_2} \times \frac{dz_2}{dw_2} \quad , \quad \frac{dL}{db_2} = \frac{dL}{dA_2} \cdot \frac{dA_2}{dz_2} \cdot \frac{dz_2}{db_2} \right)$$

→ $$\frac{dL}{dw_1} \quad \frac{dL}{dA_2} \cdot \frac{dA_2}{dz_2} \cdot \frac{dz_2}{dF} \left(\frac{dF}{dP_1} \right) \cdot \left(\frac{dP_1}{dA_1} \right) \cdot \frac{dA_1}{dz_1} \cdot \left(\frac{dz_1}{dw} \right)$$

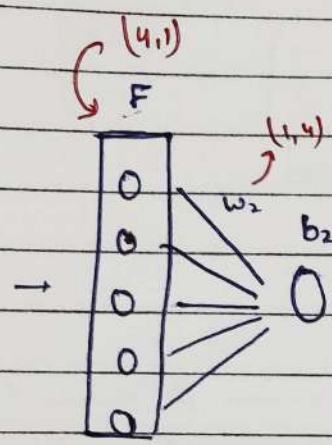
→ $$\frac{dL}{wb_1} = \frac{dL}{dA_1} \cdot \frac{dA_2}{dz_2} \cdot \frac{dF}{dP_1} \cdot \frac{dP_1}{dA_1} \quad \frac{dA_1}{dz_1} \left(\frac{dz_1}{db_1} \right)$$

PAGE NO. :
DATE : / /

now = ?

$$\frac{dL}{dw_1} = \frac{dL}{dA_2} \cdot \frac{dA_2}{dz_2} \cdot \frac{dz_2}{dw_1}$$

$$\frac{dL}{db_2} = \frac{dL}{dA_2} \cdot \frac{dA_2}{dz_2} \cdot \frac{dz_2}{db_2}$$



→ forward Prop eqⁿ

$$z_2 = w_2 f + b_2$$

$$A_2 = \sigma(z_2)$$

① → $\frac{dL}{dA_2} = \frac{d}{dA_2}\left[-y_i \log(a_2) - (1-y_i)\log(1-a_2)\right]$

A_2 → matrix

$$= \frac{-y_i}{a_2} + \frac{(1-y_i)}{1-a_2} = \frac{-y_i(1-a_2) + a_2(1-y_i)}{a_2(1-a_2)}$$

$$= \left\{\frac{(a_2 - y_i)}{a_2(1-a_2)}\right\}$$

② → $A_2 = \sigma(z_2)$

$$\frac{dA_2}{dz_2} = \sigma(z_2)(1-\sigma(z_2)) = \{a_2(1-a_2)\}$$

③ $z_2 = w_2 f + b$

$$\frac{dz_2}{dw_2} = (F) \qquad \frac{dz_2}{db_2} = (1)$$

$$\frac{\partial L}{\partial w_2} = \frac{(a_2 - y_i)}{a_2(1-a_2)} \times \cancel{(a_2)(1-a_2)} \times F = (a_2 - y_i) \cdot F$$

PAGE NO. :
DATE : / /

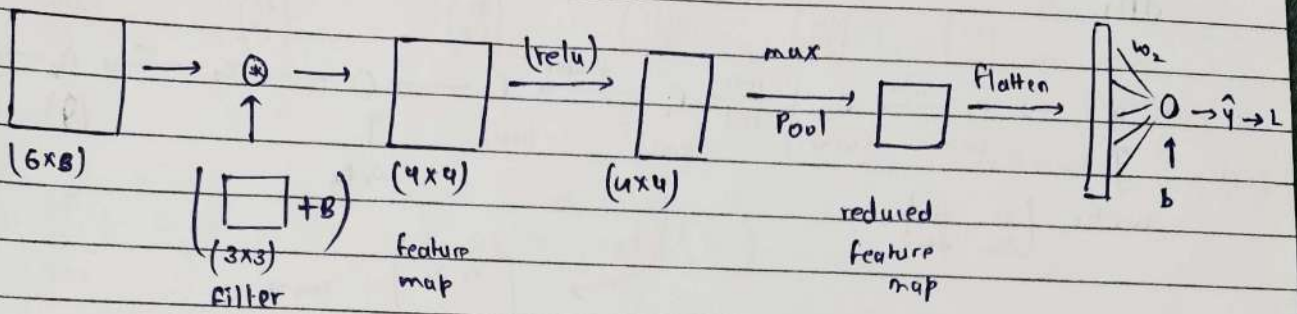
$a_2 \rightarrow$ matrix (A_2)

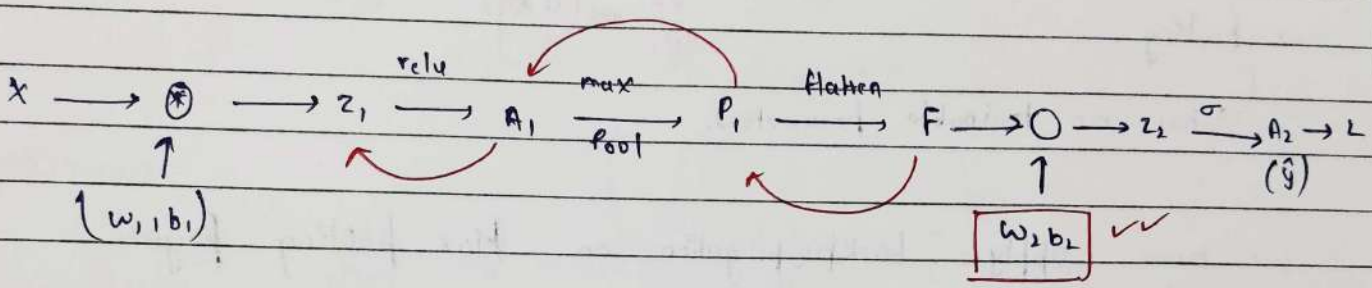
$$\frac{\partial L}{\partial w_2} = (A_2 - Y) F^T$$

$(1,4)$ $(1,1)$ $(4,1)$

$$\frac{\partial L}{\partial b_2} = (A_2 - Y)$$

Back propagation in CNN :-





→ forward propagation eqn.

$$z_1 = conv(x, w_1) + b_1 \qquad A_1 = relu(z_1)$$

$$P_1 = maxpool(A_1) \qquad F = Flatten(P_1)$$

$$z_2 = w_2 F + b_2 \qquad A_2 = \sigma(z_2)$$

$$L = \frac{1}{m} \sum_{i=1}^{n} \left[-y_i \log(A) - (1-y_i) \log(-y_i) \right]$$

already calculated

$$\frac{dL}{dw_1} = \left(\frac{dL}{dA_2}\right)\left(\frac{dA_2}{dz_2}\right) \cdot \frac{dz_2}{dF} \cdot \boxed{\frac{dF}{dP_1}} \cdot \frac{dP_1}{dA_1} \cdot \frac{dA_1}{dz_1} \cdot \frac{dz_1}{dw_1}$$

$$\frac{dL}{db_1} = \frac{dL}{dA_2} \cdot \frac{dA_2}{dz_2} \cdot \frac{dz_2}{dF} \cdot \boxed{\frac{dF}{dP_1}} \cdot \frac{dP_1}{dA_1} \cdot \frac{dA_1}{dz_1} \cdot \frac{dz_1}{db_1}$$

$$\boxed{\frac{dz_2}{dF} = w_2}$$

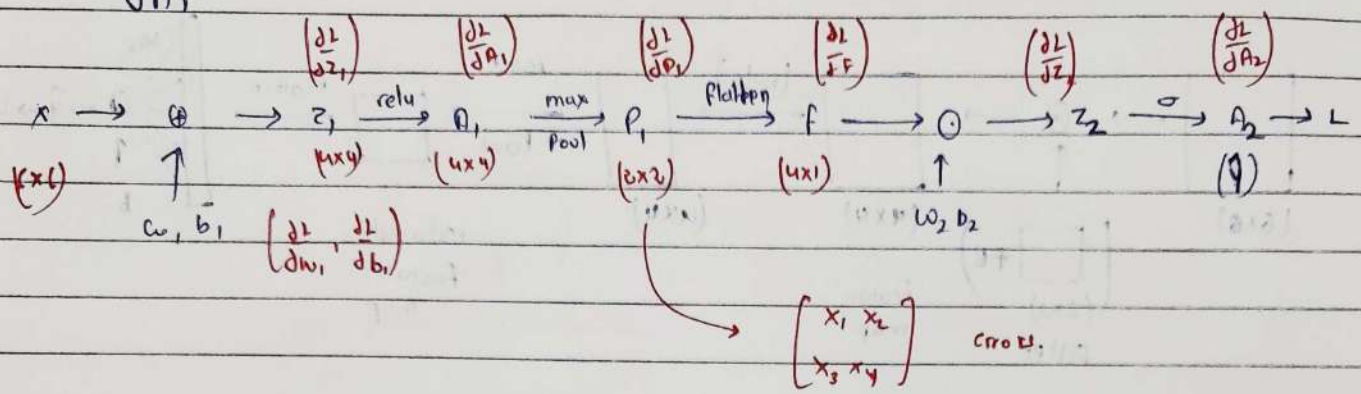
But $\frac{dF}{dP_1}$ has no trainable Parameter.

$$\frac{dF}{dP_1} = \text{reshape}(P_1 \cdot \text{Shape})$$

Pooling operation.

PAGE NO.:
DATE: / /

$\frac{\partial P_1}{\partial A_1}$



max pooling

$\frac{\partial L}{\partial A_1} \Rightarrow (4 \times 4)$

↳ has no trainable parameters.

we have apply backpropogation on Max pooling layer.

max pool convert $(4 \times 4) \longrightarrow (2 \times 2)$

eg.

$$A_1 = \begin{bmatrix} 1 & 2 \\ 3 & \textcircled{4} \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & \textcircled{8} \end{bmatrix} \\ \begin{bmatrix} 9 & 10 \\ 11 & \textcircled{12} \end{bmatrix} \begin{bmatrix} 13 & 14 \\ 15 & \textcircled{16} \end{bmatrix} \longrightarrow \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}$$

The numbers other than the max no° in their window has no role in predicting $\hat{y}$ / L

So in backpropogation when $\begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}$ will be converted

to $\begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 8 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 12 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 16 \end{bmatrix}$

position of these will be asked from A_1

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial A_2} \cdot \frac{\partial A_2}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial F} \cdot \frac{\partial F}{\partial P_1} \cdot \frac{\partial P_1}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial W_1}$$

$\frac{\partial L}{\partial A_1}$

PAGE NO.:
DATE: / /

$\frac{\partial L}{\partial P_1}$

$$\frac{\partial L}{\partial A_1} = \begin{cases} \frac{\partial L}{\partial P_{xy}}, & \text{if } A_{mn} \text{ is the max element} \\ 0, & \text{otherwise} \end{cases}$$

→ now

$\frac{\partial A_1}{\partial Z_1} = ?$ $\quad A_1 = \text{relu}(Z_1)$

$$\frac{\partial A_1}{\partial Z_1} = \begin{cases} 1 & \text{if } Z_{1\,xy} > 0 \\ 0 & \text{if } Z_{1\,xy} < 0 \end{cases}$$

→ now

$\frac{\partial Z_1}{\partial W_1} = ?$ $\quad \frac{\partial Z_1}{\partial b_1} = ?$

- Back prop on convolution

$X \longrightarrow (*) \longrightarrow Z_1$

(3,3) $\quad \uparrow (W_1, b_1)$ $\quad \left(\frac{\partial L}{\partial Z_1}\right) \rightarrow (2,2)$

(2,2)

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}$$

$$W_1 = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \qquad Z_1 = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$$

after convolution operation

$$Z_{11} = X_{11}W_{11} + X_{12}W_{12} + X_{21}W_{21} + X_{22}W_{22} + b_1$$

$$Z_{12} = X_{12}W_{11} + X_{13}W_{12} + X_{22}W_{21} + X_{23}W_{22} + b_1$$

$$Z_{21} = X_{21}W_{11} + X_{22}W_{12} + X_{31}W_{21} + X_{32}W_{22} + b_1$$

$$Z_{22} = X_{22}W_{11} + X_{23}W_{12} + X_{32}W_{21} + X_{33}W_{22} + b_1$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$$

PAGE NO.:
DATE: / /

$$\frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} = \frac{\partial L}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial b_1} + \frac{\partial L}{\partial z_{12}} \cdot \frac{\partial z_{12}}{\partial b_1} + \frac{\partial L}{\partial z_{21}} \cdot \frac{\partial z_{21}}{\partial b_1} + \frac{\partial L}{\partial z_{22}} \cdot \frac{\partial z_{22}}{\partial b_1}$$

$$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$$

$$(1) \quad (1) \quad (1) \quad (1)$$

$$= \left(\frac{\partial L}{\partial z_{11}} + \frac{\partial L}{\partial z_{12}} + \frac{\partial L}{\partial z_{21}} + \frac{\partial L}{\partial z_{22}} \right)$$

$$= \text{sum} \left(\frac{\partial L}{\partial z_1} \right)$$

of all terms of

$$\rightarrow \frac{\partial L}{\partial w_1} = ?$$

$$\frac{\partial L}{\partial w_1} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

$$\rightarrow$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{11}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_1} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{11}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{11}}$$

$$\frac{\partial L}{\partial w_{12}} =$$

$$\frac{\partial L}{\partial w_{21}} =$$

$$\frac{\partial L}{\partial w_{22}} =$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} \cdot X_{11} + \frac{\partial L}{\partial z_{12}} \cdot X_{12} + \frac{\partial L}{\partial z_{21}} X_{21} + \frac{\partial L}{\partial z_{22}} \cdot X_{22}$$

PAGE NO. :
DATE : / /

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} \cdot X_{12} + \frac{\partial L}{\partial z_{12}} \cdot X_{13} + \frac{\partial L}{\partial z_{21}} \cdot X_{22} + \frac{\partial L}{\partial z_{22}} \cdot X_{23}$$

$$\frac{\partial L}{\partial w_{21}} = \text{- - - - -}$$

$$\frac{\partial L}{\partial w_{22}} = \text{- - - - -}$$

we can say.

$$\frac{\partial L}{\partial w_1} = \text{conv}\left(X, \frac{\partial L}{\partial z_1}\right)$$

↑ matrixs ↑ matrix

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \qquad \frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

* Conclusion :-

$$\frac{\partial L}{\partial w_1} = \text{con}\left(X, \frac{\partial L}{\partial z_1}\right)$$

$$\frac{\partial L}{\partial b_1} = \text{sum}\left(\frac{\partial L}{\partial z_1}\right)$$

PAGE NO. :
DATE : / /

• Pretrained model.

→ image NET Data set.

↳ images 1.4 crores

2010	→	ML model	→	28%
2011	→	ML model	→	25%
2012	→	Alex Net	→	16.4%
2013	→	ZF Net	→	11.7%
2014	→	VGG	→	7.3%
2015	→	Google Net	→	6.7%
2016	→	RESNET	→	3.5%

Revolution

Humans → 5% error rate.

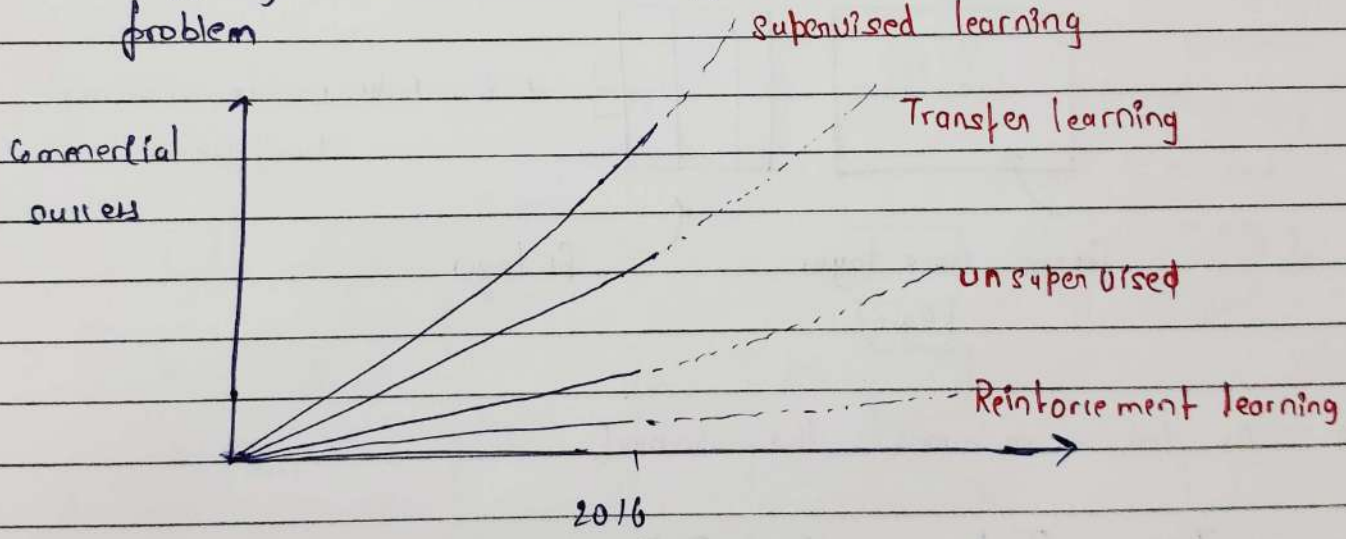
→ code is implimented in jupyter notebook.

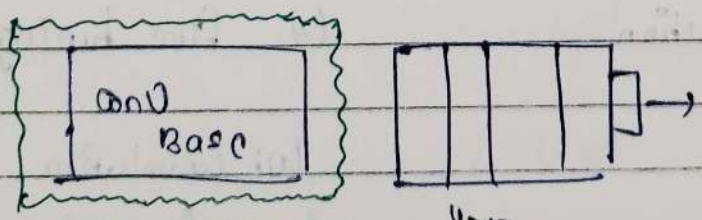
Transfer learning.

→ Pretrained model are very good. but what to do when pretrained model do not have the categories that we want to classify.

→ that were transfer learning comes in.

→ T.F learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem





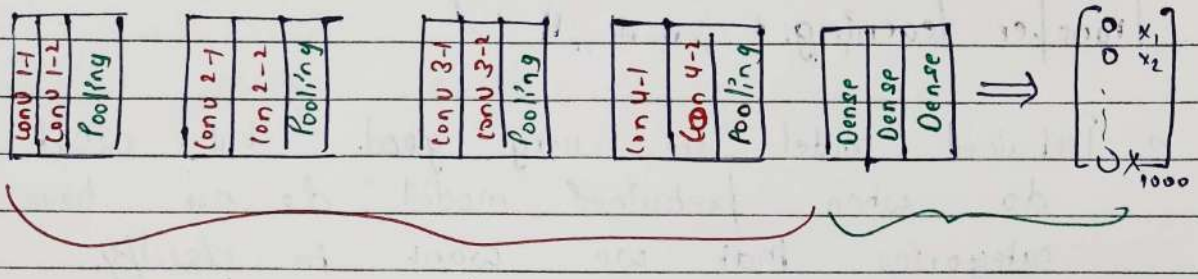
Base convolution layer. (freezed)

- will not be trained

Your Dense layer.

trained on Image net data set

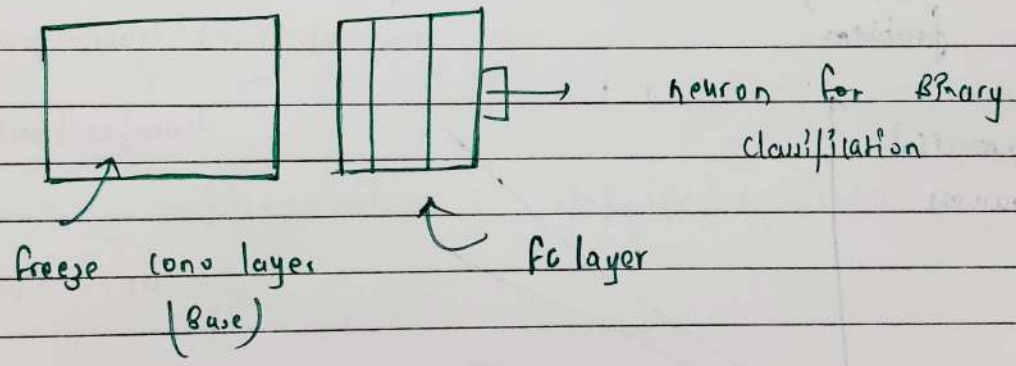
Vgg 16 model.



Convolutional Base layers

Fc layer

→ now. convolutional layer is freezed and then used to perform another problem by By taking convolutional layer as it is.



Do not re invent the wheel.

- Two Type of T.F.

(1) Feature extraction (Discussed above)

(2) Fine tuning.

last convolution layer is unfreezed
→ and then then training is done.
→ Done when.

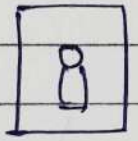
PAGE NO. :
DATE : / /

Fine tuning

→ Done when we have to do classification which is not in our training Dataset classification labels

* Non linear neural networks

↳ Why needed.

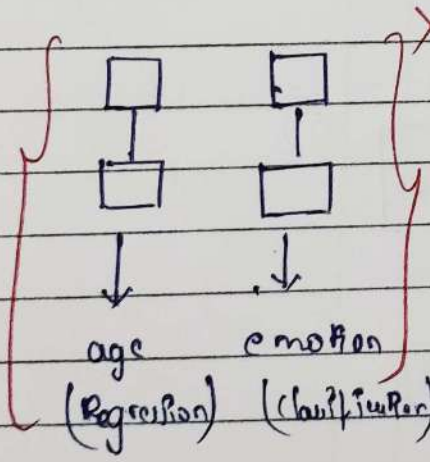
Example →  → Image dataset → humans faces → 25000 images

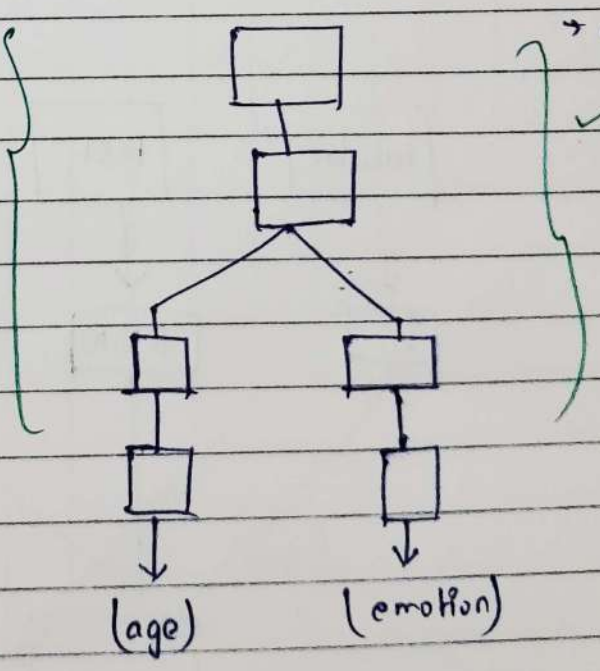
(Image)

age ? → (Regression)

emotion → happy, sad, angry. → (classification)

To solve there can be two ways





(age) (emotion)

PAGE NO. :
DATE : / /

Ex :-

Let say you are working in e-commerce company.

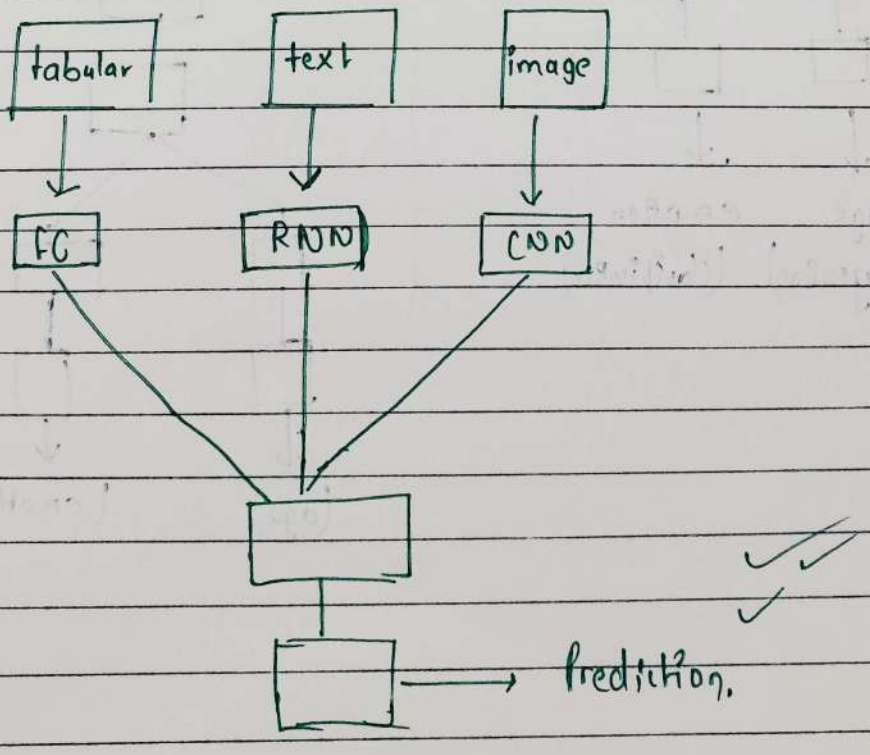
↳ certain information is given to you.

↳ Data given of product.

1) Product meta data → (tabular)
2) Product descrip. text → (text)
3) product image → (Image)

Predict price → ??

→ Good approach to solve this problem can be :. (non-linear neural network)



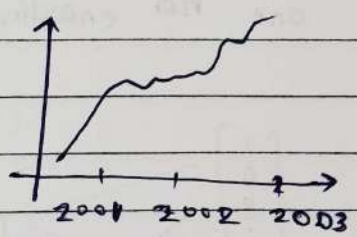
PAGE NO.:
DATE: / /

Recurrent neural network. (RNN)

Depend on sequencial data. (sequence matters).

eg. ① → text { Hi my name is Akash }

② → Time series {



③ speech

④ DNA sequence.

→ Why use ANN ?

lets say we need to do text classification using ann.

(Input)	Output
Hi my name is Akash	0
I love cambus (X)	0
India won the match	1

ANN will not understand words we need to vectorize them to feed them in neural network.

methods.

→ one hot encoding

- bag of words.

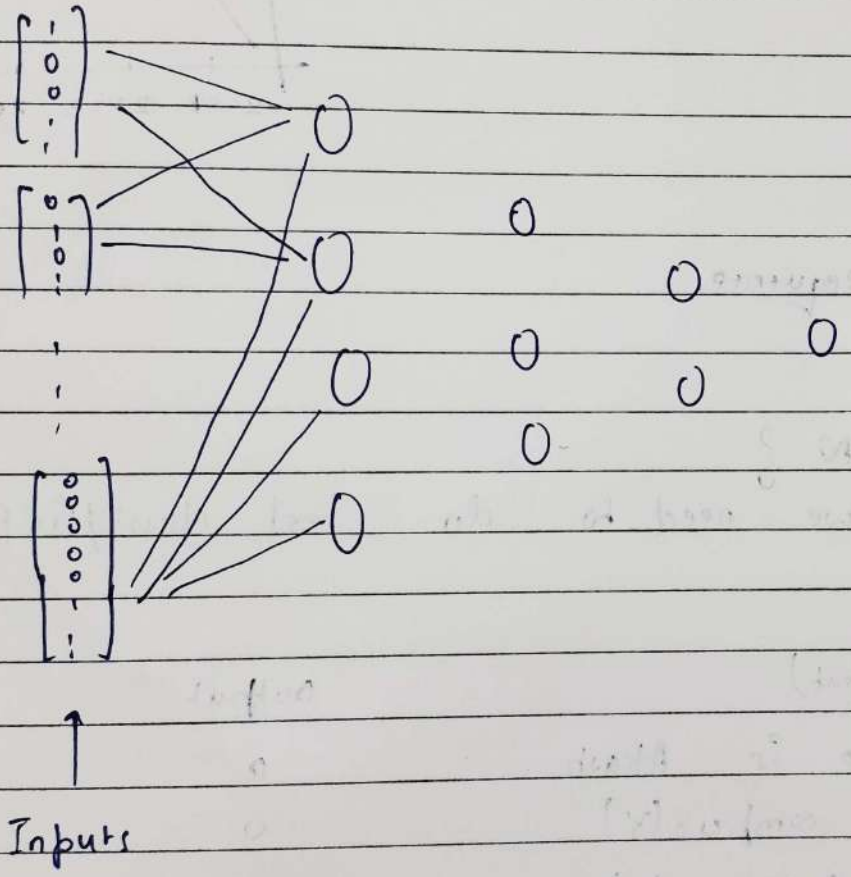
PAGE NO. :
DATE : / /

Hi my name is Akash.

[1,0,0..] [0,1,0..] [0,0,1,0..] ... [0,0,0,0,1,0..]

12 items

one hot encoding vectorization.



Inputs

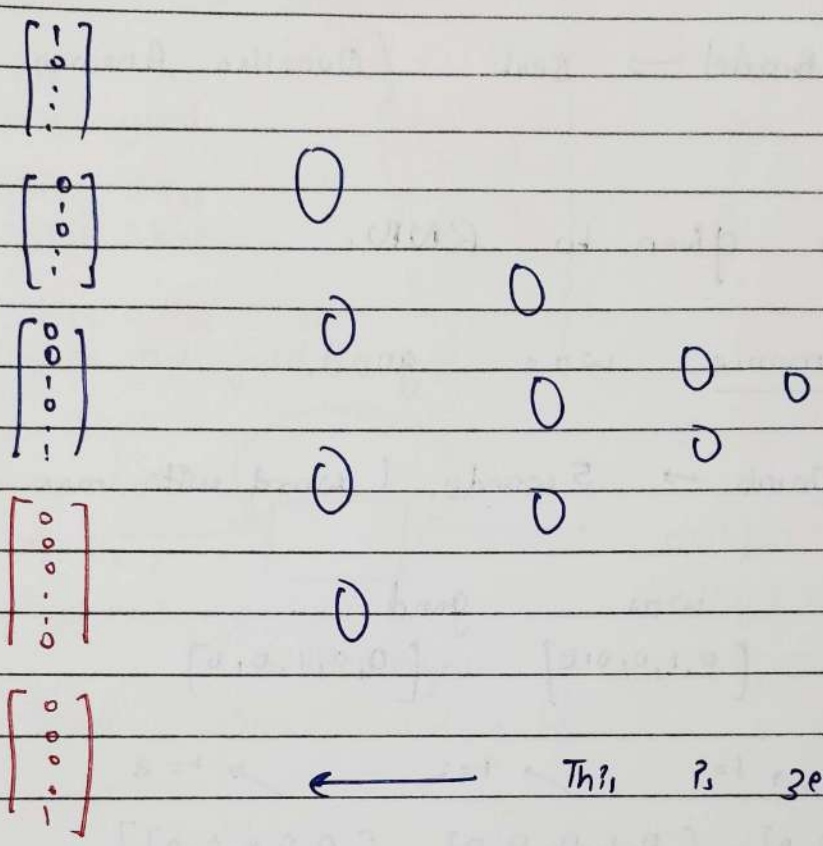
Problem → Input size can be different with every sentence

Solution → zero padding.

eg. I love Jaipur.

PAGE NO.:
DATE: / /

But input needs to be of 5 vectors. so.



← This is zero padding.

But it has problem like

(1) unnessary computation

(2) text input → varying size.

(3) prediction problem

(4) Totally disregarding the sequence info.

Projects

→ Sentiment analysis demo
→ Sentence prediction
→ Generate image caption

PAGE NO. :
DATE : / /

* Advanced Model → Bert. (Question Answer Demo)

How data is given to RNN.

eg. → movie was good.

Vocab → 5 words. (word with max digits)

movie [1,0,0,0,0] was [0,1,0,0,0] good. [0,0,1,0,0]

→ [[1,0,0,0,0] (t=1), [0,1,0,0,0] (t=2), [0,0,1,0,0] (t=3)]

↓

(3, 5)

3 → no. of time steps
5 → # of features.

(Keras) → simple RNN → batch size, time step, input features

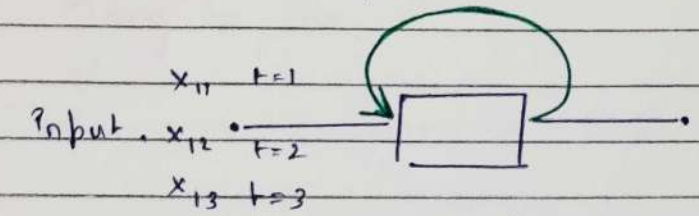
(3, 4, 5)

How RNN works.

Review				sentiments
X_1	movie	was	good	1
X_2	movie	was	bad	0
X_3	movie	was	not good	0

X_{11} X_{12} X_{13} — X_{21} X_{22} X_{23} — X_{31} X_{32} X_{33} X_{34}

RNN → ANN → a big difference



(time based)

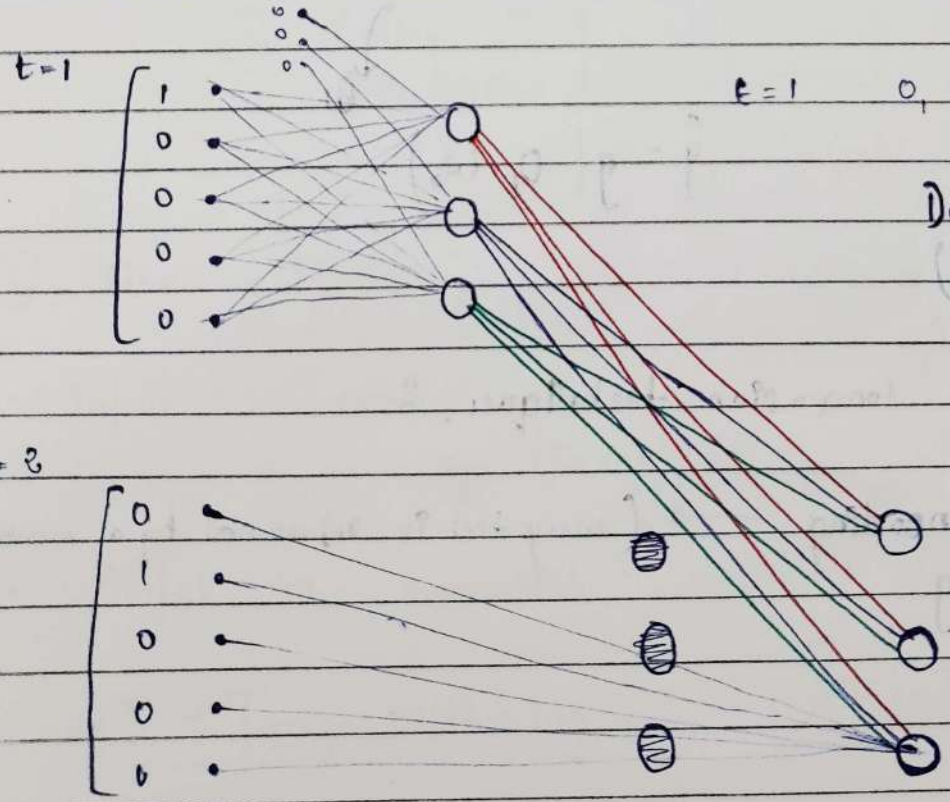
$t_1 = 1$ x_{11}

$t_2 = 2$ x_{12}

$t = 3$ x_{13}

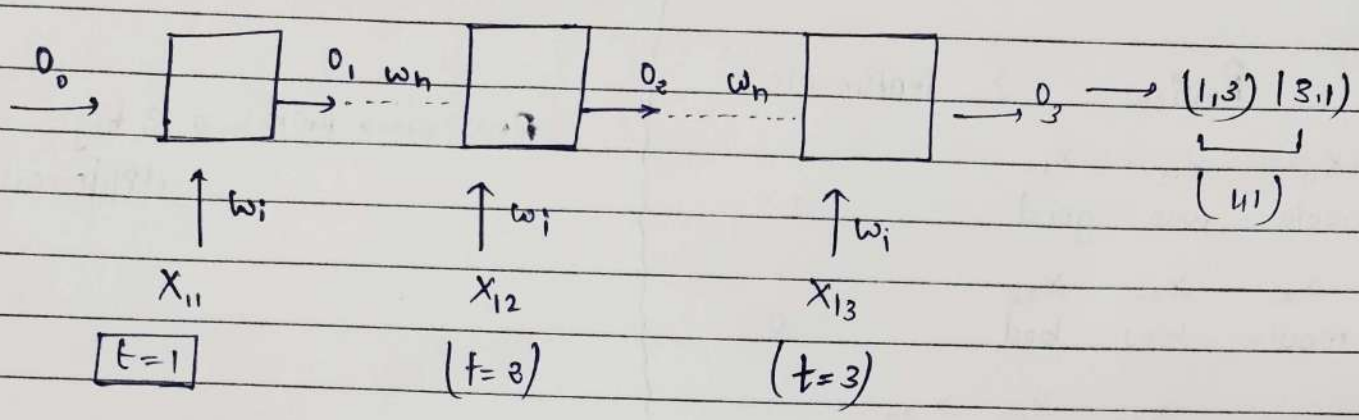
{Input is given according to time}

→ feedback mechanism :-

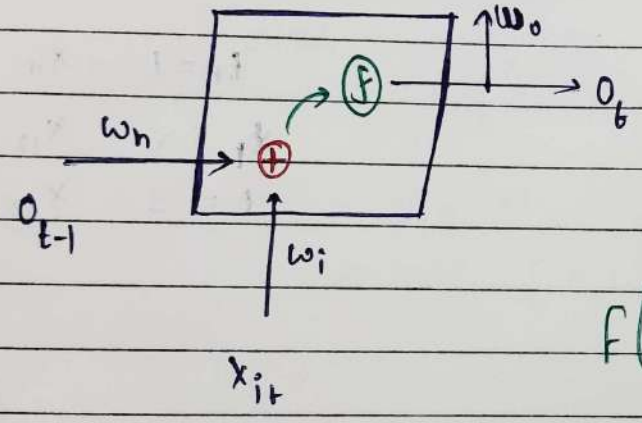


t = 1 0, random number

Default activation function = tan(h)



→ simplified Representation



$f(x_{it} w_i + O_{t-1} w_h) \rightarrow$ Vector $\rightarrow O_t$

$\hat{y} = g(O_t \cdot w_o)$

i = row #

t = time step

X_{11} → word (first)

* Text to vector conversion techniques :-

1. Integer encoding → (every text is represented by a number)
2. embedding

PAGE NO.:
DATE: / /

* Types of RNN

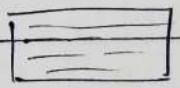
1. Many to one.
2. One to many
3. Many & many
4. One & One.

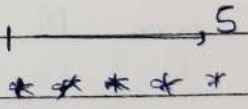
→ Many to one.

Input → Sequence → Sentence, characters, time series.

output → non-sequential → integer / num → scaler (0,1)

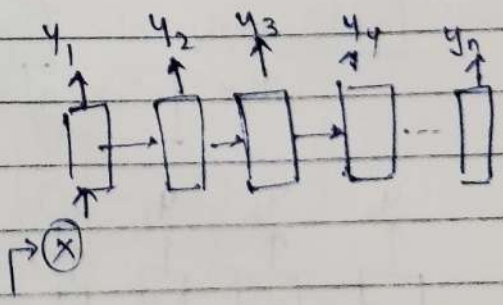
eg. ① sentiment analysis →  ⇒ (0,1)

② Rating Prediction → 



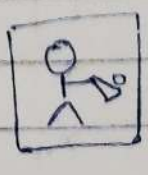
multiclass classification

② One to Many :-



- Input → normal non sequential
  eg. (Image)
- Output → sequential

eg :- → Image captioning.

 → RNN → A man is playing cricket.

→ music generation

PAGE NO.:
DATE: / /

③ Many to many →

input → sequential
output → sequential

> Seq & Seq

Same length

input seq == output seq.

eg → POS tagging → NLP

1	2	3	4
My	name	is	Akash
↓	↓	↓	↓
Pronoun ①	noun ②	Verb ③	Noun ④

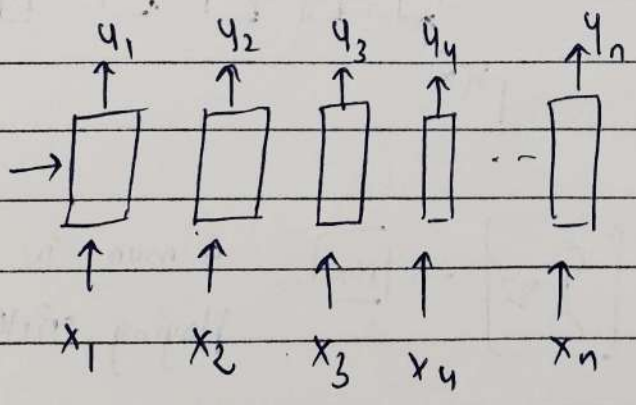
Variable length

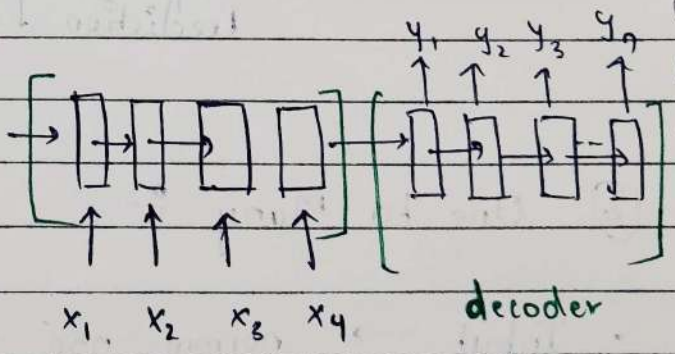
eg: machine translation.
language → other language
↓
(google translate)

→ Hi my name is Akash
→ मेरा नाम आकाश है

② Name entity Recognition (NER)

→ [lets] meets at [7pm] at the [airport]





PAGE NO.:
DATE: / /

④ One-to-One [Technically not a neural network

input → non-sequential

output → non-sequential → Image classification.

Back Propagation in RNN

Through an example of : (Many to one RNN)

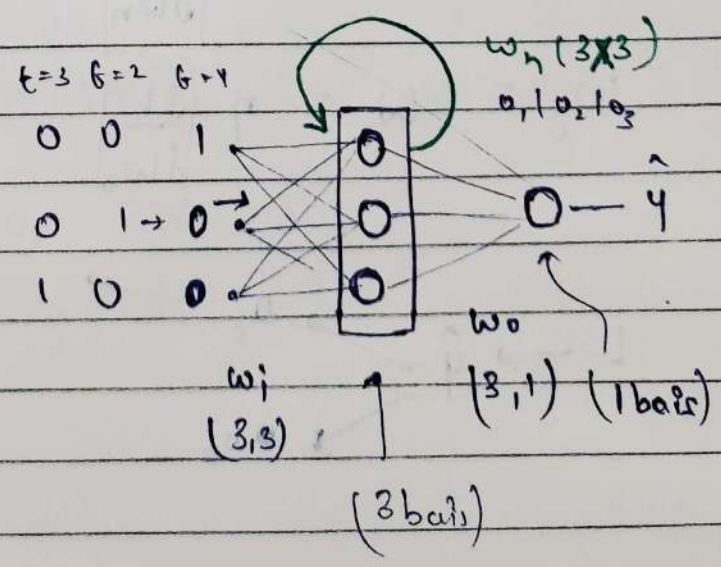
text → 1/0

→ let's take a toy Dataset.

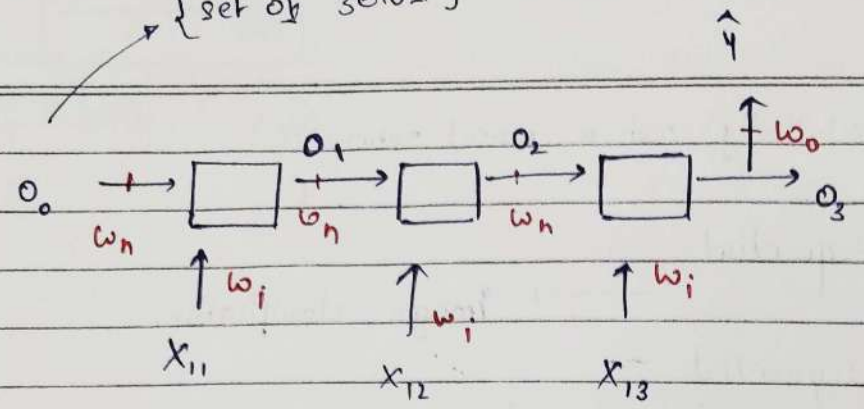
text				X				Y
Cat	mat	rat	1	x_1	[100]	[010]	[001]	1
rat	rat	mat	1 →	x_2	[001]	[001]	[010]	1
mat	mat	Cat.	0	x_3	[010]	[010]	[100]	0

Vocab → Cat mat rat

[100] [010] [001]



{set of zero's}



$$o_1 = f(x_{11}w_i + o_0 w_h)$$

$$o_2 = f(x_{12}w_i + o_1 w_h)$$

$$o_3 = f(x_{13}w_i + o_2 w_h)$$

$$y \rightarrow \hat{y} = \sigma(o_3 w_0)$$

$$L = -y_i \log(\hat{y}_i) - (1-y_i)\log(\quad)$$

→ gradient descent

$$w_i' = w_i - \eta \frac{dL}{dw_i}$$

$$w_h = w_h - \eta \frac{dL}{dw_h}$$

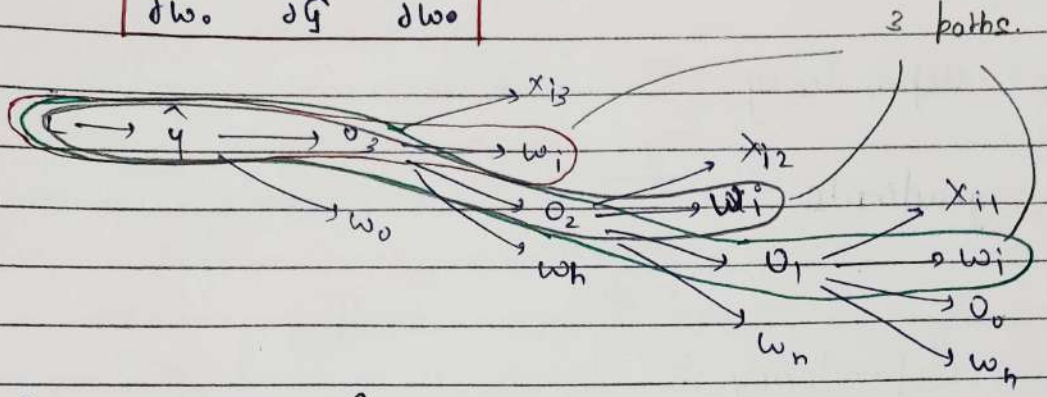
$$w_0 = w_0 - \eta \frac{dL}{dw_0}$$

$$L \rightarrow \hat{y} \rightarrow o_3 \rightarrow$$

$$\hat{y} \rightarrow w_0$$

PAGE NO. :
DATE : / /

$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_0}$$



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

$$\rightarrow \boxed{\frac{\partial L}{\partial w_i} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_i}}$$

$$\frac{\partial L}{\partial w_n} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_n} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_n} +$$

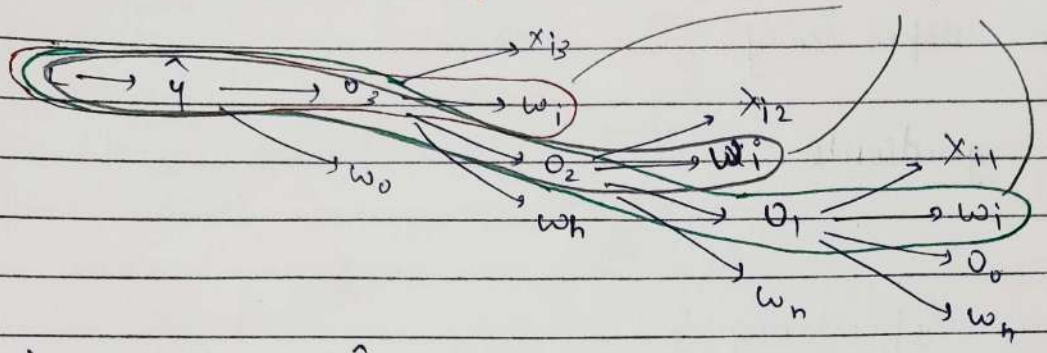
$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

$$\rightarrow \frac{\partial L}{\partial w_n} = \sum_{i=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_i}$$

PAGE NO. :
DATE : / /

$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_0}$$

3 paths.



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

$$\rightarrow \left[\frac{\partial L}{\partial w_i} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_i} \right]$$

$$\frac{\partial L}{\partial w_n} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_n} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_n} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

$$\rightarrow \frac{\partial L}{\partial w_n} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_i}$$

PAGE NO. :
DATE : / /

* Problem of RNN

① long term dependency. →

② Unstable gradients →

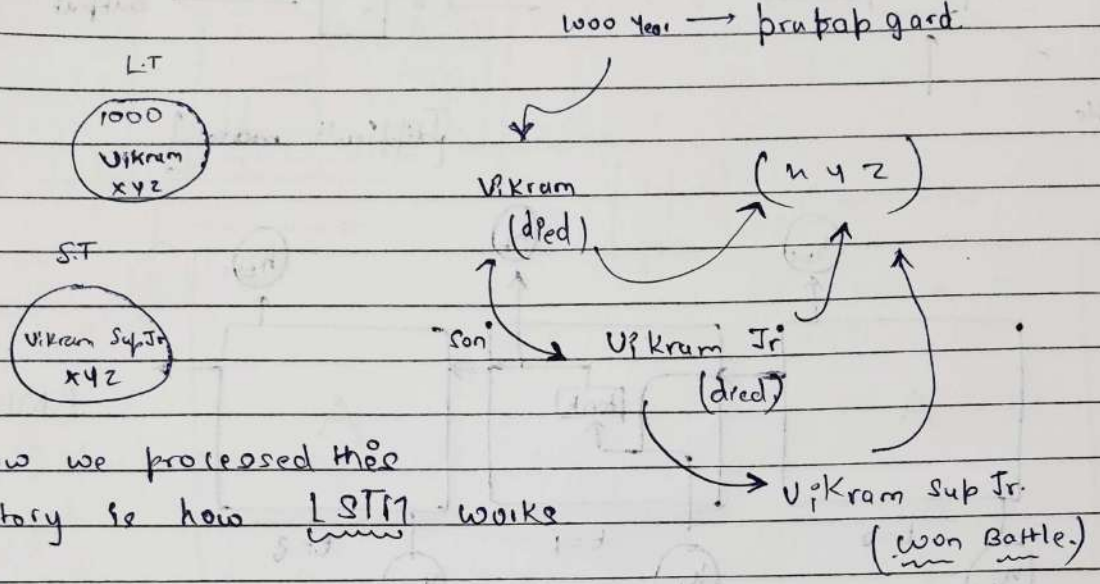
① long term dependency :-

eg :- → Marathi is spoken in Maharashtra.] short term dependence

→ Maharashtra is a beautiful place
I went there last year
But I could not enjoy there → long term dependence
Bc I could not speak Marathi

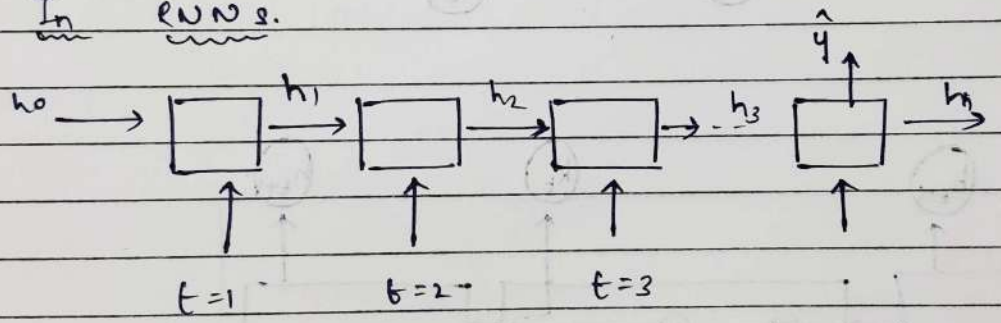
PAGE NO.:
DATE: / /

LSTM (long short Term memory)



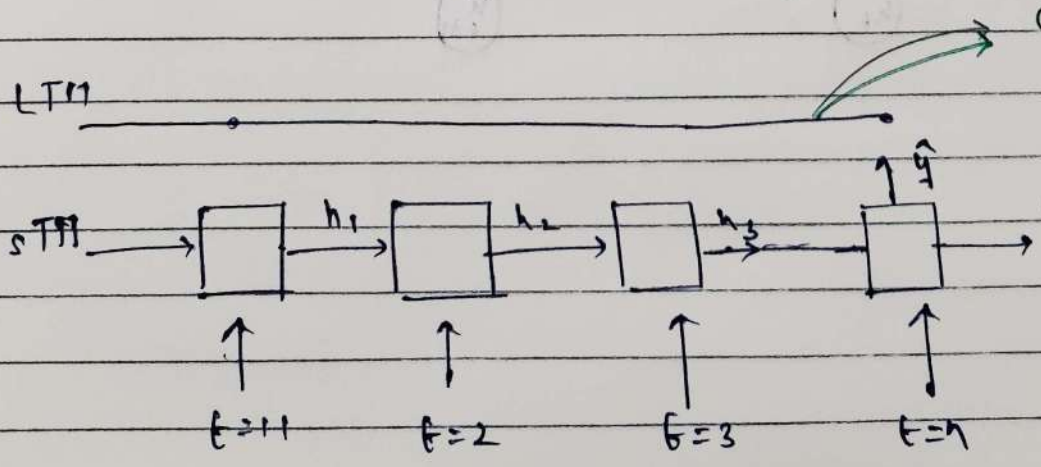
How we processed this story is how LSTM works

In RNNs.

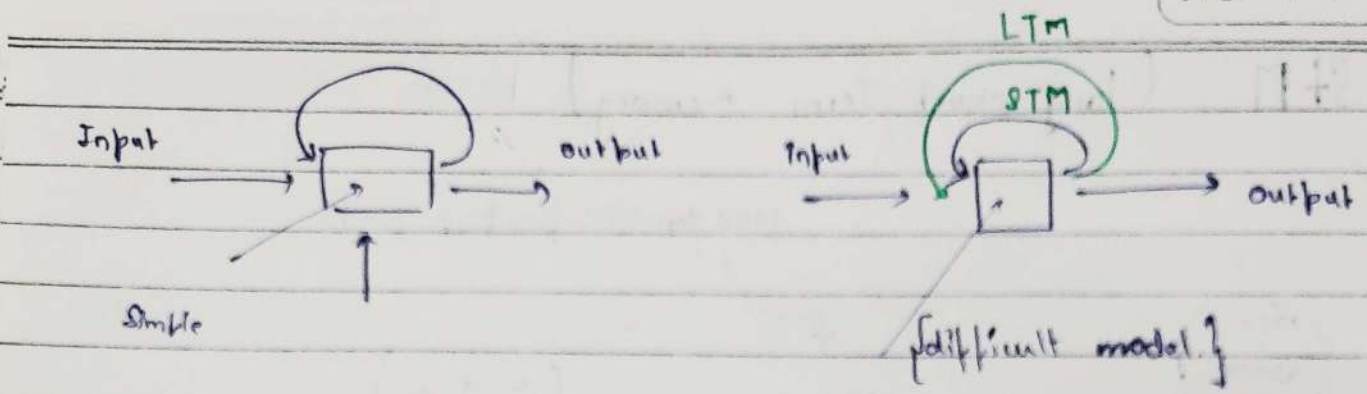


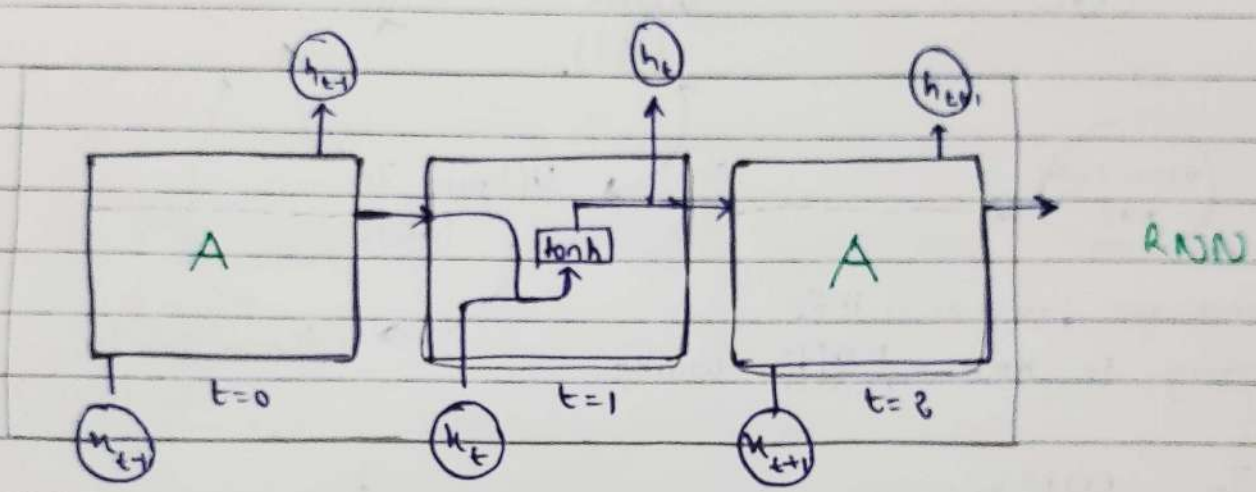
- we have no context to retain long term memory in RNN

→ so we Found a way to store long term memory.

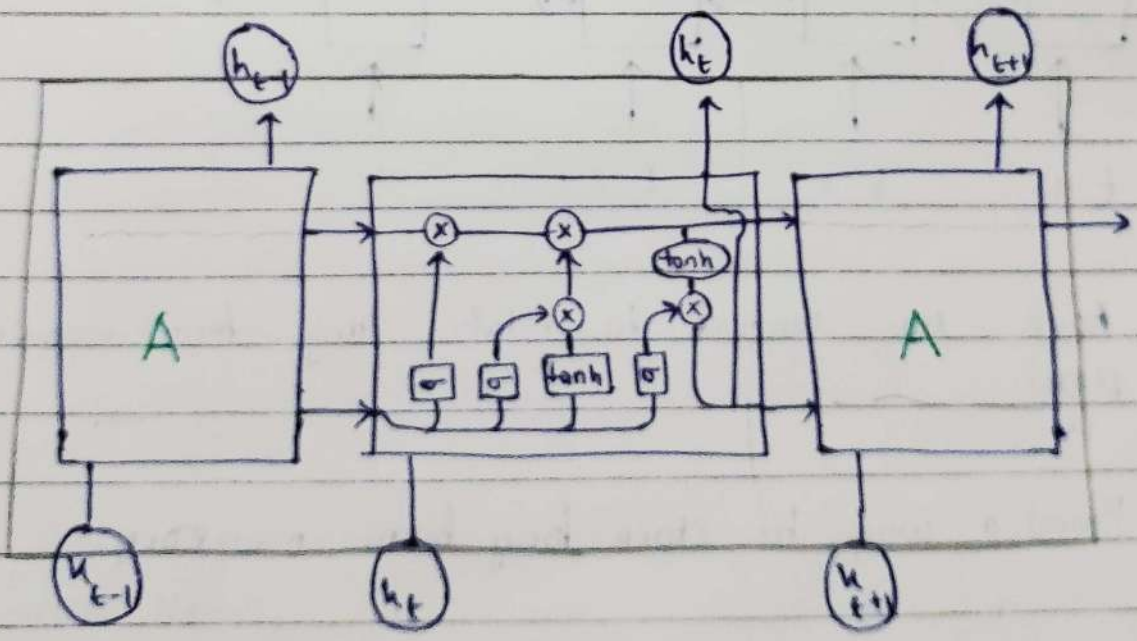


PAGE NO. :
DATE : / /

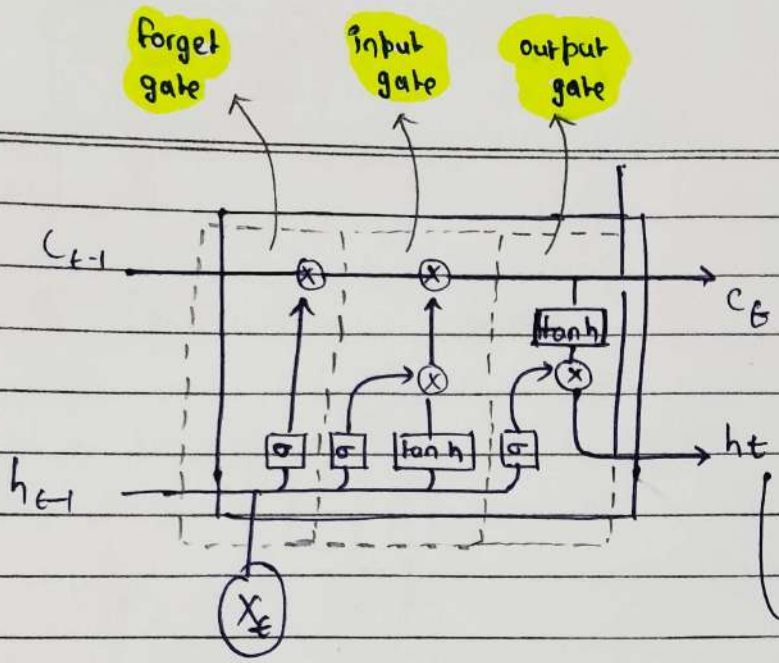




RNN



LSTM



PAGE NO. :
DATE : / /

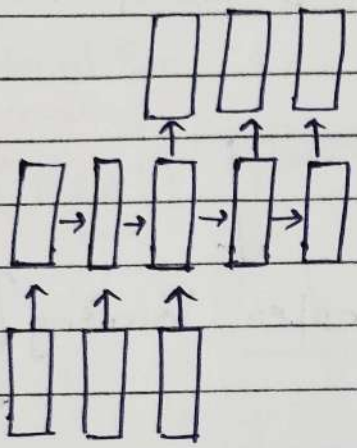
(forget gate) → {will remove items from long term memory} → {Based on current input}

(input gate) → {decides what to add in LTM} →

(Output gate) → {what should be shown as output from LTM} → {Based on input}

Large language Models

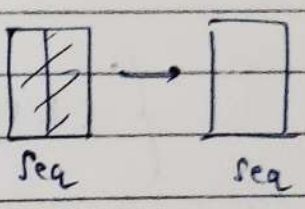
* Seq. 2 Seq tasks



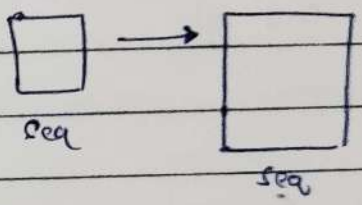
NLP

Seq 2 Seq → Machine trans

1) text summary →



2) Question Answering →



3) chatbot → input → output
(text) (text)

4) Speech to text →
↓ ↓
Seq Seq.

PAGE NO. :
DATE : / /

History of Seq. & Seq models.

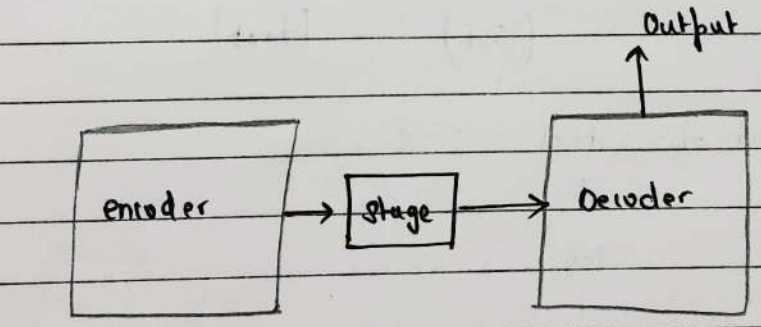
{encoder - decoder} → {Attention mech.} → {Transformer}

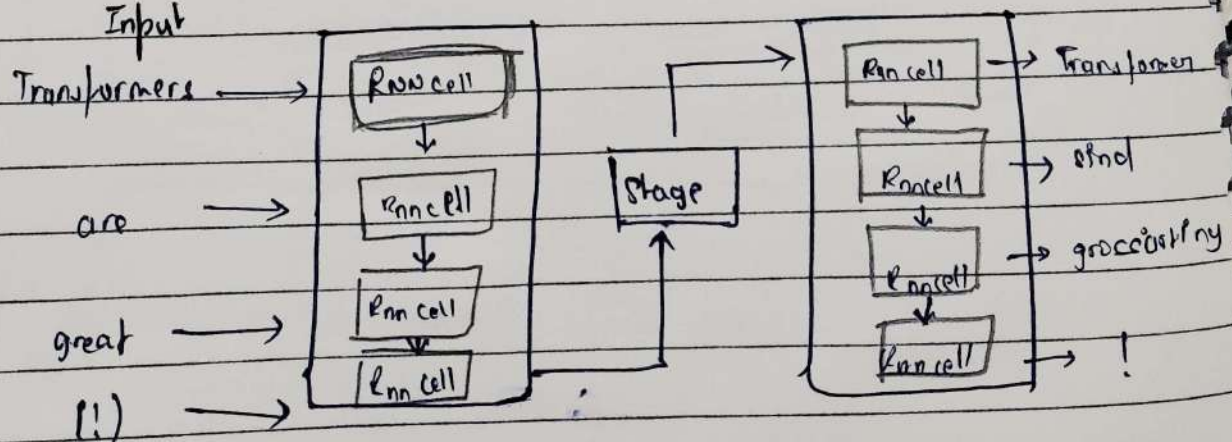
stage 1

{Transfer learning} → {LLMs}

Chat gpt.

(1) Encoder - decoder

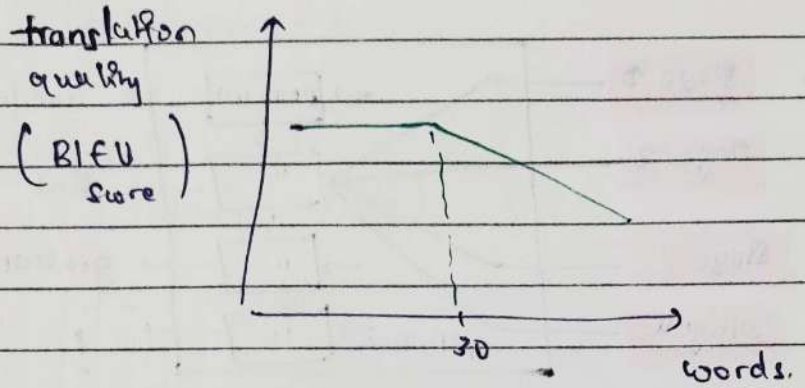




PAGE NO. :
DATE : / /

(Stage) → consist of the summary vector of the input

Problem → If very long input seq° then output is not good.



↳ A potential issue with this encoder-decoder approach is that a neural network to be able to, compress all the necessary information of a source sentence into a fixed length vector This may make it difficult for the neural network to cope with long sentences

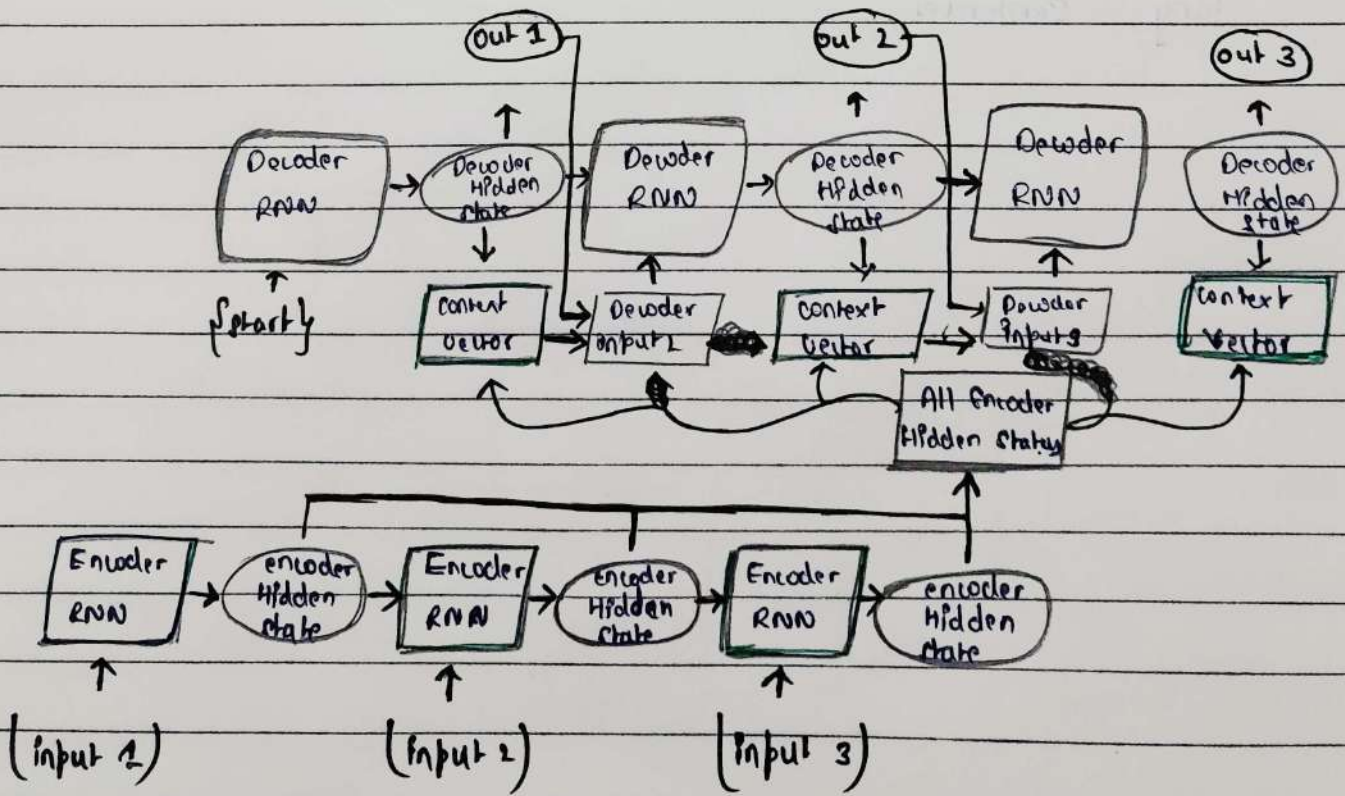
PAGE NO. :
DATE : / /

* Attention mechanism.

Paper ⇒ Neural Machine translation
By Jointly learning to Align and translate.

Attention mech°

In ~~transformer~~ there is no one single state vector at every step of predicting output it has notion of every step of input to perform better than encoder decoder

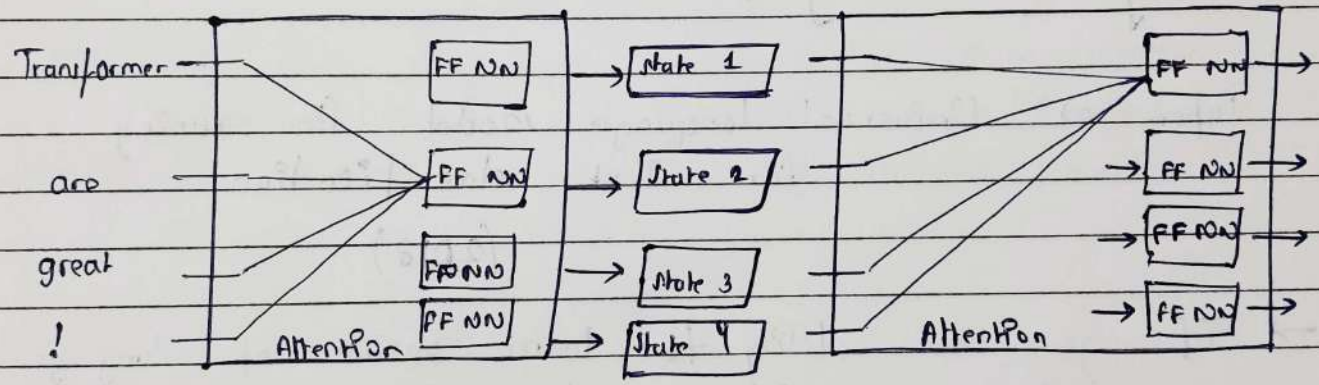


PAGE NO. :
DATE : / /

Problem :- Computational complexity is very high.
(using LSTM's are the problem)

* Transformers.

Google released a paper. (2017) : Attention is All You need.
(LSTM) X



(encoder) (decoder)

There are no (LSTM's / RNN cells.) X

↓

(Attention) ✓

(self Attention) ✓

→ It has ability to see all the words of input at every state of predicting of words.
- which is called parallel processing.

- Reduce computational complexity.

Problem →

1> Hardware → GPU → costly.
2> Time → skill not that low for training from scratch
3> Data → needs alot of data

Transfer learning :-

Paper => Universal language model fine tuning for Text classification (2018)

→ T.L gave the ability to learn basics of language and then it can be fine tuned to do any task
(No transformers used)

PAGE NO. :
DATE : / /

LLM's (2018)

Google and open AI released language models Based On transformer architecture.

Google (BERT) [Encoder only]

open AI (chat gpt) [decoder only]

→ Born trained on huge dataset

→ Tasks that they were able to DO. ⇒
(1) text Summarization
(2) question / Answering
(3) Sentiment Analysis