(AWFERA)

Course

Deep Learning

Lecture 1 + 2

Topic :-

(Overview) + (Application)

"Study of "neurals network" is called "Deep learning."

- Tunning of neural networks
- Optimization of models
- Techniques of learning of neural networks.

→ Different Types of Architecture of NN.

- Images (CNN)
- Sequence Data like Stock Exchang. weather (RNN)

→ Auto_Encoders.
→ Transformers.

What is Learning?

The process through which system or individuals improve performers or knowledge over Time base on experience.

بہت سارا Data جس سے ہم گزرتے ہیں
ماحول حالات واقعات کو Learning کہتے ہیں۔

Human learning	Machine learning
Biological brain →	Digital brain

بہت سارے انگوٹھے سے سیکھا گیا سوچنے کا اور عمل کرنے کا۔

Thinking → Perception → Action

AI:-

When machine (like (Computer, Robots) are programmed to think and make decision like humans.

→ Rule base System, Expert.
→ based Systems, Data Learn
→ Algorithms.

Sub field :-

"ML" learning focused from previous Data.

"DL" is subset of ML which learns from neural networks or Diff Types of neural networks architecture.

→ (s 1960) exist.
→ (> 1970)

Application and factors :-

Popular in different fields like

→ Financial.
→ Health Care
→ Law and others

Factors :-

* Data Availability (Tera bytes (Generate))
* Computational Powers
* Algorithm Advancements
  ↓ neural networks
  (RRN, AutoEncoders, Transformers GPT combined like, Bard)

- Super-Computer's
- Multicore-processing
- GPUs avaible.

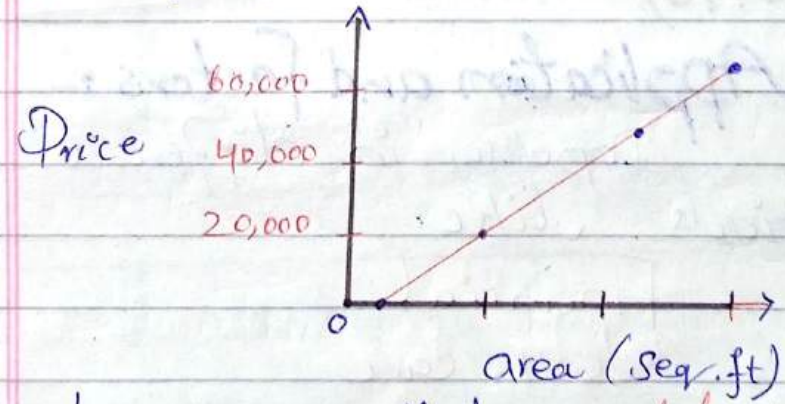
★ ↓ Pure-Men is called Super-Computer. [Must fast speed of neural networks]

Example:-

"House-Prize" prediction is an example of [Given by Area and Price.

Independ variable → x-axis = ~~Area~~

depend variable → y-axis = Price. = Predicted



Stright Line is called Models / Neurons

Neurons:-

A "neuron" in (AI) is like a simple math function. It take inputs (data) apply math function and produce (output).

Input → Processing → output

→ Math fun. $f(x) = x^2$

→ For Multiple features :-
like: Locality , facilities
↓ ↓
Location Schools, hospital
Shopping mall

Multiple features → Better Prediction

From Simple To Complex features Neural Networks. (Automic learning)

Simple To Complex features Combined Together.

"Automatic learning" "layers hidden Using in this condition"

Using of DL :- (Neural Networks)
In ML Paradiam.

→	Supervised	Labled (input / output)	Prediction
→	Un-Supervised	UnLabeled	Patterns, clusters
→	Self-Supervised	Raw-Data	Improved-Understanding

★ **Application of DL :-**

→ Spam deletion, → Self driving (Tesla) - Cars, → face (Recognization), Seepch Airports. glass of blind person.

healthcare ⇒ [dignatics, drug disease] [Translation apps]

→ NLP (Siri, Google Assistant, Alexa) ↓
Chatbots,

→ AI in Arts ⇒ Deep fake

(Lectures # 3+4)

Topics:-

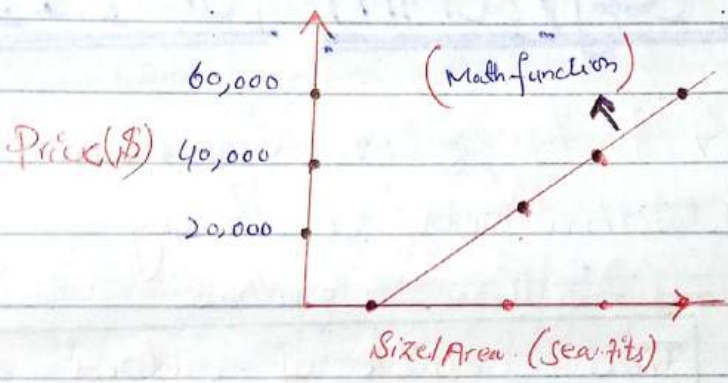
Neural Network's

→ Example : House Price Prediction

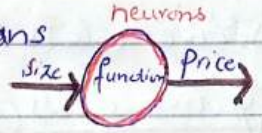
Suppose:- (Single features)

- 6 House data Set
- Input (Size), output = Price

→ how To Calculat house Price using house Size? [Linear Regression] → ML Algorithm



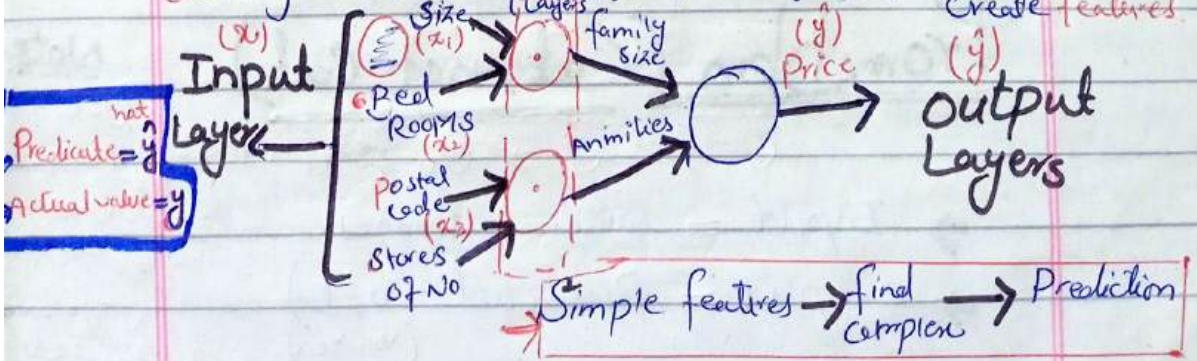
→ Simple neurons is a function.



→ Take any (x) values from Input,
→ output Predicated values. (y).

For Multiple features:- [SNN] standalone

Using many neurons arrange (Layers) find complex create features



A neuron can make one neural networks and Stack of different Layers is called "Deep Neural Networks"

Logistic Regression Algorithm for NN

A Logistic Regression Algorithm is a Binary classification algorithms

Examples
For Images cat, non cat, Spam detection Medical diagnosis

Two possible of output
- Positive class (1) = Target category
- Negative class (0) = Everything else

(Input output) pairs = (x, y)

where: x = Input, y = actual output

Problem (Cat, not cat)

Note:

→ Input = Given Images
→ output = Predicted (0, 1)
(No, Yes)

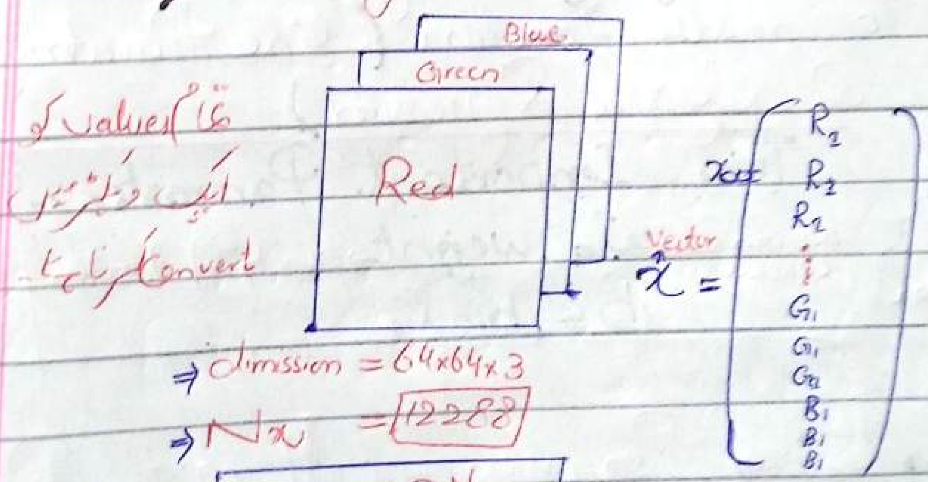
3 Channel Using Image: (8)

- Divided Images into three channels
- → (Red, Green, Blue)
- → Multiple pixels Create a Vector and then pass to the Logistic Regression models (0,1)

- کسی بھی ماڈل کو ٹرین کرنے کیلئے بہت سارے Samples چاہیے ہوتے ہیں آخر میں سب Samples پر ماڈل کو ٹرین کرتے ہیں ۔

→ ('M' Sample) Num of training examples / Samples.

→ 'x' Images has three channels



⇒ dimension = 64x64x3

⇒ N_x = 12288

Math form: $x \in \mathbb{R}^{N_x}$

★ Logistic Regression In Math form:—

اس الگورتھم کی ورکنگ یہ ہے کہ Input دیتا ہے x (image) اور ŷ value بتانا ہے

$x \longrightarrow \hat{y}$ $\hat{y}$ hat value {0, 1}

→ x = input

→ $\hat{y}$ = predicted

→ $P(y=1 \mid x)$ Probability

→ $y=1$ (+ve) class

→ x = Input value

★ Defin Parameters :-

Calculate $\hat{y}$ using 'x'.

→ What is Probability $P(y=1)$ given 'x' being a "cat picture"?

→ Probability of Space value [0,1]

→ Logistic Regression give value b/w them

→ $(x \rightarrow \hat{y})$ 'x' is a vector

• which is not control. So we have Take 'x' as Input and Performed (maths function) and convert {0,1} Edding Some values To math function (like Parameters of Logistic Regression)

Two Important Parameters:-

'w' values can Min, Max constant

We can value control

'w' = weight → branch of function

b = bias

→ **Linear Regression**

→ $y = wx + b$

Two operation Using in this case → Multiplication → Addition

$x \rightarrow (\times w) \rightarrow [+b] \rightarrow y$

input, weight, bias, output

well defined operation for vectors

Logistic Regression

$y = \sigma(wx + b)$

Applies Sigmoid function

→ $\sigma(z) = \frac{1}{1+e^{-z}}$

Example:-

Suppose:-

vector $x = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$ → Logistic Regression → $\hat{y} = w^T x + b$ ——— (A)

weights:

$w = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$, $w^T = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Transpose of w (change Rows into column)

① Using equation (A)

$\hat{y} = w^T x + b$

$\hat{y} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} + (-1)$ ∴ b = (-1)

$\hat{y} = (2 \times 1) + (2 \times 1) + (1 \times 3) + (-1)$

$= 2 + 2 + 3 - 1$

$\hat{y} = 6$

{+∞, -∞} = we just {0, 1} value

(-ve, +ve)

→ Linear Regression Convert to Logistic Regression Called "Sigmoid function" Take a any numbers and convert into {0, 1} numbers.

→ **Sigmoid function = $\hat{y} = \sigma(w^T x + b)$**

where,

→ $\sigma = \frac{1}{1 + e^{-z}}$ → $z = w^T x + b$

∴ $-\infty \leq z \leq \infty$

Case 1 :-

→ $z = w^T x + b = 0$

Using equation

→ $\sigma = \frac{1}{1+e^{-z}}$

$= \frac{1}{1+e^{(-0)}}$

$\sigma = \frac{1}{1+1}$ ∴ $e^0 = 1$

→ $\sigma = \frac{1}{2} = 0.5$

Case 2 :-

→ Larg positive Num

→ Using equation

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

$= \frac{1}{1+e^{-(\text{larg Num})}}$

$= \frac{1}{1+0}$

$\sigma = 1$

∴ $e^{-(\text{larg Num})}$

∴ $= \frac{1}{e^{\text{larg Num}}}$

∴ $\frac{1}{\text{larg Number}}$

approximation '0'

Case 3 :-

'z' is very larg Negative Numbers

→ $\sigma = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-[\text{larg -ve Num}]}}$

$= \frac{1}{1+e^{-1[-(\text{larg +ve Num})]}}$

For example) اگر (larg -ve) ہے جیسے -5 ہے → [-5M, -1(5M)]

$= \frac{1}{1+e^{+(\text{larg Num})}}$ ∴ $e^{\text{larg Num}}$ is very larg number = 0

$\sigma = \frac{1}{1+\text{large Number}}$

→ $= 0$

Hence answer in Probility Space

{Lectures # 5+6}

★ Loss Function

How Logistic Regression Train & What Means?

ہم 'w' اور 'b' کی ایسی ویلیوز
کو find کرنا چاہتے ہیں کہ جب 'x' سے multiply
کریں تو جو Answer آئے وہ اس تصویر کا
$\hat{y} \sim 1, \hat{y}=1$ — Lable Represent کرے۔
ایسا weight ہونا چاہیے جو 'x' Image کے پکسل کے
excet = 1, 0.9, 0.8 — vector سے (xby) کریں تو
جواب آئے۔

- To purpos To find out the value Actual Images lables.

Example :-

→ Cat Images x by Trained weight
= 1

→ Non-Cat Images Vector X, Trained weight
= '0, or near To '0'

Training :-

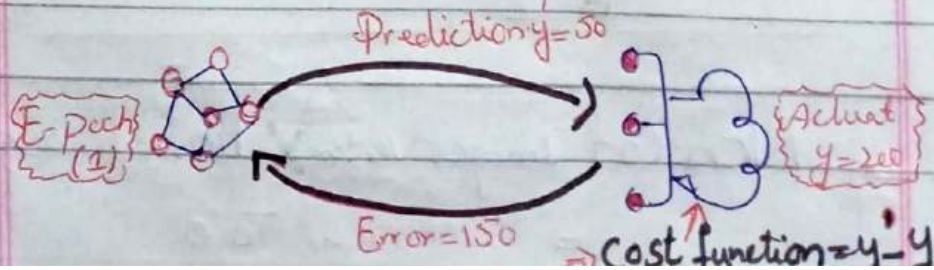
- In Neural Networks To find out such type of Input to (Convert) → output (Labled) Transform Successfuly. (Parameters) like weight, Baise.

→ $\hat{y}$ = Predicated Lable.
→ y = Actual Lable
Given → Ideal change $\hat{y} = y$

→ Given Cat Image $1 = y$ Lable. and $\hat{y}$ to find equation given by value. "Using Special function To find weight and Diase. distance."

Loss function :- (Cost)

- This function tell us how much distance b/w y and $\hat{y}$ is called "Cost" function.



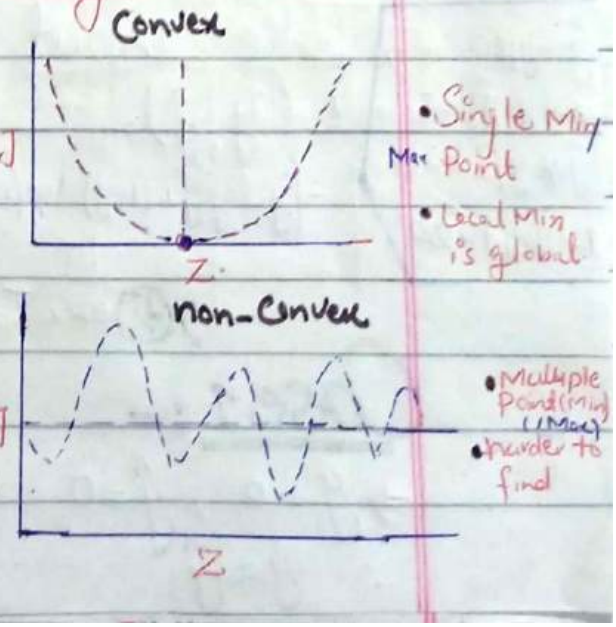
Distance / Loss function

- It is given by distance by y and $\hat{y}$. To find out the weight and bias whose value is "Minimum value".
  - Take Input $(y, \hat{y})$
  - Given output (Some Nums)
    - '0' = Min distance
    - '1' = Max distance
- Alogrithm is not Proper training. Special Category.

Convex - Non Convex function

- Distance blw y and $\hat{y}$
- To choose a special type of function is Convex function
  - y = Actual
  - $\hat{y}$ = Predicated
  - σ = Distance b/w $\hat{y}$ & y

Using $z = w^T x + b$



Notes:-

To Neural Networks training mostly function is used by ("Convex function").

→ Optimin point (Min, Max) both point use.

→ Single (one point) is Globle Minima.

→ "non Convex" function use multiple point (Local Globle Minima).

★ Logistic Regression Using function "Negative Log-Loss function:-

Note:-

$\hat{y} \approx y$ (2 Condition)
$\Rightarrow L(y, \hat{y}) = 0$
$\hat{y} \neq y$ (2-Condition)
$\Rightarrow L(y, \hat{y}) = \infty$

Take Input

$$L(y, \hat{y}) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})] \quad \text{--- (A) equation}$$

Putting values in eq (A)

$$= -[0 \log(0) + (1-0) \log(1-0)]$$
$$= -[0 + (1) \log(1)]$$
$$= -[1 \log(1)]$$
$$= -[1(0)]$$
$$= 0$$

$\because 10^0 = 1$
$\because \log_{10} 1 = 0$

Case 1:-

→ $y = 0, \hat{y} = 0$
$\therefore y \approx \hat{y}$

Case 2:-

→ When $y \neq \hat{y}$

$y = 1$, $\hat{y} = 0.1$

→ Putting values in equation (*)

$$= -[1 \log(0.1) + (1-1)\log(1-1)]$$

$$= -\left[1 \log_{10} 10^{-1} + (0)\log(0)\right]$$

→ ($\log_{10} 10^{-1}$) (-1) = base and value is same

$$= -[1(-1) + 0]$$

$$= -[-1]$$

→ $\boxed{L(y, \hat{y}) = [1]}$

① Its also called Cross-Entropy loss, binary cross-Entropy

→ To Teel its distance b/w them y and $\hat{y}$ (To improve) 'b' & 'w' show.

→ To best Training Min value

② This function is defined for one value (Image) Training set for (1 picture)

Loss Function :-

Act as a mathematical distance function that minimize the Error b/w predicated & Actual Lables To Enhance model Performance.

Case :- 'Nth' value Sample

Examples / Images. $m \approx 100$

- Collective multiple images for Training.

For 'm' Sample :-

- Adding all 'loss' of images and then Take Average.

$\hat{y} \approx y$ Using Randomly selected by (w, b)

m Sample = Add $L(y, \hat{y})$ for all

Cost function Represent by 'J'

$$\Rightarrow J(w) = \frac{\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)})}{M}$$

∴ Total Loss of Training set Sigmoid

$$\rightarrow \hat{y} = \sigma(w^T x + b)$$

cost

$$J(w, b) = -\frac{1}{M}\left[y \log \hat{y} + (1-y) \log(1-\hat{y})\right]$$

$$\rightarrow -\frac{1}{M}\left\{y \log\left(\sigma(w^T x + b)\right) + (1-y) \log\left\{1 - \sigma(w^T x + b)\right\}\right\}$$

Gradient Descent

- In this lecuture we have study to optimize our parameters like (w, b).

→ $y \approx \hat{y}$ So called "trained the Models."

ایسے find کیا ہے جب بھی Images کے طور پر Input دیے تو output کے طور پر Actual ویلیوز مل جائے۔

Asume:-

- We just using only one weights for neural networks To Train it for. Futhermore To Extand it for multiple weights, biase.

★ Meaning of this Algorithms

Difficult Sounding Name

- **Gradient** → Slop, Derivative → "Rate of change"

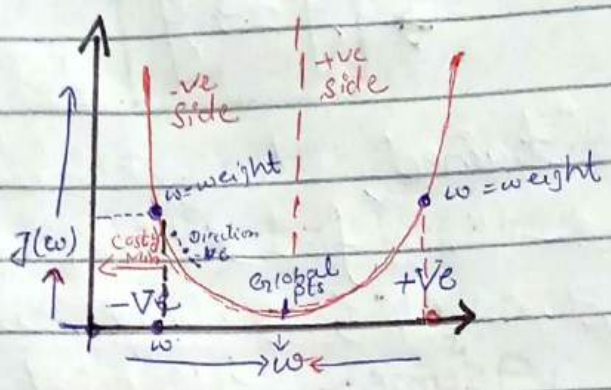
If we change Indepent variable 'dx' to Calculate how much changing in depended variable 'dy'

- **Descent** → (Moving down words) (کسی چیز کو نیچے آنا)
  - oppasitive words ascent (اوپر جانا)

اس میں ہم gradient کی بنیاد پر descent کرتے ہے کی ویلیوز (weight) کو طرف جس پر ہمارا (loss) کم ہوتا۔

Diagram :-

- We have Calculate $J(w)$ → Cost function of all training Samples.



→ w = (Updated as) w − (Change in Loss per Unit chang in w)

→ $$w = w - \frac{dJ(w)}{}$$

$\therefore dJ(w) = 0$

→ $w = w$ Updated value two (to) three times so the gradient descent Algorithms is **Converg.** (so weight trained)

Case :- ● To choose a Random value of 'w'.

● 'w' is optimize neural networks Train (Means last Minimum)

$y^{(i)} \simeq y^{(n)}$ for all Samples

$w = w$ these is not changed using Gradient descent

Case (1):- [Negative Side]

Suppose we choose 'w' randomly

$\Rightarrow \frac{dJ(w)}{dw} < 0$ (-ve Gradient)

- when we change small w slop is negative

$$w = w - \frac{dJ(w)}{dw}$$

$$= w - (-ve \text{ negative Num})$$

$$\rightarrow w = w + \text{Num Num}$$

- Some w value change (0.1)

$w = w \rightarrow$ (-ve number)

$= w + (0.1)$

$\rightarrow w = w + 0.1$

When w = Minimum = 0

$w = w - [-0]$

$\rightarrow w = w$ enshow loss

- minimized when w unchanged

$\rightarrow J(w) \approx 0$

Case 2:- (Positive Side)

- Suppose we choose w Randomly

$\Rightarrow \frac{dJ(w)}{dw} > 0$ (+ve slop)

- Unit Cost J. with.r.t Unit chang in w.

$$\rightarrow w = w - \frac{dJ(w}{dw} \longrightarrow (*)$$

$\rightarrow w = w - (+ve \text{ num}) \Rightarrow w = w - (+ve)$

Exampl... Mi... 5 ... = 4

→ Chang cost positive still same value hold so w' is more small.

→ $w = w$

→ Similarly moving w small pts near to Coneng point (Min)

$w = w - (\text{very small num})$

≈ 0

→ $w = w - 0$ cost is Minimum

- Loss is given dist
- Gradient Scend give
- Unit chang
- 'w' & 'b' optimize

- Same things is use for multiple 'w' and 'b' biase.

from equation (*)

→ $= w - \alpha \frac{dJ(w)}{dw}$

"α" is a "learning Rate"

we cannot control from 'Data'.

how much Jumbed ('w' changing)

→ "Learning Rat 'α'" = Gradient descent Rate of change of descent control

→ Jumming control value never miss any value Sequence Jummbing To control very Samall Num

X by 'w' adjust (0.2) up To Soom (Changed PTOP)

Too Low	Too high	optimal
Slow progress and Time to concergence	overshooting failure To Converg	Balance steps and smooth Convergen To Min

(Lectures No # 7+8)

Topic:- Part-1

Forward & Back-Woard Propagation

★ Important Concepts:-

Which methods we used in NN To Calculate and compulation in Groups.

Divided into Phyase:-

→ Forward Propagation.

→ BackWard Propagation.

⊛ Forward Propagation

To find inference & loss

⊛ The Process of passing input values through a Computational Graph To obtain an output.

Commonly used in NN To calculate predication.

we have find loss in predicted and actual values in NN.

Math forms:-

- The function we are analyzing

→ $J(a,b,c) = 3(a+bc)$

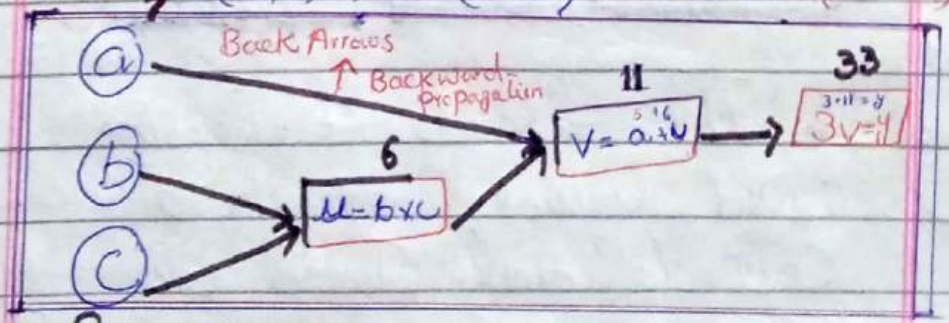
- This function involves basic arithemtic operations and can be Broken into multiples steps for Computation.

→ Simple Equation Examples:-

we using computational graphs nodes

Input Parameter

→ $J(a,b,c) = 3(a+bc)$ ∴ $b \cdot c = u$, $(a+b \cdot c) = v$



Suppose:-

We have Suppose Random values

$a = 5$, $B = 3$, $C = 2$

Back-Ward Propagation.

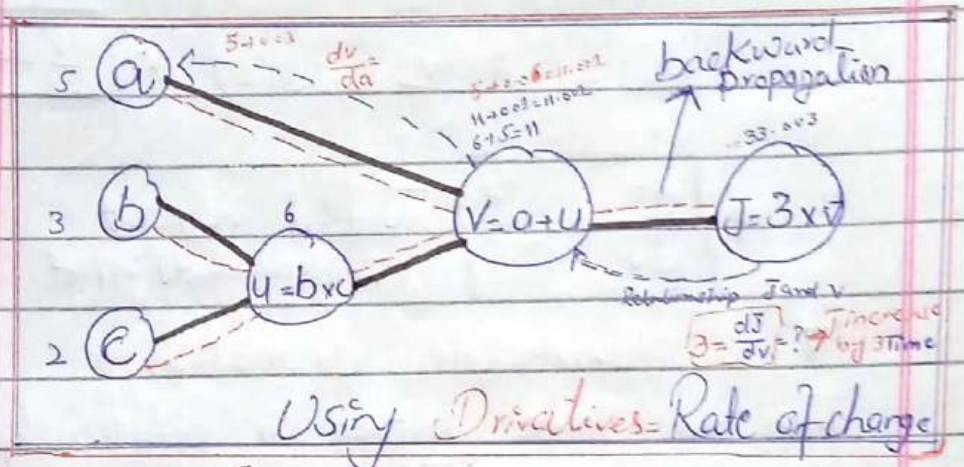
(24)

How finds changes in Input affects the loss. (Input Var & output variable)

Goals: Adjust Parameters to minimize the loss.

Here read dashed rows represent backpropagation.

To find loss and Relationship b/w two use this loss to minimize



Using Drivatives = Rate of change

Case 1:-

Suppose:- We have find Relationship if 'a' any variable unite chang then affect in 'J'.

- In basic mathematic that Relationships has find "derivatio" $\left(\frac{dJ}{da}=?\right)$ Independent variable 'a' ko change krna sey depend Variable J mai kia farq ata hain.

Note:- $\frac{dJ}{da}$ = Unite chang independent variable mai krna or dependent Variable mai kia Asar hota hy.

→ So we have $\frac{dJ}{dv} = 3$ Time

since when $v = 11$ $\quad 3(v) = J$

⇒ Unite chang $v = 11 + 0.01 \Rightarrow 3(11.001) = J$

$\Rightarrow = 11.001 \Rightarrow 33.001 = J$

→ Its means 'V' unite changes then J 3 Time change.

Now Taking derivatives

$$\frac{dJ}{dV} = 3V$$

$$\frac{dJ}{dV} = \frac{d}{dV}(3V)$$

$$\frac{dJ}{dV} = 3\frac{dv}{dV} \Rightarrow = 3(1) \quad \because \boxed{\frac{dv}{dv} = 1}$$

→ $\boxed{\frac{dJ}{dV} = 3}$ To find inderdent v Ta dependent J affect.

"V" is intermediate calculation.

Then we have intrest in Relationship b/w 'a' w.r.t 'J'.

Case:2

→ We have find 'a' independ variabe unite chang than value of changed in 'v' affect. $\left(\frac{dv}{da} = ?\right)$

→ Suppose $a = 5 + 0.01 = 5.001$

→ $V = a + u \Rightarrow V = 5.001 + 6 = \boxed{11.001}$

→ Unit change main ithe hi Value change agi gi ithe change 'a' main ki thi.

Na nahi mallipie (3, 4, 5) sey hota hy

$$\therefore \frac{dv}{da} = 1$$

→ $\frac{dv}{da} = \frac{d}{da}(a+u) \Rightarrow \frac{dv}{da} = \frac{da}{da} + \frac{du}{da}$ ∵ constant $\frac{du}{da} = 0$

→ $\boxed{\frac{dv}{da} = 1 + 0}$

Case 3:-

→ find Relational 'a' and 'J'

- We have find values using Chain Roles.

→ $$\boxed{\frac{dJ}{da} = \frac{dJ}{dv} \cdot \frac{dv}{da}}$$

putting values.

→ $\frac{dJ}{da} = 3 \cdot 1 = \boxed{3}$ see in graph.

- Unit change 'a' Resultant in 3unit of 'J'

'a' is weight and J is cost function To minimize it.

cost function when Changed 'w'

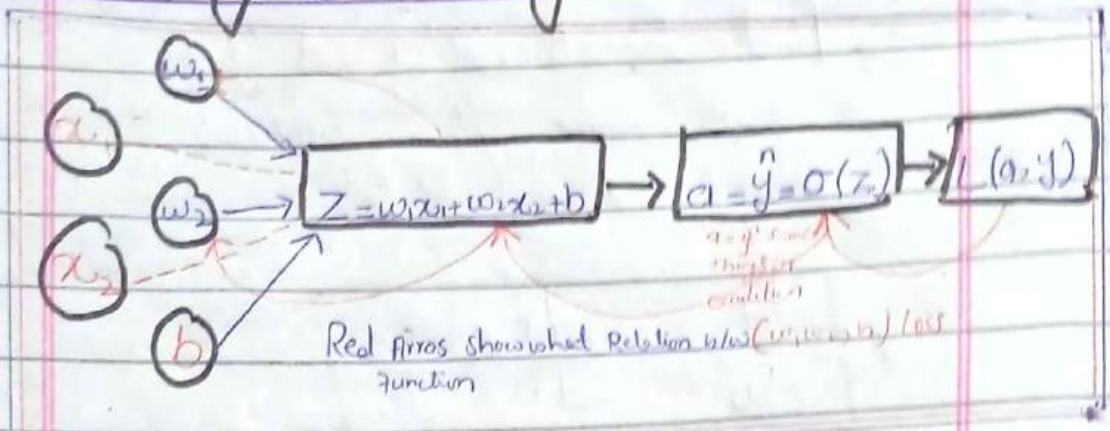
- aK Relation $\boxed{\frac{dJ}{dw^2} = ?}$ Asa button/function aajake has loss num Kam Sey Kam Krsektay. changing in 'w' update (Tune) Loss has minimize the Regic Regression Trained we have $\boxed{y \approx \hat{y}}$ find it some weight To models predicted.

Video Part 2:-

BackWard & Forward

Propagation

Logistic Regression Graph:-



- We have Calculate in Logistic Regression :-

$$z = w^T x + b$$

w is vector we written

$$\hat{y} = \sigma(z) \rightarrow \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

So $\boxed{z = w_1x_1 + w_2x_2 + b}$

Note:-
- control
- w_1, w_2 = weight not control
- x_1, x_2 = pixel pic
- b = biase
- Three Para used optimize To loss Minimize

$\boxed{y \approx \hat{y}}$

(propg- (forward) infere / Predicted value ($\hat{y}$) → loss actual value diff distance)

Case 1:- Loss function to Relation $\boxed{\hat{y} = a}$? in backward Propagation

$$\boxed{\frac{d}{da} L(a, y) = -\frac{y}{a} \frac{(1-y)}{(-a)}}$$

'a' Unit chang then' quantities

$\left[\frac{-y}{a} \frac{(1-y)}{(1-a)}\right]$.

Case 2:-

We have To calculate unite change of z then how much change in 'a', $\hat{y}$ hat

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

Calculation of Math provided by notes

clear concepts

Case 3:-

How have To Calcule Relationship blw z and w.

→ $\frac{dz}{dw_1} = x_1 \;,\; \frac{dz}{dw_2} = x_2$

Now:-

Now we Calcule Relationship blw 'w' and Loss (in backward Propagation in Logistic Regression) then we have Super power means how much change ('w') weight and loss increase or decrease (Min, Max) then Nural Network Trained.

Using Chain Role:-

→ $= \frac{dz}{dw_1} \cdot \frac{da}{dz} \cdot \frac{dL}{da}$

→ $= x_1(a-y)$

This show that w_1 chang (0.02) then loss change $x_1(a-y)$ To weight better and loss (minimize)

$\hat{y}^{(n)} \approx y^{(n)}$ We Calculate Traned NN.

Similarly this Calculation we doing for (w_2) and Biase (b)

* we have Read Gradient Dscent Algorithms.

→ $$w_1 = w_1 - \alpha \frac{dL}{dw_1}$$

Apply gradient Dscent iterative and find (w1 weight)

Backward propagation is used To updata weights.

Important Points:-

→ When we have Trained NN using both Technique like forward and backward propagation

→ But when we have Trained NN there is no need To ~~forward~~ / backward Progagation. Just (Modely deploy) using Forward Progagation and find predicted variable. (inference).

Standard Fully Connect NN :-

* Fist NN which has diff

given break throughs. This NN artichelure (s1918) Yan Yakun introduce it. We Read all basic Concepts in Previous Lectures. with using examples.

Content:-

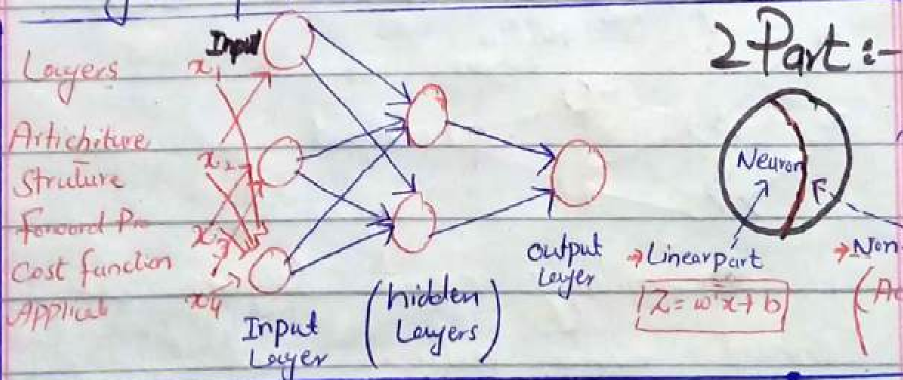
fully Connected Neural Networks (Dense) Consists of following.

- Layers
- Activation Function
- Structure/Artiechiture
- Forward Propagation
- Cost/Loss function
- Application

Example:- Important things different

Layers of Neurons.

DL میں (Linear ایک Neuron کا ایک حصہ ہے۔ اسلئے بعض دفعہ کہی کہتے ہیں



- **Input Layer:-**
  - Asscepts Raw features
- **Hidden Layers:-**
  - More then one hidden Layers in NN.
  - جتنی زیادہ hidden لیئرز زیادہ ہوگی اتنا (Complex) pattern کو find کرتا ہے۔
- **out-put Layer**
  - depending upon which type of Problem has solved Select Activation
  - Predicted

Algrithms	Activation
Binary classification	Sigmoid/Tanh
Regression Alogrithms (Continous value Score,)	ReLU/Varant
	Softmax Groups (Multi-class) programs

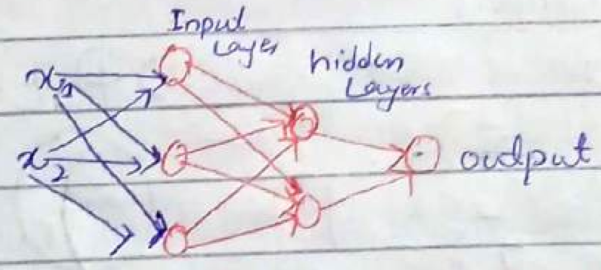
جب بائنری میں hidden Layers میں ایک کی Specification ہے تو ہمارے آخر میں (ReLU) ویزنز لیں کے تو استعمال کریں گے

Similarly:-

- Some Case in Input Layers we have used (ReLU) activation Primilarly Start layers used and output depending upon problems Relevant activation function.

Structure:-

- It is also called (fully-Connected neural Networks) or (dense NN) (کے Layers)
- ہر ایک اپنے Neurons کے ساتھ Connected ہوگی۔

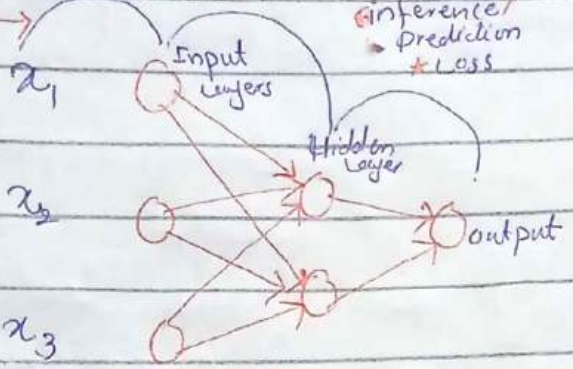


- Similarly next layers us Connect every neuron and Same in next layers is Conneted every neurons of layers.

Forward Propagation :-

If we moving in forward way is called forward calculate output and loss function

Forward Propa →



Inference = Prediction + Loss

Problem = Loss
binary = -ve log
Regression = MSE loss

Equation forms :-

hidden Layer Equation

→ $z^{(2)} = w^{(2)} \cdot a^{(1-2)} + b^{(2)}$

→ $a^2 = g(z^{(2)})$ ∴ g = Activation function

Note. where,

dimensions ⇒ $W^{(2)} \in \mathbb{R}$: (hidden neuron × input features) is the weight matrix for layer 2

3 × 3 Matrix Rows col.

$A^{(1-2)}$ is : the activation output from

$b^{(2)}$: the previous Layers

$b^{(2)}$: is the biase

$g[z]^2$: is the activation

Rows × colomn Structe Data → Applied (SNN)

Application :- Examples

Computer Vision problem :-

- Hand written digits ki pics hoti hy or classifer krtay hy digits (0 To 9) kon so digits

→ Real Estate Analytics :-

- House Price predition

→ Speech Recognition

→ NLP (Sentiment +ve, -ve)

→ financel forard, Storg detecting

(Lecture No#09)

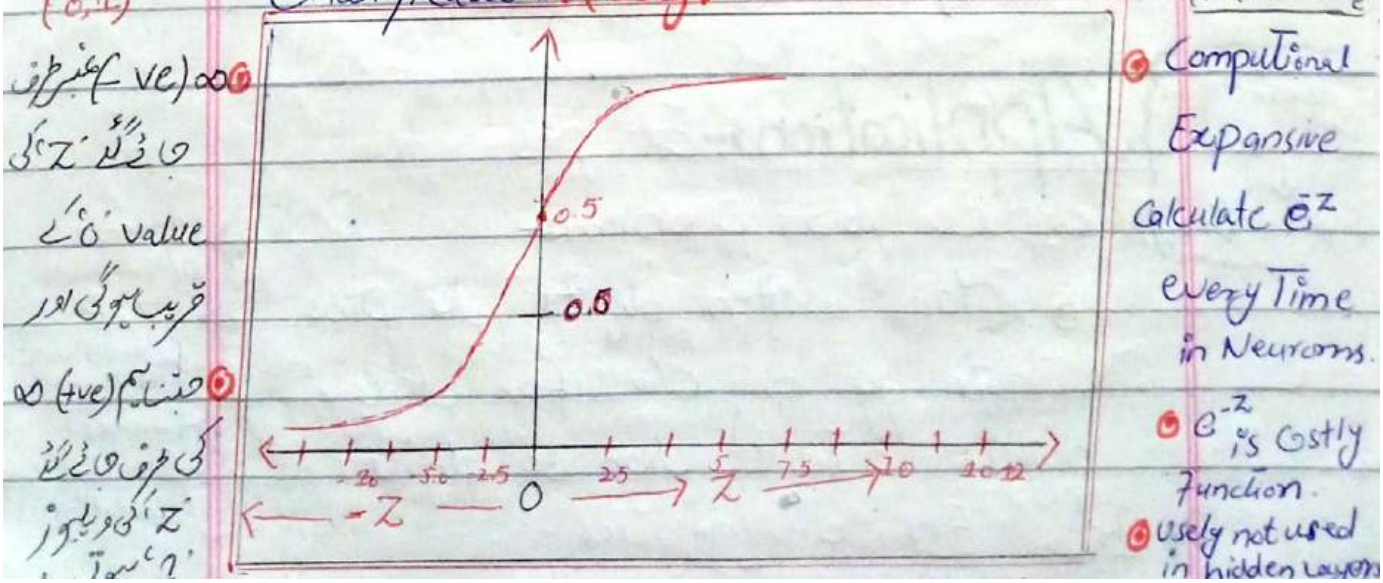
Topic:- Key- Activation Function

- The most popular activation function is as:
  - Sigmoid → Tanh
  - ReLU → SoftMax.
  - ReLU Variants
    - Leaky ReLU
    - Parameteric
    - Exponential ReLU(ELU)

Sigmoid Activation Function:-

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

- Sigmoid maps any input value **z** To Rang **b/w 0 and 1**, which is particularly useful for output probabilities in binary Classification ← (usage)



Main Issue

- Computational Expansive
- Calculate e^{z} every Time in Neurons.
- e^{-z} is costly function.
- Usely not used in hidden layers

Main Issues:- (Cons)

- very Expensive for Computational power
- Non-Zero Centralized output
- It mean z value = 0 then output value = 0 (both 'z' but not Sigmoid)

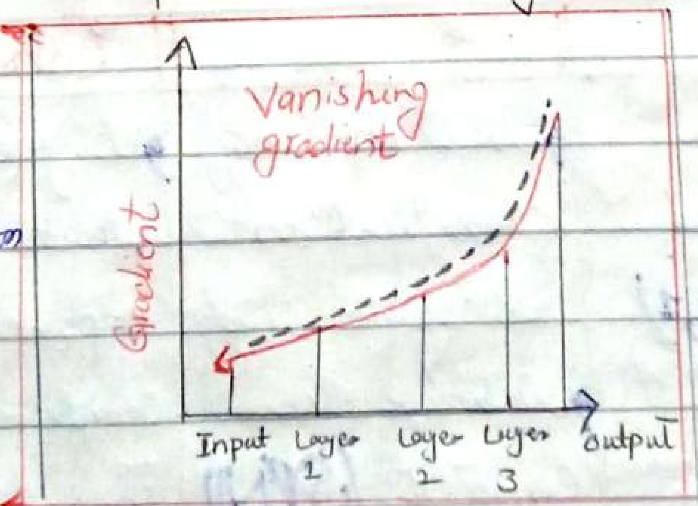
Is activation function mai ye sab ho To wo wala neural network jaldi converge hojata hy. (Gradient descent)

- Always used in output layer not used in hidden Layers

Useage in Binary Classification.

Note:- Particularly Problem Create: (Vanishing Gradient) A network learning issue where deeper layers struggle To update weight due To exponentially shrinking gradients (Mostly used when Multiple hidden Layers).

Z کی ویلیوز کے لیے یہ فنکشن "Stright line" ہے اس لیے derivative صفر ہوتا ہے جس کا مطلب ہے backpropagation میں
$w = w - \alpha \frac{d}{dz}$
$w = w$ not updated by neuron

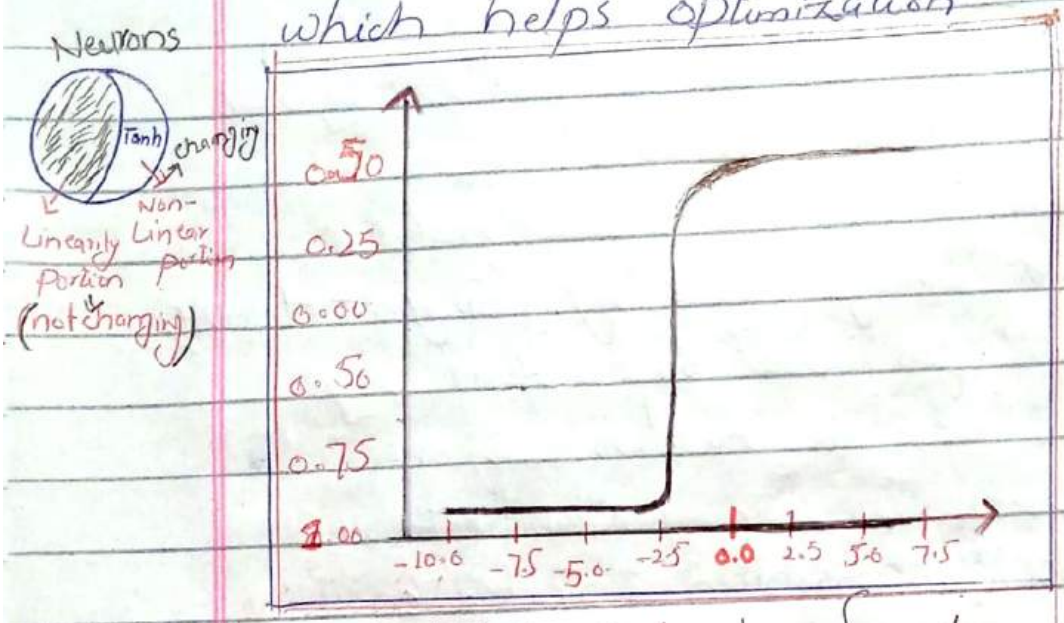


It is faced by Vanishing gradient. ∴ So not used in hidden Layers.

Vanshing Gradient

Tanh Activation Function:-

- Is a Scaled version of Sigmoid Providing outputs in the Range of (-1, +1) making Zero-centred which helps optimization



Tanh Activation function

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

Also: → $2\sigma(2z) - 1$ is very Similarly To Sigmoid but some changing

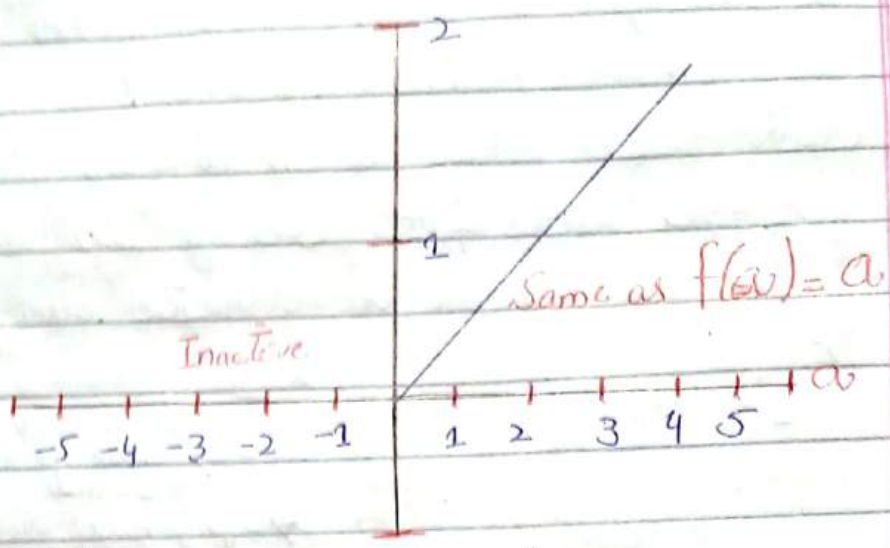
Pros:-		Cons:-
→ It is a Zero-centred → Its Rang (-1, 1) output. any problem which is show in (-ve, +ve) Class Then output, Layer used (Tanh)	→ Used in binary classification we have study → Machin learning Concepts (SFN) loss.	→ Using (e^{-z}) exponent which is very expensive for Computonal power in output layers → Some Recurrent hidden layers used it. → Vanishing Gradient

ReLU (Rectified Linear Unit)

- This most popular contemporary used activation function in (NN) (mostly used nowadays)

Rectified Linear Unit

→ ReLU(a) = max(0, a)



-ve value = 0
+ve value = As written as

If we give input as a neurons then give output as same as Input.

(ReLU) output the input value of Z for all (+ve) values, and zero ('0') for all (-ve) negative values. It's widely used in hidden Layers To avoid the vanishing gradient problem.

Cons:-

Died (Dying ReLU problem)

Neurons → 'on die' if input ve/value → Then neuron o/p no learning

Pros :-

→ It is more efficient in compulional power

→ Just compare number of both max find

→ no calculation deactived on (-ve) numbers (Neural dropout Provided)

→ use in "hidden Layers" contemporary

→ (1000) Layers present easily used (ReLU) function (NN complex).

→ no vanishing gradient

RELU Variactions:-

Leaky ReLU:-

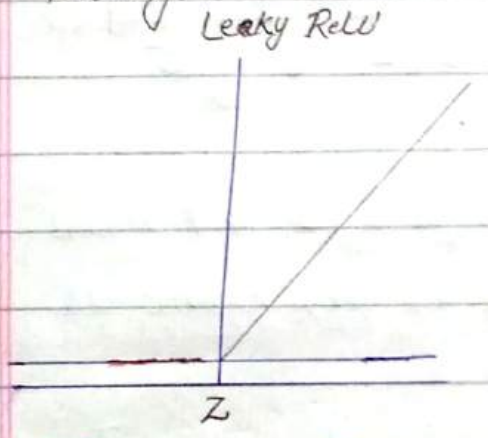
$$g(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases}$$

Point: $z > 0$ All variation changed To (-ve) Portion and (+ve) as it's as. for all. not changed

→ Leaky ReLU$(z) = \max(\alpha z, z)$

we mulliple by (-ve) numbers with very small number $\alpha = 0.01$

Leak ReLU allows a small -ve slop for -ve values of 'z' preventing "dying ReLU" Problem where neurons become inactive during Training.



★ Example = -10
ReLU Return = 0
$-10 \rightarrow -10 \times 0.01 = -0.1$
⇒ dying ReLU problem Solved.

Cons: Speech, Images, Language

The value of α is fixed to $(0.1) \Rightarrow \alpha = -0.01$

Parametric ReLU:-

Value is also a "Parametric" w; b learn data by itself by α = Data learned

is a Similar To Leaky ReLU, but instead of using a fixed value for α, it introduce a learnable parametric for (-ve) input.

$$g(z) = \begin{cases} z & z > 0 \\ \end{cases}$$

Exponential ReLU

It is also (-ve) number address. Using exponential function $\alpha(e^z - 1)$ $z \leq 0$ it is ensure output is '0' centred

It is exponential expensive for Training Time

ReLU Variations:-

Leaky ReLU:-

$$g(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases}$$

Leaky ReLU(z) = Max(αz, z)

Point: z > ... All variation changed To (-ve) Portion and (+) as it's as. For all not changed

we multiple by (-ve) numbers with very small number [α = 0.01]

Leaky ReLU allows a small -ve slope for -ve values of z preventing "dying ReLU" Problem when neurons become inactive during Training.

* Example: z = -10
  ReLU Return = -10 → -10 × 0.01 = -0.1
* dying ReLU problem Solved.

Cons: Speech, Images, ...

The value of α is fixed (0.01) → α = 0.01

Parametric ReLU:-

is a Similar To Leaky ReLU, but instead of using a fixed value for α, it introduce a learnable parametric for (-ve) input.

$$g(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases}$$

Exponential ReLU

* It is also (-ve) number address. Using exponential

$$g(z) = \begin{cases} z & z > 0 \\ \alpha(e^{z} - 1) & z \leq 0 \end{cases}$$

it is ... output is "0 centred"

* It is exponential expensive for Training Time

RELU Variactions-

Leaky ReLU:-

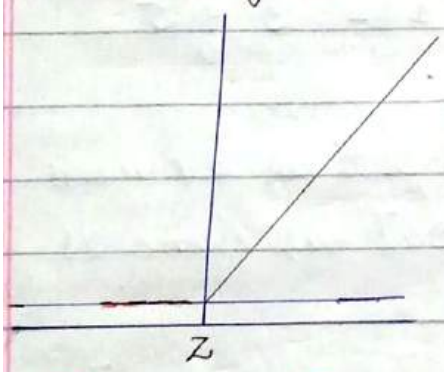
$$g(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases}$$

Leaky ReLU(z) = Max(αz, z)

Point: z > 0
All variation changed To (-ve) Portion and (+ve) as it's as. for all. not changed

we mulliple by (-ve) numbery with very small number $\alpha = 0.01$

Leak ReLU allows a small -ve slop for -ve values of 'z' preventing "dying ReLU" Problem where neurons become inactive during Training.



Example = -10
ReLU Reluran = 0
-10 → -10 × 0.01
= -0.1
⇒ dying ReLU problem Solved.

Cons: Speech, Imges, Language

The value of α is fixed ($\frac{1}{2}$) → $\alpha = -0.01$

Parametric ReLU:-

is a Similar To Leaky ReLU, but insted of using a fixed value for α, it introduce a learnable parametric for (-ve) input.

$$g(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases}$$

Exponential ReLU

It is also (-ve) number address. Using exponential function.

$$g(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases}$$

it is ensure output is 'o' centred

It is exponential expensive for Training. Time