16/08/22

Assesment Methodology :

CT1 - 20

CT2 - 20

Programming Assignment - 20 (Group of 2 or 3)

End Sem - 40

Machine Learning :

→ Training a machine to think like humans

2 phases :

* Training - used to help the system learn
* Testing - used to test if the system has learnt useful features

If machine recognizes the digit '0' as '6' or '9'
⇒ it has not learnt to distinguish 0, 6 or 0, 9
→ retrain the model with more examples.

We need to extract meaningful features from the data

Earlier → expert system (Rule based system)
we provide the rules to recognize sth
(Eg): 7 → has a horizontal and vertical line connected at 1 point

⇓

then ⌜ will also be recognized as 7
so we need to give clear & explicit rules
⇒ Difficult and complex

provided by experts

In ML, we extract the features and provide the model along with its label (corresponding output). — in case of supervised models only

I/p data → any format
(audio, video, image, text, ...).

I/p data

Training set — Testing set
(both come from the same distribution)

Can use 80-20, 70-30 or even 60-40 split (any split is fine in general)
↳ can be random split as well
input data + label ⇒ Ground truth

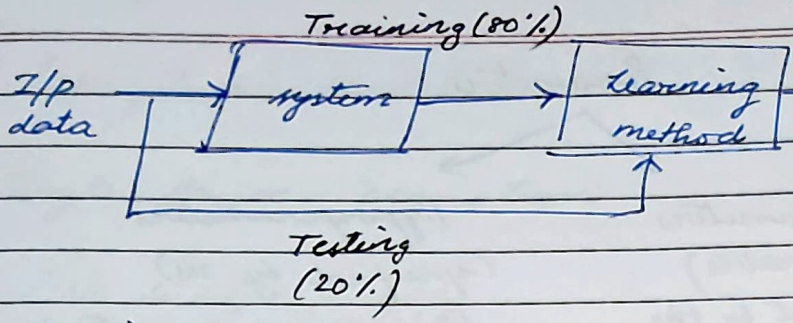
Training set

ground truth	features F_1	F_2	...	F_n	label L
observations O_1					
O_2					
⋮	⋮	⋮	⋮	⋮	⋮
O_n					

⇒ supervised learning (contains labels).

To avoid misclassification and ~~gener~~ perform well on unseen data → use diverse training set with many samples.

Testing data - unseen data for the system.

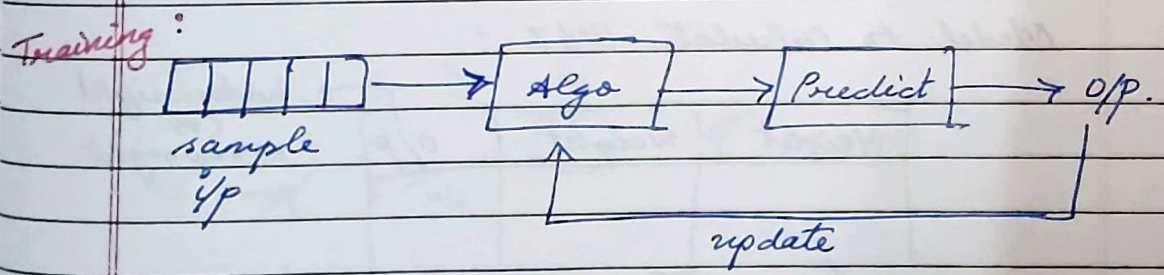


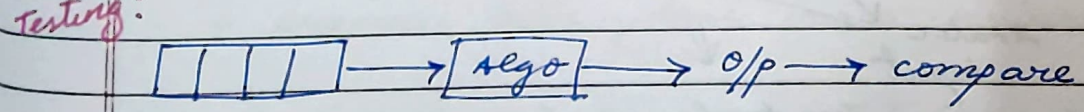
✓ While training, the system comes up with a learning method using the train features
✓ While Testing, we directly get the O/p of for the test set from the method learnt by the system

If test set accuracy is ↓, we need to change the learning method

Train set - contains both data (features) & label

Test set - contains only the data (features)





Algo → depends on the dataset and the parameters we use

Parameter

Parameters (tunable) learnt by the algo itself

Hyperparameters (specified by us)
(Eg): learning rate
#epochs
number of layers in NN
optimizer to use

Parameters → also affect accuracy

19/08/22

NN : 3 ~~layers~~ types of layers

Input (1 layer) | Hidden (≥1 layer) | Output (1 layer)

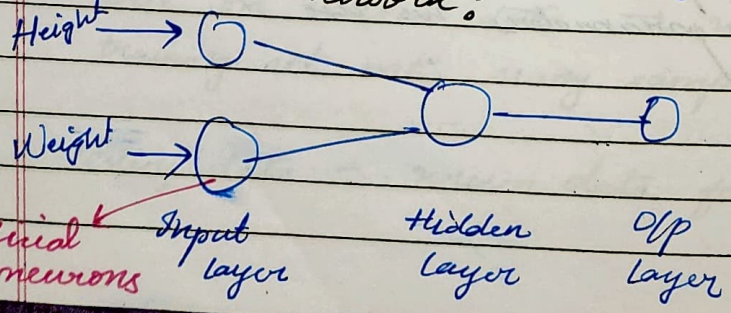
Model to calculate BMI :

Height	Weight	O/P → underweight (or) overweight

static input

Input layer has 2 neurons
→ 1 for height attribute
another for weight attribute

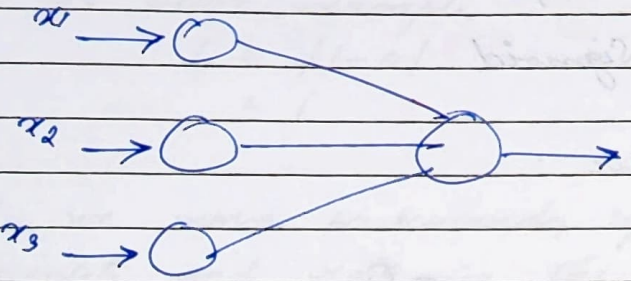
Artificial Neural Network :



Perceptron Model :

1. Single layer perceptron :

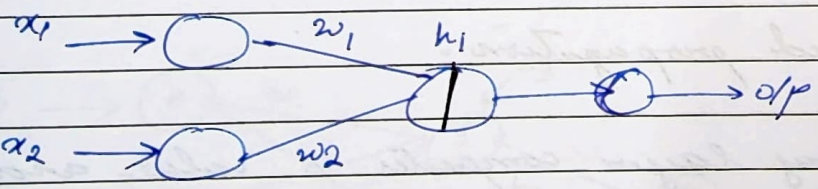
✓ No hidden layer



Weights :

Responsible for transforming the info layer by layer

Output (Input layer) = Input (hidden layer).



2 processes take place in a neuron.

① linear operation on inputs (summing the product of inputs and weights).

$h_1 = \sum_i w_i x_i = w_1 x_1 + w_2 x_2$

w_i are randomly initialized

If all $w_i = 0$ then our model always computes 0 only

⇒ To avoid this we use bias.

$\therefore h_1 = \sum w_i x_i + b$

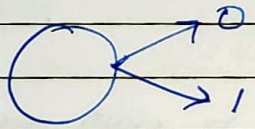
② Apply activation function

$y = f(h)$. ⇒ Brings the values in the range 0 to 1 (always!)

↓ activation fn

(Eg): ReLU, Sigmoid

Output layer:



(for binary classification).

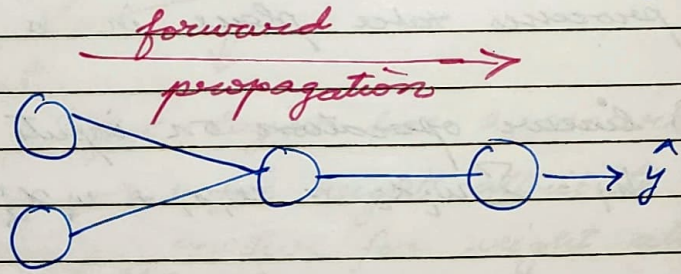
If values are $\begin{pmatrix} 0.75 \\ 0.3 \end{pmatrix}$ ⇒ class 0 is more probable

($\geq 0.5 \Rightarrow 1$

$< 0.5 \Rightarrow 0$).

Forward propagation:

✓ Every layer computes a value and forwards it to the next layer



If original label $y = 0$ but predicted label $\hat{y} = 1$ then we have to retrain the model by updating the weights (bias, inputs are constants!)

Loss function :

Calculates the loss = difference in predicted and actual value.

① $L = |\hat{y} - y|$

For above example

$L = |1 - 0|$

$= 1$ $\Rightarrow$ 100 % error !

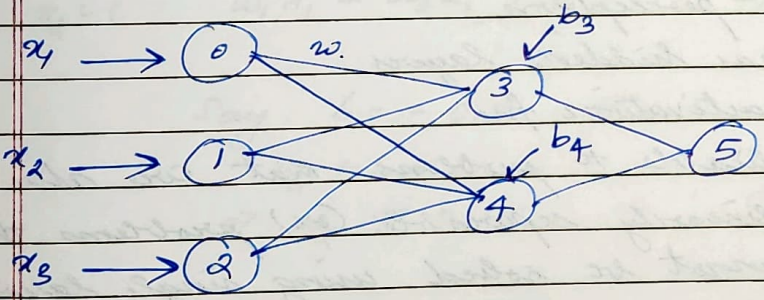
So we move backwards, update ~~at~~ the weights and retrain the model.

Optimizers :

Responsible for updating weights

(Eg): Adam, RMSprop, SGD (Stochastic Gradient Descent).

2-layer perceptron

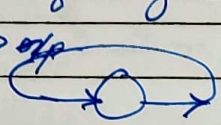


bias - associated with each node

Multilayer - lower chances of error ~~too~~ of ~~even~~

Topologies

- Complete (every node in 1 layer is connected to every node in the next layer).
- Recurrent (there are self weights ⇒ o/p)
- forward
- backward



1 forward + 1 backward prop → 1 epoch

22/08/22

ANN with >1 hidden layer → DNN

Single layer perceptron :-

<i> No hidden layer

<ii> So, no activation function

<iii> applicable to problems that are linearly separable.

Multi layer perceptron :-

<i> It has hidden layers

<ii> Has activation fns

<iii> applicable to problems that are also non-linearly separable (or) problems that cannot be solved using single layer perceptron model

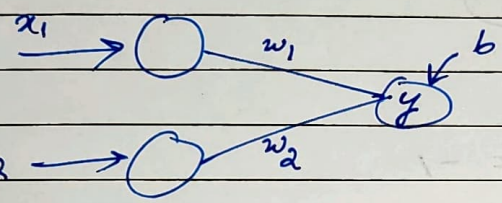
Mcculloch - Pitt Algorithm (for single layer perceptron model).

$$\text{o/p} \begin{cases} 1 & \text{if } wx + b > 0 \\ 0 & \text{if } wx + b \leq 0. \end{cases}$$

1. Logical AND

x_1	x_2	y	using Heaviside fn
0	0	0	$w_1x_1 + w_2x_2 + b = 0$
0	1	0	$w_1x_1 + w_2x_2 + b = 0$
1	0	0	$w_1x_1 + w_2x_2 + b = 0$
1	1	1	$w_1x_1 + w_2x_2 + b = 1$.

Model



Using Mcculloch Pitt Algo

$x_1=0, x_2=0$ $\quad w_1x_1 + w_2x_2 + b \leq 0 \Rightarrow b \leq 0$

$x_1=0, x_2=1$ $\quad w_1x_1 + w_2x_2 + b \leq 0 \Rightarrow w_2 + b \leq 0$

$x_1=1, x_2=0$ $\quad w_1x_1 + w_2x_2 + b \leq 0 \Rightarrow w_1 + b \leq 0$

$x_1=1, x_2=1$ $\quad w_1x_1 + w_2x_2 + b > 0 \Rightarrow w_1 + w_2 + b > 0$.

Say $b = -0.5$, then

$w_2 + b \leq 0$

$\Rightarrow w_2 \leq 0.5$ $\qquad$ Say $w_2 = 0.4$.

$w_1 + b \leq 0$

$\Rightarrow w_1 \leq 0.5$ $\qquad$ Say $w_1 = 0.4$.

$w_1 + w_2 + b > 0$

$\Rightarrow w_1 + w_2 > 0.5$.

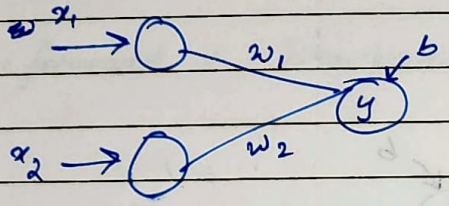
Check:

$0.4 + 0.4 - 0.5$

$= 0.3 > 0$ ✓

2. Logical OR

x_1	x_2	y		
0	0	0	$w_1x_1 + w_2x_2 + b = 0 \leq 0$	$b \leq 0$
0	1	1	$w_1x_1 + w_2x_2 + b = 1 > 0$	$w_2 + b > 0$
1	0	1	$w_1x_1 + w_2x_2 + b = 1 > 0$	$w_1 + b > 0$
1	1	1	$w_1x_1 + w_2x_2 + b = 1 > 0$	$w_1 + w_2 + b > 0$



Say $b = -1$.

$w_2 + b > 0$

$w_2 > 1 \Rightarrow$ Say $w_2 = 2$

$w_1 + b > 0$

$w_1 > 1 \Rightarrow$ Say $w_1 = 2$

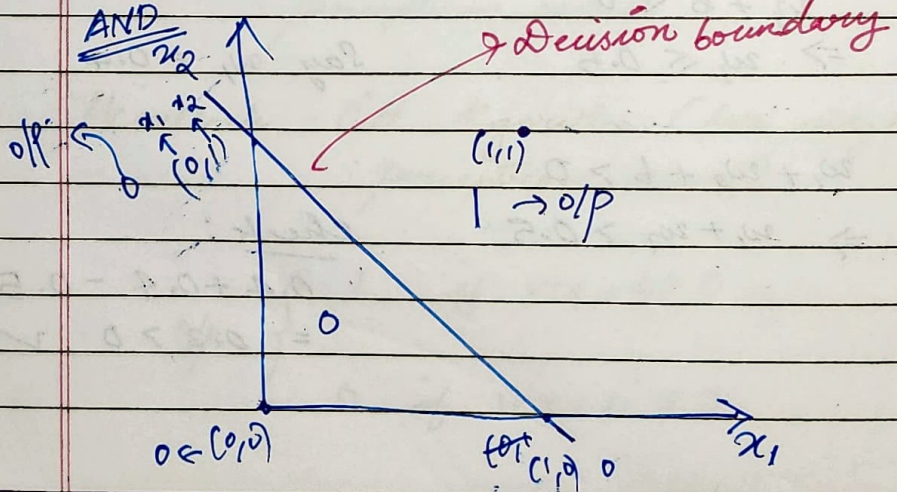
$w_1 + w_2 + b > 0$

$2 + 2 - 1 > 0$

$3 > 0$ is true

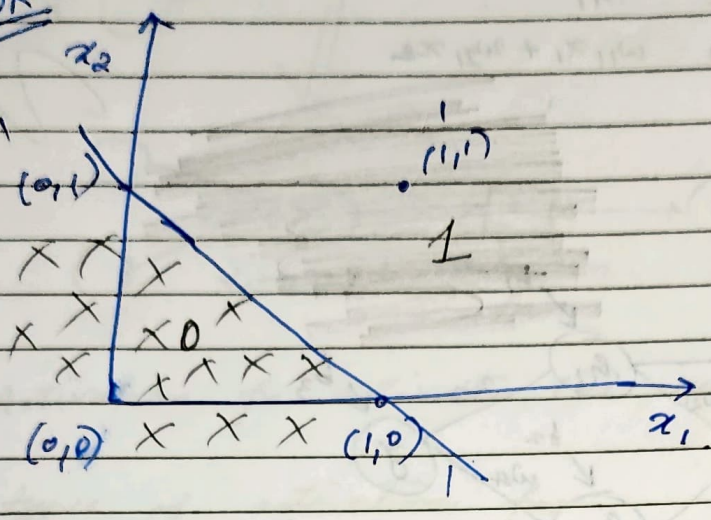
∴ One possible solution is

$b = -1, w_1 = 2, w_2 = 2$



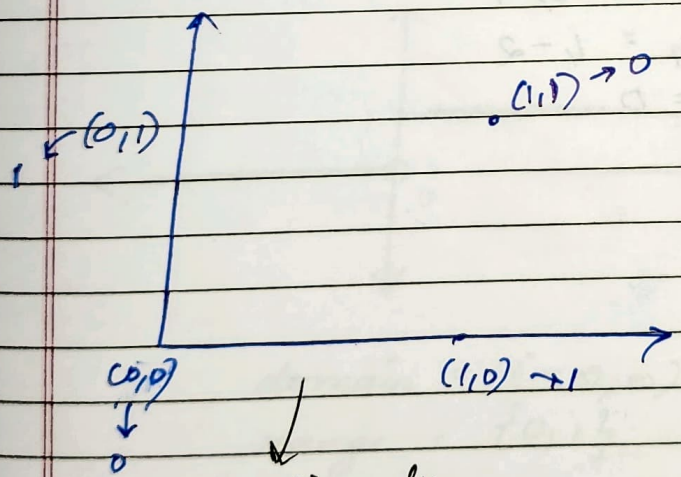
If decision boundary is a line, then problem stmt at hand is linearly separable.

OR



3. XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



non-linearly separable problem

→ No linear boundary possible

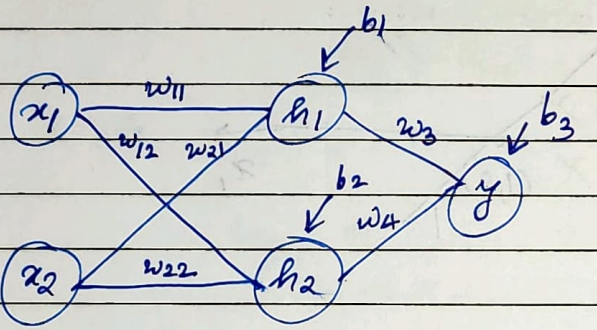
So XOR is not linearly separable & hence cannot be ~~implemented~~ solved using single layer perceptron model

Linear

Multi layer perceptron model

1. XOR

x_1	x_2	y	h_1
0	0	0	$w_{11}x_1 + w_{21}x_2$
0	1	1	
1	0	1	
1	1	0	



$h_1 \rightarrow w_{11}x_1 + w_{21}x_2 + b_1$

$h_2 \rightarrow w_{12}x_1 + w_{22}x_2 + b_2$

suppose $w_{11} = w_{12} = w_{21} = w_{22} = 1$.

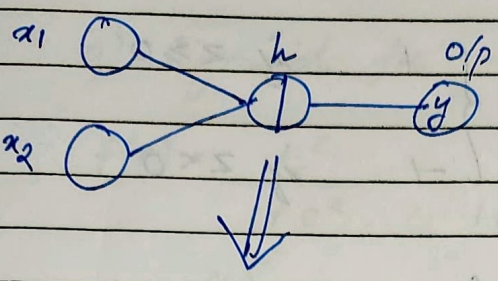
$b_1, b_2 = 0, -1$.

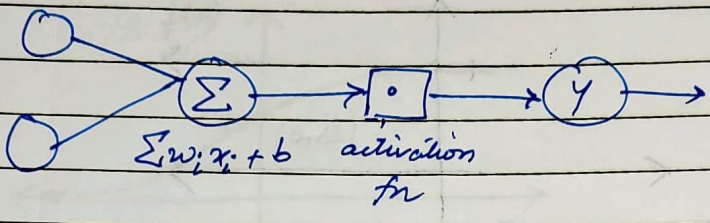
$w_3, w_4 = 1, -2$

$b_3 = 0$.

27/08/22

Activation functions





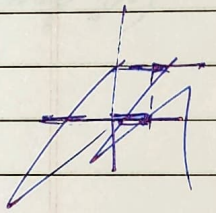
Activation fn activates the neurons.

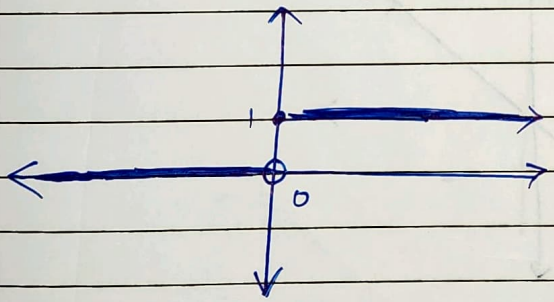
2 types of activation fns:

1. Linear fn

a) Step function

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





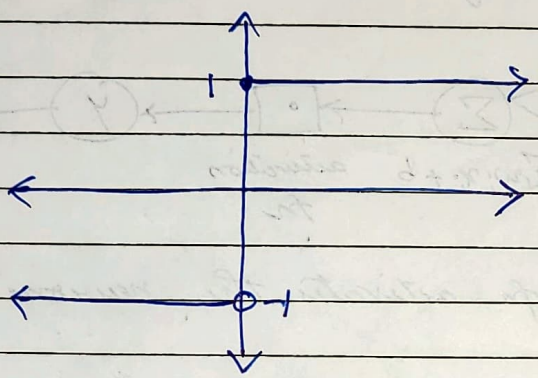
domain = $(-\infty, \infty)$

range = $\{0, 1\}$.

Used in binary classification.

b) Signum function :

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

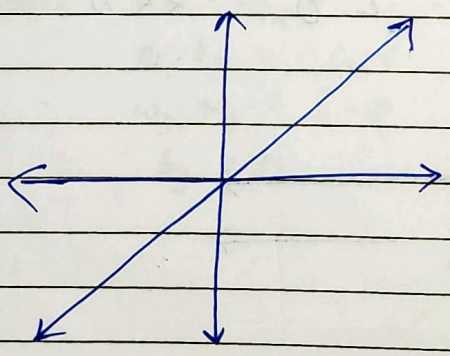


domain = $(-\infty, \infty)$

range = $\{-1, 1\}$.

c) linear function

$f(z) = z$.



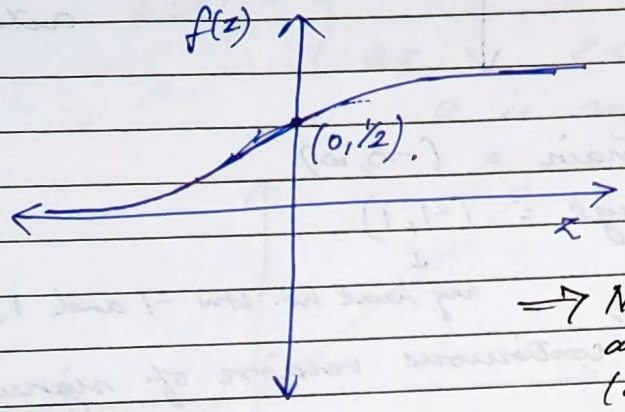
domain = $(-\infty, \infty)$

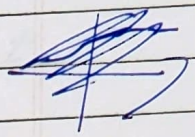
range = $(-\infty, \infty)$.

2. Non-linear function

a) Sigmoid fn

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





⇒ Non-zero centered activation fn (does not touch 0).

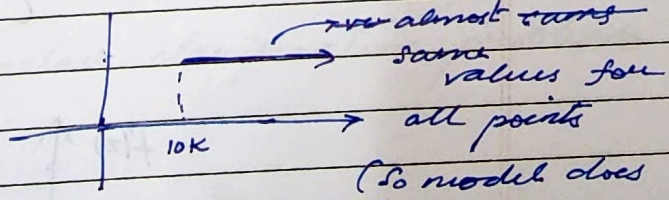
domain = $(-\infty, \infty)$

range = $[0,1]$ (continuous version of step fn)

↓ any real no. btw 0 & 1.

Problem: Local convergence

Say $z \in [10000, \infty)$.



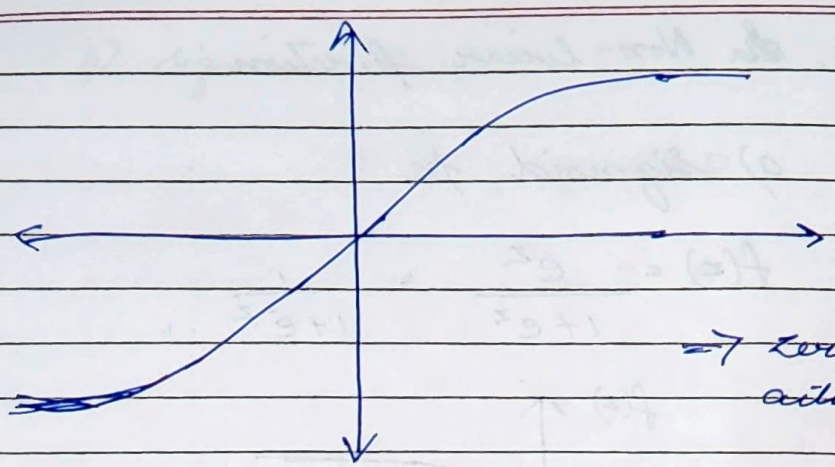
(So model does not learn).

⇓

So we use tanh

b) tanh function
(or) hyperbolic tangent function

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



⇒ zero-centered activation fn

domain = $(-\infty, \infty)$

range = $(-1, 1)$.

↓

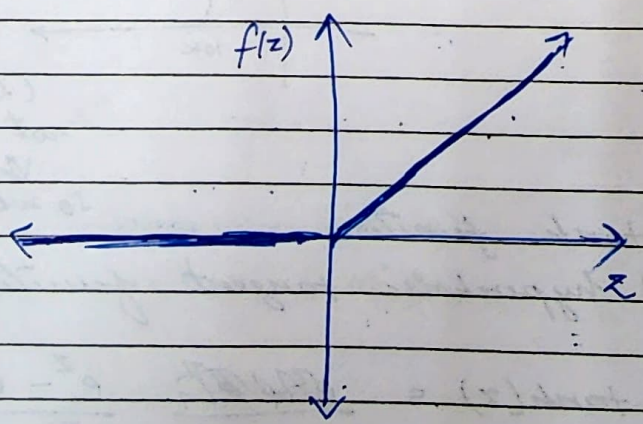
any real no. btw -1 and 1.

(continuous version of signum)

c) ReLU (Rectified linear unit).

tanh, sigmoid graphs are non linear. So we introduce linearity by using the foll. fn

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



domain = $(-\infty, \infty)$

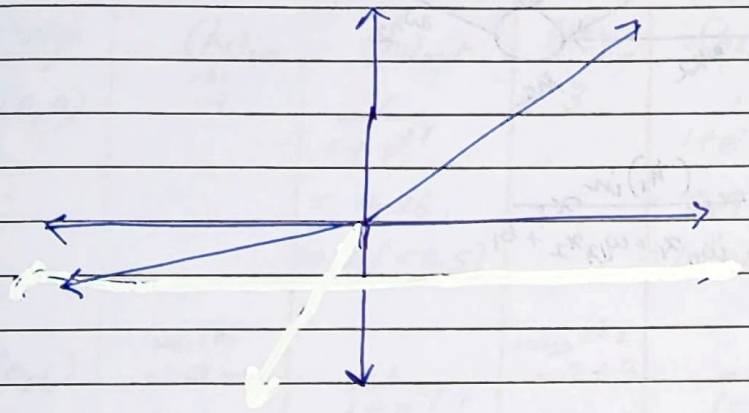
range = $[0, \infty)$.

d) Leaky ReLU

If ReLU does not ~~the~~ give good performance we can use leaky ReLU.

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

a is any constant



domain = $(-\infty, \infty)$

range = $(-\infty, \infty)$.

Softmax activation function

✓ Used for multi-class classification problems

$$f(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \ldots + e^{z_n}}$$

where $z_1, z_2, \ldots, z_n$ are the n outputs

$$f(z_2) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + \ldots + e^{z_n}}$$

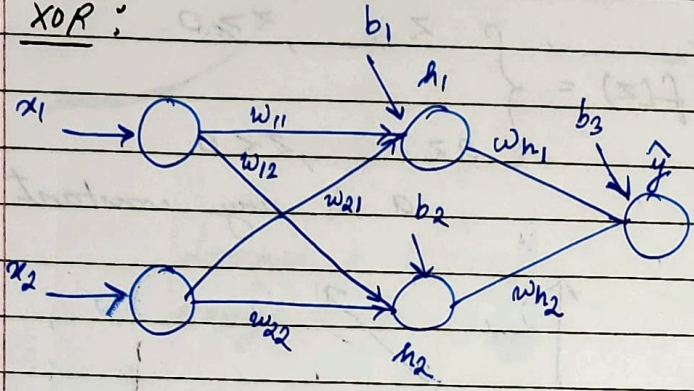
$$\vdots$$

$$f(z_n) = \frac{e^{z_n}}{e^{z_1} + e^{z_2} + \ldots + e^{z_n}}$$

$$f(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

26/08/22

XOR :



I/P		O/P	~~x_1~~ $(h_1)_{in}$ ~~x_2~~
x_1	x_2	y	$w_{11}x_1 + w_{21}x_2 + b_1$
0	0	0	
0	1	1	
1	0	1	
1	1	0	

$$(h_1)_{in} = w_{11}x_1 + w_{21}x_2 + b_1$$

$$(h_1)_{out} = Act((h_1)_{in})$$

$$(h_2)_{in} = w_{12}x_1 + w_{22}x_2 + b_2$$

$$(h_2)_{out} = Act((h_2)_{out})$$

$$y_{in} = w_{h_1}(h_1)_{out} + w_{h_2}(h_2)_{out} + b_3$$

$$y_{out} = Act(y_{in})$$

1) Given $w_{11} = w_{21} = 2$

$w_{12} = w_{22} = -2$

$[b_1, b_2] = [-1, 3]$.

$w_{h_1} = w_{h_2} = 2$

$b_3 = -3$.

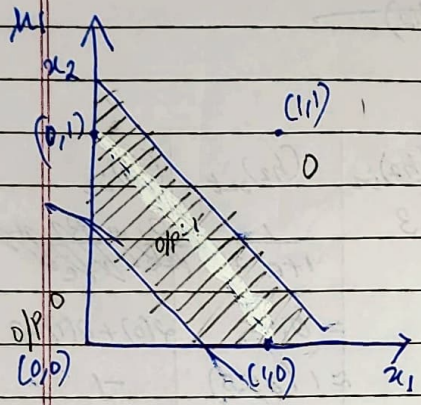
Use sigmoid activation fn and solve

~~I/p $(h_1)_{in}$ $(h_1)_{out}$ (h_2)~~

~~(0,0) b_1 $\frac{1}{1+e^{-b_1}}$~~

I/p	$(h_1)_{in}$	$(h_1)_{out}$	$(h_2)_{in}$	$(h_2)_{out}$	$(y)_{in}$	y_{out}
(0,0)	$=b_1$ -1	$\frac{1}{1+e^{+1}}$ $= 0.26$. ≈ 0 (<0.5)	3	$\frac{1}{1+e^{-3}}$ $= 0.95$ ≈ 1 (≥ 0.5)	$2(0)+2(1)-3$ -1	$\frac{1}{1+e^{+1}}$ 0.26 0.
(0,1)	$w_{21}+b_1$ $-2-1$ $=1$	$\frac{1}{1+e^{-1}}$ $= 0.75$ ≈ 1	$w_{22}+b_2$ $-2+3$ $=1$	$\frac{1}{1+e^{-1}}$ $=0.75$ ≈ 1	$2(1)+2(1)-3$ $=1$	$\frac{1}{1+e^{-1}}$ $=0.75$ ≈ 1
(1,0)	$w_{11}+b_1$ $+2-1$ $=1$	$\frac{1}{1+e^{+3}}$ $= 0.75$ ≈ 1	$w_{12}+b_2$ $-2+3$ $=1$	$\frac{1}{1+e^{-1}}$ $= 0.75$ ≈ 1	$2(1)+2(1)-3$ $=1$	$\frac{1}{1+e^{-1}}$ $=0.75$ ≈ 1,
(1,1)	$2+2-1$ 3	$\frac{1}{1+e^{-3}}$ $= 0.95$ ≈ 1	$-2-2+3$ -1	$\frac{1}{1+e^{1}}$ $= 0.26$ ≈ 0	$2(1)+2(0)-3$ $2-3$ $=-1$	$\frac{1}{1+e^{1}}$ 0.26 ≈ 0

	$(h_1)_{out}$	$(h_2)_{out}$	y
0,0	0	1	0
0,1	1	1	1
1,0	1	1	1
1,1	1	0	0
	↓	↓	↓
	OR	NAND	AND(h_1, h_2)



Given $w_{11} = w_{12} = w_{21} = w_{22} = 1$

$b_1, b_2 = 0, -1$

$w_{h_1}, w_{h_2} = 1, -2$

$b_3 = 0$.

ReLU activation

	$w_{11}x_1 + w_{21}x_2 + b_1$	ReLU	$w_{12}x_1 + w_{22}x_2 + b_2$	ReLU	$w_{h_1}h_1 + w_{h_2}h_2 + b_3$	ReLU
(0,0)	0	0	-1	0	0	0
(0,1)	$1+0=1$	1	$1-1=0$	0	$1+0=1$	1
(1,0)	$1+0=1$	1	$1-1=0$	0	$1+0=1$	1
(1,1)	$1+1+0=2$	2	$1+1-1=1$	1	$2-2+0=0$	0

	$(h_1)_{out}$	$(h_2)_{out}$
0,0	0	0
0,1	1	0
1,0	1	0
1,1	2	1

29/08/22

y - ground truth
$\hat{y}$ - predicted value

Loss function :

1. Mean Squared Error :

$$MSE = \frac{1}{m} \sum (y_i - \hat{y}_i)^2$$

m - number of samples
y_i - original output
$\hat{y}_i$ - predicted output

2. Mean Absolute Error :

$$MAE = \frac{1}{m} \sum |y_i - \hat{y}_i|$$

Linear regression

$$y = mx + c$$

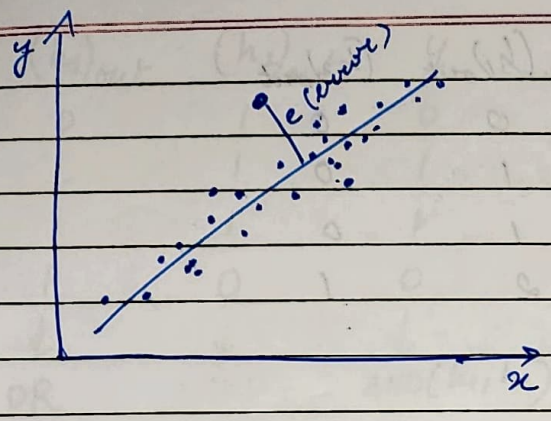
or

$$y = \alpha_0 + \beta x$$

In DL

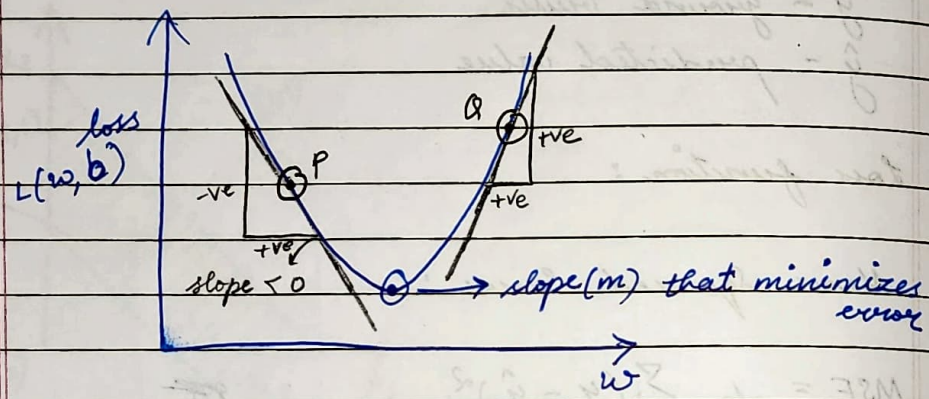
$$y = \sum w_i x_i + b$$

w x c



$e = (y - \hat{y})^2$

$= [y - (mx+c)]^2$



Gradient Descent

It is an iterative optimization algorithm to find the minimum of a function.

$$\text{slope} = \frac{\Delta Y}{\Delta X}$$

$$W_{new} = W_{old} - \eta \left(\frac{\partial L}{\partial w}\right)$$ → slope

η - learning rate

for point P

$$W_{new} = W_{old} - \eta \left(-\frac{\partial L}{\partial w}\right)$$ → -ve slope

$$= W_{old} + \eta \left(\frac{\partial L}{\partial w}\right)$$

↳ magnitude of slope

For point 4

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

(We need to reduce W to minimize error).

$$\hat{y} = mx + c$$

Error is (Loss) $L = \frac{1}{n}(y - \hat{y})^2$

$$= \frac{1}{n}(y - (mx + c))^2$$

$$\frac{dL}{dm} = \frac{2}{n}(y - \hat{y})(-x) = \frac{-2x}{n}(y - \hat{y})$$

$$\frac{dL}{dc} = \frac{2}{n}(y - \hat{y})(-1) = \frac{-2}{n}(y - \hat{y}).$$

$$m_{new} = m_{old} - \eta \frac{dL}{dm}$$

$$= m_{old} + \frac{2x}{n}\eta(y - (mx + c))$$

$$c_{new} = c_{old} - \eta \frac{dL}{dc}$$

$$= c_{old} + \frac{2\eta}{n}(y - (mx + c))$$

3. Binary Cross-entropy loss:

$$L = \frac{-1}{m} \sum \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

Used for classification problems

MSE	BCE
✓ for linear regression	for logistic regression classification problems
✓ does not have the ability to update weights	has the ability to update weights
✓ applicable only to simple problems	applicable to complex problems,

Multi-class classification

uses probabilistic distributions

3 classes

(Eg): cat – 0.9 (prob) → max ⇒ cat – 1
dog – 0.7 ⇒ dog – 0
rabbit – 0.5 rabbit – 0

⇓

data point ∈ cat class

✓ inherently finds probabilistic distributions of predicted values

Variants of Gradient Descent :

1. Stochastic Gradient Descent

Calculate loss and update weights aft for every data point

2. Mini batch GD

Divide dataset into batches and update weights for every batch

30/08/22

Binary Cross entropy Loss (BCE)

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

1.

y	$\hat{y}$
1	0.95
1	0.1

Find MSE, BCE

$(y-\hat{y})^2$

y	$\hat{y}$	MSE	BCE
1	0.95	0.0025	$-1 \log 0.95 = 0.051$
1	0.1	0.81	$-1 \log 0.1 = 2.302$

point 1 is almost accurate point 2 is almost inaccurate

there is significant difference btw the values allowing us to penalize 2nd prediction better

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

1. $y = 1, \quad \hat{y} = 0$

$$L = -y_i \log(\hat{y}_i) \Rightarrow \text{if target is 1}$$
$$= -\log \hat{y}_i$$

2. $y = 0, \quad \hat{y} = 1$

$$L = -(1-y_i) \log(1-\hat{y}_i) \Rightarrow \text{if target is 0}$$
$$= -\log(1-\hat{y}_i)$$

Categorical Cross entropy

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

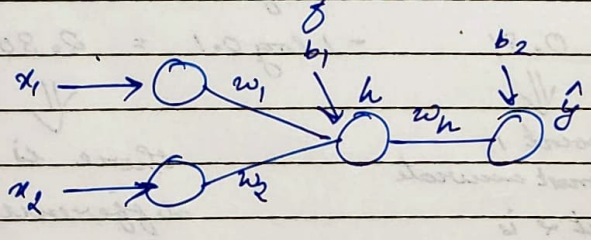
Backpropagation:

2 steps:
- ✓ weight updation
- ✓ chain rule of calculus

Weight updation:

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

Chain rule of Calculus



$$h_{in} = w_1 x_1 + w_2 x_2 + b_1$$
$$h_{out} = Act(h_{in})$$

$$y_{in} = w_h \times h_{out} + b_2$$
$$y_{out} = Act(y_{in}).$$

This is the forward pass

We now calculate error

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

Now we backpropagate & update weights

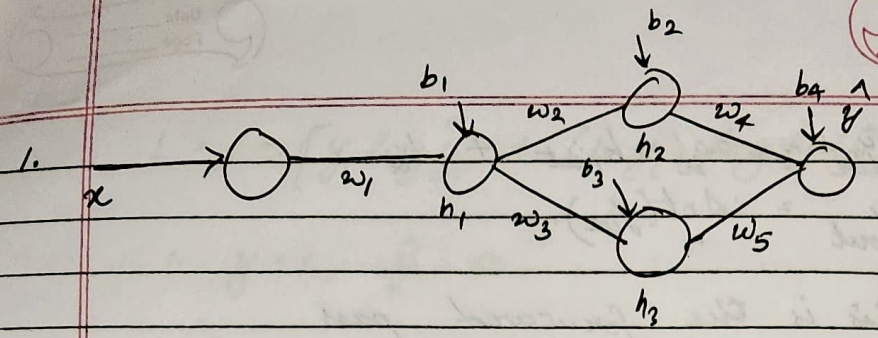
$$(w_h)_{new} = (w_h)_{old} - \eta \frac{\partial L}{\partial w_h}$$

$$\frac{\partial L}{\partial w_h} = \frac{\partial L}{\partial y_{out}} \cdot \frac{\partial y_{out}}{\partial y_{in}} \cdot \frac{\partial y_{in}}{\partial w_h}$$

~~$(w_h)_{new} = w_h$~~

$$(w_1)_{new} = (w_1)_{old} - \eta \frac{\partial L}{\partial w_1}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{out}} \times \frac{\partial y_{out}}{\partial y_{in}} \times \frac{\partial y_{in}}{\partial h_{out}} \times \frac{\partial h_{out}}{\partial h_{in}} \times \frac{\partial h_{in}}{\partial w_1}$$



$(h_1)_{in} = w_1 x + b_1$

$(h_1)_{out} = Act(h_{1in})$

$(h_2)_{in} = w_2 (h_1)_{out} + b_2$

$(h_2)_{out} = Act(h_{2in})$

$(h_3)_{in} = w_3 (h_1)_{out} + b_3$

$(h_3)_{out} = Act(h_{3in})$

$y_{in} = w_4 (h_2)_{out} + w_5 (h_3)_{out} + b_4$

$y_{out} = Act(y_{in})$.

$$(w_4)_{new} = (w_4)_{old} - \eta \left(\frac{\partial L}{\partial y_{out}} \times \frac{\partial y_{out}}{\partial y_{in}} \times \frac{\partial y_{in}}{\partial w_{4old}} \right)$$

$$(w_2)_{new} = (w_2)_{old} - \eta \left(\frac{\partial L}{\partial y_{out}} \cdot \frac{\partial y_{out}}{\partial y_{in}} \cdot \frac{\partial y_{in}}{\partial h_{2out}} \cdot \frac{\partial h_{2out}}{\partial h_{2in}} \cdot \frac{\partial (h_2)_{in}}{\partial w_2} \right).$$

$$(w_1)_{new} = (w_1)_{old} - \eta \left[\left(\frac{\partial L}{\partial y_{out}} \cdot \frac{\partial y_{out}}{\partial y_{in}} \cdot \frac{\partial y_{in}}{\partial (h_2)_{out}} \cdot \frac{\partial (h_2)_{out}}{\partial (h_2)_{in}} \cdot \frac{\partial h_{2in}}{\partial h_{1out}} \cdot \frac{\partial h_{1out}}{\partial (h_1)_{in}} \cdot \frac{\partial h_{1in}}{\partial w_{1old}} \right) + \left(\frac{\partial L}{\partial y_{out}} \cdot \frac{\partial y_{out}}{\partial y_{in}} \cdot \frac{\partial y_{in}}{\partial (h_3)_{out}} \cdot \frac{\partial (h_3)_{out}}{\partial (h_3)_{in}} \cdot \frac{\partial (h_3)_{in}}{\partial (h_1)_{out}} \cdot \frac{\partial (h_1)_{out}}{\partial (h_1)_{in}} \cdot \frac{\partial (h_1)_{in}}{\partial w_{1old}} \right) \right]$$

(path via h_2 + path via h_3).

Vanishing Gradient Problem

If we use sigmoid activation fn say

sigmoid → 0 to 1

derivative → 0 to 0.25.
range

If we use a long chain rule (as we did for w_1) then

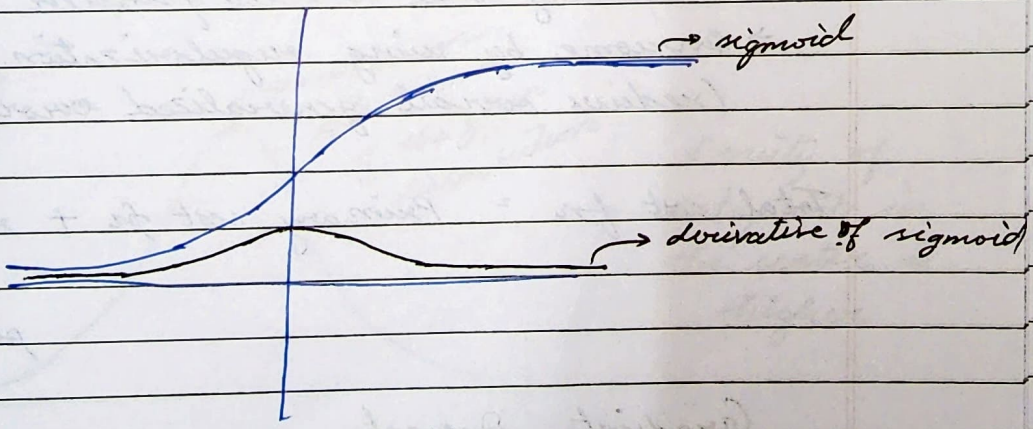
$0.25 \times 0.11 \times \ldots \times 0.2$ (say) (product of all derivatives).

will be very small

$\Rightarrow (w_1)_{new} \cong (w_1)_{old}$

$\Rightarrow$ Model will not learn (we are stuck at a local minima)

$\Rightarrow$ Vanishing gradient problem



02/09/22

Things to consider when creating a model

1. Cost fn (minimization problem → min. errors/loss)
2. Output unit
3. hidden unit
4. architecture

Cost fn → decided based on type of problem

Regression - statistical measures like MSE, MAE

Classification - Binary entropy (binary classification)
Categorical entropy (Multiclass classification)

Overfit:

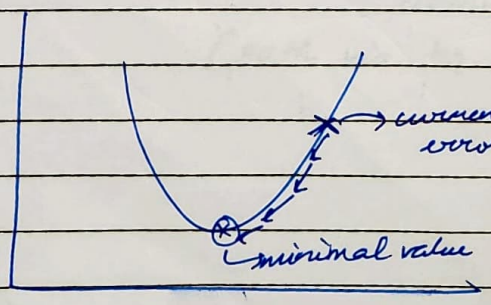
Model performs well on train data but not very well on test/unseen data

⇒ Overcome by using regularization. (reduces overall generalized error).

Total cost fn = Primary cost fn + regularization
↓
penalty.

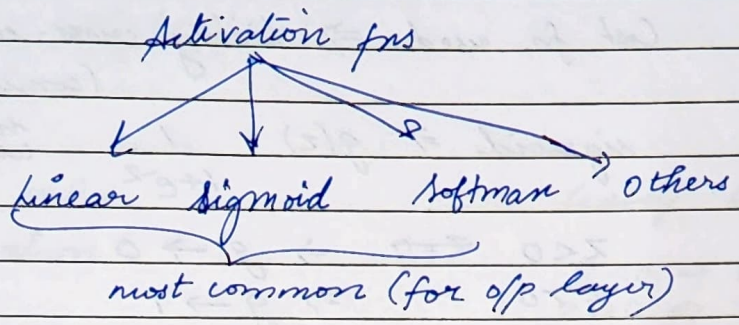
Gradient Descent

used for
✓ Reducing the error by updating weight



o/p activation need not be the same as hidden layer activations.

⇒ mostly softmax used for o/p activation.



* Choosing activation fns (for o/p layer):

Depends on $P(y|x, \theta)$ ⇒ distribution of o/p conditioned on inputs.

If distribution is normal / gaussian

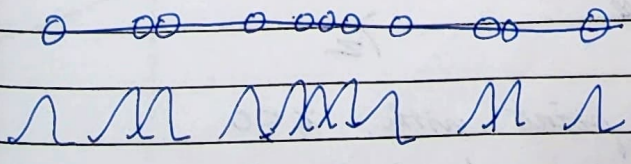
⇒ use linear activation

Density of datapoints in the center is higher

(data is linear/continuous)

↓

regression



If data label is either 0 or 1 ⇒ use sigmoid activation

⇓

binary classification.

Cost fn used ⇒ binary cross-entropy

(consists of Prob of success (class 1) & failure (class 0))

sigmoid ⇒ $g(z) = \frac{1}{1+e^{-z}}$

$z < 0 \quad z = 0 \Rightarrow g \to 0$

$z > 0 \Rightarrow g \to 1.$

softmax for multiclass classification problem

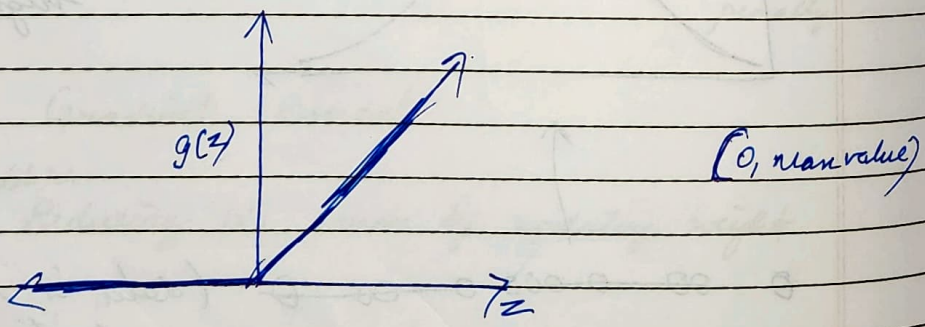
⇒ most common

* Activation (for hidden)

- ReLU
- sigmoid
- tanh

most common

ReLU



For all points with $z < 0$

⇒ we get o/p 0 always

⇒ next layer always gets the i/p 0

leads to saturation

so we use other variants like leaky ReLU

y y/n

tanh

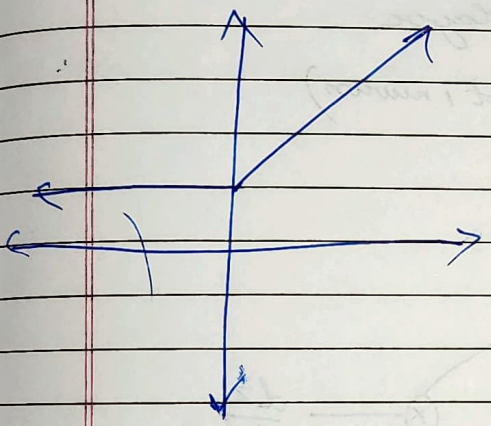
$g(z) \Rightarrow$ sigmoid?

tanh $\Rightarrow$

$$2(g(2z)) - 1$$

Variants of ReLU

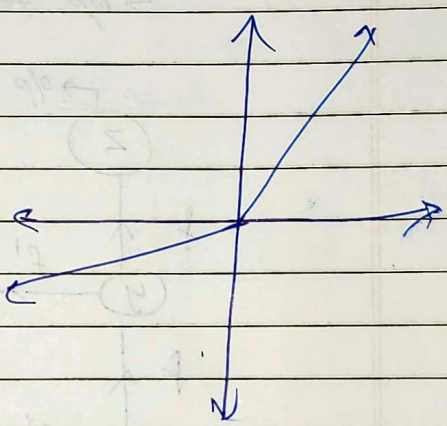
Absolute value rectification



Leaky ReLU.

$z < 0 \Rightarrow 0.01z$

$z \geq 0 \Rightarrow z$

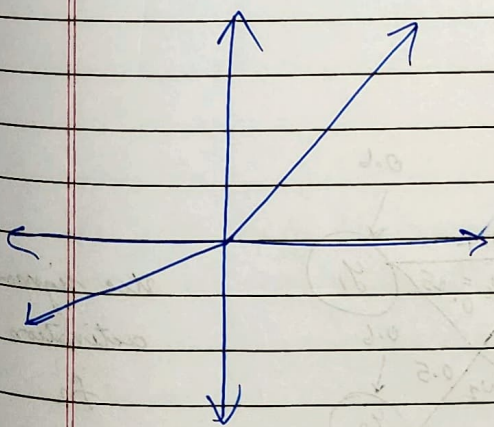


Parametric ReLU

$z < 0 \Rightarrow az$

$z \geq 0 \Rightarrow z$

'a' is a parameter



Maxout

$\max(W_1^T x + b_1, W_2^T x + b_2)$. (uses 2 sets of weights & biases?!)

05/09/22

Initial weights - assigned randomly

Updation of weights - done using approaches/algos like Gradient Descent (Backpropagation).

2 steps:

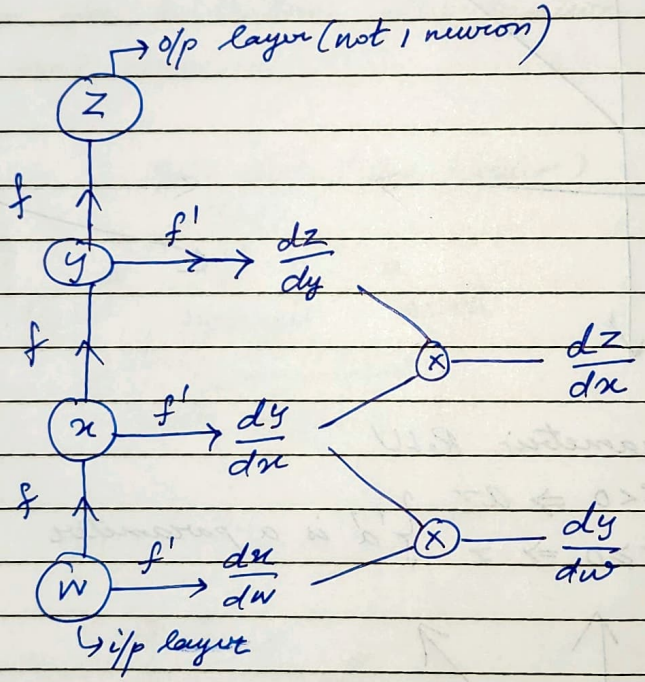
Forward propagation of weights

Backward propagation of errors.

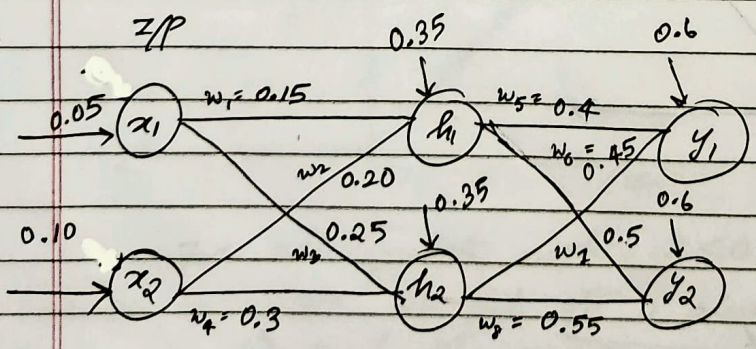
Loop.

$y = f(x, \theta)$

↳ o/p for any layer



1. I/P



Use sigmoid activation fn

Target (expected) T

y_1	y_2
0.01	0.99

$(h_1)_{in} = (h_1)_{in} = w_1 x_1 + w_2 x_2 + b_1$

$= 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35$

$(h_1)_{in}$ t_0 =

$(h_2)_{out} = \frac{1}{1+e^{-}}$

We got $E_{total} = 0.2984$ (MSE).

⇒ not good enough
so we backpropagate and update weights

* stopping conditions (for training a model):

✓ If error in current iteration is much better than previous iteration (convergence occurs)

✓ If max iterations have reached
⇒ will not give the best model (weights might still have to be updated).

06/09/22

Updation:

$$W = W - \alpha \Delta$$

(learning rate, gradient)

$$E_{total} = \frac{1}{2}(T_1 - (y_1)_{out})^2 + \frac{1}{2}(T_2 - (y_2)_{out})^2$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial T_1} \times \frac{\partial T_1}{\partial}$$

$$= \frac{\partial E_{total}}{\partial (y_1)_{out}} \times \frac{\partial (y_1)_{out}}{\partial (y_1)_{in}} \times \frac{\partial (y_1)_{in}}{\partial w_5}$$

$$= \left[\frac{1}{2} \times 2 \times (T_1 - (y_1)_{out}) \times -1\right]\left[\frac{-1}{(1+e^{-y_1})^2} \cdot e^{-y_1} \cdot -1\right]$$

$$\left[(H_1)_{out}\right].$$

$$= +((y_1)_{out} - T_1)\left(\frac{e^{-y_1}}{(1+e^{-y_1})^2}\right)(H_1)_{out}$$

$\alpha = 0.5$ = ~~———— = 0.01~~

$T_1 = 0.01$

$T_2 = 0.99$

$(y_1)_{out} = 0.7513$

$(H_1)_{out} = 0.592$

$\left(1 - \frac{1}{1+e^{-y_1}}\right)$

$$\frac{\partial E_{total}}{\partial w_5} = \left((y_1)_{out} - T_1\right)\left(\frac{1}{1+e^{-y_1}} \cdot \frac{e^{-y_1}}{1+e^{-y_1}}\right)(H_1)_{out}$$

$$= \left((y_1)_{out} - T_1\right)\left((y_1)_{out}\left(1-(y_1)_{out}\right)\right) H_{1out}$$

$$= (0.7513 - 0.01)(0.7513 \times (1-0.7513)) \times 0.592$$

$$= 0.08216.$$

$$(w_5)_{new} = (w_5)_{old} - \alpha \cdot \frac{\partial E_{total}}{\partial w_5}$$

$$= 0.4 - 0.5 \times 0.08216$$

$$= 0.4 - 0.04108$$

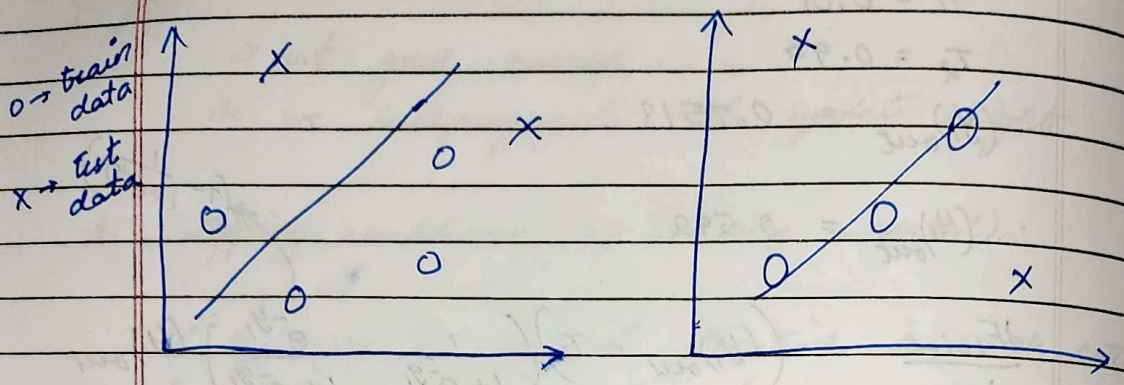
$$= 0.35892$$

Repeat this for all w

After updation of weights, check the error (eg performing forward propagation) to see if it is acceptable

Else repeat.

Regularization

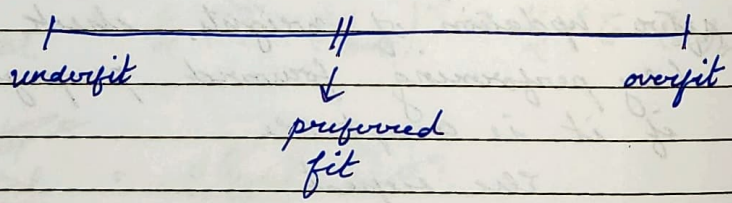


⇒ Poor performance on both train & test set
⇒ underfitting

⇒ good performance on train set but poor performance on test set ⇒ Overfitting
Use regularization

Total cost fn = primary cost fn + Regularization parameter (penalty).

We need to find a central point



Methods :
- ✓ L^2 ⇒ penalty = $(\text{slope})^2$
- ✓ L^1 ⇒ penalty = $|\text{slope}|$
- ✓ Data augmentation
  - *. flipping
  - *. rotation
  - *. skewing (introducing noise / distortion)
- ✓ Early stopping (stop training when error is minimal or convergence occurs)

Even if #samples in ~~dat~~ train data is low, overfitting can ~~ca~~ occur

⇒ Use more data or data augmentation (if more data is not available).

Be careful when using data augmentation

(Eg): 9 $\xrightarrow{\text{rotate by 180°}}$ 6

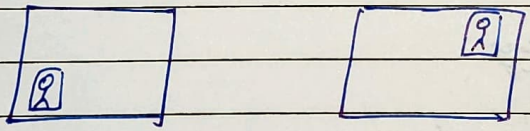
d $\xrightarrow{\text{flip vertically}}$ b

} data is itself changed (not preferred leads to misclassification)

9 $\xrightarrow{\text{blur}}$ 9 (acceptable).

(cont. of Reg. methods) ✓ Parameter sharing

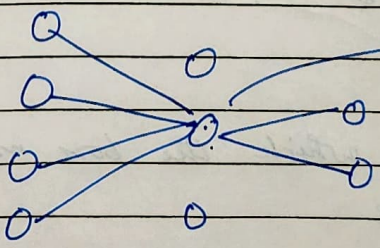
Consider 2 images in which the same person is present at different locations



To recognize the person, we can use the same parameters (can be saved & shared).

✓ Dropout

Randomly omit nodes



→ if this node is dropped out then o/p from this node is not propagated to the next layer (~~exists to~~ reads ~~computed~~)

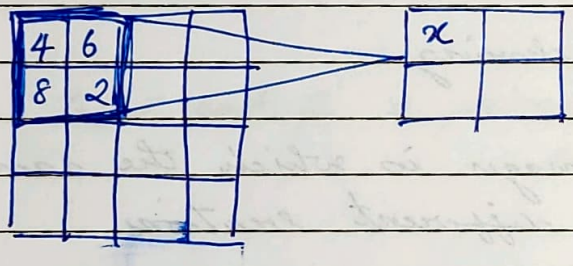
Leads to reduction in computation

If one node causes overfitting then dropping it will help prevent dropout since we don't know which node is responsible for dropout, we randomly drop out some % of the nodes.

✓ Boosting & Bagging

Ensemble - Train many models & combine the results from them

09/09/22 ~~Unit 1~~



Max Pooling:

$x = \max(4, 6, 8, 2)$

$= 8$

Avg. Pooling:

$x = \frac{4+6+8+2}{4}$

$= 5.$

Stride:

steps by which the box moves in X and Y direction

MNIST 0-5