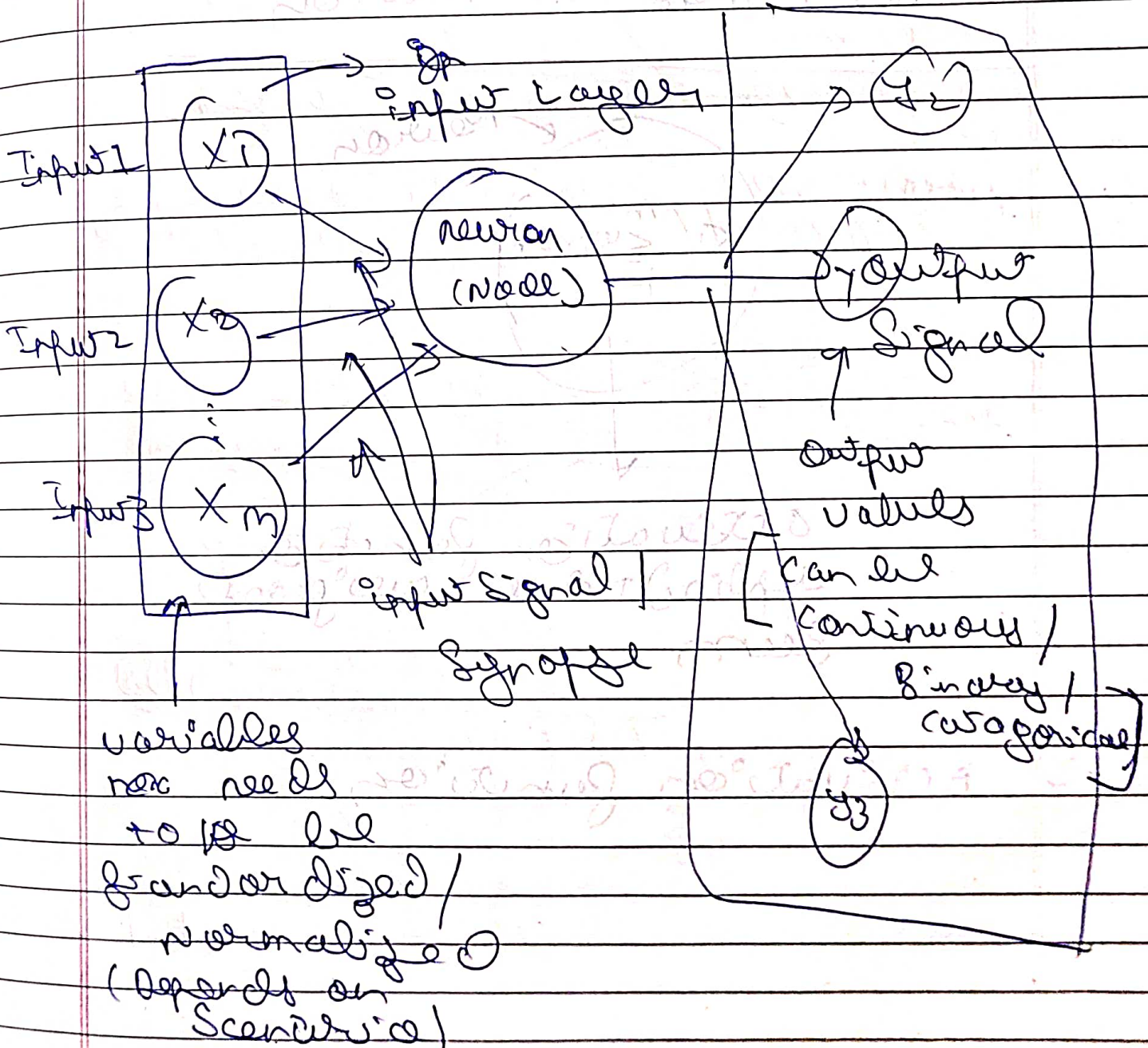
PAGE No
DATE / /

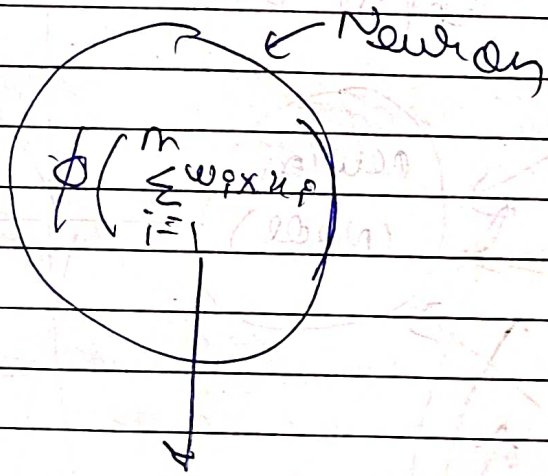
Aaw 12

Artificial Neural Networks



all Synapses have weights
By weights Neural networks
decide which signal
is important and which
is not

what happens in neuron



activation function
applied to a weighted
sum

→ Activation function

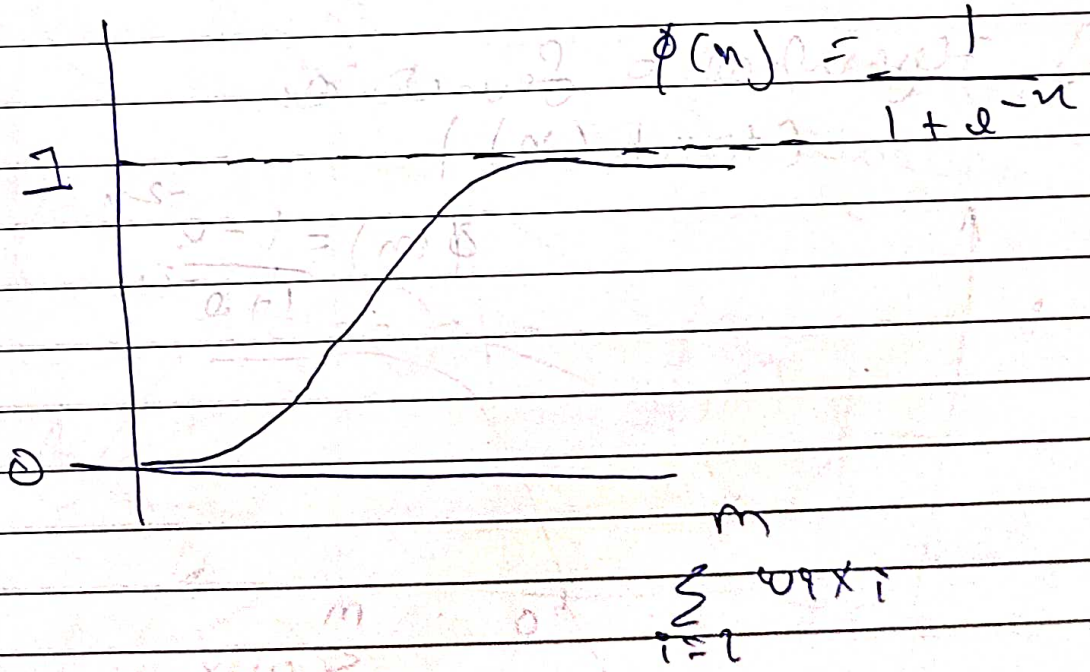
PAGE No.
DATE / /

Types

1] threshold function

$\phi(n) = \begin{cases} n < 0 \rightarrow 0 \\ n \geq 0 \rightarrow 1 \end{cases}$

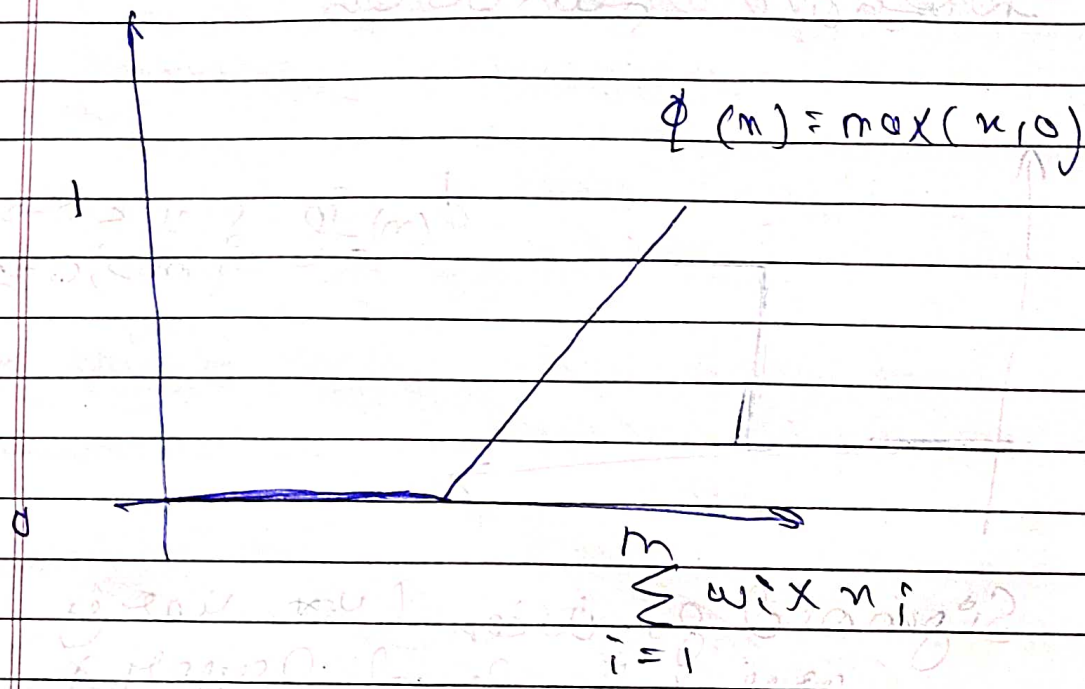
2] Sigmoid function (used very often in final layer)



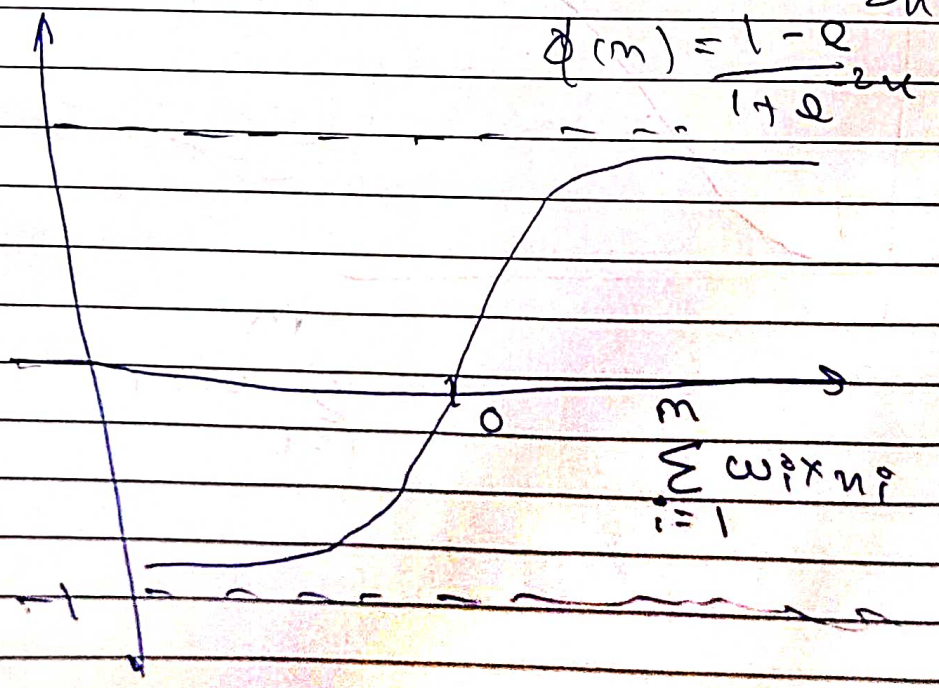
$\phi(n) = \frac{1}{1 + e^{-n}}$

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

3) Rectified function
(f on g the not type)



4) Hyperbolic function
(tanh(n))



PAGE No.
DATE / /

① Suppose we have a binary input variable

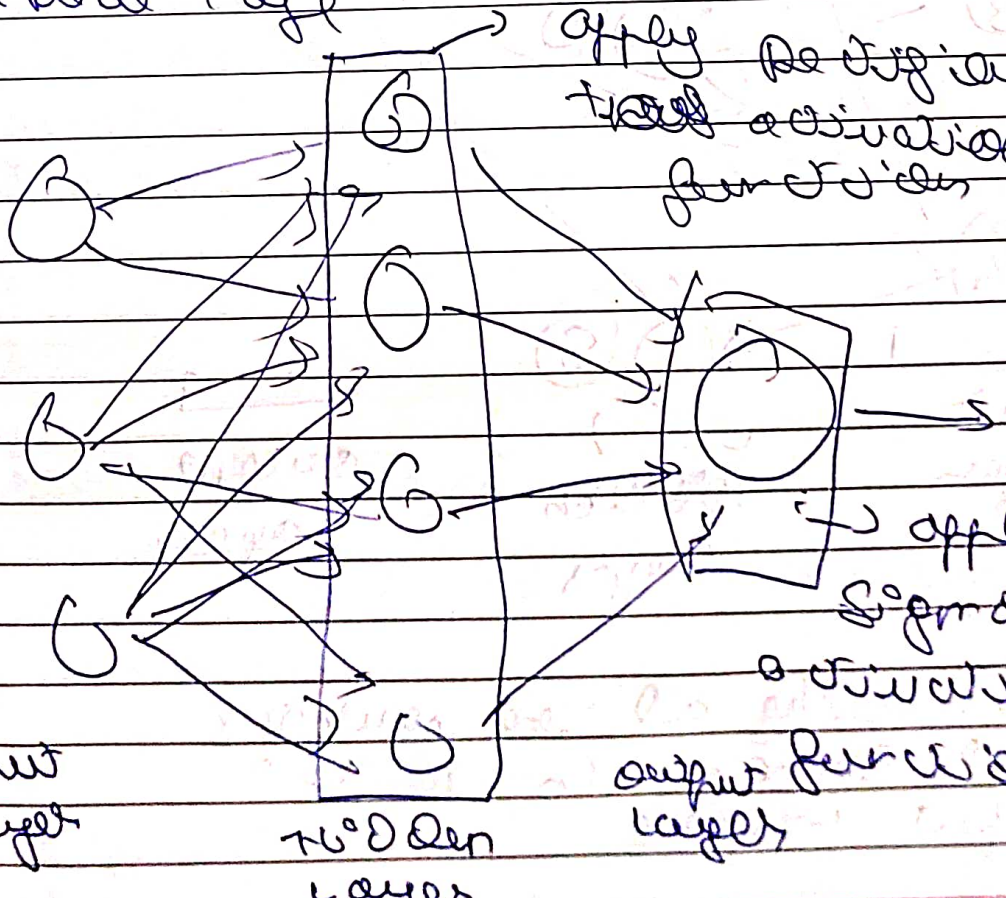
→ threshold function $y = \phi\left(\sum_{i=1}^{n} w_i x_i\right)$

$y = 0$ or 1

→ Sigmoid function $P(Y=1) = \phi\left(\sum_{i=1}^{n} w_i x_i\right)$

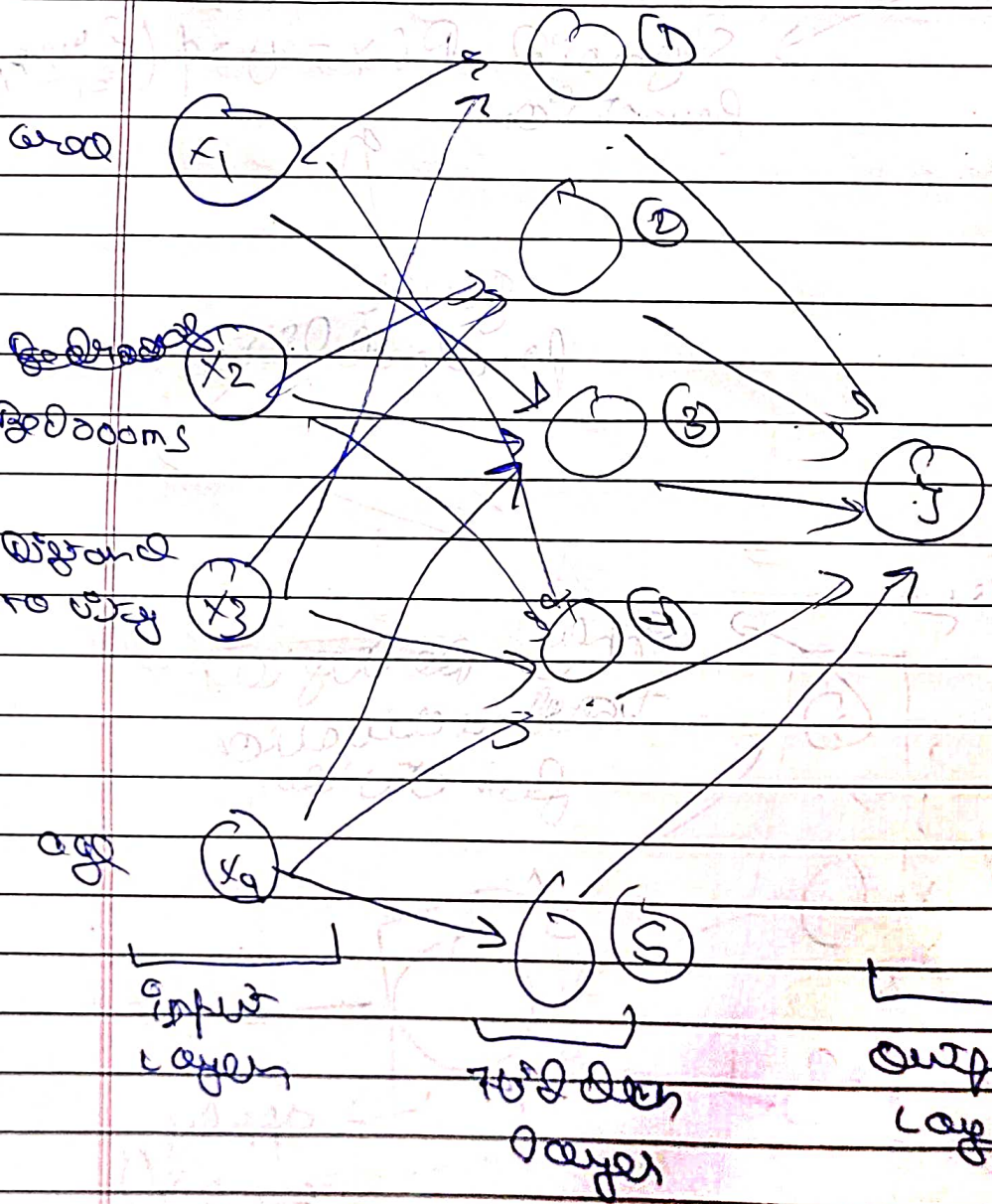
↑ gives probability

② a hidden layer



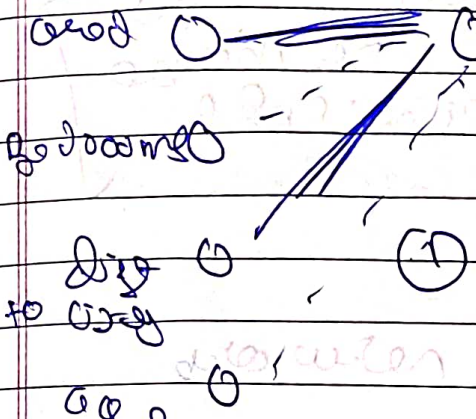
How do neural network work

Suppose we want to predict property valuations



To actually all the neurons are connect to all the input layers

But for a neuron only some connections are important



Here area and city are important because there might be correlation, that as we go for more city area increases

for (3)

in a area a lot of family might prefer a property with good area, more bedrooms and less age

for (5)

indicate some correlations

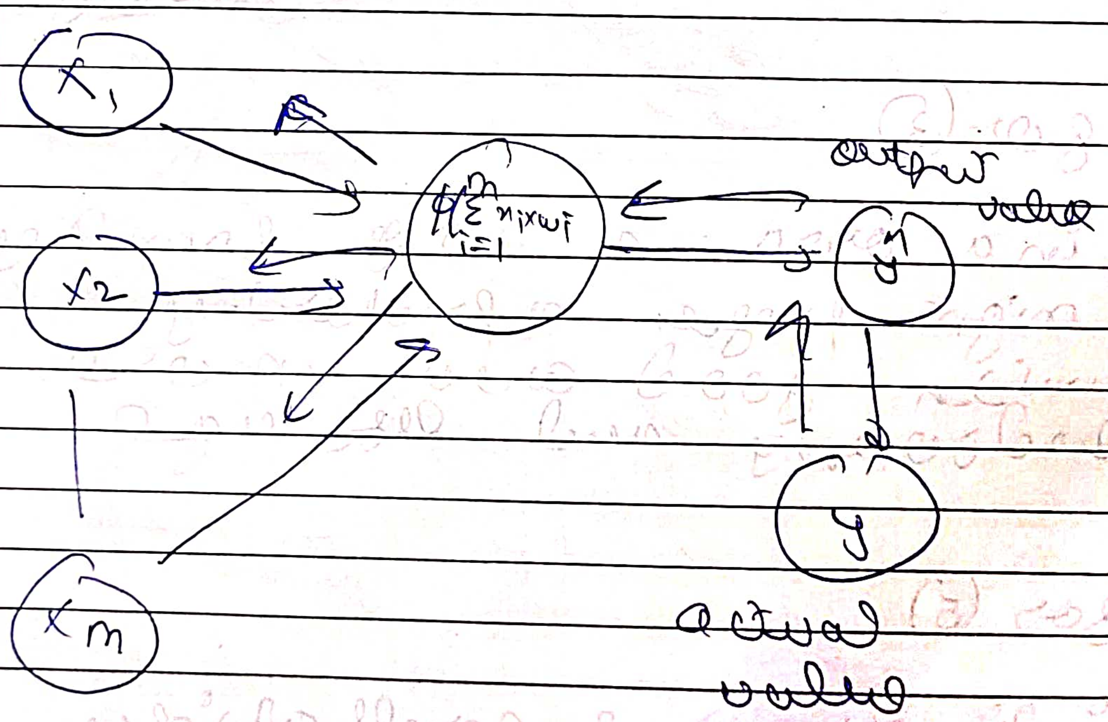
we know that more age less value

Now however if it is Historic it suddenly becomes more valuable

so ③ only activates when nonlinear property

[Basically hidden layer allows us to make our model flexible]

→ How do neural network learns



for a value is predicted then it is compared with the actual value

after this the cost is calculated

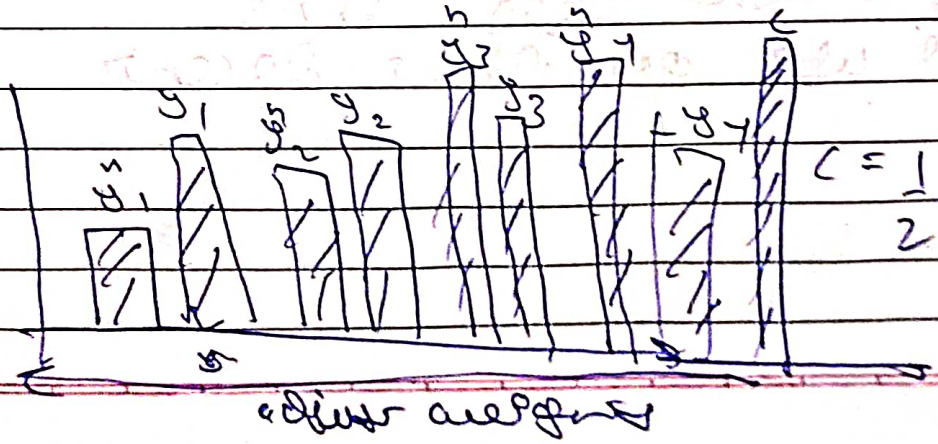
$$C = \frac{1}{2}(\hat{y} - y)^2$$

(can also be any other function)

now we want to minimize the cost function so we send this info back to neural network which then re-adjusts weights

[this is for one row]

if we have multiple rows let say we have 5 rows



PAGE No.
DATE / /

after all 4 pairs are predicted we calculate C and give it back to neural network

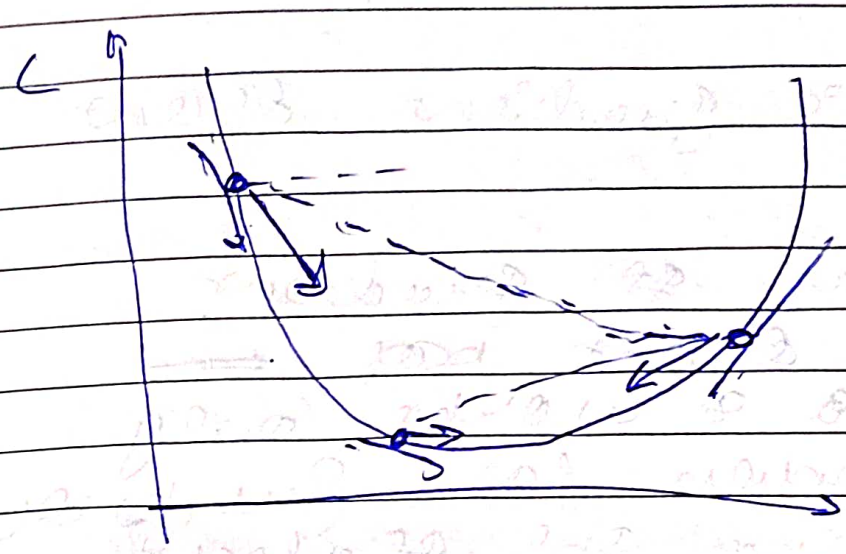
This process of going back is called Back propagation

⇒ Back propagation (flow of gradient)

⇒ Gradient Decent

: If we try to brute force all the weights the model will take exponential time to train

to avoid this we use gradient decent



$C = \frac{1}{2}(a^{y} - b^{y})^2$

for 1-D

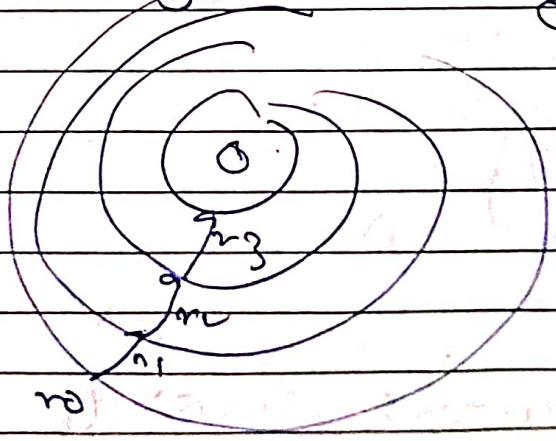
we find slope of the tangent point

if slope is -ve go towards Right

if slope is +ve go towards left

this is better than brute force

for 2-D



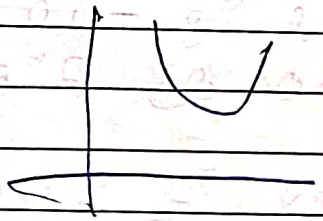
gradient descent

PAGE No.
DATE / /

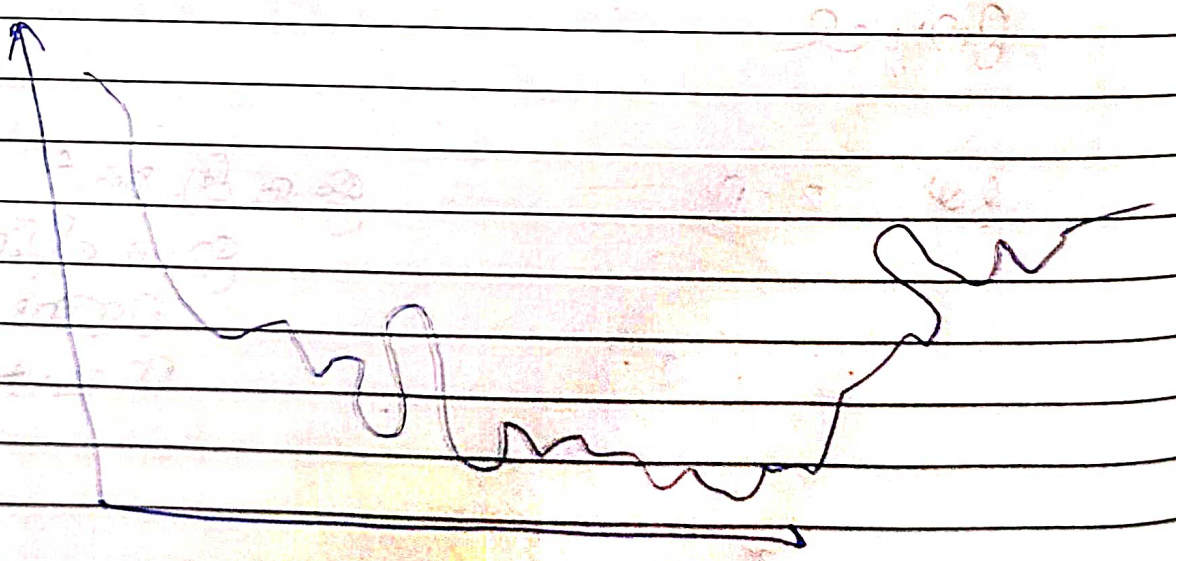
→ Socastic gradient descent

normal gd. gradient descent cost always only on convex cost- function (i.e having one global minima)

$\frac{1}{2}(\hat{y}-y)^2 =$



if function is like this



it won't work

PAGE NO.
DATE / /

in this type weights are readjusted after every data is processed [not in batch batch]

~~the Backpropagation~~

Steps followed

1] weights are assigned randomly with values close to 0

2] input layer observations of each dataset in input layer where each feature in is a node

3] forward - propagation:- from lyr - to - right the neurons are activated in a way that impact of each neuron is limited by weights

4] compare predicted results to actual results

5] Back - propagation

6] Repeat from 1 to 5

7] Do Forward and Backward propagation multiple times again and again

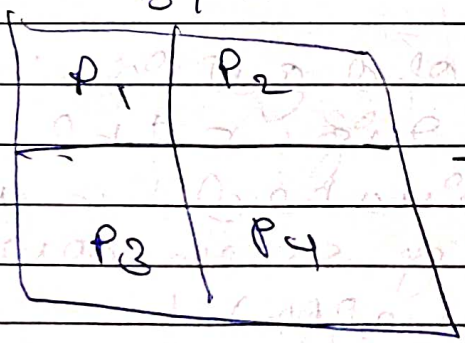
(One time is one epoch)

Convolutional neural Networks

How do CNN see images

Suppose we have a B/W image and a coloured image of 2×2 px

B/W

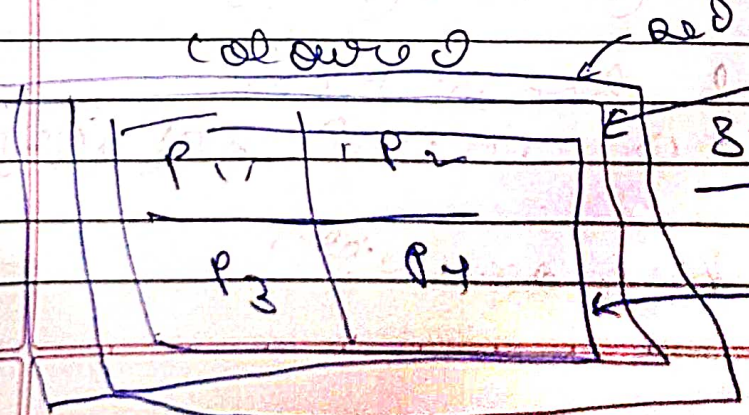


2D array

0 → Black

(255 → white)

coloured



Steps followed

1] ~~(onv2d~~ Convolution

2] max Pooling

3] flattening

4] full - connection

Convolution

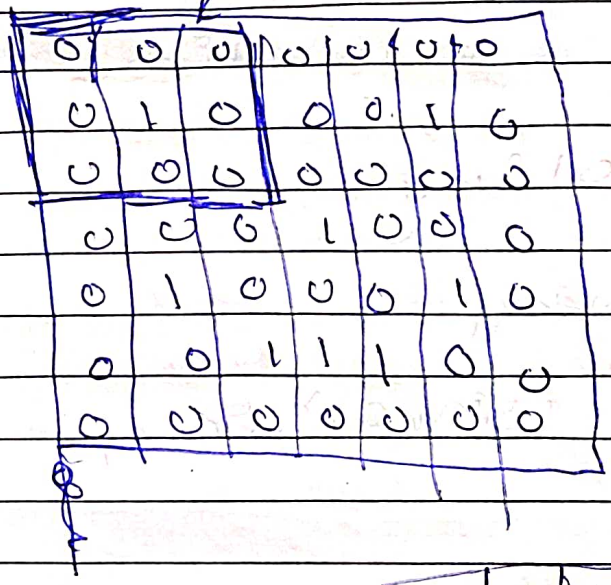
Convolution function

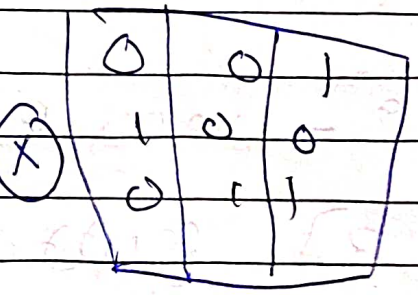
$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

First when we have our image we Reduce it's size and only keep certain important features this is done using a feature detector and after doing this we generate a feature map

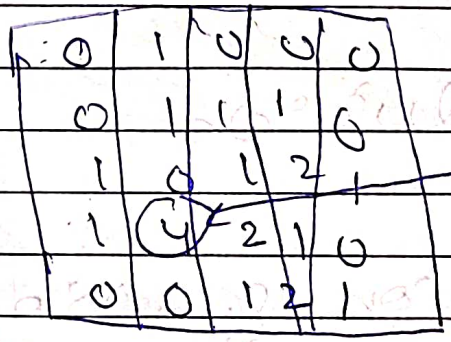
eg

Q: Do this we take all 3×3 squares





Feature detection / filter



perfect match

→ we multiply each element by corresponding (not matrix multiplication and summing results up)

Help to get rid of unnecessary stuff

we get multiple feature maps the CNN detects through training which is important

→ Rectified linear unit (ReLU)

we apply the rectifier activation function to our feature maps this is done to increase non linearity (we do this as images are themselves non-linear)

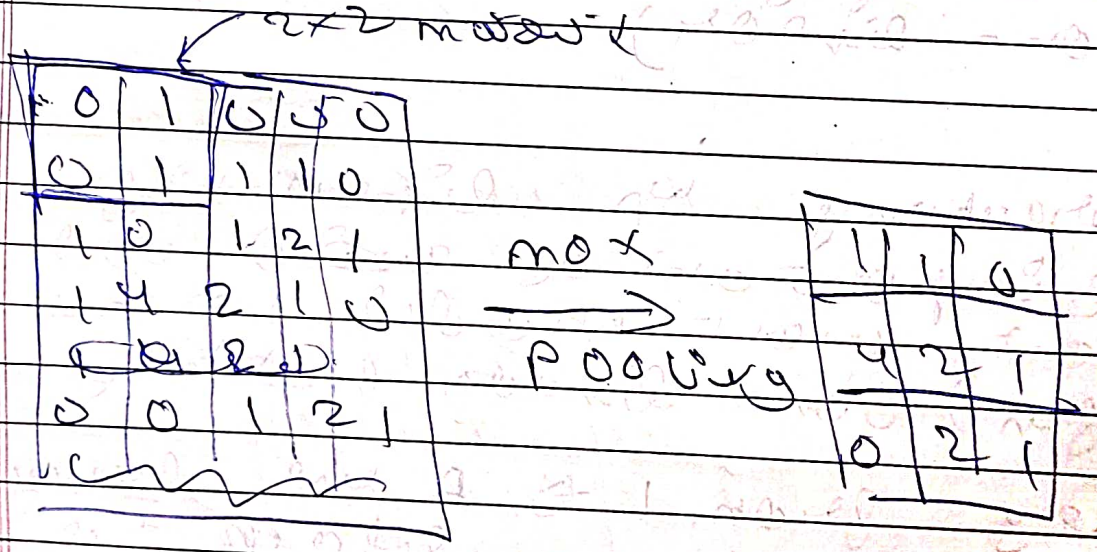
Breaking up linearity is important as we don't want model to find some linear functions

(e.g from L → R we always go from black to white)

→ max Pooling

Suppose we want to identify a cat But ~~the~~ in one photo it is inverted in other it is rotated

So we should not care about the position of features we want spatial variance



for any 2×2 matrix we take out max element

[there are also other pooling like sum pooling but we prefer max pooling]

-> Flattening

we flatten the pooled featured map and make it into a column network

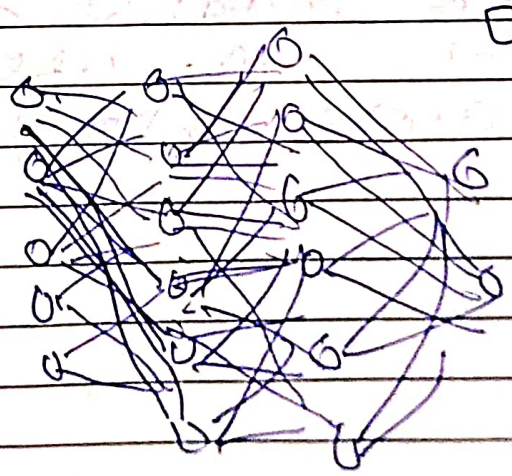
1	1	0
4	2	1
0	2	1

flatten

1
1
0
4
2
1
0
2
1

-> Full connection

Flatten



Everything is fully connected

following some steps by ANN

input is given → loss / cost function → error if Backpropagated by or Backprop given →

1] weights are adjusted

2] feature descriptor are adjusted

→ Softmax and cross-entropy

In the neural network the probabilities given by the output layer want add up to one but to make it add up to one we use Softmax and cross entropy

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$ (helps to make probability add up to one)

↓

Softmax function

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

↑

Cross-entropy function

$$H(p,q) = -\sum_x p(x) \log q(x)$$

[This is the cost/loss function used to detect error and back propagate]

Step 1: We have 2 neural network to differentiate between cat and dog

NN1 → (y) → (y)

Row	Dog	Cat	Dog	Cat
~~Dog~~	0.9	0.1	1	0
~~Cat~~	0.1	0.9	0	0
	0.4	0.6	1	0

this obs is incorrectly predicted

Row	Dog	Cat	Dog	Cat
1	0.6	0.4	1	0
2	0.3	0.7	0	1
3	0.1	0.9	1	0

Classification error (BCE)

Y_3 Y_3

PAGE No.
DATE / /

Mean Squared error

0.25 0.71

Cross-Entropy

0.36 1.06

So why use Cross-Entropy
in the first two values are
very small so the log
in Cross-Entropy helps
to detect those errors

(Cross-Entropy only used
for Classification
for e.g. we should
use mean squared)