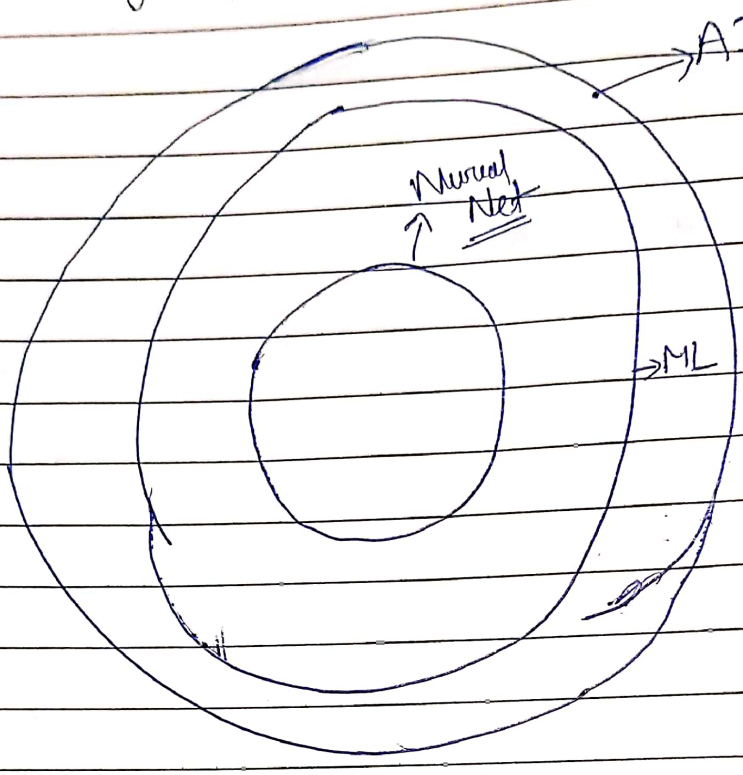
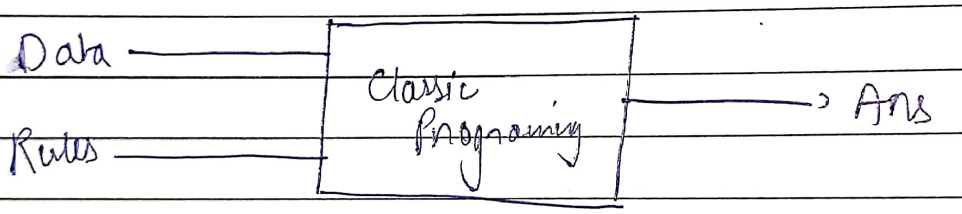
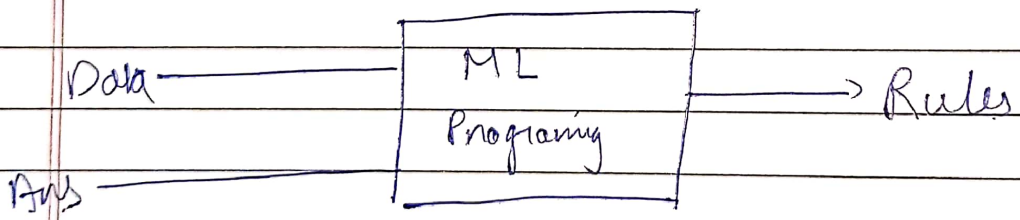
Artificial Intelligence (Personal study)

Agent: Anything that can be viewed as perceiving its environment through the sensors and acting upon that environment through actuators







Neural Net → layered

Machine learning → figures out the rules for a given input and output

Date
Page

Data

Ex Table. (for detection of Mid term2 grade)

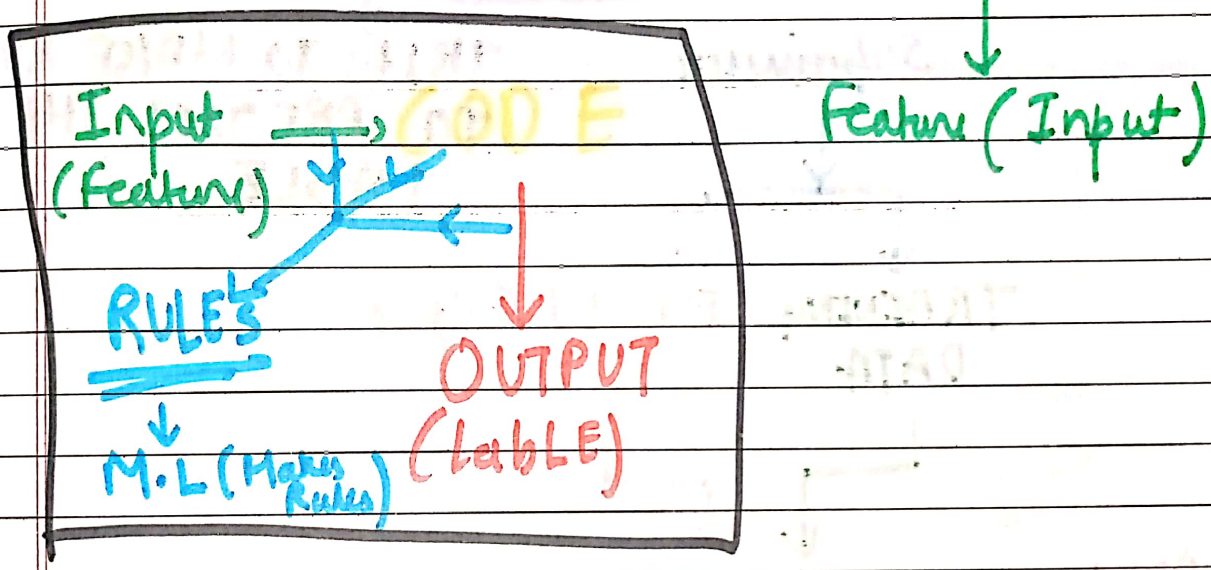
	Mid term1	Mid term2	Final
1.	70	80	77
2.	60	90	84
3.	40	50	38

Feature

Feature (Input)

Lable (OUTPUT)

Feature (Input)

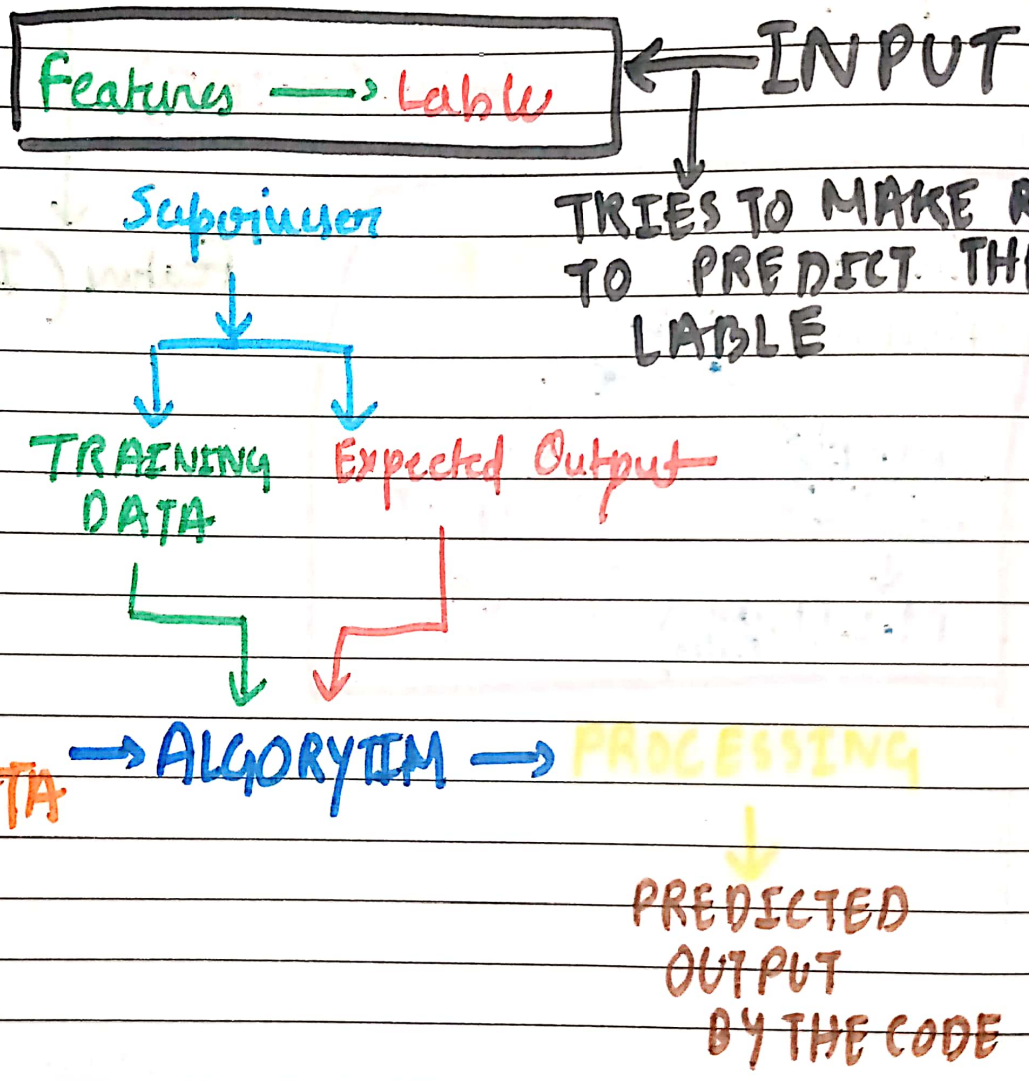


Date
Page

Different types of M.L

→ Unsupervised learning
→ Supervised learning
→ Reinforcement learning

Supervised LEARNING



IF,

PREDICTED OUTPUT != Expected OUTPUT

Tweaks Made by the Supervisor to get the correct output

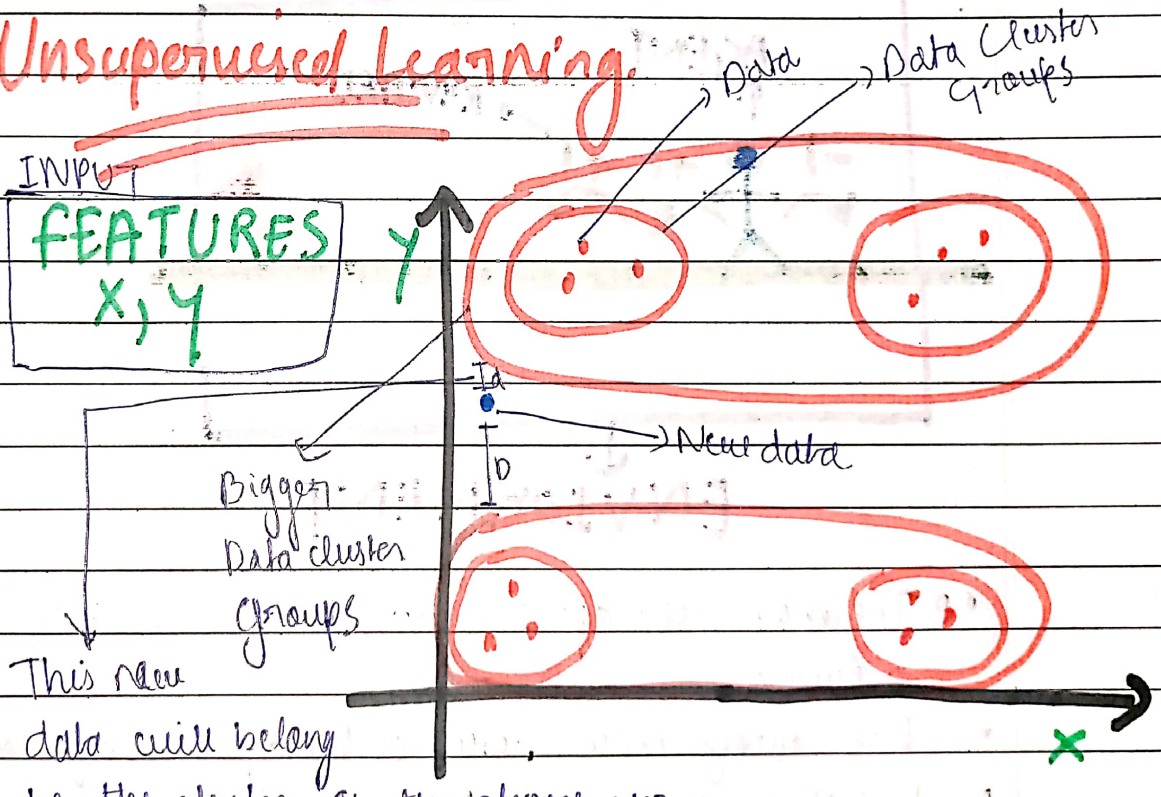
THIS IS REAPEATED UNTIL,

PREDICTED OUTPUT == Expected OUTPUT

Unsupervised Learning.

INPUT

FEATURES X, Y



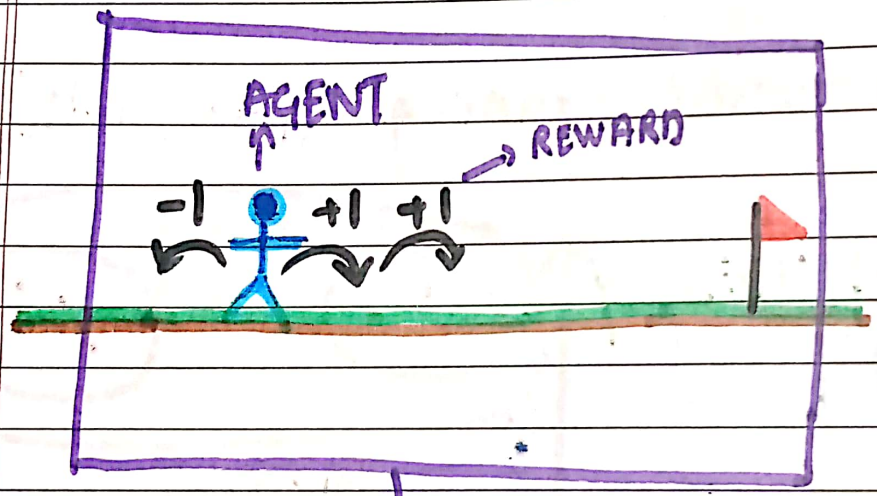
This new data will belong to the cluster on the above as the data is nearest to the top cluster.

→ Escencially groups data and data clusters and figures out where new data will go

REINFORCEMENT LEARNING

- NO DATA
- You ONLY have a AGENT
- An Environment

For example (goal is to reach the flag)



- The more the agent moves towards the flag it gets a Reward
- Each wrong move results in the agent loosing some reward
- The goal of the agent is to get maximum reward which can be only done by reaching the goal.

***GOAL OF THE AGENT IS TO GET MAXIMUM REWARD**

Import Tensorflow as tf

INTRODUCTION TO TENSOR Flow

>>> pip install TENSORFlow

↓

A collection of Data

↓

Making Math easy.

↓

for our.

ai

Tensor :→ Vector generalised in higher dimentions

↓ ↓

Datatypes Represent dimensi of the data

↓

float32, int32, string etc.

How to create tensors

Example

string = tf.Variable("this is a string", tf.string)

number = tf.Variable(324, tf.int16)

floating = tf.Variable(3.567, tf.float64)

Date
Page

Rank and/Degree of tensor.

RANK/DEGREE

↓

DIMENTION

1D ← or scalar — rank0_tensor = tf.Variable("Str", tf.string)

2D ← rank1_tensor = tf.Variable(["Hey", "Hello"], tf.string)

3D ← rank2_tensor = tf.Variable([["test"]], tf.string)

Dimentions can be known by counting the number of nested list

To find the Rank we can use :-

```
tf.rank(rank2_tensor)
```

OUTPUT

<tf.Tensor: shape=(), dtype=int32, numpy=2>

Rank ←

Example = tf.Variable([["A", "B"], ["C", "D"]])

Shape

2D

Demention

Shape of a tensor

- Simply No. elements in each demention

To get shape we use:

Example.shape

OUTPUT

TensorShape([2, 2])

2 elements in 2nd elimation.

2 elements in first Dimation

Reshaping a tensor

tensor1 = tf.ones([1, 2, 3])

Step1

[ ]

Makes big list

Fills two lil list with 3 ones

3, [[1. 1. 1] [1. 1. 1]]

Step 3

Step2

Makes two lil list inside big list

[[ ], [ ]]

tensor2 = tf.reshape(tensor1, [2, 3, 1])

Takes tensor1

Before reshape

[[1.1.1], [1.1.1]]

Reshapes it to [2, 3, 1] shape

After reshape [2, 3, 1] shape

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

2 big list

3 inside list

[2, 3, 1]

1 element each inside

No fist value will be taken as 1

```
tensor3 = tf.reshape(tensor2, [3, -1])
```

-1 tells tensor to calculat the data Demention

$[3, -1] = [3, 3-1] = [3, 2]$

Tensor2 before modification

$[[1.] [1.] [1.]] [[1.], [1.], [1.]]$

New Tensor3

$[[1. 1.] [1. 1.] [1. 1.]]$

Types of Tensor:

- Variable → Mutable
- Constant
- Placeholder → Immutable
- SparseTensor

Evaluating a tensor

Get value of a tensor (uses a graph)

↓

we need sessions to eval tensor

```
With tf.session() as sess:
    tensor.eval()
```

↓

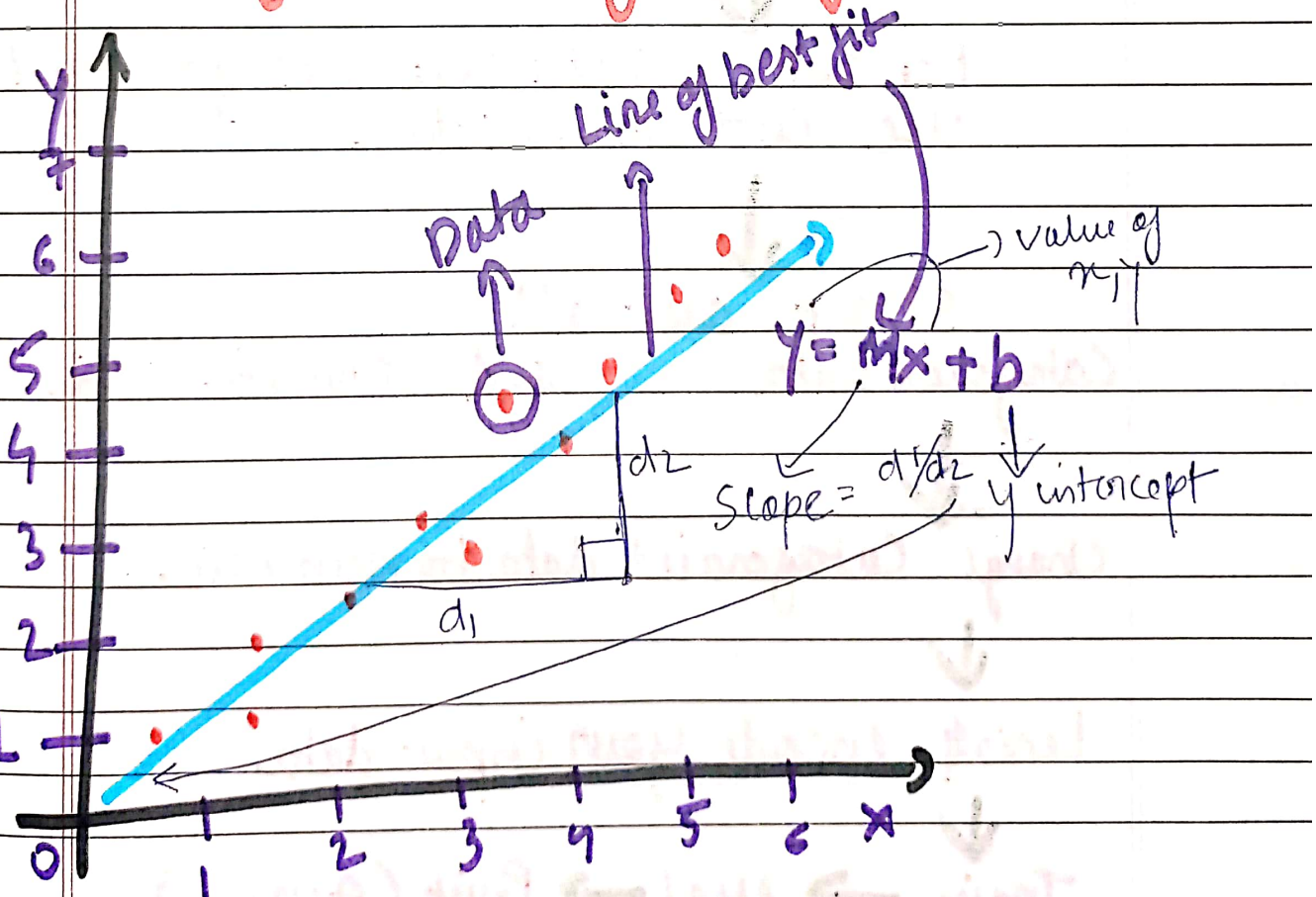
Name of tensor

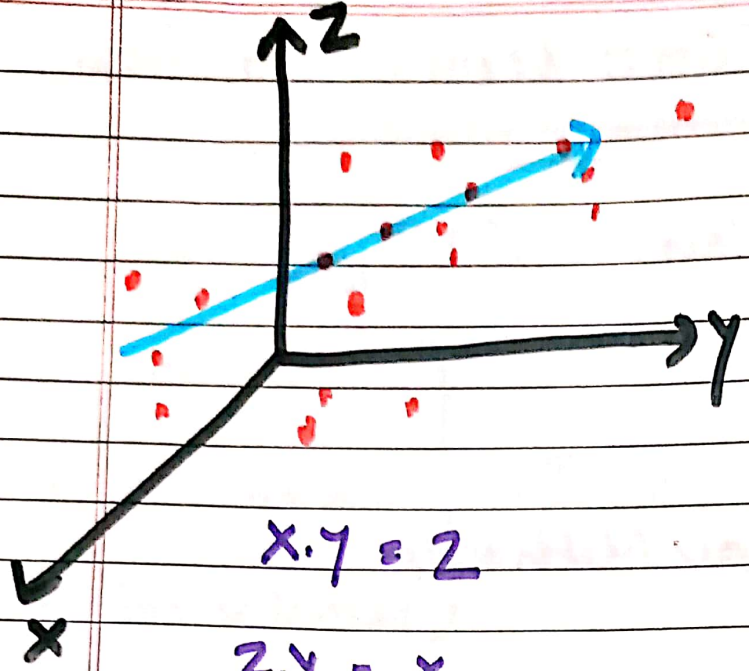
Tensorflow CORE learning algorithms

- Linear Regression
- Classification
- Clustering
- Hidden Markov Methods

Linear Regression

Takes linearly correlated Data to find the line of Best fit and predict using the line of best fit





$x.y = z$

$z.y = x$

In CODE

Take a .csv (datasheet)

↓

Pop out all the elements you need to use into a variable

↓

Short out all Categorical Data and Numeric data.

↓

change Categorical data to numeric.

↓

Label encode your input data.

↓

Train → Test → Print (Accuracy)

CLASSIFICATION

Date
Page

CLASSIFIES DATA INTO DIFFRENT DATA SET (CLASSES)

TAKE IN CODE

→ CLASSIFIER TYPE

- DNNCLASSIFIER (DEEPNUREAL NET)
- LENIAR CLASSIFIER

Take the csv file

↓

specify the class types

↓

pop out the coloumbs you need

↓

Make an input function

↓

Make feature coloumb

↓

Make the model

```
Classifier = tf.estimator.DNNClassifier(
    feature_columns = my_feature_columns,
    hidden_units = [30, 10]
    n_classes = 3)
```

- hidden_units = [30, 10] → Number of Nodes in 2nd layer; Number of Nodes in 1st layer of Neural Net
- n_classes = 3 → Number of Classes

Date
Page

↓

Train Model

```
classifier.train(
    input_fn = lambda: input_fn(train, train_y,
    Training = True, steps = 5000)
```

Number of times to be repeated

lambda defines a single line anonymous function in python

↓

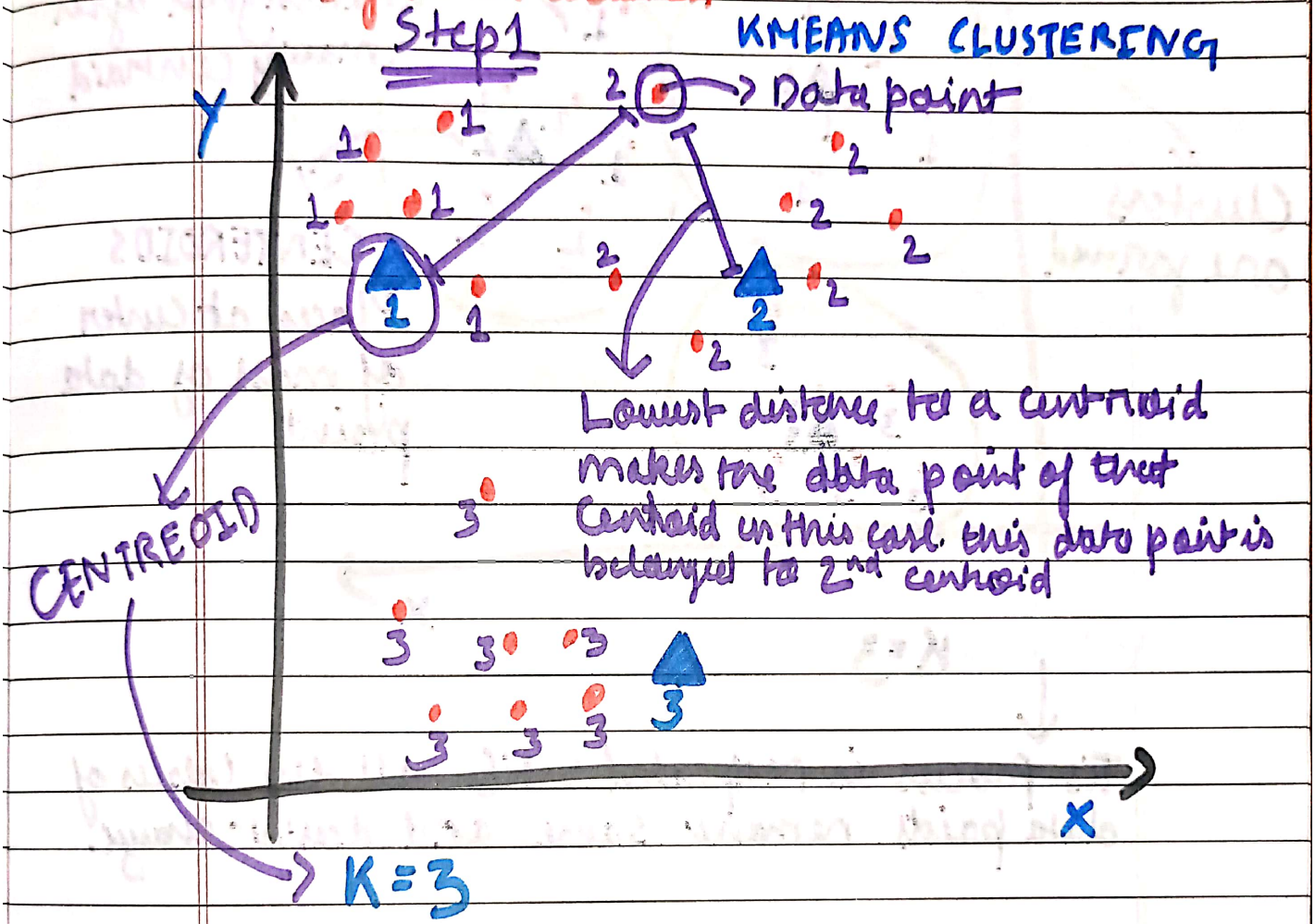
eval

↓

Predict / Accuracy

CLUSTERING

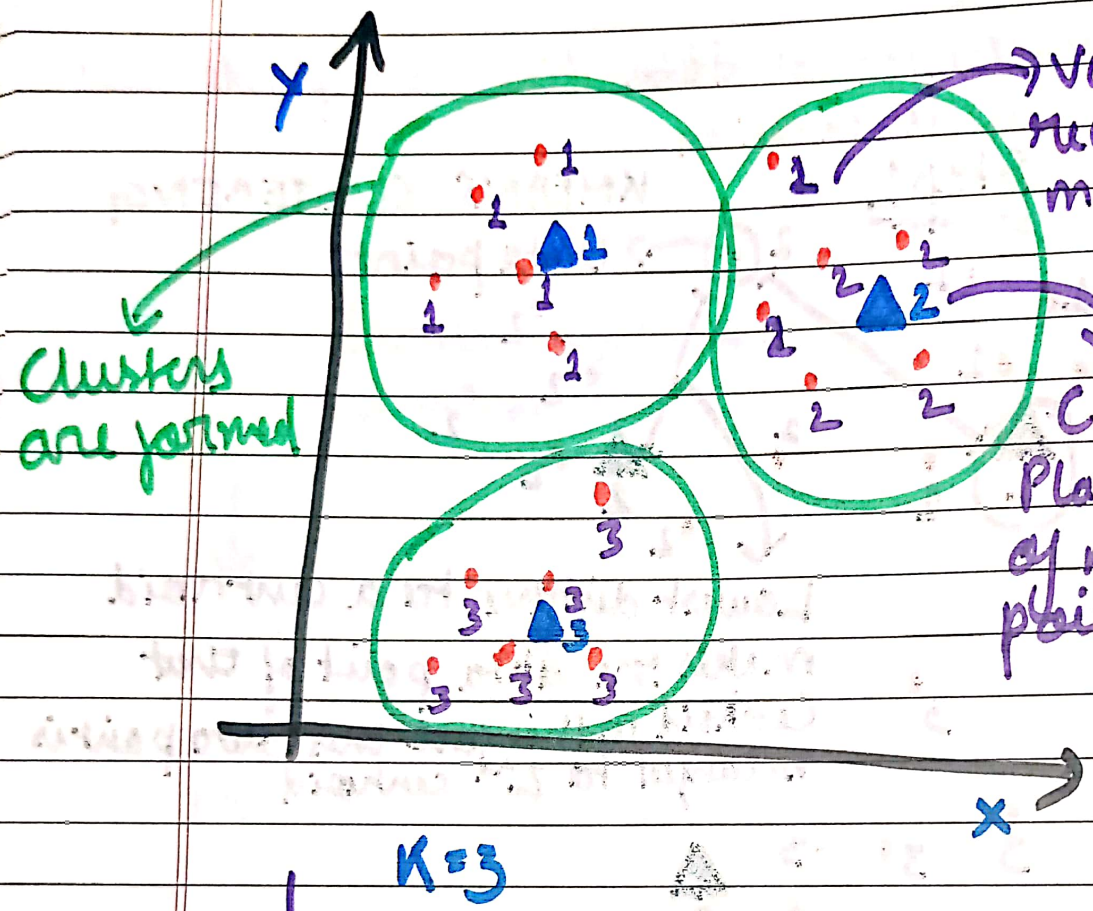
- Unsupervised learning algorithm.
- Used when you only have input info and No labels (output) info.
- Finds cluster of ~~data~~ like data points and tells you its location



K = Number of clusters

Step 2

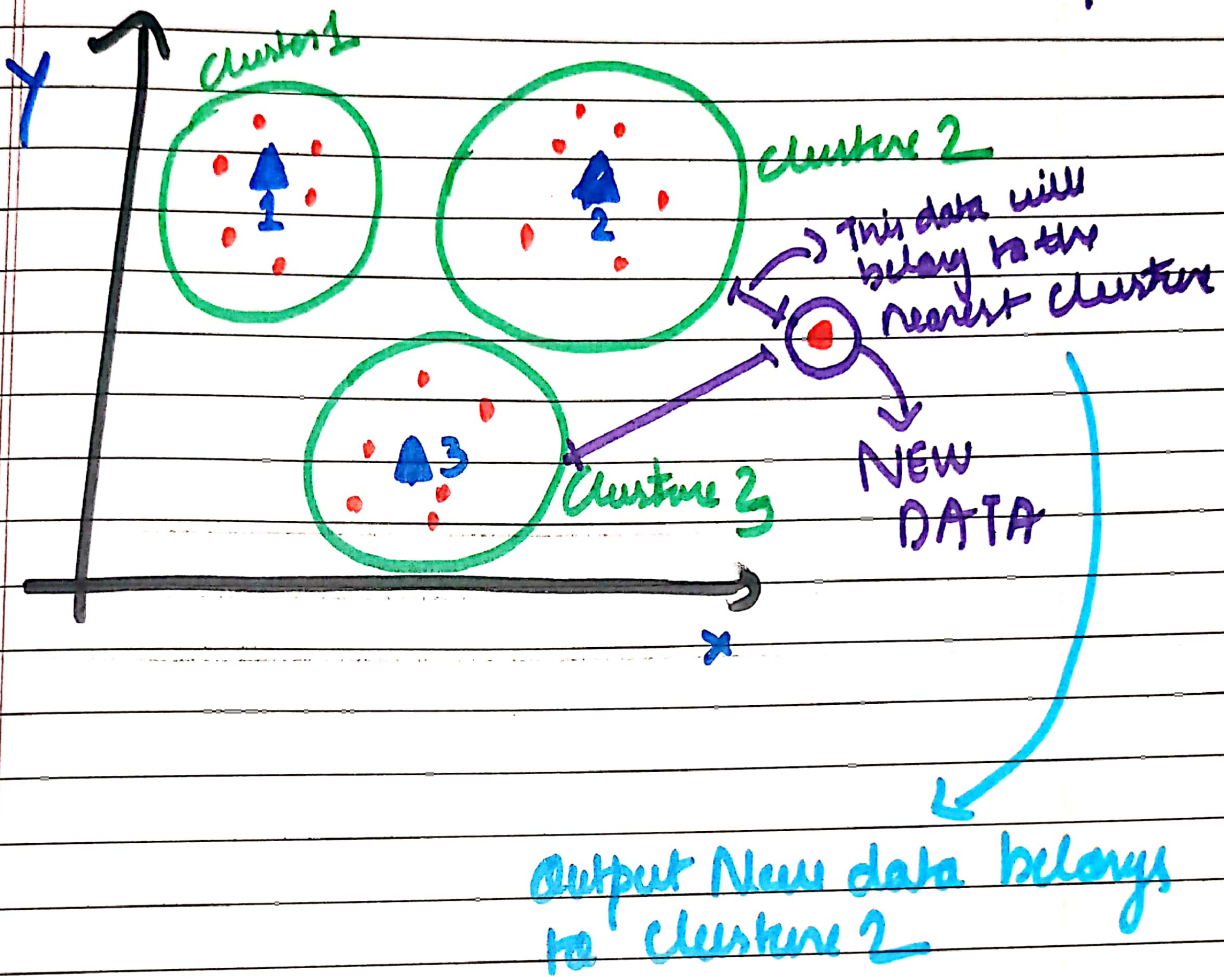
Finds center of mass of Datapoints and places centroids in the middle of them



This process is repeated until all the labels of data points remain same and doesn't change

Step 3

Once clusters are formed the new data are assigned to the nearest cluster and labeled the label will be the output



HIDDEN MARKOV MODEL

- finds probability of someting happening based on observations

↓

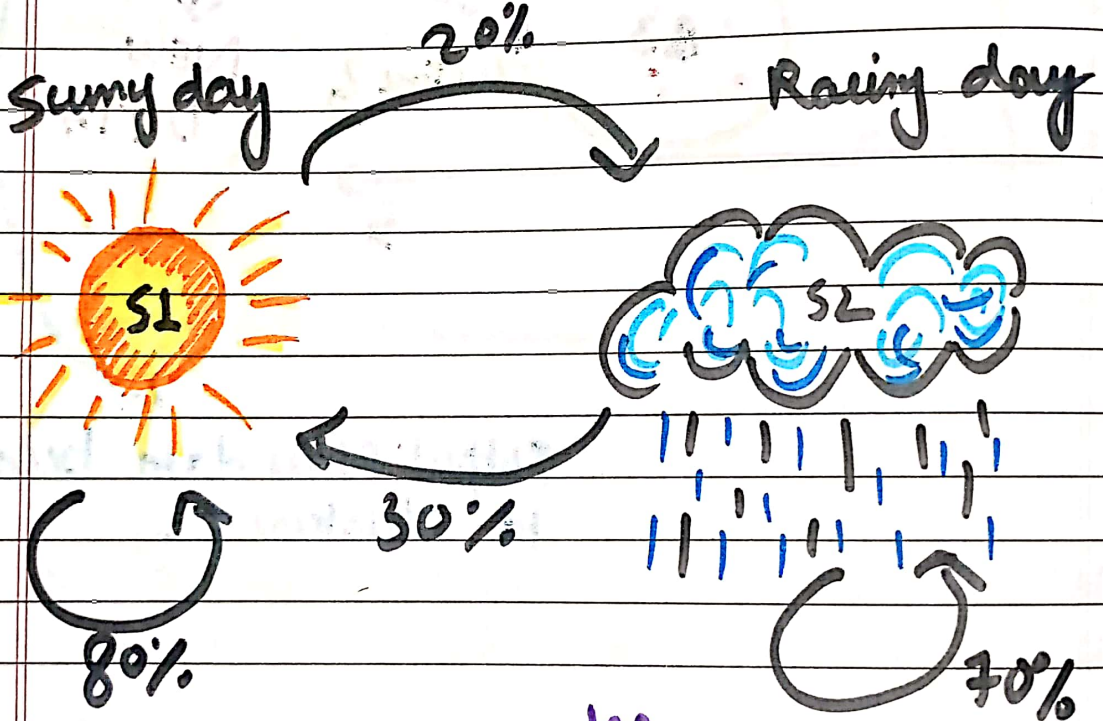
- States

↓

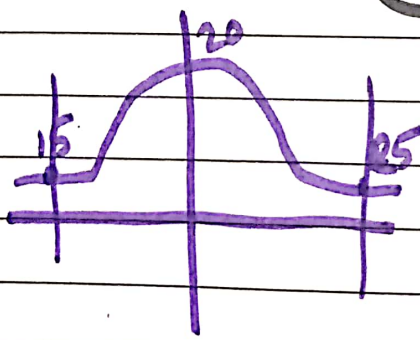
- Observation Distribution

↓

- Transition Distribution



Observation:
Mean: 20
Min: 15
Max: 25



Observation:
Mean: 5
Min: -5
Max: 15

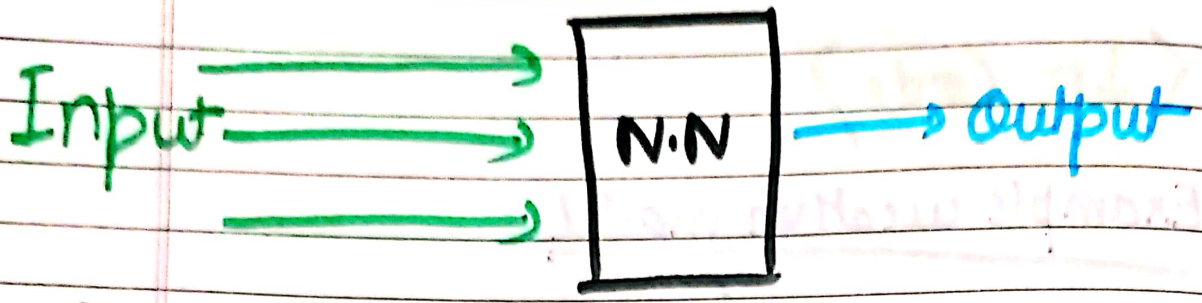
Purpose: Predict future event based on passed events

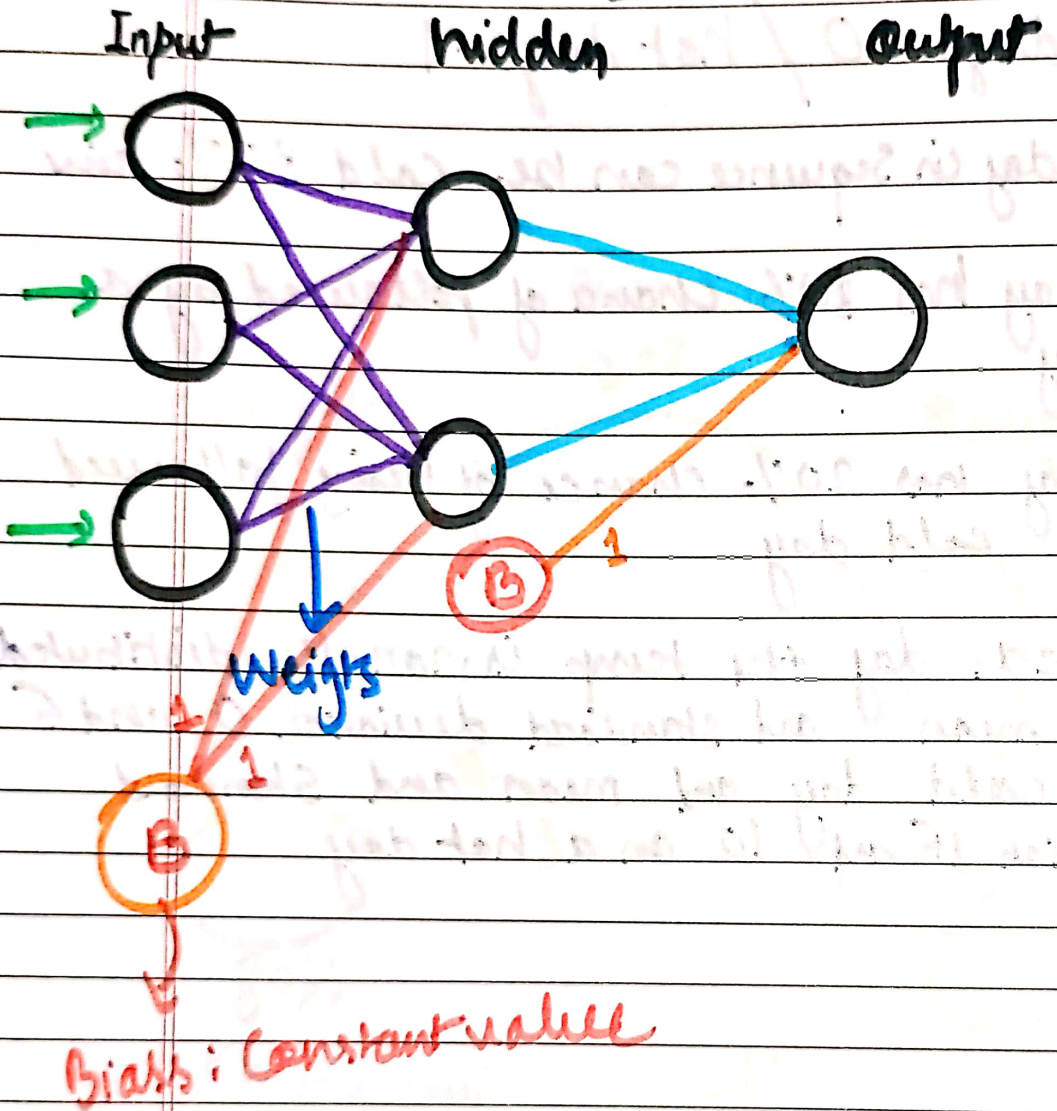
<In Code>

Example weather model

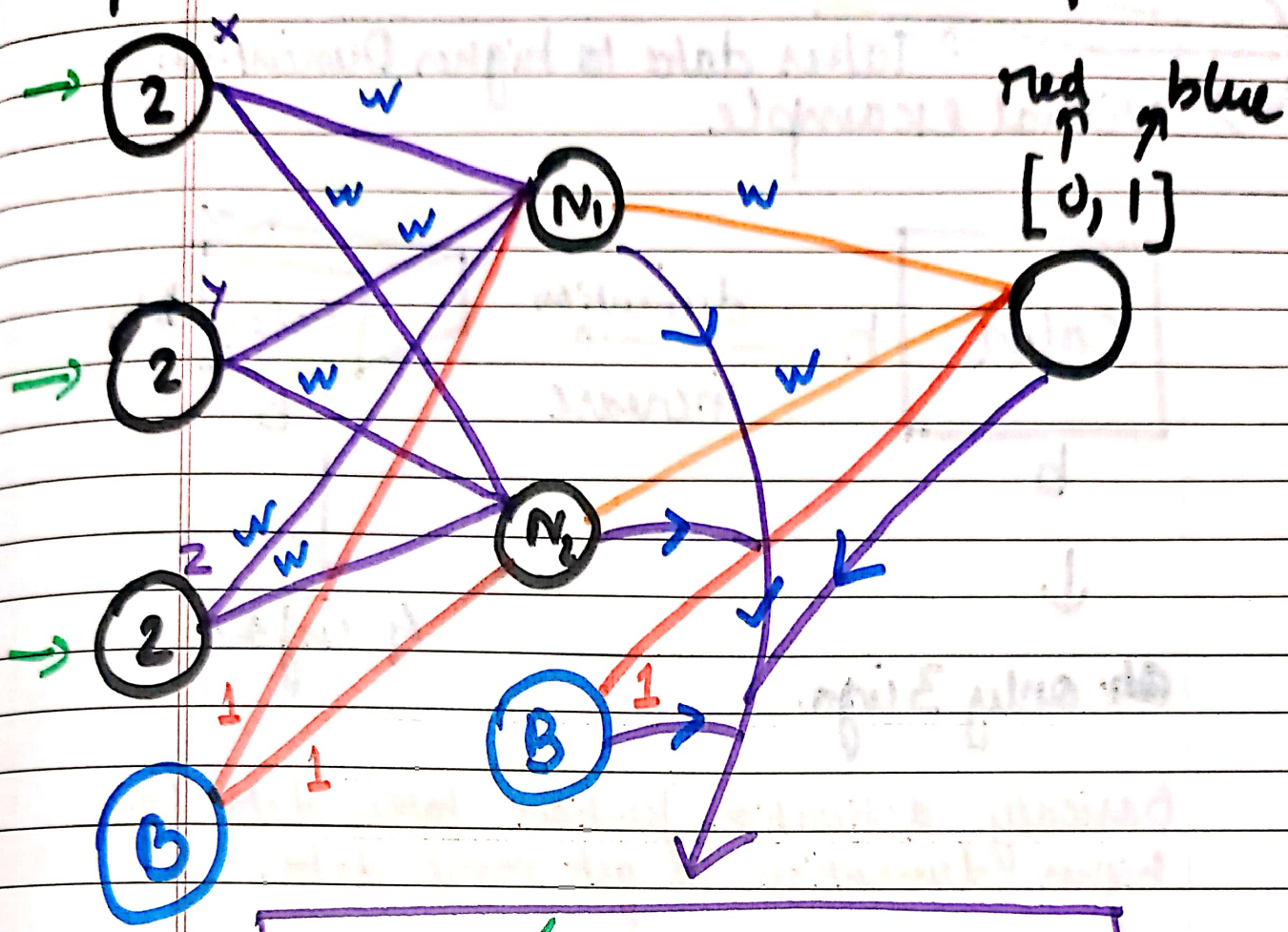
1. Cold days = 0 / hot days = 1
2. First day in sequence can be cold 80% time
3. Cold day has 30% chance of followed by a hot day
4. Hot day has 20% chance of being followed by a cold day
5. On each day the temp is normally distributed with mean and standard deviation 0 and 5 on a cold day and mean and standard deviation 15 and 10 on a hot day

NEURAL NETWORKS





Example: $(X, Y, Z) \rightarrow$ [red, blue]



$$N_x = F\left(\sum_{i=0}^{n} w_i x_i + b\right)$$

activation function

$= w x(2) + w y(2) + w_2(2)$

$= V$

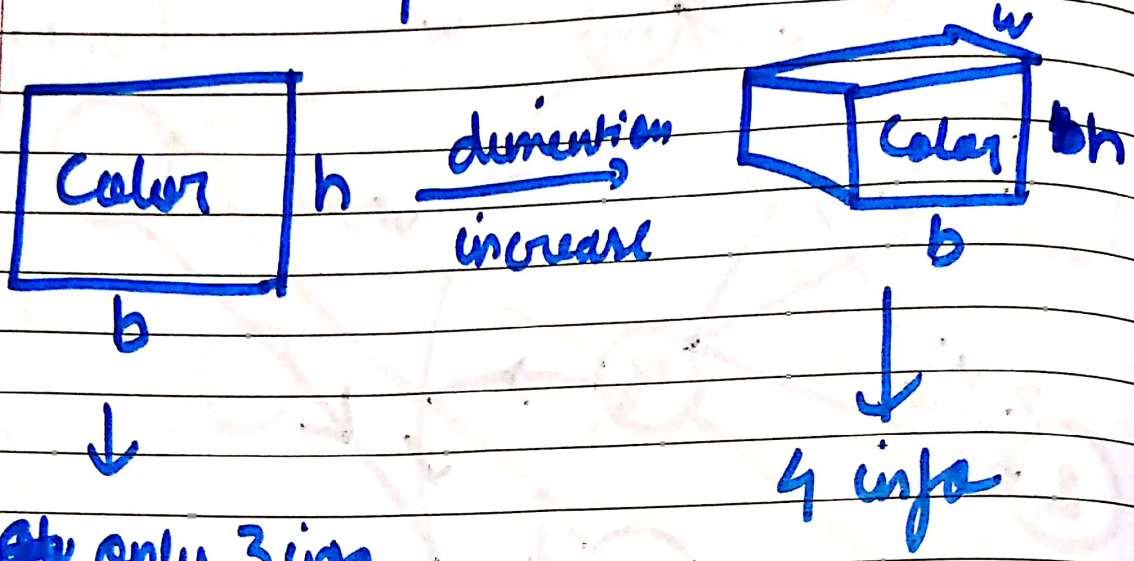
↳ Next page

Date
Page

Activation funtion

→ Takes data to higher Dimention

visual example



only 3 info

4 info

Basically activation funtion takes data to higher dimention to get more data.

Loss funtion

Calculates how far output is from expected output and suggest changes