K-Nearest Neighbours (KNN)

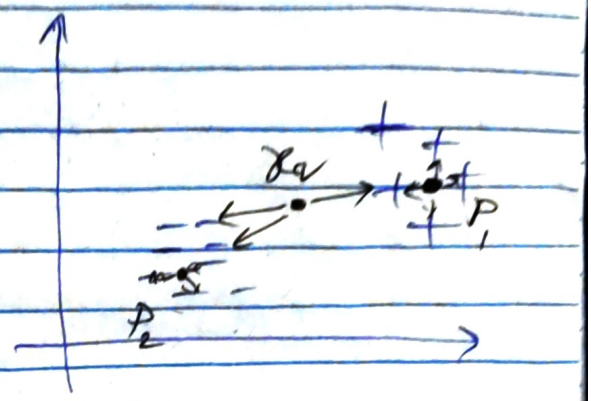
KNN :-
↳ classification

$k = 3$

P_1 :→

we need to take

3 nearest points to P_1



$P_1 \Rightarrow [+ve, +ve, +ve]$

⇒ all the data points near to P_1 are +ve's; So P_1 belongs to +ve data point

(Note:- Maximum Voting

P_2 :→

$P_2 = [-ve, -ve, -ve]$

all are -ve points. i.e it has maximum votes for -ve point (Majority Voting)

⇒ P_2 belongs to -ve data point

Task:- Given any query point x_q we want to find its class by looking at its K-Nearest neighbours.

↳ We will consider K-Nearest data points to x_q (which are nearest to x_q) by euclidean distance and we will do Majority Voting.

Algorithm:→

↳ Given → D_{Train} , K, x_q

D_{Train} ↓ training data

K ↓ # of Neighbours

x_q ↓ find the class for the query point.

Lets $K=3$ (How to find best K we will discuss later)

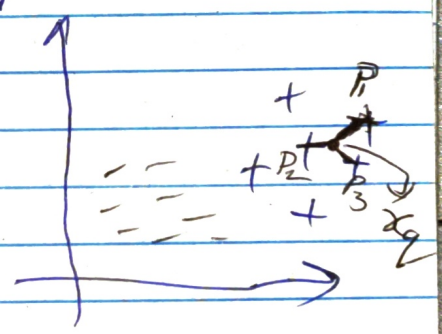
Step 1:→ $K=3 \Rightarrow$ 3 nearest neighbours ~~for~~ to x_q

(Here we are not training any data, directly we are testing it)

$\Rightarrow$ We will get K-Nearest Neighbours to x_q

Step 2:→

we will get the class for K-Nearest points & do the Majority Voting



Here

K-Nearest Neighbours are P_1, P_2 and P_3.

$\Rightarrow$ $P_1, P_2, P_3 \rightarrow$ belongs to +Ve class.

$\Rightarrow$ $x_q = [+ve, +ve, +ve]$

+ve is having 3

As per Majority Voting x_q is positive

Q) How to find the K-Nearest Neighbours

A → Euclidean distance/...

Here Computer has to know which are the nearest neighbours.

↳ So it will calculate distance from ~~each~~ query point to each data point.

↳ Now i will sort ~~it~~ the distance in Ascending order.

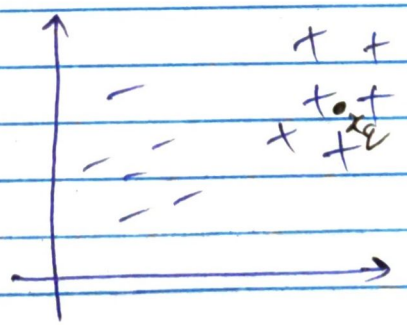
↳ Now i know which data points having minimum distance to the query point(

↳ Now identify the class it belongs to and do the Majority Voting.

Ex:→

1 → 150
2 → 5
3 → 0.5
4 → 100
5 → 50
⋮
999 → 0.1
1000 → 0.9

Sort asc



→ For each datapoint, i am calculating the distance from x_q.

(Here i am having 1000 datapoints

$= [0.1, 0.5, 0.9, 5, \ldots\ldots, 150]$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

999 3 1000 1

data points

$= [999, 3, 1000]$

$\downarrow \quad \downarrow \quad \downarrow$

[+ve +ve +ve] $\Rightarrow$ +ve = 3, −ve = 0 } → Majority Voting

$\downarrow$

$x_q = +ve$

KNN:→

Given → D_{Train}, K, x_q

1. Initialize distance = [ ]

2. for each datapoint in D_{Train}:

Compute distance of x_q from dataset

(and we will store in an array)

distance.append (distance, index of datapoint)

O/P of 2.→

[(150,1), (5,2), (0.5,3), (100,4), ... (0.1,999), (0.9,1000)]

(refer to the prev page)

3. Sort the distance list according distances

[(0.1,999), (0.5,3), (0.9,1000), ... (150,1)]

Index of D_{Train}

Distance of x_q from Data point in D_{Train}

4. pick First K tuples from distance list

K nearest neighbors = distance[:3]
(lets K=3)

5. Initialize

pos_count = 0
neg_Count = 0

for neighbour in nearest_neighbors:
if class is +ve
↳ pos_count += 1
Else
neg_Count += 1

6. if (pos_count > neg_Count):
+ve class
Else
-ve class.

** Note:- So, In this algorithm there is no test data.
↳ It Completes training data (memorizes the complete training data)
↳ No, need of train test seperation

entire training data ↳ KNN → memorizes the complete data & Stores it in Ram

Testing m,c ← Logistic Reg → Min log Σ Signed distance

Testing m,c ← Linear Reg → Min Sum of Squared error