K-Nearest Neighbours

How classification works?

~~$y_i \rightarrow$ Text $\rightarrow$ vector~~

prediction of a class wheather +ve or -ve

$$\boxed{y_q = f(x_q)}$$

Notations

✧ Each vector by default is column vector

✧

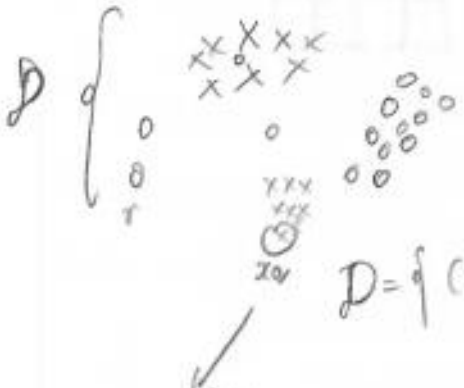
$$D_n = \{ (x_i, y_i)_{i=1}^{n} \mid x_i \in \mathbb{R}^d, y_i \in \{0, 1\} \}$$

(0: -ve, 1: +ve; | : such that)

K-Nearest Neighbours:

2D - toy datasets :

Binary classification



X : +ve data pt.

O : -ve data pt.

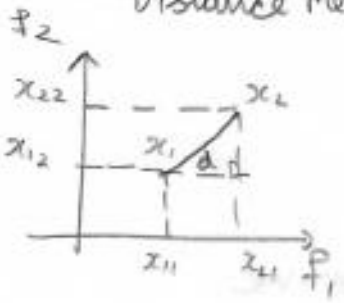
$$D = \{ (x_i, y_i) \mid x_i \in \mathbb{R}^2 \quad y_i \in \{0, 1\} \}$$

(0: -ve, 1: +ve)

query pt.

Take pt close to this pt $\quad x_q \rightarrow y_q$

"Distance Measures



$f_1 \quad f_2$

$x_1 = (x_{11}, x_{12})$

$x_2 = (x_{21}, x_{22})$

d = len of shortest line from x_1 to x_2

✧ Euclidean distance

$$d = \sqrt{(x_{21}-x_{11})^2 + (x_{22}-x_{21})^2}$$

$$= \underbrace{||x_1 - x_2||}_{\text{length}}$$

$x_1 \in \mathbb{R}^d, x_2 \in \mathbb{R}^d$

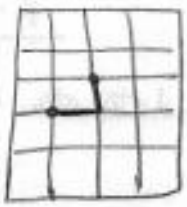
Euc-distance: $||x_1 - x_2||_2 = \left(\sum_{i=1}^{d} (x_{1i} - x_{2i})^2\right)^{1/2}$

$||x_1 - x_2||_2 \longrightarrow$ L2 norm

$||x_1||_2$ = euc dist from origin $= \left(\sum_{i=1}^{d} x_{1i}^2\right)^{1/2}$

✧ Manhattan distance:

$$\sum_{i=1}^{d} \overbrace{|x_{1i} - x_{2i}|}$$



L1-norm vector $(x_1 - x_2)$

$||(x_1 - x_2)||_1$

$||x_1||_1 = \sum_{i=1}^{d} \overbrace{|x_{1i}|}^{\text{absolute}}$

L_p norms → Minkowski distance

$$\|x_1 - x_2\|_p = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p\right)^{1/p}$$

$P = 2$ → Minkowski distance → Euclidean distance

$P = 1$ → Minkowski distance → Manhattan distance

$$\|x_p\| = \sum_{i=1}^{d} \left(|x_i|^p\right)^{1/p} \qquad P \neq 0, \; P > 0$$

$$\text{Euc-dist}(x_1, x_2) = L2 \text{ norm of } (x_1 - x_2)$$
$$= \|x_1 - x_2\|_2$$

Distance are between 2 pts → $\text{Eucdistan}(x_1, x_2) = \|x_1 - x_2\|$
Norms is for a vector → L2, L1 norms

☆ Hamming distance

Test passing and boolean Vectors → Binary Bars

$x_1 = [0, 1, 1, 0, 1, 0, 0, \ldots]$
$x_2 = [1, 0, 1, 0, 1, 0, 1, \ldots]$

Hamming dist (x_1, x_2) = # locations / dimensions where binary vectors differ.

In x_1 & x_2 Hamming = 3

$x_1 =$ a b c a d e f g h i k ← Gene code / seq
$x_2 =$ a c b a d e g f h i k ←

Hamming dis $(x_1, x_2) = 4$

☆ Cosine distance & Cosine Similarity

Similarity · distance x_1, x_2

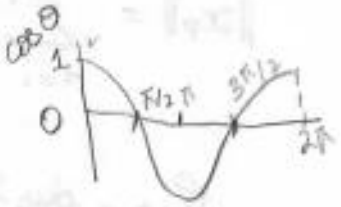
↓ dec ⟶ ↑ inc

↑ inc ⟶ ↓ dec (opposite)

$$1 - \text{cossim}(x_1, x_2) = \text{cos-distance}(x_1, x_2)$$

↓

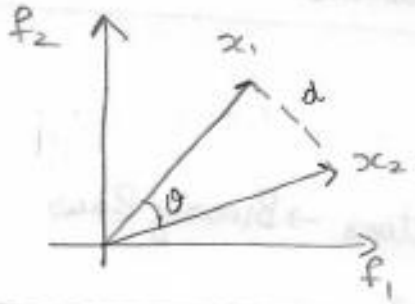
$[-1, 1]$



Cos-sim$^{\text{distance}}(x_1, x_2)$ very similar $= +1$

Cos-sim$^{\text{distance}}(x_1, x_2)$ very dissimilar $= -1$

☆ what is cosine similarity



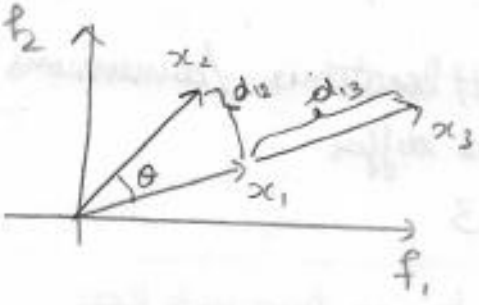
d = euclidean distance

Cos-similarity $= \cos\theta$

(x_1, x_2)

θ : angle between x_1 & x_2

? Difference between Euclidean distance and cosine similarity



Cos-sim $(x_1, x_2) = \cos\theta$

Cos-sim $(x_1, x_3) = \cos 0° = 1$

↓

but Euclidean distance

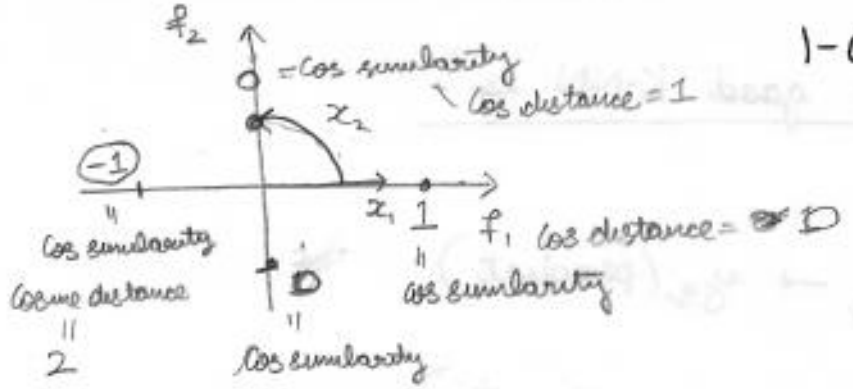
$d_{13} > d_{12}$

↳ Cos-distance $(x_1, x_3) = 1 - 1 = 0$

Cos-distance $(x_1, x_2) = 1 - \cos\theta$

NOTE

$d_{13} > d_{12}$ on the other hand Cos distan $(x_1, x_3) <$ Cosdis (x_1, x_2)



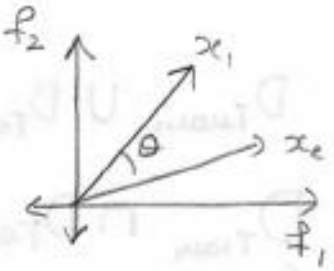
$1 - \text{cos sim}(x_1, x_2) = \text{cost dist}(x_1, x_2)$

angle (x_1, x_2)	dist
$0 - 90°$	$0 \to 1$
$90° - 180°$	$1 \to 2$
$180° - 270°$	$2 \to 1$
$270° - 360°$	$1 \to 0$

$1 - \cos 90° = 1$ (cos distanc
$1 - \cos 180° = 2$ (cos sim dis
$1 - \cos 270° = 1$ (cos simila
$1 - \cos 360° = 1$ (cos dis

$$\cos\theta = \frac{x_1 \cdot x_2}{\|x_1\|_2 \, \|x_2\|_2}$$

$\|\cdot\|_2$: L2 norm



① If x_1 & x_2 are unit vector

$$\|x_1\| = \|x_2\| = 1$$

$$\boxed{\cos\theta = x_1 \cdot x_2}$$

? Relationship between Euclidean & cos-similarity / distance

$$\left[\text{Euclidean dist}(x_1, x_2)\right]^2 = 2\,(\underbrace{1 - \cos(\theta)}_{\text{cos dist}}) = 2\,(\text{cost dist}(x_1, x_2))$$

$\downarrow$ cos sim

If x_1 & x_2 are unit vectors

Using
From $\|A-B\|^2 = (A-B)^T(A-B) = \|A\|^2 + \|B\|^2 - 2A^TB \Rightarrow \|A\| = \|B\| = 1$ (cosine similarity)

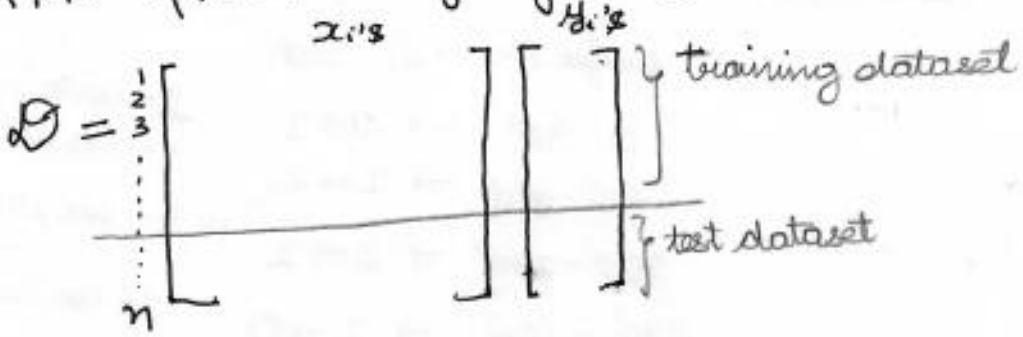
$$= 2 - 2A^TB = 2(1 - A^TB) = 2(1 - \cos(A,B))$$

✧ 'Measuring how good K-NN is'

Amazon

Given $x_q \rightarrow y_q$ (predict)

Measure: KNN $\rightarrow$ (K-NN + majority vote)



$\mathcal{D}_n \rightarrow \mathcal{D}_{Train}$

$\mathcal{D}_n \rightarrow \mathcal{D}_{Test}$

$$D_{Train} \cup D_{Test} = D_n$$

$$D_{Train} \cap D_{Test} = \phi$$

For example

$\mathcal{D}_{Train} \rightarrow n_1 \quad n_1 + n_2 = n$

$\mathcal{D}_{Test} \rightarrow n_2$

Split $\mathcal{D}_n$ (x_i) $\rightarrow$ $\mathcal{D}_{Train}$ (70%), $\mathcal{D}_{Test}$ (30%) — randomly

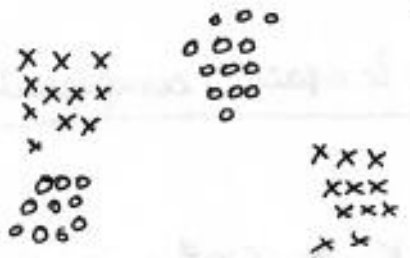
$n_1 \approx 0.7n$

$n_2 \approx 0.3n$

D_n $\rightarrow$ n_1 $\mathcal{D}_{Train}$ $(x_i, y_i)_{i=1}^{n_1}$ $\rightarrow$ KNN

D_n $\rightarrow$ n_2 $\mathcal{D}_{Test}$ $(x_i, y_i)_{i=1}$

D_{train}



D_{train}

---

each point x_i in $D_{test} = \{(x_i, y_i)_{i=1}^{n_2}\}$

$x_q = x_1$ for each point in D_{test}

y_q = Blue (+ve)

→ x_q = pt

→ use D_{train} & KNN to predict y_q

Algorithm:

```
count = 0;
for each pt in D_test:
        x_q = pt
        use D_train & K-NN to determine y_q

        if y_q == y_pt
                count += 1
```

number of points for which D_{train} + KNN gave correct class label.

$$\text{Accuracy} = \frac{\text{count}}{n_2}$$

↙ # pts in D_{test}

$0 \leq Acc \leq 1$

Accuracy = 0.91 ⇒ 91 % of times

$x_q \rightarrow y_q$

Test / Evaluation time & space complexity :

$x_q \rightarrow y_q$

Input : D_{Train}, K, $x_q \in \mathbb{R}^d$; output : y_q

KNN pts = [ ]

n pts (samples) ; d dimensional (features)

for each x_i in D_{train} :

$O(d)$ - Compute $d(x_i, x_q) \rightarrow d_i$

$O(k)$ - Keep the smallest K-distances → KNN pts [ ] → (x_i, y_i, d_i)

Time complexity

K is small → 5 or 10

```
count_pos = 0 ; count_neg = 0
for each x_i in KNN pts
    if y_i is +ve
        count_pos += 1
    else
        count_neg += 1
if count_pos > count_neg
    return y_q = 1 → +ve
else
    y_q = 0 - ve
```

Small ↑ O(K) ↓ O(1)

Time complexity :- $O(nd) + O(1) + O(1)$

$O(nd)$

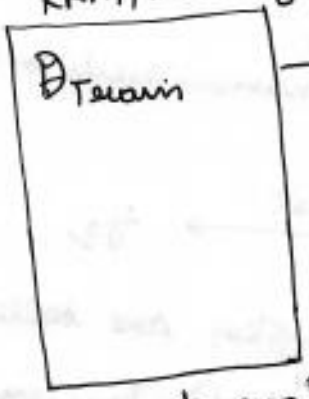
if d is small

if $d << n$

$O(n) \rightarrow$ Time

deploy :

$x_q \rightarrow y_q$



Space complexity
$O(nd)$
Testing phase

It will take lot of memory. at evaluation time ~~or~~ and run time.

✧ evaluation / Test phase
Time complexity :- $O(nd)$
Space complexity :- ~~same~~ space that us need to evaluate $O(nd)$

KNN limitations

(Amazon) Fine Food reviews :

Production → live system

Time :- $O(nd)$
Space :- $O(nd)$

$n \approx 364K$ (Samples)
$d \approx 100K$ (features)
→ D_{Train}

$n \times d = 36,400M$ = 36 GB → RAM

64GB of RAM needed

① Large space complexity

2.) Time Complexity : 36 Billions Computations

Review of product in amazon → 1 msec → +ve / -ve

This number needs to be small.

x_q → fast → y_q

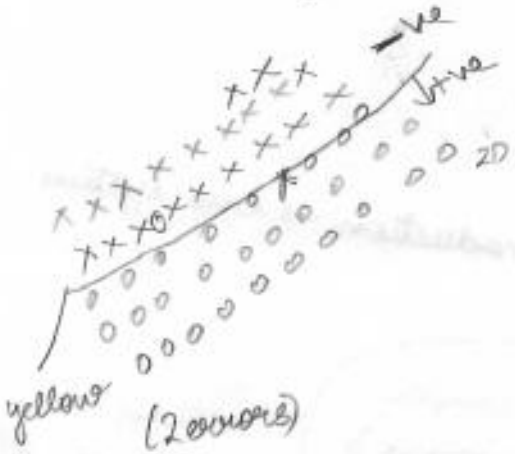
- such system are called low latency system.

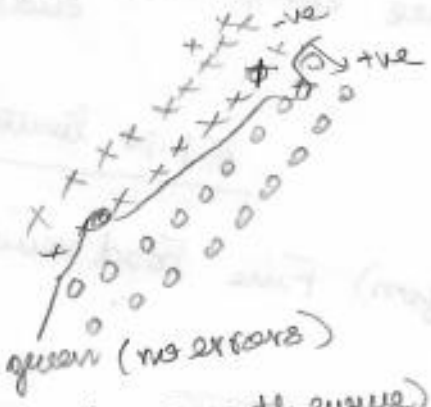
O(nd) O(nd) Time complexity is high.

Decision surface for KNN as K Changes

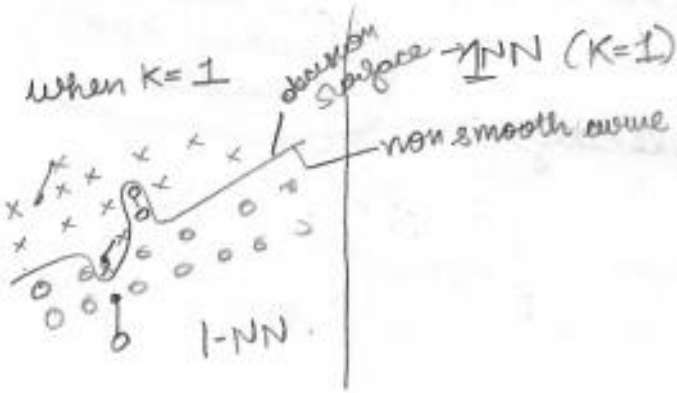
? How to pick right K ?

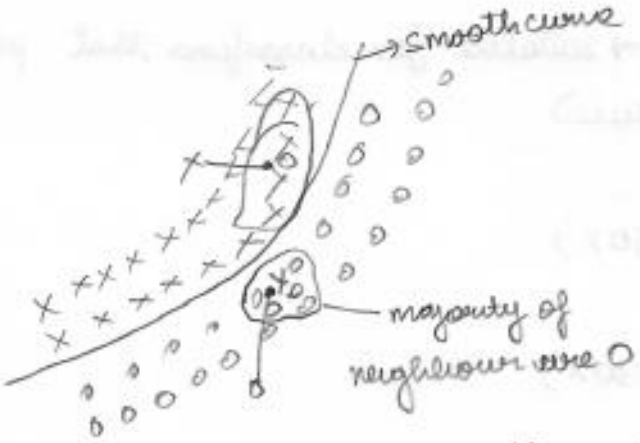
'K' is also refered as hyperparameter.





decision surfaces yellow and green.





Let k=5.

5-NN

5 neighrest neighbour' majority vote

NOTE: K increases smoothness of curve increases

In K nearest neighbour. the smoothness of the decision surfaces increases as K increases

lets say K=n

K = 1, 2, 3, 4, ... n

max value of K= n.

n is total number of pts

n = Total number of points

1000 ↙

600 ← n_1 = +ve

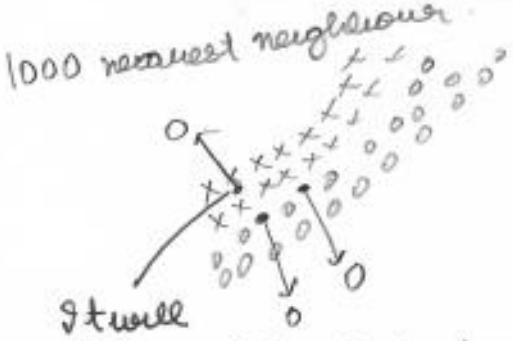
400 ← n_2 = -ve

$n_1 + n_2 = n$

lets say $n_1 > n_2$

600 > 400

1000 nearest neighbour



It will be misclassified as O number of $n_1 > n_2$ so it will be considered O. instead of x.

K = n = 1000

n_1 = +ve = 600

n_2 = -ve = 400

$n_1 > n_2$

600 > 400

NOTE: When K becomes n than the everything becomes majority class.

Overfitting & underfitting

K = 1

K = 5

Need For Cross Validation → suitable for classifiers that predict labels (positive or negative)

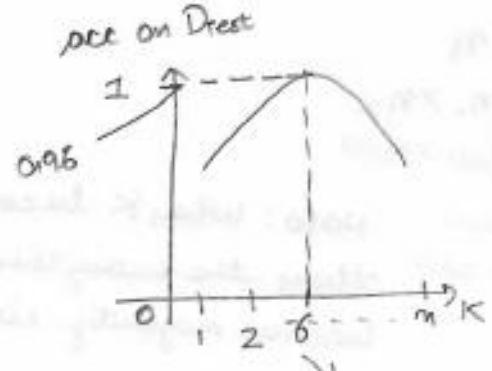
How to determine K

One idea

→ num of correctly classified pts

	Train	accuracy D_{test}
K = 1	D_{train}	0.78
K = 2		0.82
K = 3		0.85

$$D_{Test} = \left\{ (x_i, y_i) \mid x_i \in \mathbb{R}^d, y_i \in \{0,1\} \right\}_{i=1}^{n_{test}}$$



(Typically)
Calculating value of K using test data

best accuracy on my D_{test} when using D_{train} as training data

K = 6 (6-NN)

96%

{ using d_{train} & 6-NN on amazon food review dataset get accuracy of 96%

Small problem

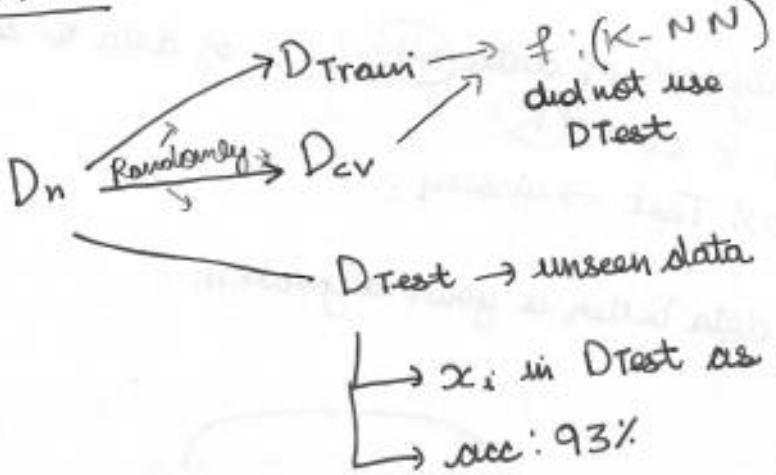
Objective → D_n → f (D_{train}) (6-NN) → accuracy ($D_{testing}$)

future, unseen point $x_q \rightarrow (y_q)$

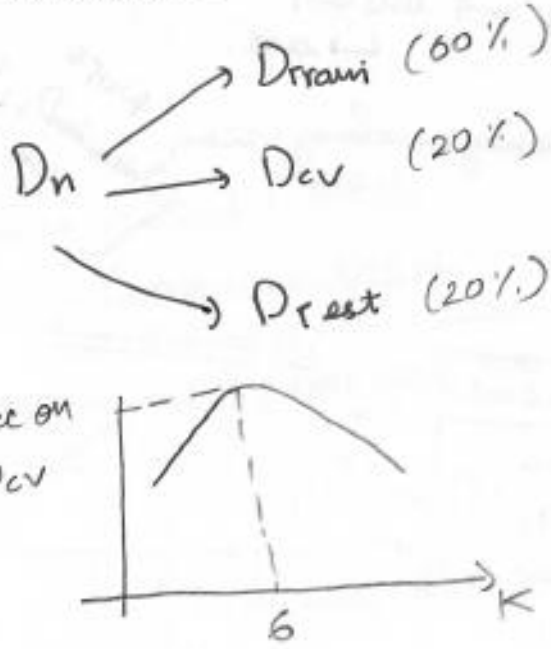
Objective → accurate on future, unseen point ~~than it~~

Generalization → when algorithm does well on future unseen point not only for D-test.

CROSS-VALIDATION (CV)



Yes { Can I now say that 6-NN has an accuracy of 93% on unseen data ?



NOTE first we were finding value of k using test set but now we use cross validation to find best value of K

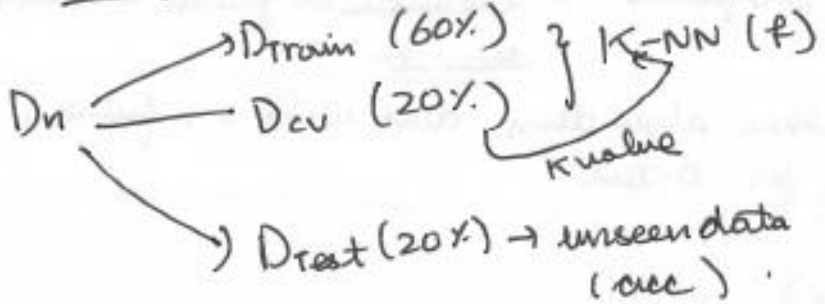
→ k = 6

DTest

93% accuracy on test data on future on unseen data

What is right K'

K-fold cross validation



Problem: only 60% of data (NN) } 80% of data to compute NN data
20% CV → (K)
20% Test → unseen

Rule: More training data better is your algorithm.

K-fold CV:

① D_n → D_{train} (80%) → f(K-NN)
D_n → D_{Test} (20%) → unseen → acc

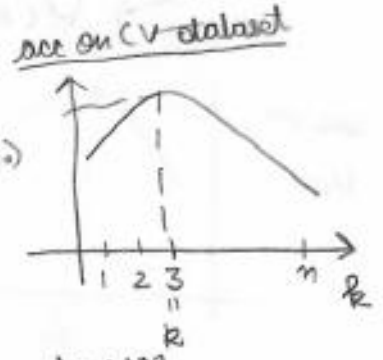
② D_{train} randomly breaking D_{train} into 4 equal size parts: D_1 | D_2 | D_3 | D_4 (1 2 3 4)

(NOTE Intersection of D_1, D_2, D_3 & D_4 are disjoint)

③

	Train	CV	acc on CV
K=1	D_1 D_2 D_3	D_4	a_4
K=1	D_1 D_2 D_4	D_3	a_3
K=1	D_1 D_3 D_4	D_2	a_2
K=1	D_2 D_3 D_4	D_1	a_1

avg (a_1, a_2, a_3, a_4) = acc of K=1



acc on K-NN for k=1 on CV dataset

D_1 D_2 D_3 D_4 each of 20% data

4 times

	train	CV	acc on CV
K=2	$D_1 D_2 D_3$	D_4	a_4
K=2	$D_1 D_2 D_4$	D_3	a_3
K=2	$D_1 D_3 D_4$	D_2	a_2
K=2	$D_2 D_3 D_4$	D_1	a_1

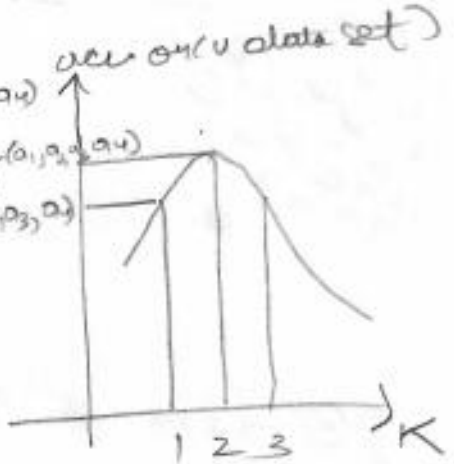
} avg of (a_1, a_2, a_3, a_4)

$a_{K=2}$

K=3	$D_1 D_2 D_3$	D_4	a_4
K=3	$D_1 D_2 D_4$	D_3	a_3
K=3	$D_1 D_3 D_4$	D_2	a_2
K=3	$D_2 D_3 D_4$	D_1	a_1

} avg of (a_1, a_2, a_3, a_4)

$a_{K=3} = \text{avg}(a_1, a_2, a_3, a_4)$



4 times

→ This is called 4 fold-
- Cross validation

Suppose K=3 you got the best accuracy.

f : 3-NN ← Dtrain

→ avg-acc on 4 factor.

K' fold CV → Dtrain → NN

→ K in K-NN

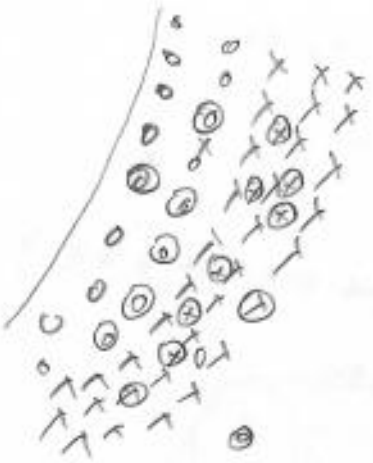
What is right K'

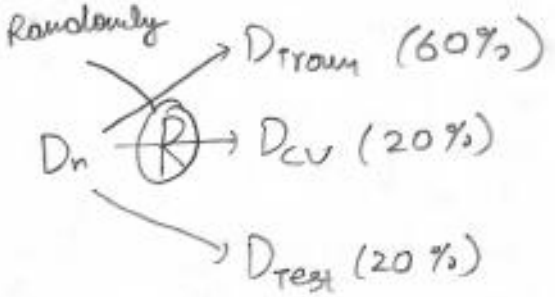
K' = 4; K=10 ; K'=100

↓

rule of thumb 10 fold CV.

Visualizing Train, CV & test datasets





x : +ve datapoint in D_{train}
o : -ve datapoints in D_{train}

⊗ → +ve pt from D_{cv}
⊙ → -ve pt from D_{cv}

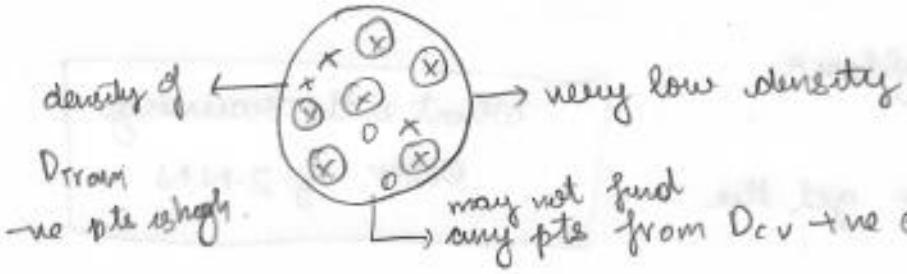
NOTE

① D_{train} & D_{cv} donot overlap perfectly when randomly sampled

② If there are many +ve pts from D_{train} in a region, then it is high likely to find many +ve/-ve pts from D_{cv} in that region.

③ If there are very few +ve/-ve pts in a region from D_{train}, then it is very unlikely to form +ve/-ve from D_{cv} in that region

density of +ve pts from D_{train} is high



How to determine overfitting vs underfitting:

{k fold CV or D_{cv} → best K → neither overfit nor underfitting

Q How can we be sure that we are not underfitting nor overfitting

$$\text{accuracy} = \frac{\#\text{ correctly class pt}}{\text{Total \# pts}} = 0.93 \quad \text{(Suppose)}$$

error = 1 - accuracy = 1 - 0.93 = 0.07 = 7%.

D_{CV} D_n → D_{train} → K-NN

→ D_{cv} — find right K → K-NN

→ D_{test}

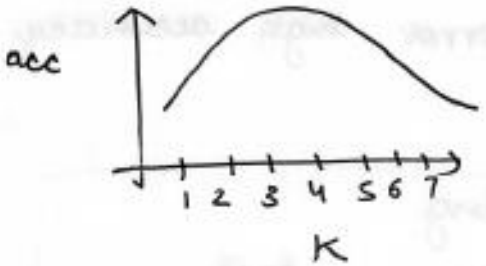
error on D_{cv} = 1 - accuracy

↑ acc. on D_{cv}

{ for each pt x_i in D_{cv}

x_q = pt

y_q — y_i



Training error:

{ D_{train} → NN

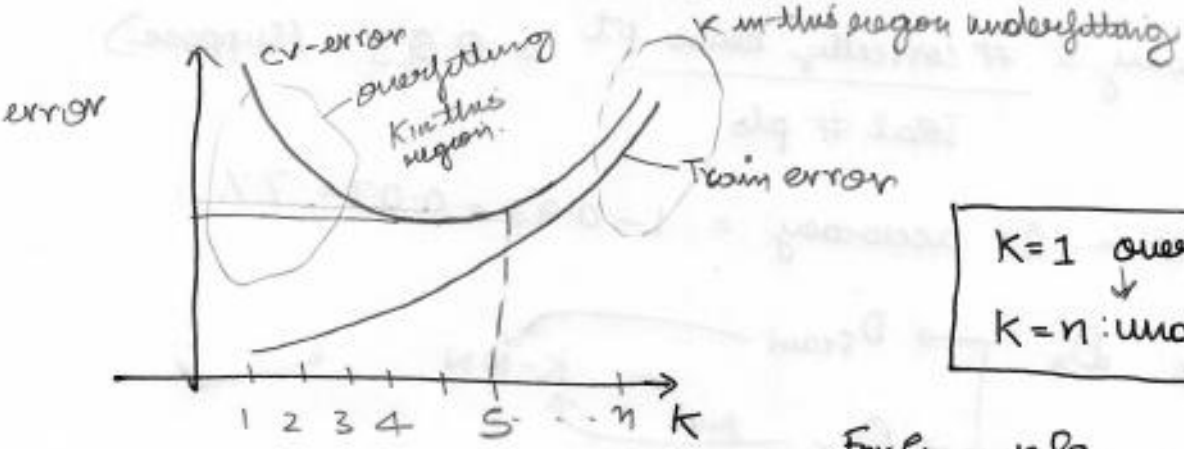
D_{cv} → K

f [K-NN]

for each x_i in D_{train}
- find 2 nearest neighbour to x_i from D_{train}
- majority vote to get the class label
- if y_1 == class label
  accurate
  - else
    error.

What is the training error of 2-NN

error; ~~train~~ error, Validation - error



low error high accuracy

K=1 overfit
↓
K=n : underfit

For Example

5-Nearest Neighbour

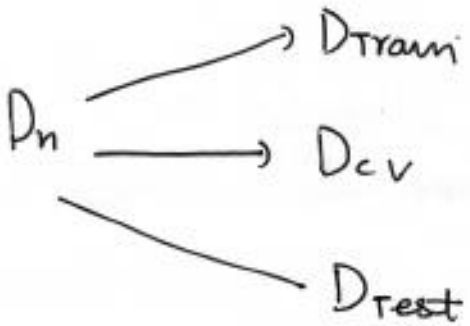
lowest error from Graph

K is selected using cross validation with highest accuracy.

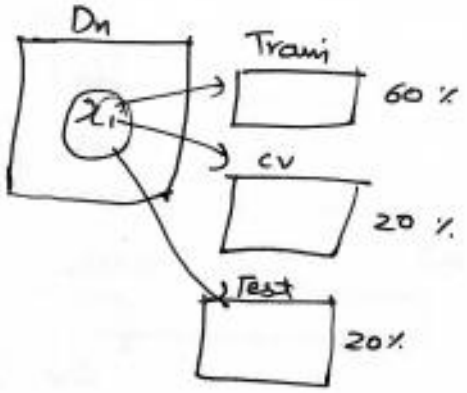
Underfitting
Train error → high
Cross validation error → high

Overfitting
Train error → low
Cross validation error → high

Time Based splitting

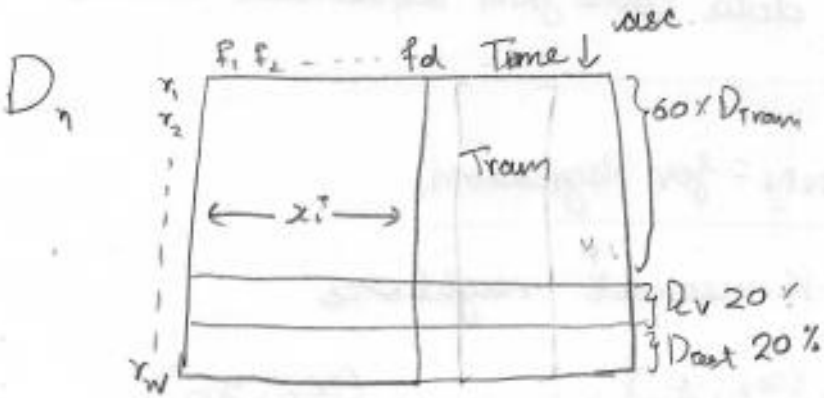


Randomly split



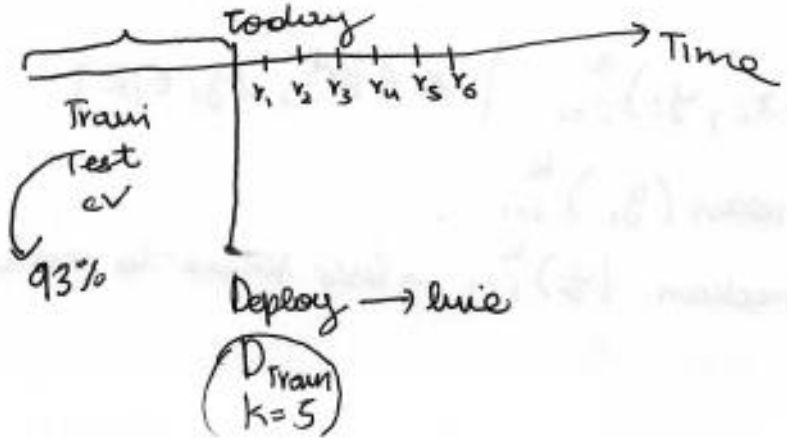
✧ Time based Splitting

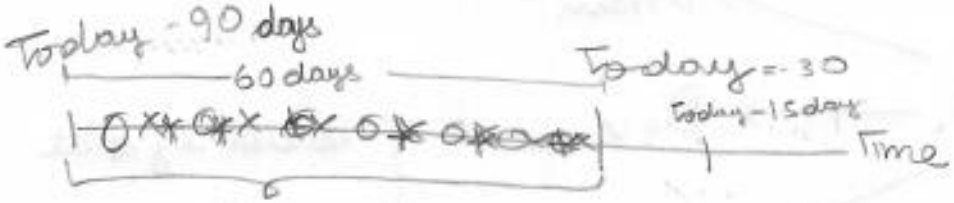
better than Random-splitting for some data sets

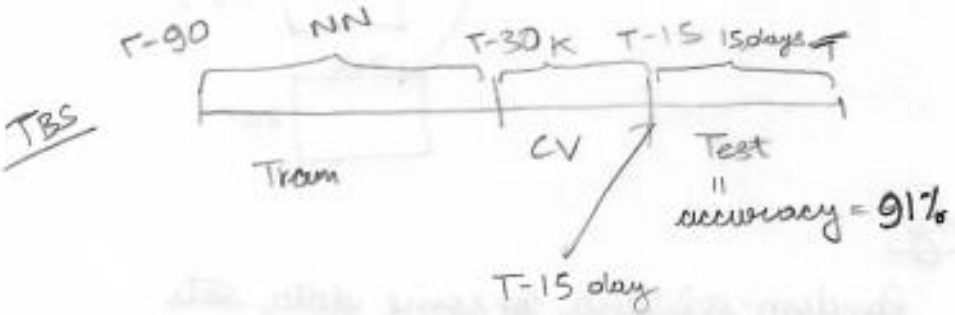


① Sort D_n in asc order of time

TBS → can only be done if Time stamp is given







TBS :- In this Test data come from sequential model

KNN - for Regression

given x_q, find K-nearest neighbors

① $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$

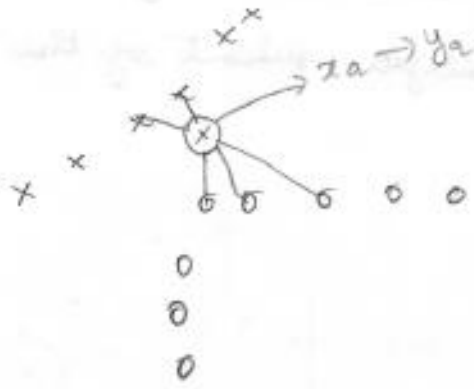
② $y_q \leftarrow y_1, y_2, y_3, \ldots, y_k$.

$$D = \{ (x_i, y_i)_{i=1}^{n} \mid x_i \in \mathbb{R}^d ; y_i \in \mathbb{R} \}$$

$\rightarrow y_q = \text{mean} (y_i)_{i=1}^{k}$

$\rightarrow y_q = \text{median} (y_i)_{i=1}^{k}$ $\rightarrow$ less prone to outliers

Weighted K-NN



$0 \rightarrow$ +ve
$x \rightarrow$ -ve

SNN K=5

Let distance

$x_q \rightarrow x_1, y_1, d_1, 0.1$ -ve
$x_2, y_2, d_2, 0.2$ -ve
$x_3, y_3, d_3, 1.0$ +ve
$x_4, y_4, d_4, 2.0$ +ve
$x_5, y_5, d_5, 1.0$ +ve

→ distance more near

One way

$w_i = \frac{1}{d_i}$ one way

$\begin{cases} d_i \uparrow, w_i \downarrow \\ d_i \downarrow, w_i \uparrow \end{cases}$

	y_i	distance(d_i)	w_i
$x_q \rightarrow x_1$	-ve	0.1	10
x_2	-ve	0.2	5
x_3	+ve	1.0	1
x_4	+ve	2.0	0.5
x_5	+ve	4.0	0.25

$w_i \rightarrow$

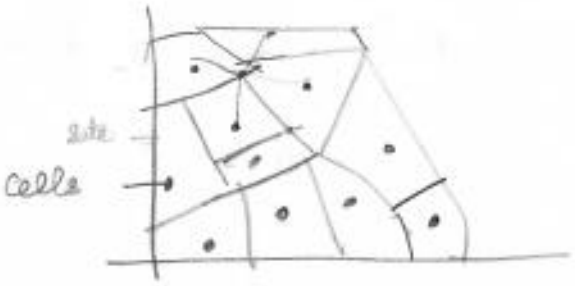
Majority rule

y_q = +ve which is wrong

but by using weighted K-NN we can predict right posi-tion

Voronoi diagram

Used partitioning of a plane into regions based on distance to points in a specific subset of the plane.



Voronoi cells

cell : region.

Binary Search Tree (BST)

K. NN : { O(n) if d is small → Time
O(n) K is small
↓
space

Time : $O(n) \rightarrow O(\lg n)$ → Kd-tree can be used to reduce it

$n = 1024 \rightarrow \log(n) = 10$

prob : given a sorted array, find a number is present in the array or not.

1	2	3	4	5	6	7	8	
1	2	6	8	12	15	17	18	Sorted

10 → median

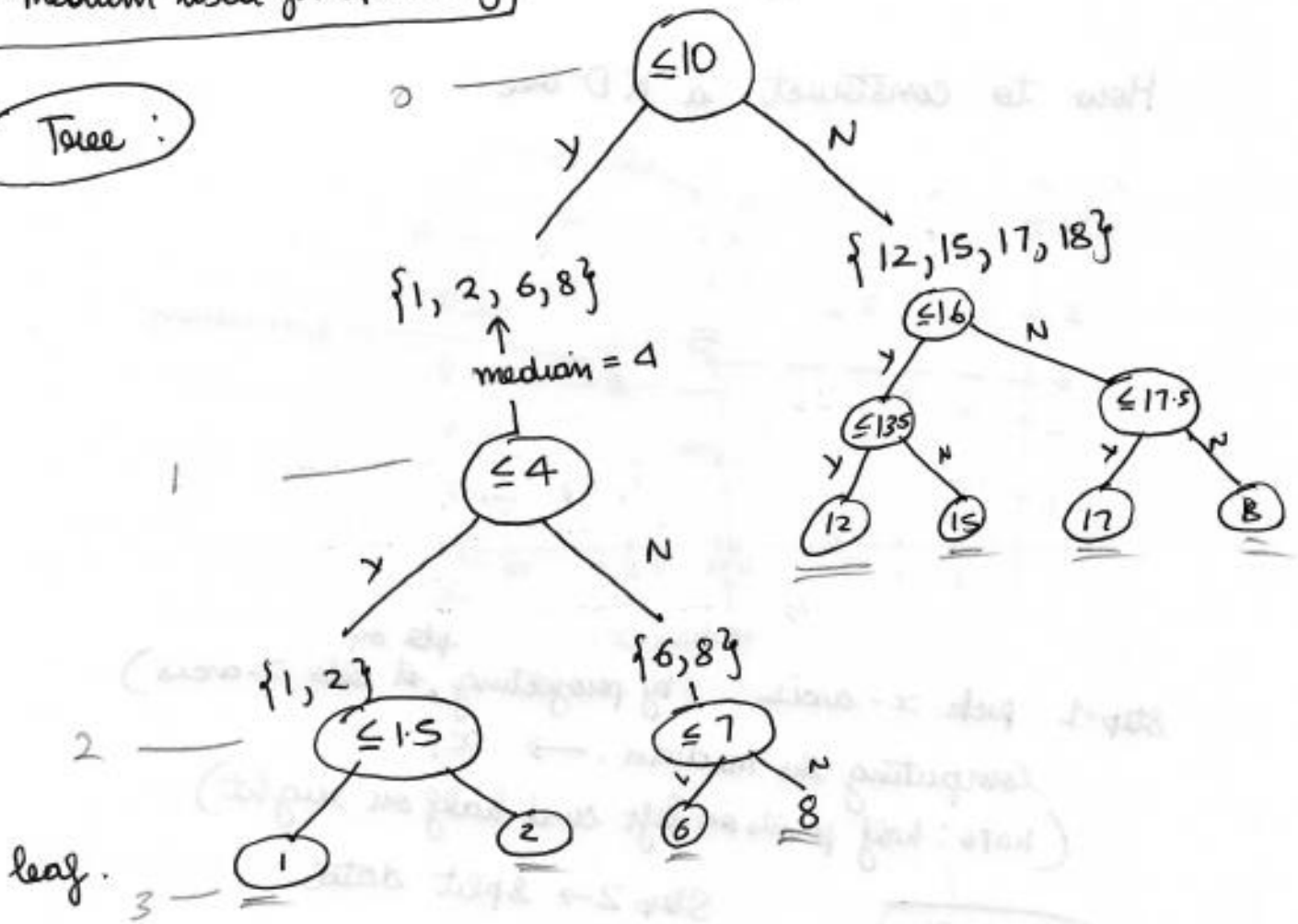
1.) Construct a BST

median = 10

NOTE median used for splitting

Flow chart

Tree :



Depth of tree = 3

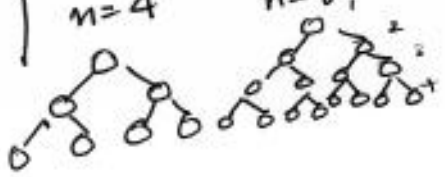
B.ST :- Time complexity to search is $O(\log n)$

↓

depth of binary search tree

{ n elements
depth BST = $O(\log n)$

2^2 n=4 2^3 n=8, 2^4 n=16 2^5 n=32



$\log_2 8 = 3$

$\log_2 32 = 5$

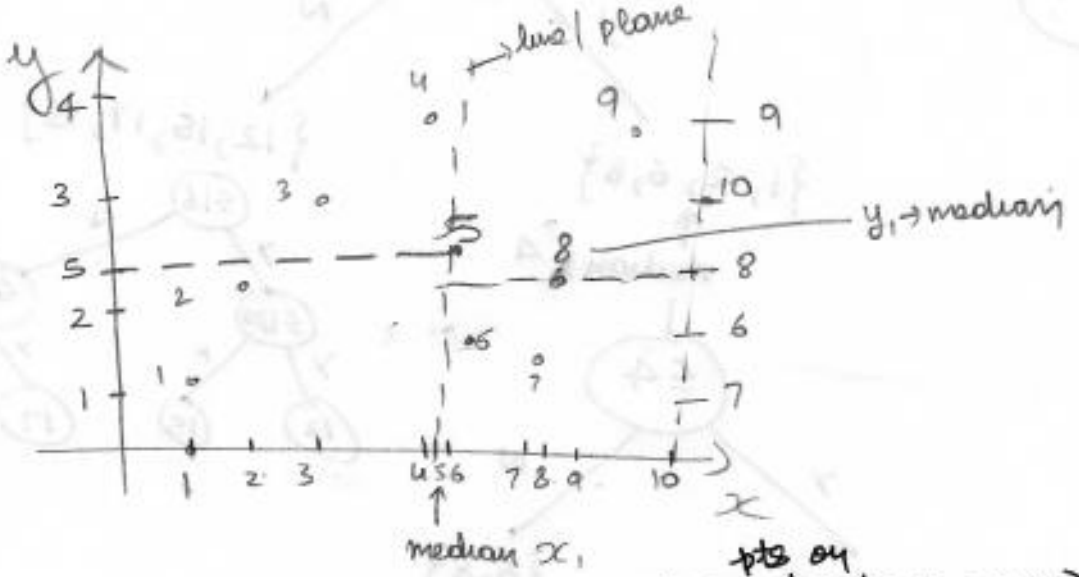
Kd-tree :- (2D, 3D, ... nD)

BST :- → sort numbers → Tree

scalars

'1D'

How to construct a KD tree

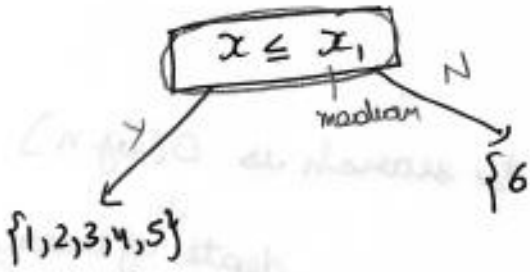


Step→1 pick x-axis (by projecting pts on x-axis)

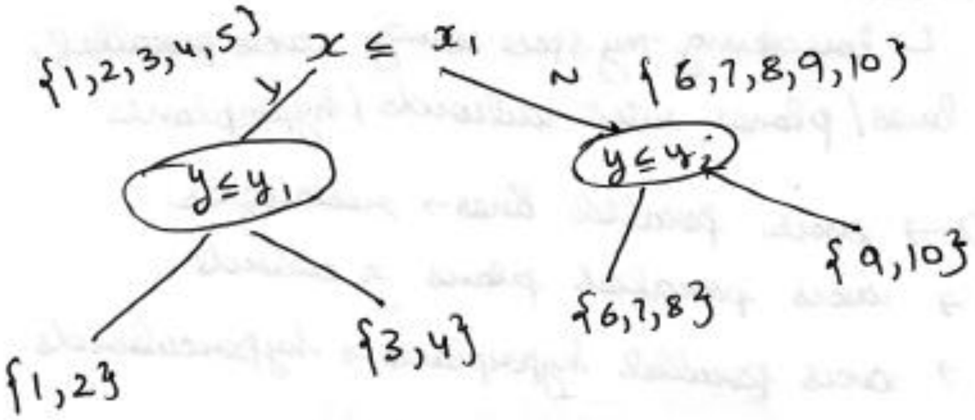
computing the median → x_1

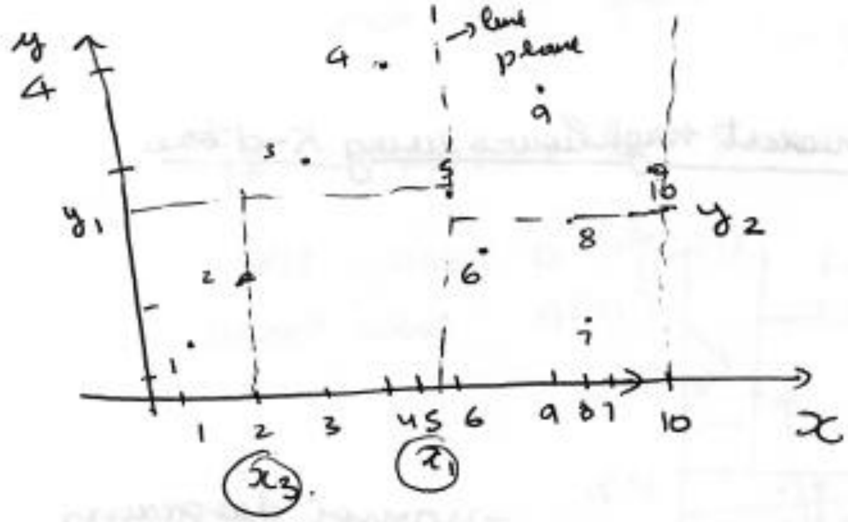
(NOTE: half points on left and half on right)

Step 2→ Split data

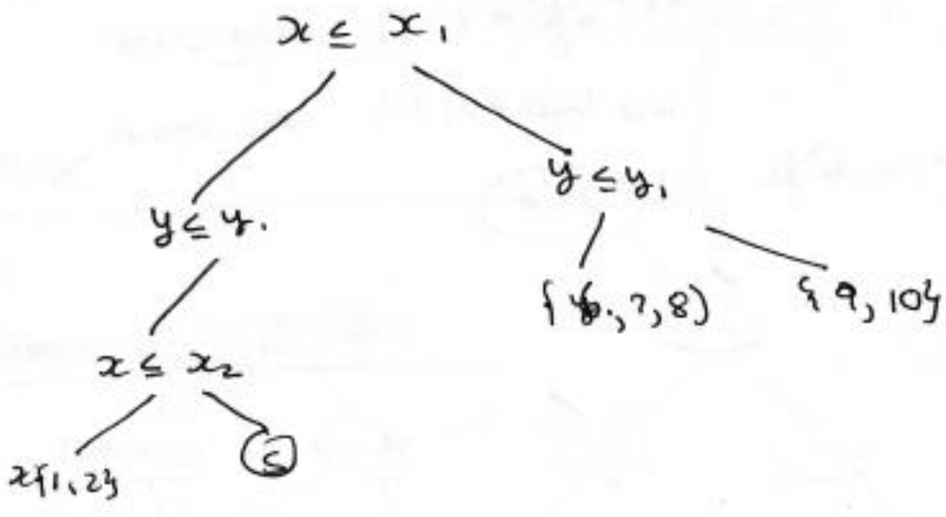


Step 3→ Use Y axis (project pt on Y axis). Find median of Y axis on left and right side





These line are axis parallel.



... not small Kd tree will not work well.

Kd tree

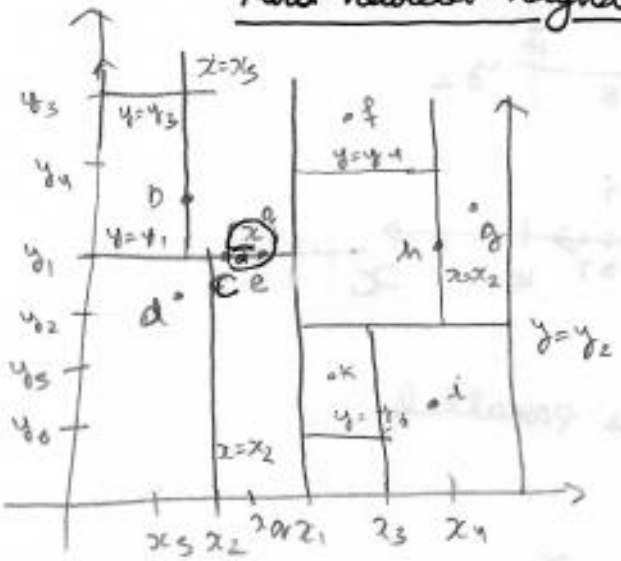
↳ breaking my space using axis parallel lines / planes into cuboids / hyperplanes

2D → axis parallel lines → rectangles

3D → axis parallel planes → cuboids

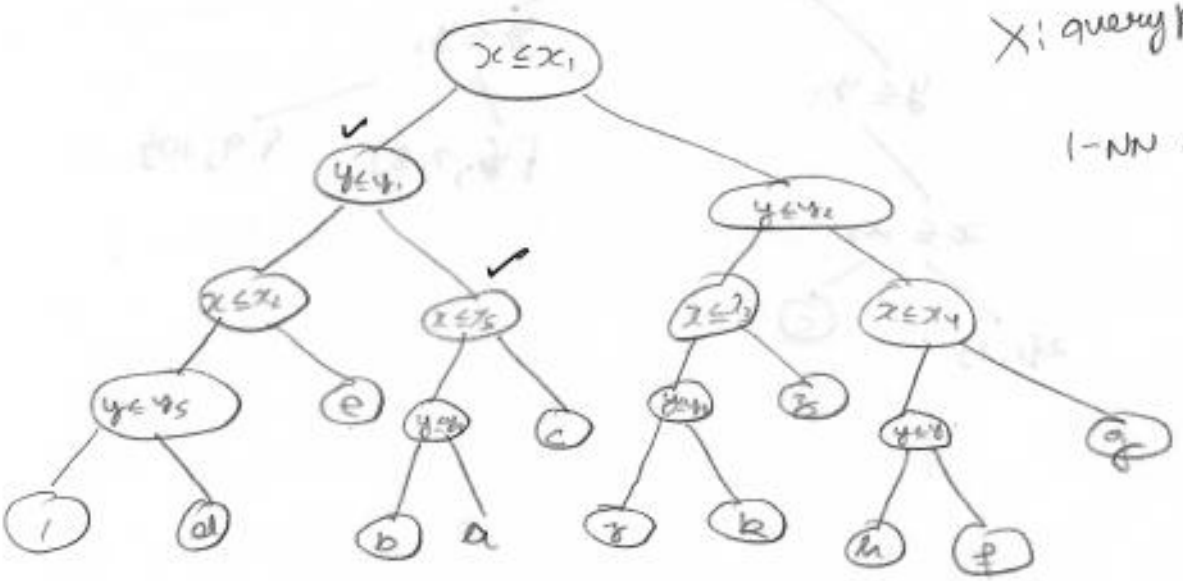
nd → axis parallel hyperplanes → hypercuboids.

Find nearest neighbours using K-d tree



- voronoi diagram

Kd tree



X: query point (a)
(x_a, y_a)

1-NN of q

Circle / hyper-sphere : rad-d
C-Q

comparisons to find 1-NN

best case : $O(\lg(n))$
worst case : $O(n)$

check in every box.

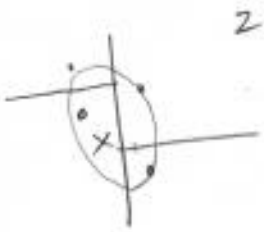
K-NN	
best case	$O(K * \lg n)$
worst case	$O(K * n)$

if d is small

☆ Limitations of Kd tree

What happens if d is not small.

$d=10$; $2^d=1024$
$d=20$ $2^d \approx$ million



d↑ worst case # adj cells ↑

$O(\log n)$ (n=1 million) $d=20$
$2^d \approx n$.

If d is not small Kd tree will not work well.