1. NLP AND KNN

Tfidf :-

$$TF(w_i, r_j) = \frac{\#\text{ of times } w_i \text{ occurs in } r_j}{\text{Total } \#\text{ of words in } r_j}$$

$$0 \le TF(w_i, r_j) \le 1 \leftarrow \text{Probability}$$

$$IDF(w_i, D_i) = \log(N/n_i)$$

$N \rightarrow$ # of documents

$n_i \rightarrow$ # doc containing w_i

$$n_i \le N \Rightarrow N/n_i \ge 1$$

$$\log\left(\frac{N}{n_i}\right) \ge 0 \qquad \log(1) = 0$$

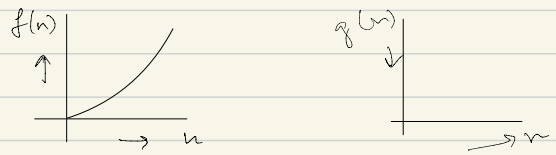
if $n_i \uparrow$; $\frac{N}{n_i} \downarrow$; $\log\left(\frac{N}{n_i}\right) \downarrow$

① $IDF \ge 0$

② If w_i is more freq in my corpus than IDF decrease

$$\frac{1000}{10} > \frac{1000}{20}$$

Note



monotonic increasing function

monotonic decreasing function

$\uparrow TF(w_j, d_i) * idf(w_j; D_c) \uparrow$

w_j is frequent in d_i ↳ w_j is rare in D_c

↙

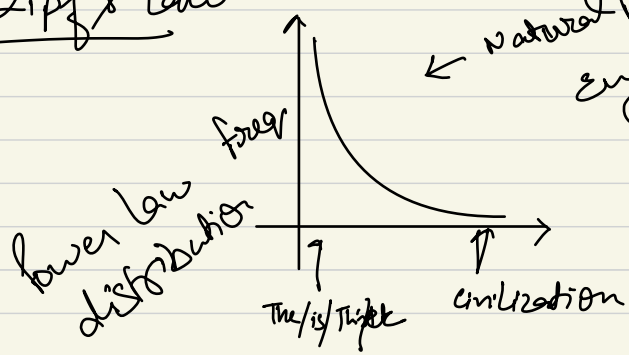
More weight to rare words

⇒ Limitation

↳ Semantic meaning is not considered

Why we use log for idf

Zipf's Law



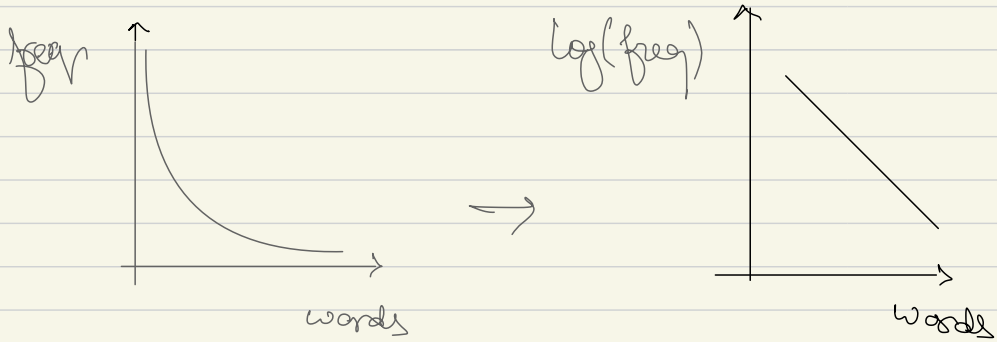
← natural property of any language

→ Decreasing order of frequency

$X \sim$ power law

↳ log normal distribution

↳ Gaussian (box - Cox transformation) → $\log(x)$



practical reason

	$\frac{N}{n_i}$		$\log_e\left(\frac{N}{n_i}\right)$
The →	1	$(n_i = N)$	0
its →	1	difference is 1000	0
⋮			
Civilization →	1000	1 in 1000	~ 6.9

diff is just 7

Here idf dominates low frequency words

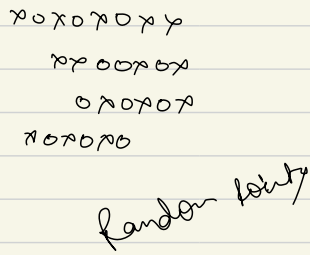
We have huge difference that's why we use log.

Classification and Regression

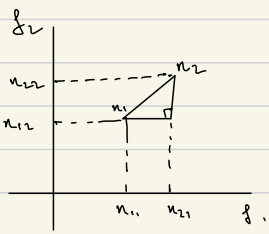
Finding $f(x)$ based on "x" and getting "y" i.e. label class

$$y = f(x)$$

KNN fails



Distance Measure



d = shortest len for x_1 & x_2

Euclidian distance $= \sqrt{(x_{21} - x_{11})^2 + (x_{22} - x_{12})^2}$

$= \| x_1 - x_2 \|$

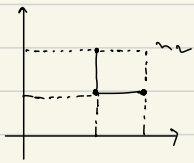
$$\|x_1 - x_2\|_2 \quad \left(\sum_{i=1}^{d} (x_{1i} - x_{2i})^2 \right)^{1/2}$$

$\|x_1 - x_2\|_2 =$ L_2 norm

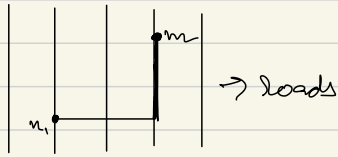
$\|x_1\|_2 =$ euc distance from origin $= \left(\sum_{i=1}^{d} x_i^2 \right)^{1/2}$

Manhatan distance

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



It is like block (RCF colony)



$\|x_1 - x_2\| =$ L_1 norm.

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

Lp - norms → Mankorski distance

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

$p \neq 0$

$p > 0$

$p = 2$ → Minkowski distance = Euc distance

$p = 1$ → " " = manhaten dist

Hamming distance (boolen vect)

↳ Binary BOW

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

Hamming distance (x_1, x_2) = # location of dim where no. of distance diff

$x_1 = abcadefghik$
$x_2 = acbadegfhik$ U/C in genecode

hamming distance $(x_1, x_2) = 4$

Cosine Similarity and Cosine distance

If Similarity ↑ ↓ Distance ↓ ↑

Distance increase the similarity b/w points decrease and vice versa

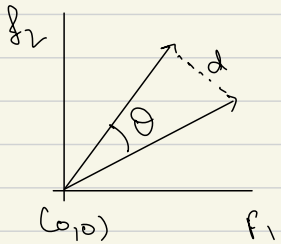
$$\boxed{1 - \cos_{sim}(x_1, x_2) = \cos_{dis}(x_1, x_2)}$$

Very similar 2 points

Cosine sim = +ve

Very dis similar 2 points

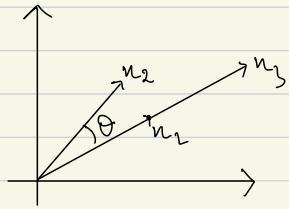
Cose sime = -ve



d = euc dist

Cos sim = cos θ (x_1, x_2)

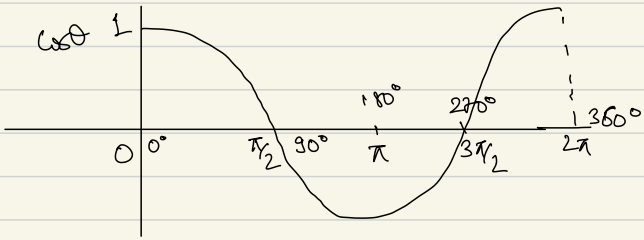
θ = angle b/w x_1, x_2



Cos sim(x_1, x_2) = cos θ

Cos sim $(x_1, x_3) = 1$

θ $x_1, x_3 = 0°$



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

↳ L_2 norm

① If x_1 & x_2 are unit vector

$$\|x_1\|_2 = \|x_2\|_2 = 1$$

$$\cos\theta = x_1 \cdot x_2$$

relationship b/w euc-dist & cos sim

Euc dist

$x_1 = A$

$x_2 = B$

$$\|A-B\|^2 = (A-B)^T(A-B)$$
$$= \|A\|^2 + \|B\|^2 - 2A^TB$$

If $\|A\|^2 = \|B\|^2 = 1$

So $2(1-\cos(A,B))$

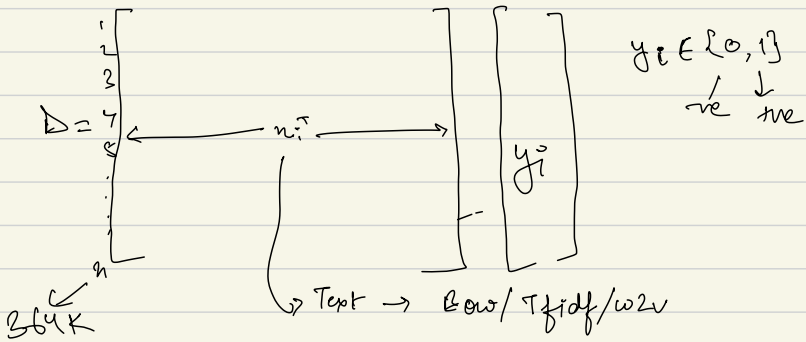
If x_1 & x_2 are unit vector

$$[\text{euc dist}(x_1, x_2)]^2 = 2(\underbrace{1-\cos\theta}_{\text{cosine dist}})$$

↳ 2cos-dist (x_1, x_2)

Measure how good KNN is

Amazone food Review



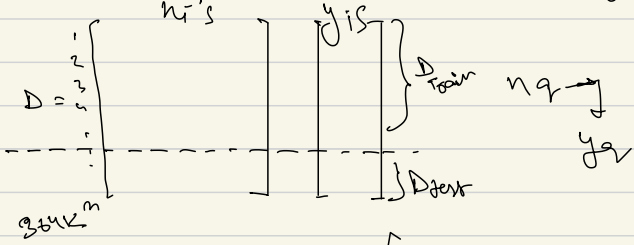
$y_i \in \{0, 1\}$
-ve → +ve

Problem: Given a food review (x_q), what is its polarity (+ve/-ve)

$x_q \rightarrow \text{Text}_q \xrightarrow{\text{Conv.}} x_q$

$x_q \rightarrow y_q$

Measure K-NN → (K-NN + Majority)



$D_n \rightarrow D_{train}, D_{Test}$

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

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

$D_n \rightarrow D_{Train} \rightarrow n_1$

$D_n \rightarrow D_{test} \rightarrow n_2$

$n_1 + n_2 = n$

Split D_n $\xrightarrow{R}$ D_{Train} 70% } randomly
$\xrightarrow{R}$ D_{test} 30%

n_1 D_{Train} $(x_i, y_i)_{i=1}^{n}$ $\longrightarrow$ K-NN

D_n

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

for each point in D_{test}:

$x_q = pt$

use D_{Train} & KNN to determine y_q

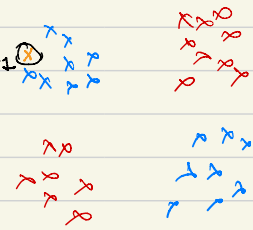
if $y_q == y_{pt}$ (Actual label)

count += 1

end

Count = # pts for which D_{Train} + KNN gave a correct class label

D_{Train} n_1 x_1



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

$x_q =$ x_1 for each pt in D_{Test}

$y_q =$ Blue = +ve $\rightarrow x_q = pt$

$\rightarrow$ use D_{Test} & KNN to predict y_q

Accuracy $= \frac{\text{count}}{n_2}$ $\rightarrow$ # pts for which D_{Train} + KNN gave a correct class label

$0 \leq Acc \leq 1$

$Acc = 0.91 \Rightarrow 91\%$ of times

Conclude: KNN on data using D_{Train} gives us 91% of accuracy

Test Evaluation time / Space Complexity

$x_q \rightarrow y_q$

K is small
↳ 5 or 10

d → large

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

KNN pts = [ ]

→ n pts ; d-dim

for each x_i in D_{Test} — n

- $O(d)$ – compute $d(x_i, x_q) \rightarrow d_i$
- $O(1)$ – Keep the smallest K distance (x_i, y_i, d_i) → KNN pts [ ]

Complexity → $O(nd)$

$Count_{pos} = 0$; $Count_{neg} = 0$

for each x_i in KNN pts — Small q — $O(K)$ → $O(1)$

- if y_i is +ve
  - $count_pos += 1$
- else
  - $count_neg += 1$

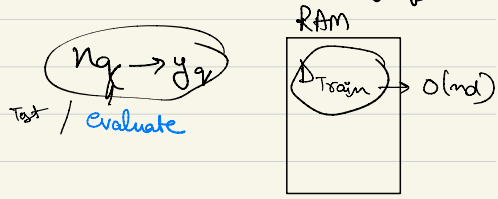
$O(1)$

- if count pos > count neg
  - return $y_q = 1 \rightarrow$ +ve
- else
  - return $y_q = 0 \rightarrow$ −ve

Time Complexity $= O(nd) + O(1) + O(1)$
$\approx O(nd)$

If dis small
if $d \ll n$
$\boxed{O(n)}$ → time

Amazone Food Review dataset
$n \approx 34K$
$d \approx 100K$ (BoW)
300 (+ tfidf)



Note: Never count input data size
But while deploying we do conside every data input also

Time complexity :- $O(nd)$

Space-compl :- Space we need to evaluate

KNN limitation

when I have to productionize whole thing I need to store data in memory

$n \approx 364K$ → Dtrain
$d \approx 100K$

$364100M = 36GB$

② Time Complex : 36 Billions Computation

Input Review $\xrightarrow{1ms}$ +ve/-ve

Low Latency system: $n_q \xrightarrow{fast} y_q$

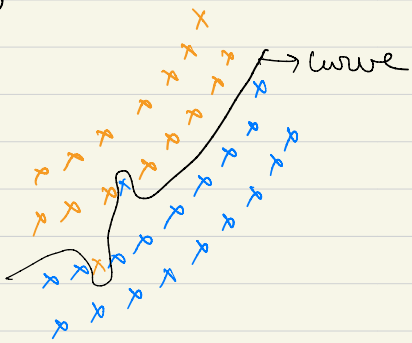
Medicine $n_q \xrightarrow{1min}$ → OK y_q

KNN → Simple implementation → $O(nd)$ $O(nd)$
↳ Kd-tree
↳ LSH

Decision surface for KNN as K changes.

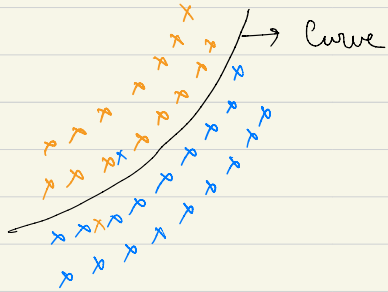
K in KNN is hyperparameter

If K = 1



My curve will look like this

K = 5



My curve will be like this

Note: As K increases Smoothness of curve increases

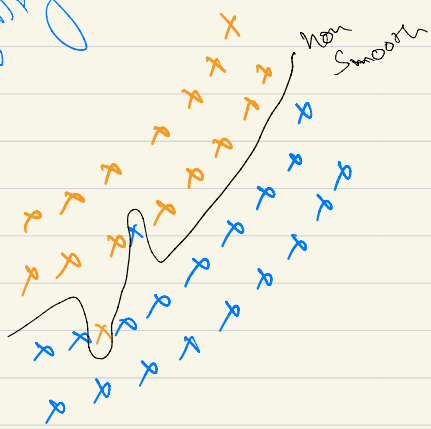
If K = n

$n_1 \Rightarrow$ blue points
$n_2 \Rightarrow$ Red points

$n_1 > n_2$
So every decision will be n_1

Overfitting & Underfitting

Overfitting



- over work
- no mistake
- make nearly perfect boundary

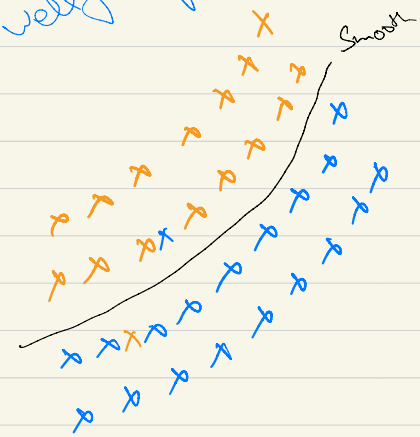
It can be at $K = 1$ in KNN Case

Underfitting

imperfect

every query points belong to the majority of cars.

Well fitting



→ well fit

→ less prone to noise

→ 80% cut

→ $K = 5$

Need of Cross validation

How to determine K

D_n — D_{Train} (70%)
D_n — D_{Test} (30%)

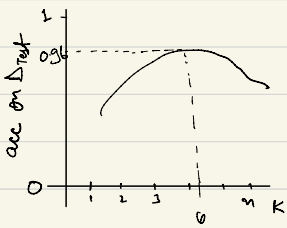
One-Idea

	Train	Accuracy on D_{Test}
K=1	D_{Train}	0.78
K=2	"	0.82
K=3	"	0.85

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

x_q y_q

$x_q = x_1$, y_q
$x_q = x_2$
$x_q = x_3$
⋮
$x_q = x_{n_{Test}}$



(Typically)

→ Not follow this every time

K=6 gives me the best acc on D_{Test} when using D_{Train} as training data

K=6
Accuracy = 0.96 = 96%

Using D_{Train} & 6-NN on data, I get an accuracy of 96%

Small problem

D_n → f (6-NN) → acc (96%)
(D_{train}, D_{test})

and we broke up D_n in D_{test} & D_{train}

future, unseen point (need to work well) x_q → y_q (acc)

- If it not work good on future point than it is useless
- If it work good on future unseen point it said to $\boxed{\text{generalize well}}$

Obj: perform well on "unseen" pts
↓
Not in Train & Test

K ← Test-set

f

K-NN
D_{test} D_{train}

acc on D_{Test} = 96%

Can I say that my acc on future data is 96%? ✗ NO

Cross validation

D_n → D_{Test} → K
D_n → D_{Train} → NN

D_n → D_{Train} → NN → f: K-NN
D_n → D_{CV} → K → f: K-NN
D_n → D_{Test} $\xrightarrow{f}$ acc

didn't use D_{Test}

$D_n \rightarrow D_{Train}$, $D_n \rightarrow D_{CV}$ $\rightarrow f(K\text{-}NN)$

$D_n \rightarrow D_{Test} \rightarrow$ Unseen data

each point n_i from D_{test} as n_q $\xrightarrow{\text{meaning}}$ n_i in D_{test} as n_q

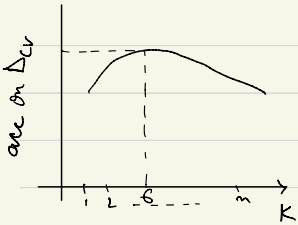
$\hookrightarrow$ acc: 93%

Can I now say that my 6-NN has an accuracy of 93% on unseen data ✓ yes

Cross validation is just about making some data seen some data unseen

			also
$D_n \xrightarrow{R}$	D_{Train}	(60%)	70
	D_{CV}	(20%)	10
	D_{Test}	(20%)	20

Randomly



f: 6-NN $\rightarrow$ CV, $\uparrow$ Train

acc on Test data

$\hookrightarrow$ 93%

Using D_{Train} as train data, I found 6-NN to have acc of 93% on future/unseen data

generalization accuracy

K fold cross validation

$D_n \xrightarrow{R}$ D_{Train} (60), D_{CV} (20), D_{Test} (20)

D_{Train}, D_{CV} → K-NN (P)

K	Train Data	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
K=2	D_1, D_2, D_3	D_4	a_4

4 times

average that make acc for K=1

4 times

K' = 4 fold CV

Problem. only 60% of data NN

20% CV → K

20% Test → unseen

only using 80% of data to compute NN

f : (3-NN) ← D_{Train}

avg-acc on 4 K-fold

n_q $\xrightarrow{NN}$ D_{Train} for NN → 3 NN

> More the Training data, the better is your model

acc, D_{Test} → acc of 3-NN on D_{test} → 93% on unseen Data

K fold CV

1. D_n → D_{Train} (80%) → f: K-NN

   D_n → D_{Test} (20%) → unseen → acc

2. ← D_{Train} →

D_1	D_2	D_3	D_4
1	2	3	4

   ← randomly

   4 equal size part

K' fold CV → D_{train} → NN → K in K-NN

> What is the right K'

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

rule of thumb: 10-fold CV

No Scientific reason, we just select 10

Time it take to compute the optimal or best K in K-NN increase by K' time if we use K'-fold CV

10

Visualizing Train, Cv & test dataset

$D_n \xrightarrow{R} D_{Train}$ (60%), D_{CV} (20%), D_{Test} (20%)

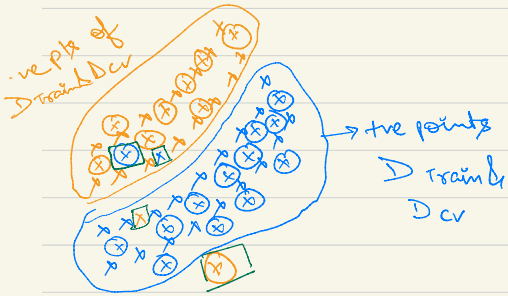
Randomly

x : -ve datapoin in Dtrain

x : +ve datapoint in Dtrain

Train, cv, test : (x_i, y_i)

↑ class labels



(x) : +ve points from D_{cv}

(x) : -ve points from D_{cv}

[pt] → outliers/Noise/error

How to Determine overfit & underfit

D_n — D_{Train}, D_{CV}, D_{Test}

D_{Train} → K-NN

D_{CV} → K-NN

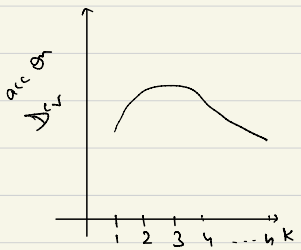
error on D_{CV} = 1 − acc on D_{CV}

acc on D_{CV}

for each x_i pt in D_{CV}

x_q = pt

$y_q \rightarrow y_i$



Training Error

for each x_i in D_{Train}

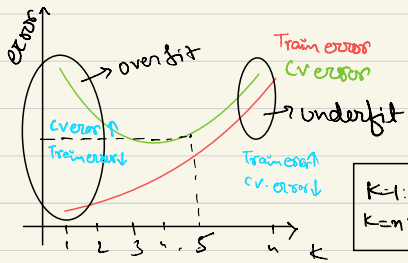
- find 2 nearest valour to x_i from D_{train}
- Majority rule to get the class labels
- If y_i == class labels accurate

else

error

$D_{Train} \rightarrow$ Training

$D_{Train} \rightarrow$ acc/error



K=1: overfit
K=n: underfit

error = 1 − accuracy

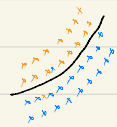
Train error is high, CV error is high	underfit
Train error is low, CV-error is high	Overfit

OR

Train acc is low, CV acc is low	underfit
Train acc is high, CV acc is low	overfit

K=1

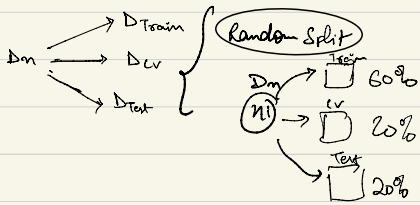
Train error is low
CV error is high



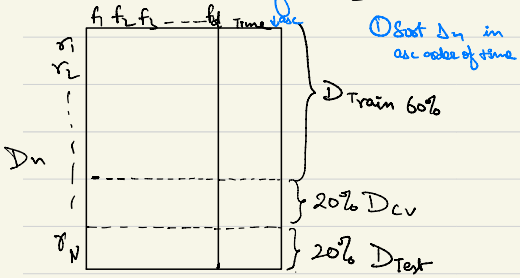
K = 5

Train error is no more zero
CV error is low

Time base Splitting



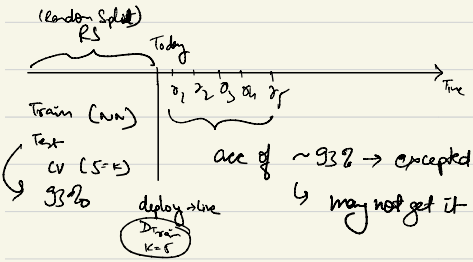
Time base Splitting (TBS)



① Sort Dn in asc order of time

TBS → (Time) is necessary field

RS



acc of ~93% → expected

↳ may not get it

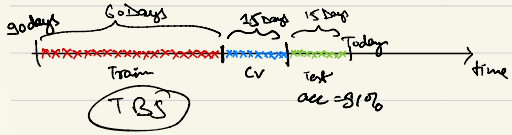
When we use random sampling on live data then based on Test data acc that ~~is~~ will be expected in future or upcoming day, may not get good results from model

With time, my products and their review changes

Train (Red) ×

Test (Green) ×

CV (Blue) ×



In this I will be living in past but my future results will be or more likely to be accepted accuracy

If I trained my model 15 days back & tested it on last 15 days data, my acc is 91%

Whenever time is available & if thing / behave or data change over time

$\boxed{\text{Time based splitting is preferable}}$

KNN for Regression

classification
2 classes

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

find KNN majority class

$f(x_q) \rightarrow y_q \rightarrow$ class labels

Regression

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

$x_q \rightarrow y_q \rightarrow$ number

extend

classification $\rightarrow$ Regression

① given (x_q), find K-nearest-neighbor

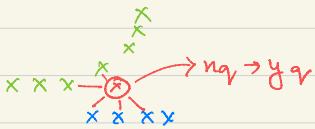
$(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)$

Regression

$$y_q \leftarrow \overbrace{y_1, y_2, y_3, \ldots y_k}^{\text{Not 0 or 1}} \; \mathbb{R}$$

$$\boxed{\begin{aligned} y_q &= \text{mean}(y_i)_{i=1}^{K} \\ y_q &= \text{median}(y_i)_{i=1}^{K} \rightarrow \text{less prone to outliers} \end{aligned}}$$

Weighted KNN



5-NN

		Distance	
$x_q \rightarrow$	x_1, y_1, d_1	0.1	-ve
	x_2, y_2, d_2	0.2	
	x_3, y_3, d_3	1.0	+ve class
	x_4, y_4, d_4	2.0	
	x_5, y_5, d_5	4.0	

	y_i	distance d_i	weight w_i	
$x_q \rightarrow$ x_1	-ve	0.1	10	
x_2	-ve	0.2	5	$\rightarrow 15$
x_3	+ve	1.0	1	
x_4	+ve	2.0	0.5	$\rightarrow 1.75$
x_5	+ve	4.0	0.25	

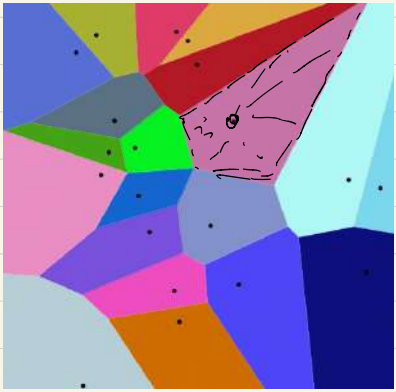
$$w_i = \frac{1}{d_i}$$

one way, $d_i \uparrow ; w_i \downarrow$

Simplest $d_i \downarrow , w_i \uparrow$

So y_q = -ve

Voronoi Diagram



In mathematics, a **Voronoi diagram** is a partition of a plane into regions close to each of a given set of objects. In the simplest case, these objects are just finitely many points in the plane (called seeds, sites, or generators). For each seed there is a corresponding region consisting of all points of the plane closer to that seed than to any other. These regions are called Voronoi cells. The Voronoi diagram of a set of points is dual to its Delaunay triangulation.

It is same as RNN, $\underline{\text{st}}$ used in many fields like virtual art, science & tech.

Binary Search tree

A:

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

← Sorted

Ques:- find an element in an array

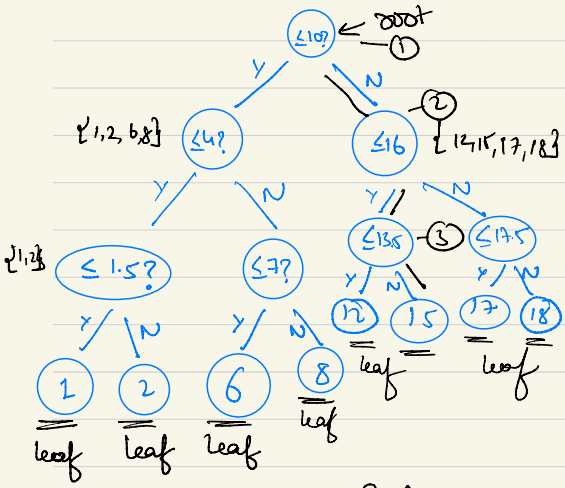
$q = 15$

① Construct BST

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

10

find median



8 elements

3 comparisons

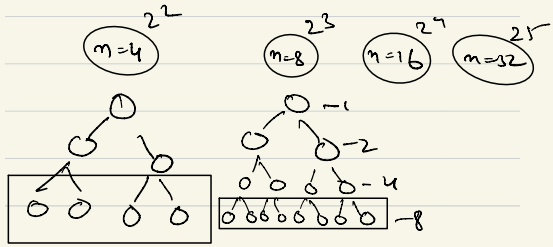
Time complexity to search is $O(\log n)$

⇓

depth of tree

{ n elements

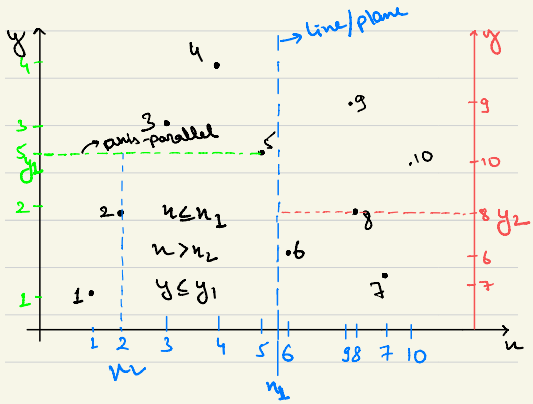
depth BST = $O(\log n)$



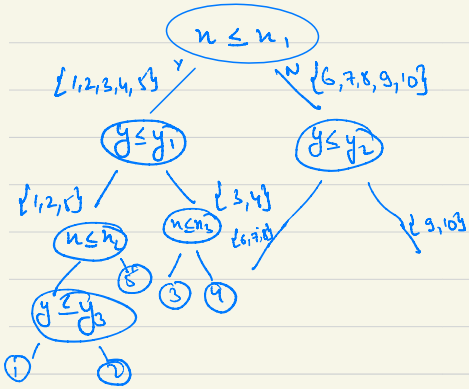
$\log_2 8 = 3$

$\log_2 32 = 5$

Kd tree



① pick x axis project point onto x axis, compute median



Kd tree

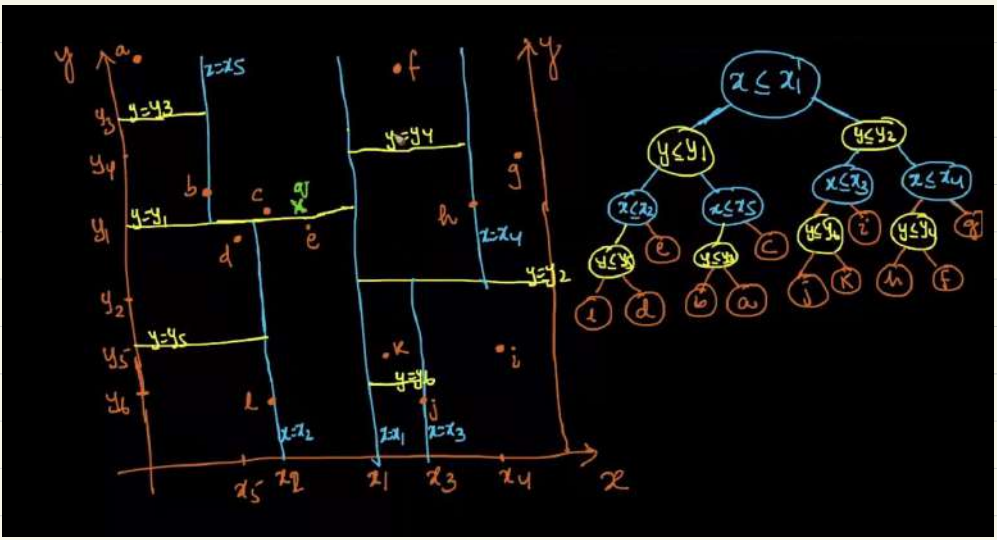
↳ breaking space using axis parallel line/plane into rect/cuboid/hyperbolic

2D → axis parallel line → rectangle

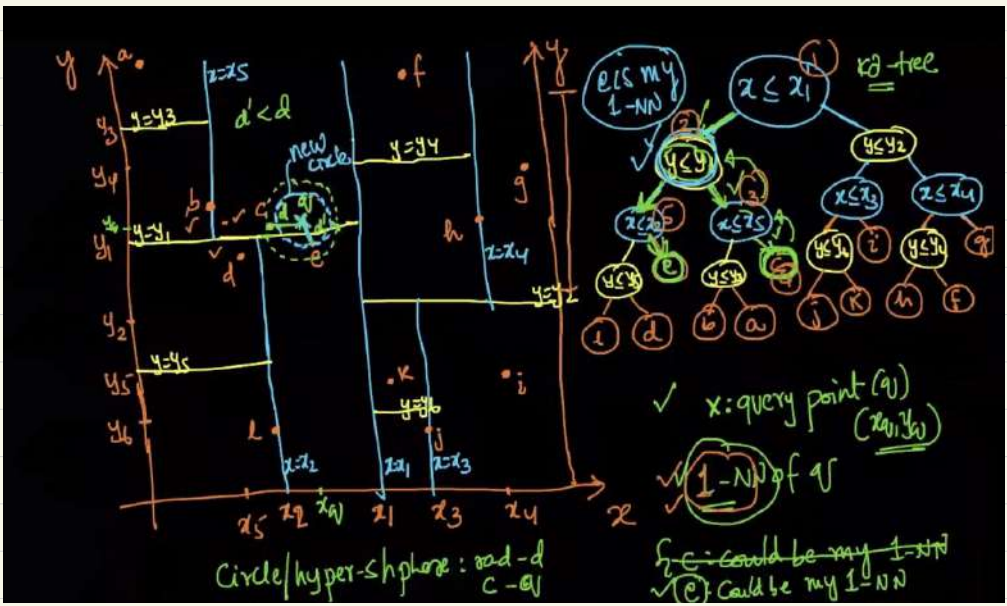
3D → " " plane → cuboids

nD → hyperplane → hyperCuboids

Find Nearest Neighbours using KD



Kd tree



Time Complexity

Comparison to find 1-NN

- best case: $O(\log(n))$
- worst case: $O(n)$

perfect analysis :- very complex

We make cirular where radis is distance b/w query point and some region point if something comes in between that circle than we revert back and goest to other region

Limitation of Kd tree

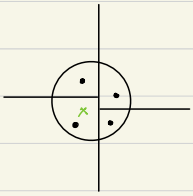
1. When d is not small {2,3,4,5}

$d=10$; $2^d = 1024$

$d=20$; $2^d \approx 1$ Million

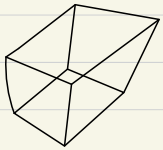
$d\uparrow$ worst case # adj. cells $2^{d\uparrow}$

$n =$ 1 Million if $d=20$



2D

$2 \to 4$ (2^2)



3D

$3 \to 8$ (2^3)

When d is not small {2,3,4,5}

Time Complexity: $O(\log n) \leftarrow$ 1NN

$O(2^d \log n) \leftarrow$ d-dim

which is worst than brut force

When d is 10, 11, 12 ---

Time Complexity increases drastically

② $O(\log n)$ when d is small

↳ when data is uniformly distributed

Unif : $O(\log n)$

Not Unif : $O(n)$ simple implementation

Kd tree :- Computer Graphics

↳ 2D, 3D

ML: 10D, 20D, 1000D

xx

Hashing Vs LSH

LSH: Locality Sensitive Hashing

Hashing

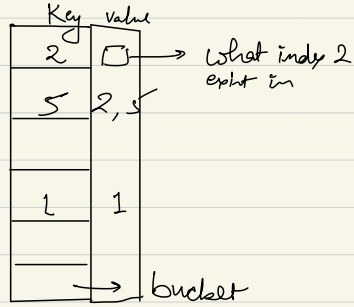
a: | 2 | 1 | 5 | 6 | 7 | 5 | 8 |

Unordered

hashtable (dict)

hash fn

h(n)

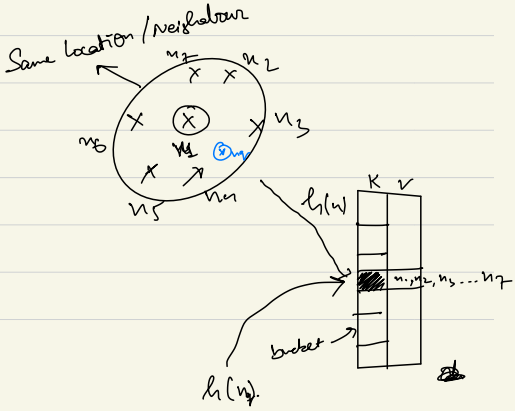


what index 2 exist in

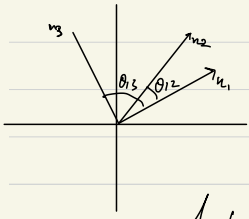
give value from array to hash fn at it will tell where or in which bucket this value goes

to check if 5 exist in array we can linear search it with complexity of O(n), where as in hash fn its complexity is 1

LSH



LSH for Cosine Similarity

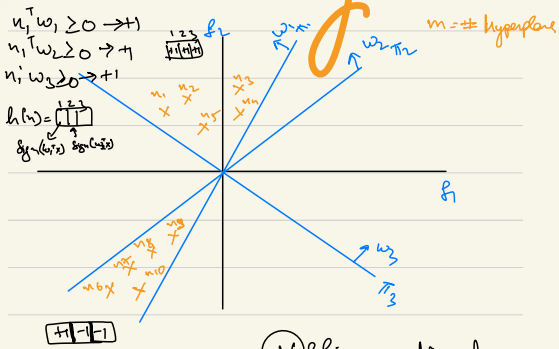


$\theta_{12} \sim 0 \Rightarrow n_1, n_2$ are similar / close

(small)

$h(x_1) = h(n_2) = \text{Key}$

K	V
$h(n_1)$	n_1, n_2



$m = \#$ hyperplane

$\forall$ slice $\rightarrow$ random value

↳ frequency

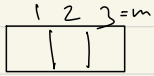
LSH : randomized algo

↳ It will not always gives you the correct

↳ But it is very high chance I give you

hash table

$n \rightarrow h(n)$



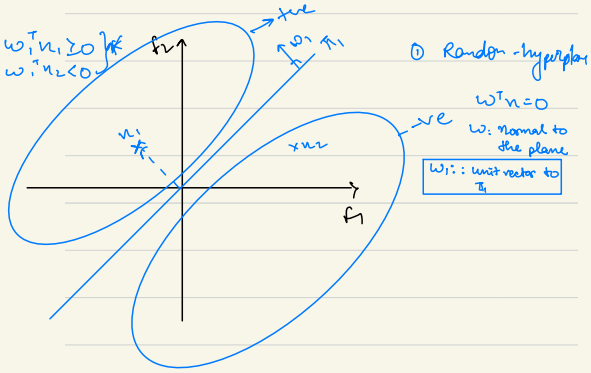
K	V
$h(n_1)$	$[n_1, n_2, n_3 \ldots n_5]$

$h(n_1)$ ←

← Time : $\sim O(m d n)$

Space $O(nd)$

hash table/Dict

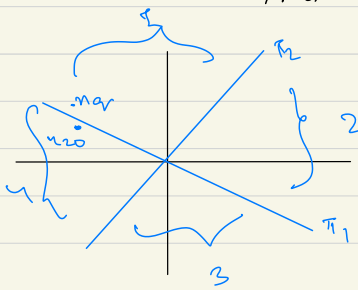


① Random-hyperplane

$w^T n = 0$

w: normal to the plane

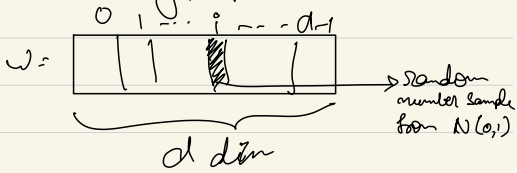
w_i: : unit vector to π_i



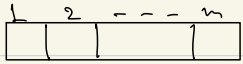
n_{20} is very close to n_1

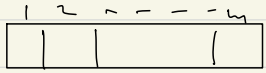
(*) could miss a NN in cos-sim as you measure

Generate Random hyperplane

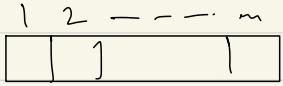


to resolve this issue we builds "L" hash table with m hyperplanes

① $h_1(x) =$ 

② $h_2(x) =$ 

⋮

$h_L(x) =$ 

$x_q \to h_1(x_q) \to$ key $\to$ hashtable $\to \{x_1, x_2, x_3, x_4\}$

$\hookrightarrow h_2(x_q) \to$ key $\to$ hashtable $\to \{x_1, x_2, x_3, x_5\}$

$\hookrightarrow$

$\hookrightarrow$

⋮

$\hookrightarrow h_L(x_q) \to$ _____ $\{\ \}$

Now we can look over x_q in set union of all the data points

$m = O(\lg n)$

L is usually small

Time complex to query give L hashtable

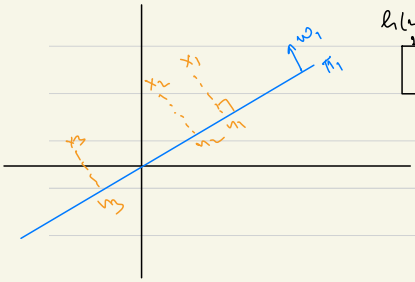
$O(md\,L)$ L is small

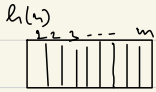
as number of hyperplane ↑ ↓

slice ↑ ↓

point per slide ↓ ↑

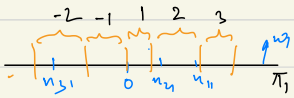
LSH for euclidean Distance





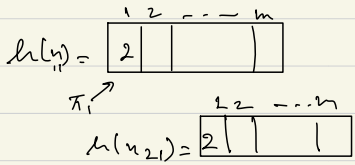
+1 or -1 in cosine sim

project points on π_1



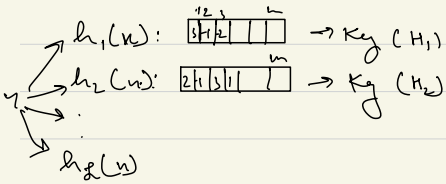
breaking plane into region (a)

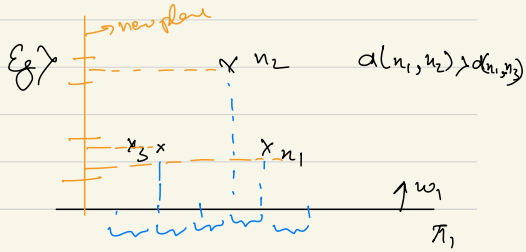
$a = 5$



Cos-sim LSH: [1 | -1 |]

euc dist LSH: [2 | -1 | 3 | 2 |]





$d(n_1, n_2) > d(n_1, n_3)$

$\{n_1, n_2\} \rightarrow$ same region but n_1 is very far from n_2

LSH is not perfect

↳ probabilistic / randomized algo

K-NN Probabilistic class label

n_q: 4-ve pt & 3+ve pt : $y_q = -ve$ {majority vote}

n_q': 7 (-ve pts) : $y_q' = -ve$

more certain

Quantify this uncertainity

$$P(y_q = -ve) = \frac{4}{7} = \left(\frac{\#\ -ve\ pt}{\text{Total}\ \#\ pt}\right)$$

$$P(y_q' = -ve) = \frac{7}{7} = 100\%$$