* **Avg word2vec :→**

r_1: w_1 w_2 w_1 w_3 w_4 w_5

n_1 words

$$V_1 = \frac{1}{n_1}\left[\underbrace{w_2V(w_1)}_{d\text{-}dim} + \underbrace{w_2V(w_2)}_{d\text{-}dim} + w_2V(w_1) + \cdots + w_2V(w_5)\right]$$

* **tf-idf - word2vec :→**

r_1 : $w_1 w_2 w_1 w_3 w_4 w_5$

tfidf :	w_1	w_2	w_3	w_4	w_5	w_6	w_7
	t_1	t_2	t_3	t_4	t_5	0	0

t stands for tf * idf.

$$\text{tfidf-}w_2V(r_1) = \frac{\sum_{i:\text{words}} \left(t_i * w_2V(w_i)\right)}{\sum_{i:\text{words}} t_i}$$

* **Classification :→** is to check that given a new review text, determine / predict if the review is positive or negative.

Its like finding a function.

$$\boxed{y = f(x)} \longrightarrow \text{central concept.}$$

x → review text

y → +ve/-ve

Training stage

$(x_i, y_i) \rightarrow$ Training $\rightarrow \boxed{F}$ algo

$x_q \rightarrow \boxed{F} \rightarrow y_q$ → Testing / evaluation stage.

$$D_n = \underbrace{\{}_{\text{set}} (x_i, y_i)_{i=1}^{n} \underbrace{|}_{\text{Such that}} x_i \in R^d, y_i \in \{\underbrace{0}_{-ve}, \underbrace{1}_{+ve}\} \}$$

Classification algorithm takes these x_i and y_i values as its training data set and then when you provide it any x_i it will apply the function it formed and tell you if the review is positive or not

* Classification Vs Regression :→

When $y_i \in \{\underbrace{0}_{-ve}, \underbrace{1}_{+ve}\}$, when y_i has only two values, then it is called 2 - class classification/ Binary classification,

But in MNIST dataset,

$$y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

↓

10 class / multiclass classification.

When you've $y_i \in R$

y_i is no more part of a small finite set of classes

then it is called a Regression problem.

So, the only diff in classification & Regression is the value of y_i

if $y_i \in \{0, 1\} \rightarrow$ or smaller finite number of values

its a classification problem

if $y_i \in R \rightarrow$ Regression

starting of KNN

★ K- Nearest Neighbours :→

$x_q \longrightarrow$ k-NN of x_q

$\downarrow$

$x_1, x_2, \ldots x_k$

$\downarrow$

$y_1, y_2, \ldots y_k$

$\downarrow$

majority vote

$\downarrow$

y_q · ∴ Ans

→ KNN continues

* ## Failure cases of KNN :→

a) when ~~x~~ x_i is so far away from all data points then, its difficult for KNN to solve it.

b) If data is so much concentrated and close and randomly ~~be~~ ~~...~~ spread. mixedup data then,

there is no useful info in it.

and KNN fails and most of the algorithm fails

If KNN works,

apply majority rule.

$K = odd$ $\{y_1, y_2, y_3\}$

$+ \ + \ + \longrightarrow \oplus \rightarrow y_q \rightarrow +ve$

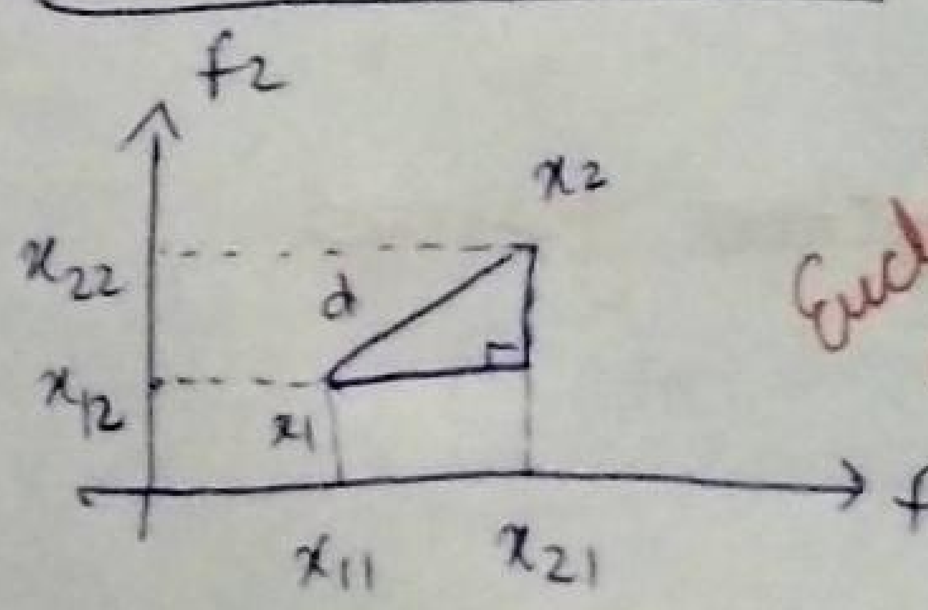
$+ \ + \ - \longrightarrow \oplus \rightarrow y_q = +ve$

$\cancel{4}$ if $K = 4$

do not work in this case

$+ + - - \longrightarrow \oplus$ or $\ominus$

* ## Distance Measures :→



Euclidean distance

$x_1 = (x_{11}, x_{12})$, $x_2 = (x_{21}, x_{22})$

d = len of shortest line from x_1 to x_2

$$d = \sqrt{(x_{21} - x_{11})^2 + (x_{22} - x_{12})^2} = \underbrace{||x_1 - x_2||}_{\text{len}}$$

For - d - dimensions :→

$x_i \in R^d$, $x_2 \in R^d$

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

↓

L2 Norm of a vector

* Manhattan dist :→ $\sum_{i=1}^{d} \underbrace{|x_{1i} - x_{2i}|}_{\text{absolute value}}$

$\|x_1 - x_2\|_1$ → L_1 norm of vector $(x_1 - x_1)$

* Minkowski dist :→ generalised form.

↓

Lp norms of vector

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

for $p = 2$ → minkowski dist ⟶ Euclidean dist.

$$\|x_1\|_p = \left(\sum_{i=1}^{d} |x_{1i}|^p\right)^{1/p}, \quad p \neq 0, \; p > 0$$

Lp norm

⇒ Distances are b/w two points

⇒ Norms are for the corresponding vector formed.

* Hamming dist :→ for boolean vectors.

x_1, x_2 → boolean vector → Binary Bag of words.

$x_1 = [0, 1, 1, 0, 1, 0, 0, \ldots]$
$x_2 = [1, 0, 1, 0, 1, 0, 1, \ldots]$
} → Hamming dist is 3.

Hamming dist (x_1, x_2) = no. of locations / dimensions where binary vectors differ.

Also used for strings.

x_1 = abcadefghik , x_2 = acbadegfhik

haming dist $(x_1, x_2) = 4$

Hamming dist is useful in case of Gene-code / seq
AGTC

x_1 = AAGTCTCAG ...
x_2 = AGATCTCGA ...

Hamming dist = 4.

* Cosine - similarity & cosine - distance :→

Similarity	distance	
↓ dec	↑ inc	opposite relation.
↑ inc	↓ dec	

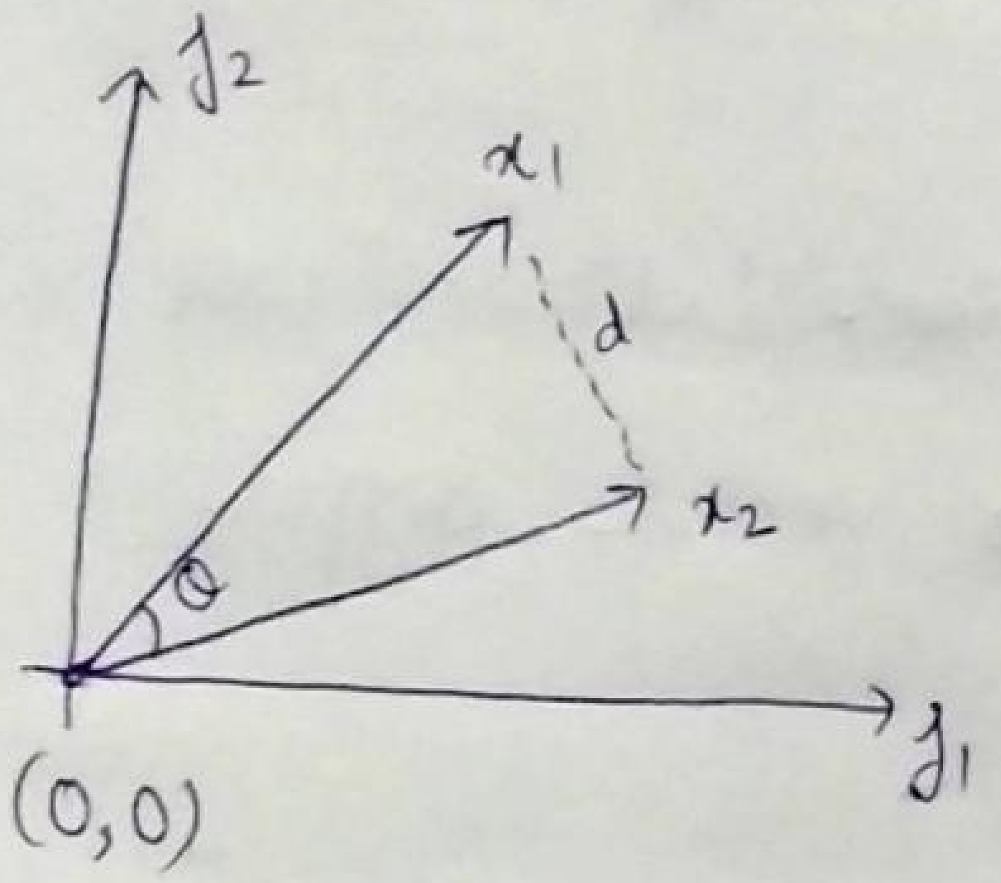
$$\boxed{1 - \cos\text{-}Sim(x_1, x_2) = \cos\text{-}dist(x_1, x_2)}$$

Scanned by TapScanner

let cos-sim $(x_1, x_2) \rightarrow [-1, 1]$

let v. similar cos-sim $(x_1, x_2) = +1$

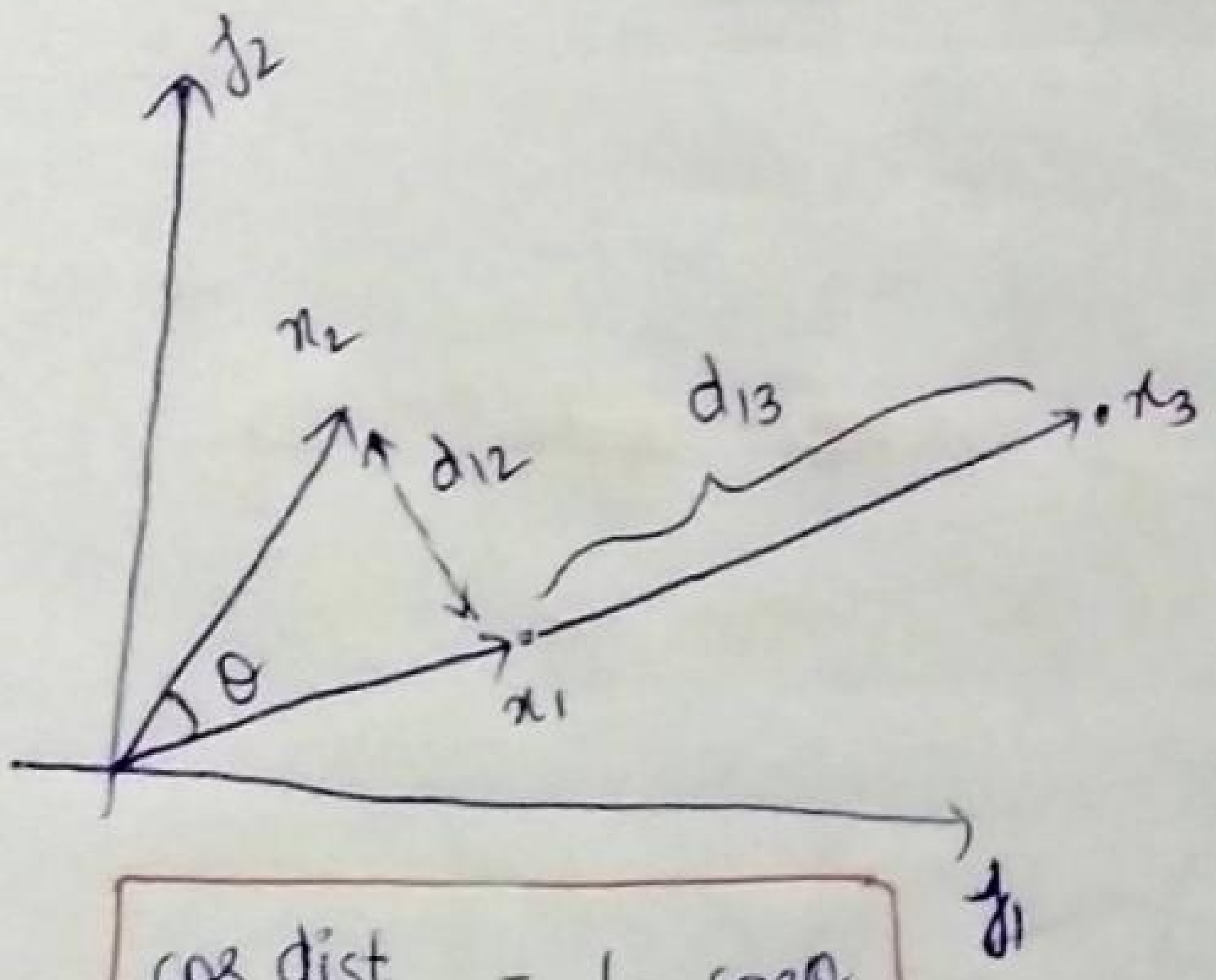
v. dissimilar cos-sim $(x_1, x_2) = -1$



d = euc-dist

$\left\{ \underset{(x_1, x_2)}{\text{cos-sim}} = \cos\theta \right\}$

θ: angle b/w x_1 & x_2



$\boxed{\underset{(x_1, x_2)}{\text{cos dist}} = 1 - \cos\theta}$

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

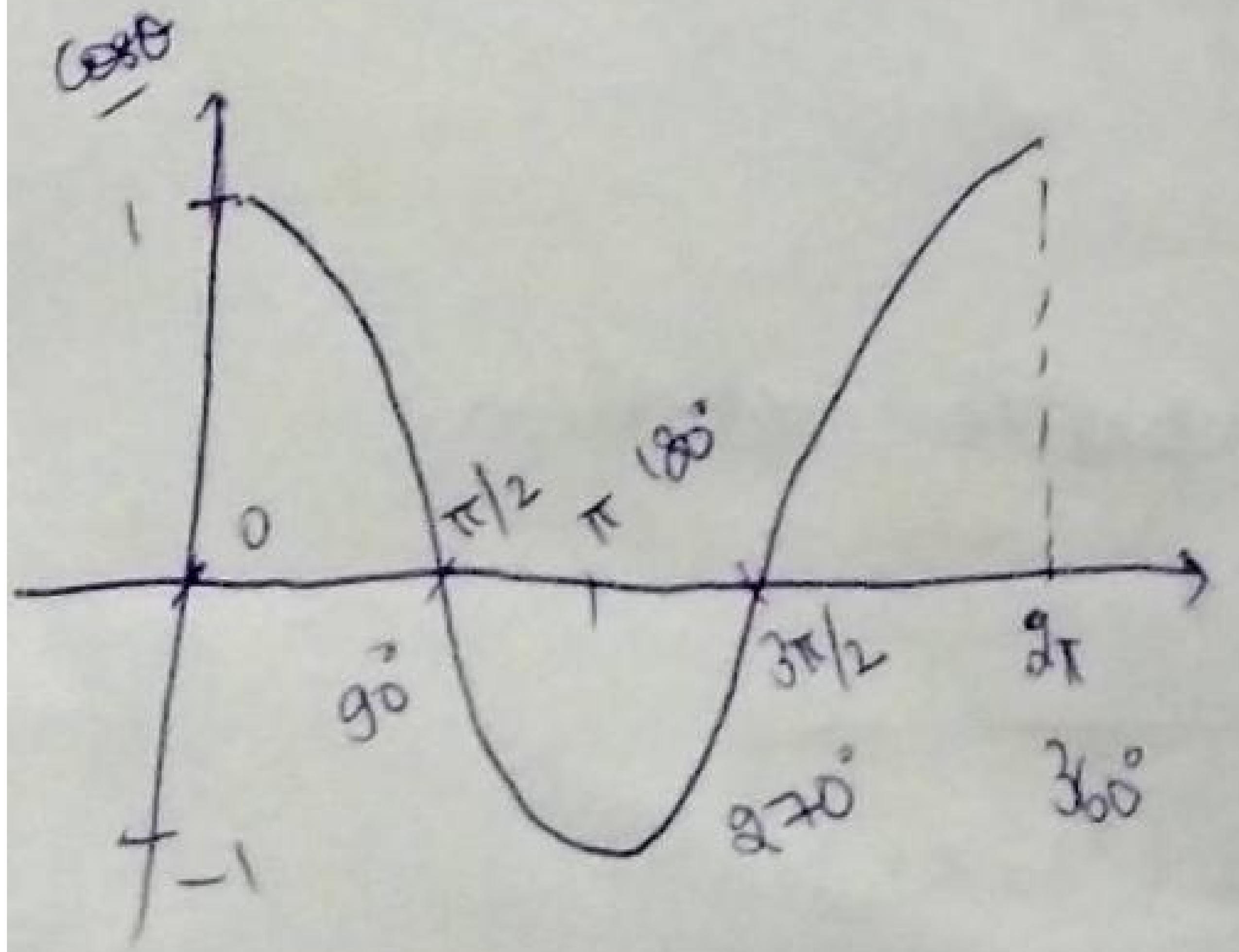
cos-sim $(x_1, x_3) = 1$

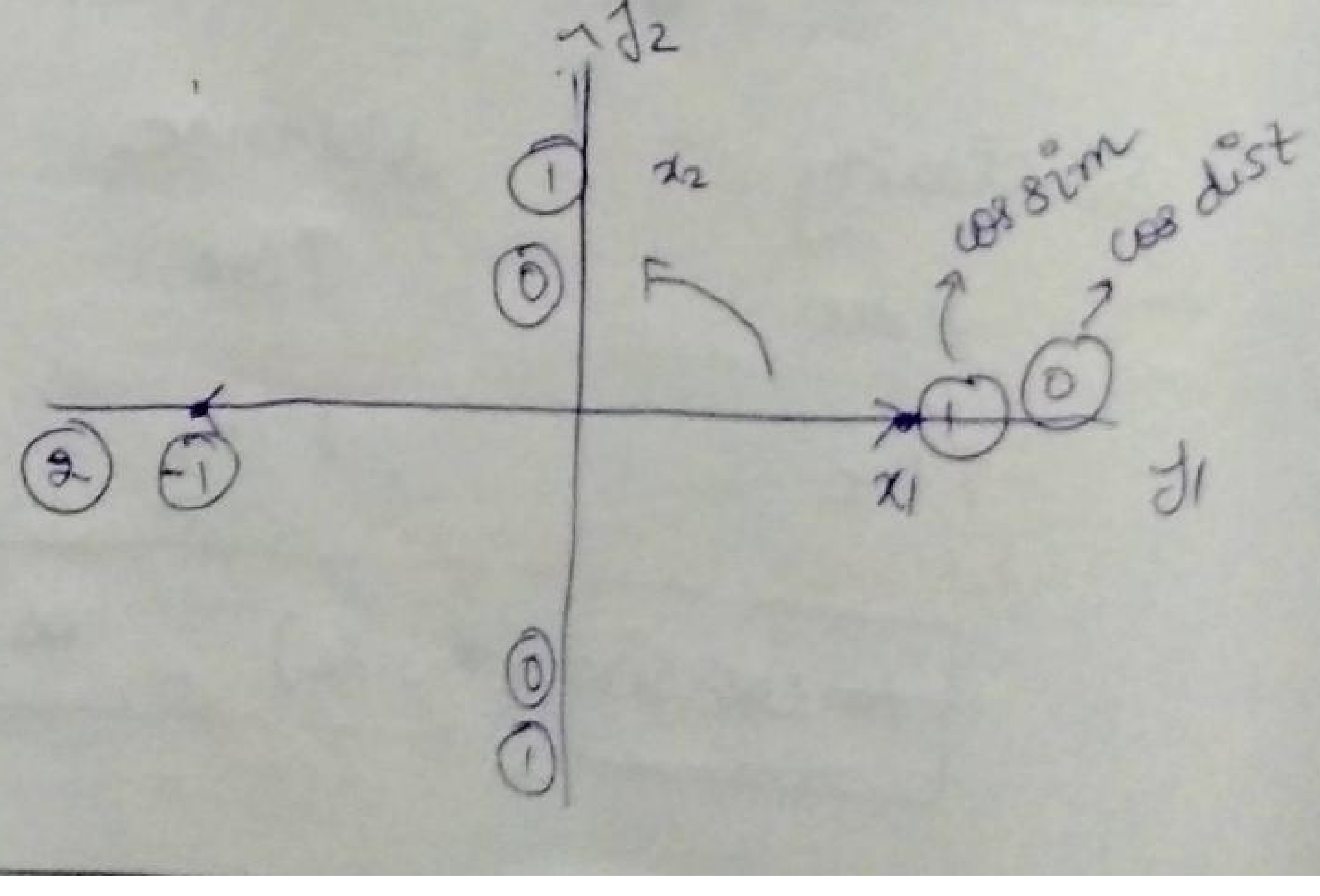
as $\theta_{x_1, x_3} = 0°$, $\cos 0° = 1$

$\boxed{\text{cos-dist}(x_1, x_3) = 1 - 1 = 0}$

So, even if euclidean distances

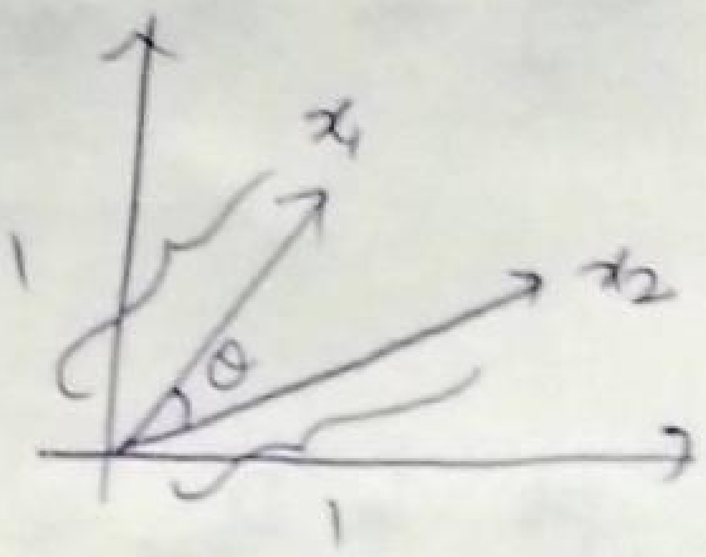
$$\left\{ \begin{array}{l} d_{13} > d_{12} \\ \text{cos-dist}_{13} < \text{cos-dist}_{12} \end{array} \right\}$$





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

$$\underset{\text{cos-sim}}{\cos\theta} = \frac{x_1 \cdot x_2}{\underbrace{\|x_1\|_2}_{l_2 \text{ norm of } x_1} \|x_2\|_2}$$



① If x_1 & x_2 are unit vectors

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

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

② Relationship b/w euc-dis & cos-sim (cos-dist)

imp

if x_1 & x_2 are unit vectors, where θ = angle b/w x_1 & x_2

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

$$[\text{euc-dis}(x_1, x_2)]^2 = 2 \text{ cos-dist}(x_1, x_2)$$

* How to measure how good KNN is?

data points

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

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

$n_1 + n_2 = n$

How to split $D_n \xrightarrow{R} D_{train}$ 70%

$D_n \xrightarrow{R} D_{test}$ 30%

Randomly

One way is to split it randomly

count = 0;

for each pt in D_{Test}:

$x_1 \rightarrow y_1$

x_q = pt

use D_{Train} & K-NN to determine y_q

if $y_q == y_{pt}$

count += 1

⇒ count = no. of points for which D_{Train} + KNN gave a correct class label.

$$\text{Accuracy} = \frac{\text{count}}{n_2}$$ = no. of pts for which D_{Train} + KNN gave correct class label.

no of points in D_{Test}

$$\boxed{0 \leq \text{Accuracy} \leq 1}$$

if Accuracy = 0.91 ⇒ 91% of times.

Test - Evaluation time & space complexity :→

$x_q \longrightarrow y_q$

k is small
↳ 5 or 10 etc.

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

KNNpts = [ ]

O(nd)

for each x_i in D_{Train} : n points → can be large enough
d-dim → can be large enough

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

time complexity

distance b/w x_i & x_q

KNNpts [ ]

O(1) ← Keep the smallest k- distances = (x_i, y_i, d_i)

cnt-pos = 0 ; cnt - neg = 0

for each x_i in KNN pts :

small ↑ O(K) ||| O(1)

if y_i is +ve
cnt_pos += 1
else
cnt-neg += 1

O(1)

if cnt_pos > cnt_neg
return $y_q = 1$ → +ve
else $y_q = 0$ → -ve

Time complexity → $O(nd) + O(1) + O(1)$

$= \boxed{O(nd)}$

if d is small

if $d << n$

$\boxed{O(n)}$

* Space complexity :→ Space that is needed to evaluate

$x_q \longrightarrow y_q$

$\boxed{O(nd)} \rightarrow$ to store D_{Train}

* Limitations of KNN (Simple implementation)

(Amazon) Fine food Reviews :→

Production ⟶ live

Time :→ $O(nd)$

Space :→ $O(nd)$

$n \simeq 364k$

$d \simeq 100k$ → DTrain

36,400M $\simeq$ 36GB → too large space needed.

(2) Time-complexity → 36 Billions computations.

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

↓

low-latency

↓

when you get y_q very very fast

$x_q \xrightarrow{fast} y_q$

$O(nd)$ is too bad

↓ =

few seconds

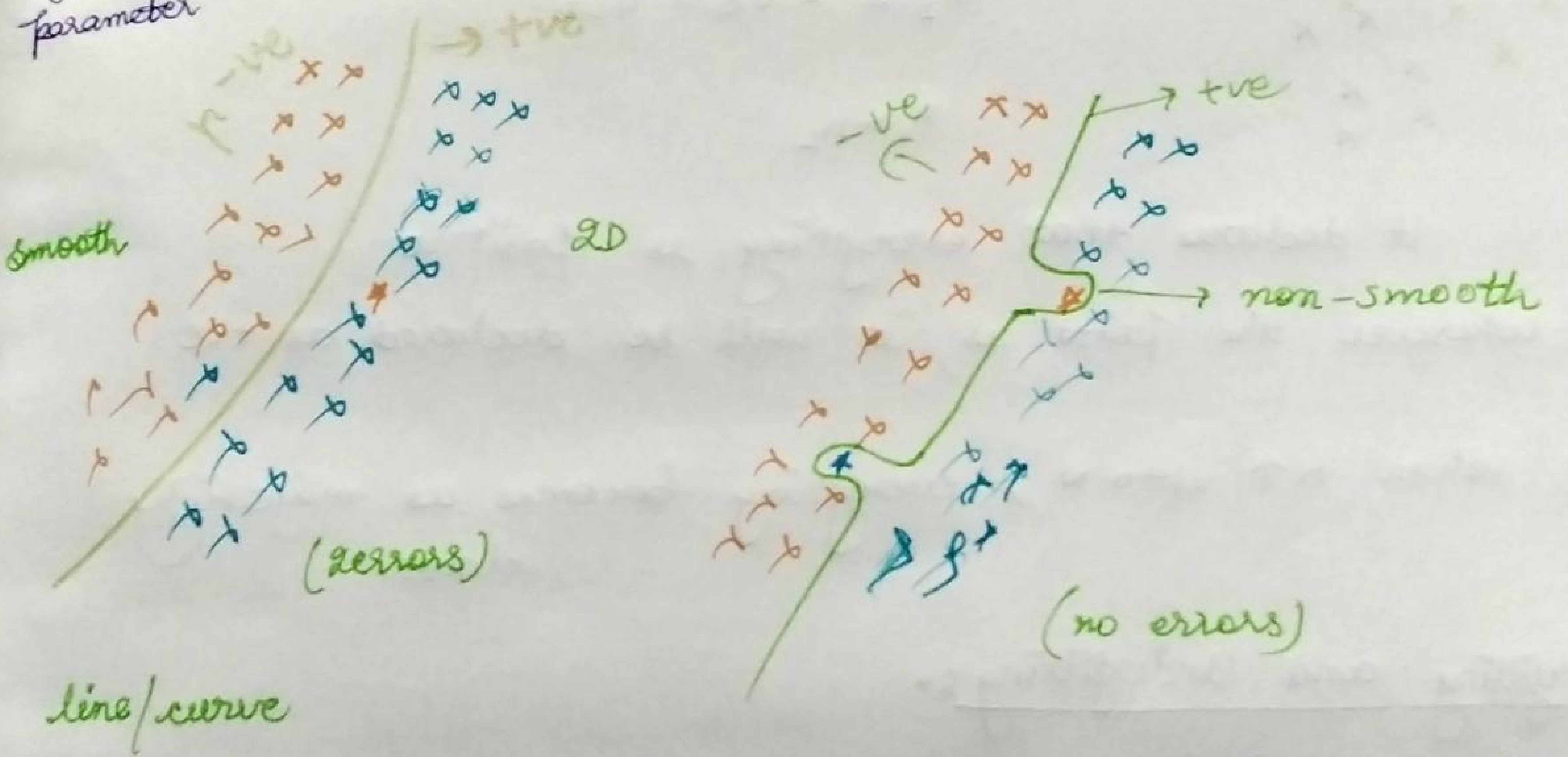
So, only reason not to use KNN is its terrible time and space complexity of O(nd).

otherwise its really simple & intuitive & elegant.

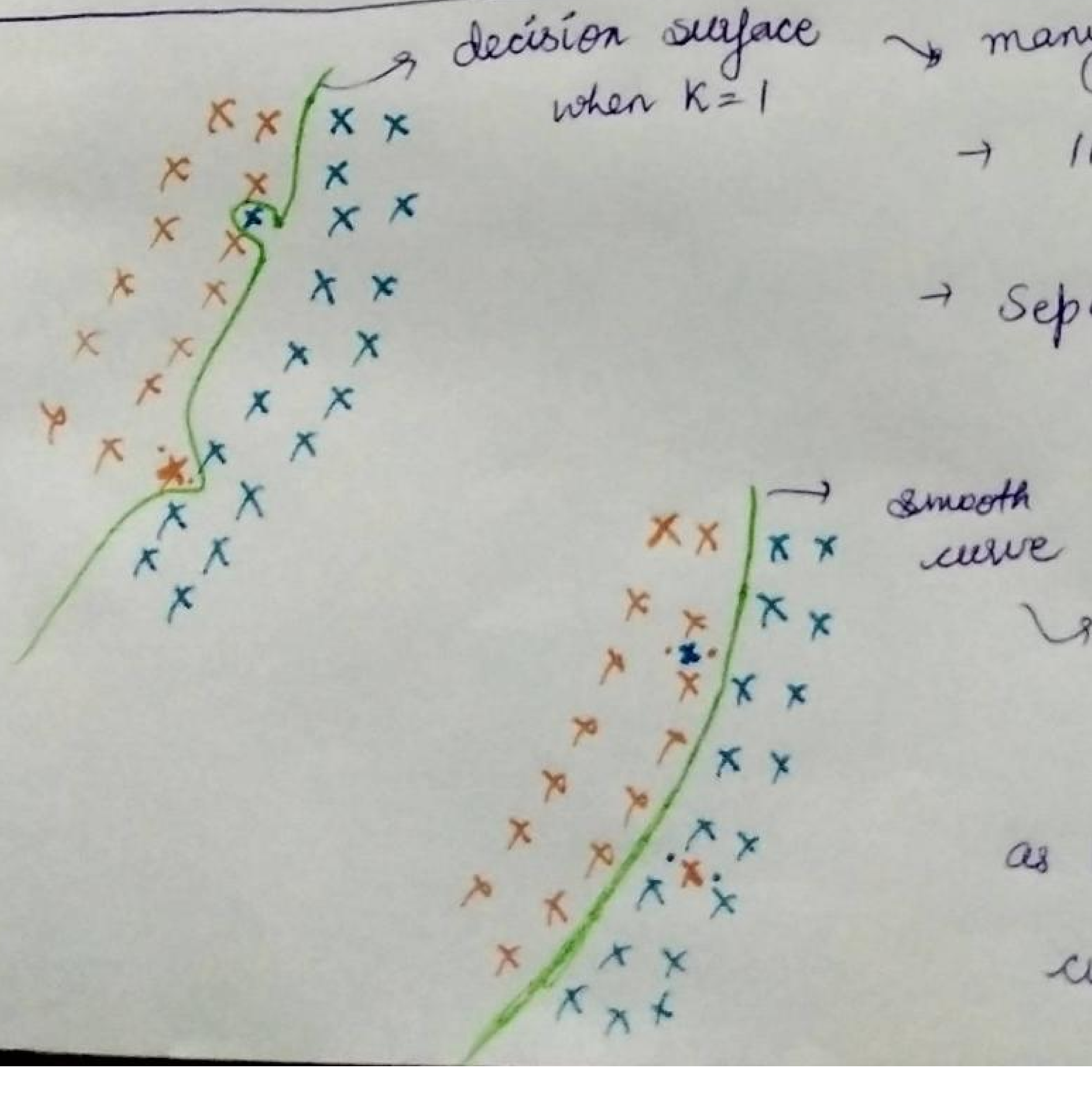
* Decision surface for KNN :→

'k' in KNN ← hyper parameter

Let K = 3, 5, 7, 15, ...



These curves are called decision surfaces.



→ many query points

→ 1NN, (K=1)

→ Seperate regions using a curve

↳ Let K=5 majority rule.

as K↑, smoothness of the curve increases.

Scanned by TapScanner

So, In KNN, the smoothness of the decision surface incre...
as K increases.

<u>worst case</u> :->

1000- nearest neighbours (NN)

$k = n = 1000$

$n_1 = +ve = 600$

$n_2 = -ve = 400$

$n_1 > n_2$

It declares that everything is positive.

So, wherever the point is, it will be declared as +ve.

So, when K↑ upto N, everything becomes as majority class.