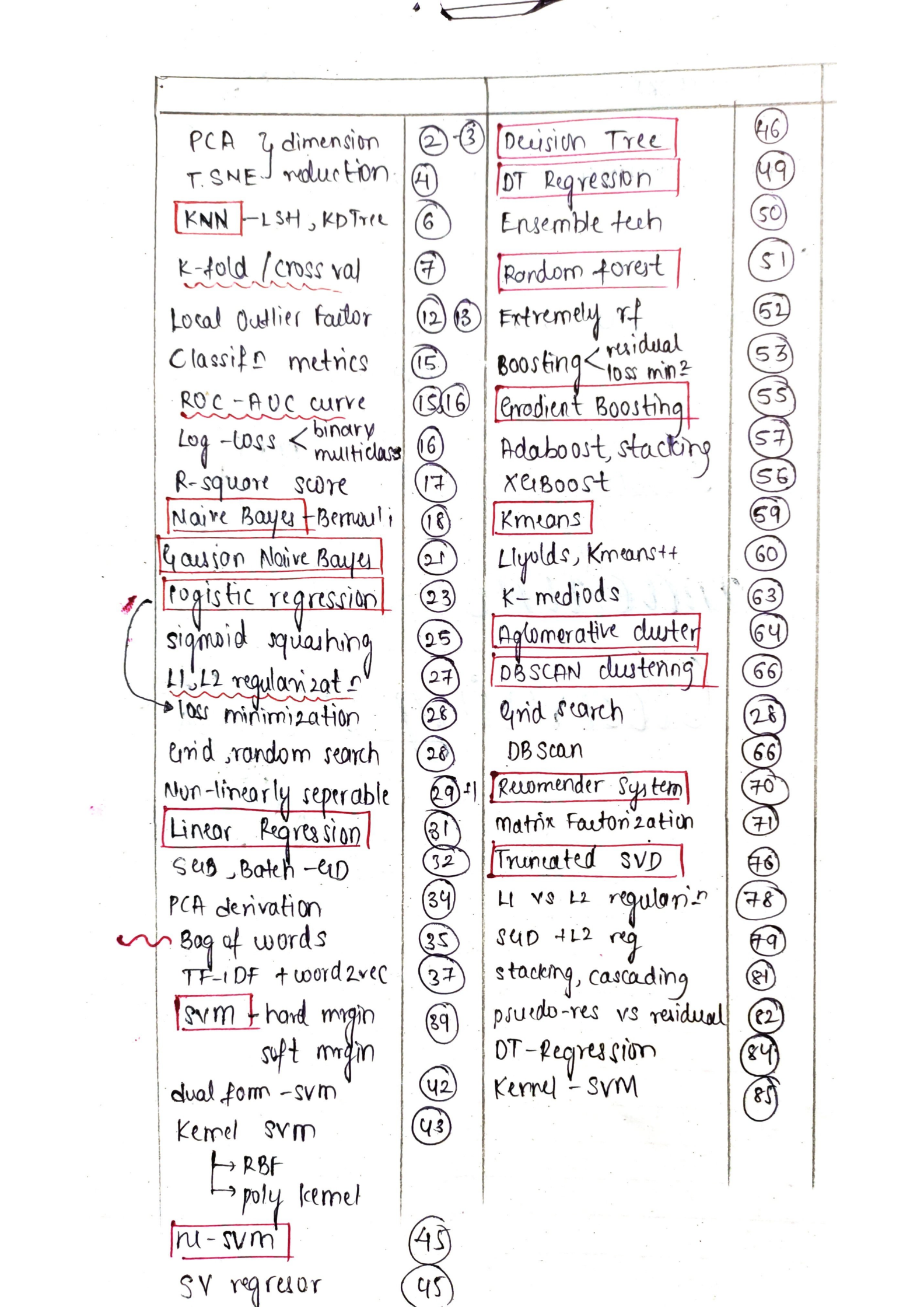

PCA & dimension reduction
T.SNE
2-3
KNN - LSH, KDTree
6
K-fold / cross val
7
Local Outlier Factor
12 13
Classifn metrics
15
ROC - AUC curve
15 16
Log-loss < binary multiclass
16
R-square score
17
Naive Bayes - Bernoulli
18
Gausion Naive Bayes
21
Logistic regression
23
sigmoid squashing
25
L1, L2 regularizatn
27
loss minimization
28
Grid, random search
28
Non-linearly seperable
29 +1
Linear Regression
31
SGD, Batch - GD
32
PCA derivation
34
Bag of words
35
TF-IDF + word2vec
37
SVM - hard mrgin
soft mrgin
39
dual form - SVM
42
Kernel SVM
43
RBF
poly kernel
nu-SVM
45
SV regresor
45
Decision Tree
46
DT Regression
49
Ensemble tech
50
Random forest
51
Extremely rf
52
Boosting < residual loss minz
53
Gradient Boosting
55
Adaboost, stacking
57
XGBoost
56
Kmeans
59
Llyolds, Kmeans++
60
K-mediods
63
Aglomerative cluter
64
DBSCAN clustering
66
Grid search
28
DBScan
66
Recomender System
70
matrix Factorization
71
Truncated SVD
76
L1 vs L2 regularizn
78
SGD + L2 reg
79
stacking, cascading
81
psuedo-res vs residual
82
DT-Regression
84
Kernel - SVM
85

Data Normalization

→ convert all data to $0 \le x_i \le 1$

$$x_i = \frac{x_i - \mu_{in(x)}}{x_{max} - x_{min}}$$

$$\boxed{x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}}$$

$x = x_{max} = 1$

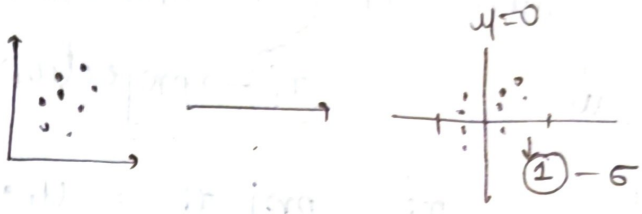
$x = x_{min} = 0$

Data standardization

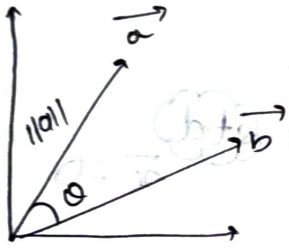
→ apply z-score

→ $\boxed{x_i = \frac{x_i - \mu}{\sigma}}$

→ $\sigma = 1, \mu = 0$



Dot product



→ $a \cdot b = [a_1 b_1 + a_2 b_2 \cdots]$

$\boxed{a \cdot b = a^T \cdot b}$

→ $\boxed{a \cdot b = \|a\| \|b\| \cos\theta}$

$\|a\| = \sqrt{a_1^2 + a_2^2 + \cdots a_n^2}$

$\|a\|$ → distance frm (0,0)

* if $\theta = 90$,

→ $a \cdot b = 0$

* if $\vec{a}$ is unit vector

$\vec{a} \cdot a^T = 1$

Covariance

$$\begin{matrix} & f_1 & \cdots & f_n \\ x_1 & & & \\ \vdots & & & \\ x_n & & & \end{matrix}$$

$\mu(f_1)$ $\mu(f_2)$

$$\boxed{Cov(f_1, f_2) = \frac{1}{n} \sum (x_i - \mu_1)(y_i - \mu_2)}$$

$$Cov(f_1, f_2) = \frac{1}{n} \sum (x_{i1} - \mu_1)(x_{i2} - \mu_2)$$

* if data is std → $\mu = 0, \sigma = 1$

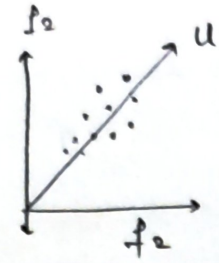
$$Cov(f_1 f_2) = \frac{1}{n} \sum x_i y_i$$

$$\boxed{S = \frac{1}{n} f_1^T f_2}$$ (n-1)

$$\therefore \boxed{S_{d \times d} = X^T_{d \times n} \cdot X_{n \times d}}$$

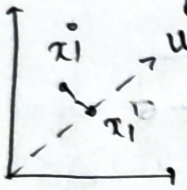
$S_{d \times d}$ — covariance matrix

(2) **Principal Component Analysis**



* find direction of a unit-vector such that variance is maximized.
* ie variance can be explained by fewer features.

d=2



unit vector u_1 ; x_i : data point

x_i' → projection of x_i on $\vec{u_1}$

$$x_i' = \text{proj}_{\vec{u_1}} x_i = \frac{u_1 \cdot x_i}{\|u_1\|} = u_1 \cdot x_i$$

$$\boxed{x_i' = u_1^T x_i}$$

→ find $\vec{u_1}$ such that var (x_i') is maximized

$$\text{var}(x_i') = \frac{1}{n}\sum_{i=1}^{n} \left(x_i u_1^T - u_1^T \cdot \overline{x_i'} \right)^2$$

as std $\bar{x} = 0$

$$\therefore \text{maximize } \frac{1}{n}\sum_{i=1}^{n} \left(u_1^T x_i \right)^2$$

such that $u^T \cdot u = 1$

Eigen values

$S_{d \times d}$ (covariance matrix)

for any matrix, (d×d) mat → (d) eigen val/vector

values $\lambda_1, \lambda_2 \; -- \; \lambda_d$

vector $v_1 \; v_2 \; -- \; v_d$

$\lambda_1 v_1 = S v_1$, also, $v_i \perp v_j$

$\lambda_n v_n = S v_n$ $\quad v_i \cdot v_j = 0$

$\forall$ all $i = 1 \to n$

we can proove that, $\vec{u_1} = \vec{v_1}$ → largest eigen vetor

let d=2, $\lambda_1 \lambda_2$ $\quad \lambda_1 = 3, \lambda_2 = 2$

λ_1 can explain $\frac{3}{5}$ % of variance

$\lambda_1 = 3$

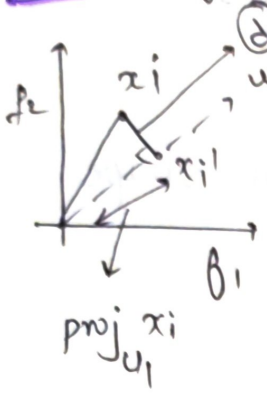
$\lambda_2 = 0$

% var exp by $\lambda_1 = \frac{3}{3} = 100\%$.

$\therefore$

λ_i can explain $\frac{\lambda_i}{\text{sum}(\lambda_n)}$ % of var

PCA using distance minimization



$\rightarrow$ minimize $\sum_{i=1}^{n} d_i^2$ $\rightarrow$ $d_i^2 + x_i' - \|x_i\|^2$

$d_i^2 = \|x_i\|^2 - x_i' = x_i^T \cdot x_i - u_i^T \cdot x_i$

$$\text{minimize} \sum_{i=1}^{n} \left(x_i^T x_i - u_i^T x_i \right)$$

PCA for dim reduction

consider $X_{15K \times 784}$ $\rightarrow$ top (200) largest Eigen vetors

$V = [\uparrow v_1 \; \uparrow v_2 \; \uparrow v_3 \; \cdots \; \uparrow v_{200}]$

$v_i = (200 \times 1)$ size

$$X_{15K \times 784} \times V_{784 \times 200} \rightarrow X_{15K \times 200}$$

workflow

1. standardize data
2. covariance mat $\rightarrow \frac{X^T \cdot X}{n-1}$
3. calculate Eigen values and vectors for (2) covariance matrix
4. pick top (n) eigen ~~values~~ vector for n-dimensions

code

```
→ from sklearn.decomposition import pca
→ pca = pca(n_components = 200)
→ p_data = pca.fit_transform(data)
```

$X_{n \times d} \rightarrow X_{n \times 200}$

plot % var

```
var = pca.explained_var_
      sum( )
cumsum = np.cumsum(var)
```

(4)

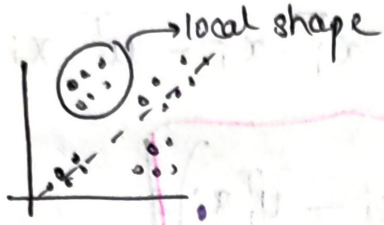
T-sne

T - distnbuted stochastic Neighborhood Embeddeding

T-sne → can preserve local and global shape.

↳ PCA cant preserve local shape

Ex:



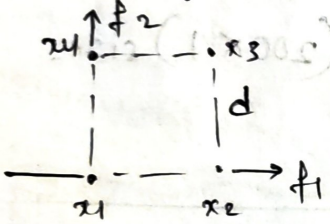
Neighborhood

↳ points which are nearer to given point

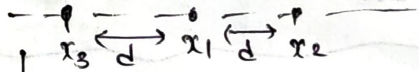
Embedding

↳ place pts from higher to lower dimensional space

Crowding problem



Embed →



stochastic

↳ probabilistic

↳ results may vary each time

$N(x_1) = x_2, x_4$

$N(x_2) = x_3, x_1$

$N(x_3) = x_4, x_2$

$N(x_4) = x_3, x_1$

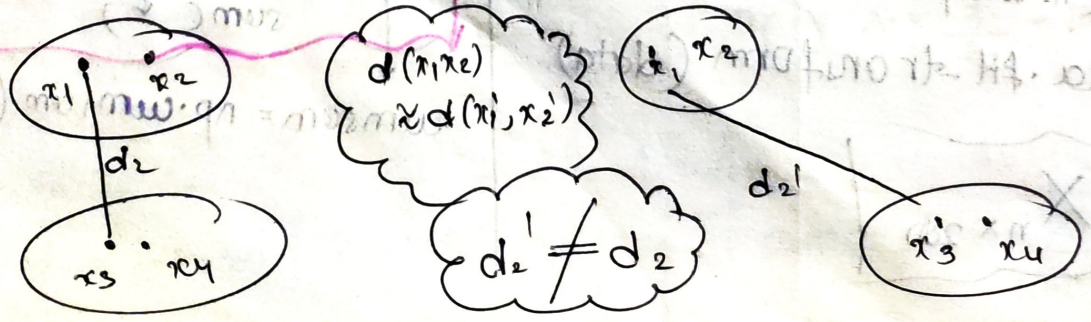
↳ you cant place x_4 on 1d-plane and preserve neigborlod with booth x_3 and x_1

↳ only 1 can be preserved

Geometric Intuition

↳ intra ~~inter~~ cluster nbrhood is preserved

↳ inter cluster nbrhud may not be preserved



5) tsne tips

1. always check multiple params
2. change it until shape stabilize
3. $2 \leq \text{perplexity} \leq n$
4. cluster sizes and intercluster distances may vary.

- iteration
- perplexity → no of nbrhood pts for which maintain neighborhood
- epsilon (learning rate)

dist – 2 pts
norm – 2 vectors

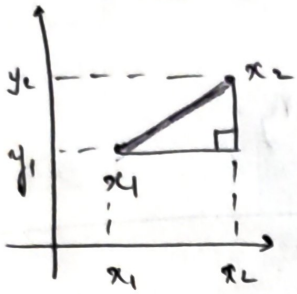
Distances

KNN

hyper-p = K

No of dimensions

L2 Euclidean → $\sqrt{\sum (x_{21}-x_{11})^2 + (y_{21}-y_{22})^2}$ →



manhatan (L1) → $\sqrt{\sum (y_2-y_1) + (x_2-x_1)}$ → absolute →

minkoski (Lp norm) → $\left(\sum_{i=1}^{d} |x_{1i}-x_{2i}|^p\right)^{1/p}$

p = 2 → Euclidean
p = 1 → manhattan

KNN steps

1. given pt as i/p
2. calculate nearest (n) pts
3. res = majority voting

Algo

i/p → D_{train}, K, $x_q \in R^d$

to find → class of (x_q) ← test pt

KNN = [ ]

for x_i in D_{train}:

compute $d(x_i, x_q)$ → O(d)

keep (K) smallest distances – (d_i, x_i, y_i) → O(K) = small

res = majority_vote (KNN)

TC = O(nd) + O(1) + O(1) → majority voting

n-pts with d-dimensions

selecting K

decision surface

* smaller K – ruged line
* as K↑ → smooth line
* if (K == n) all pts will be classified as majority class

Outlier

→ if (K) is small, more impact of outliers

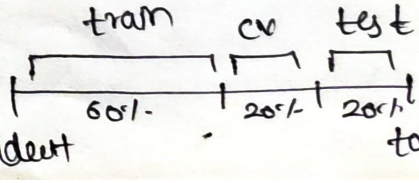
as K↑, less prone to outliers

Time Based Split

for spliting dataset

1. sort record acc to time

train – Earliest 60%
test – most recent 20%
CV – mid 20%



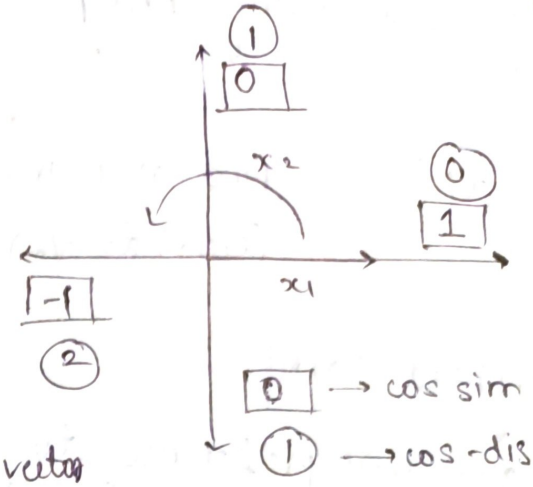
Cosine similarity and distance

→ as sim ↑ dist ↓

→ angle b/w vectors

cos-sim = cosθ

cos-dist = 1 − cosθ



cos-sim = +1 similar

cos-sim = −1 different

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

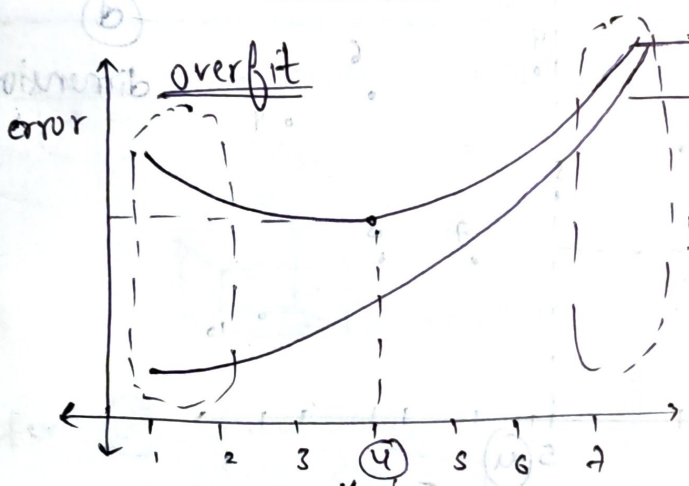
① $\vec{x_1}, \vec{x_2}$ are unit vectors

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

$$\|\vec{A} - \vec{B}\|^2 = 2(1 - \cos(a,b))$$

$Euc(a,b)^2 = 2 \cdot cos\text{-}dist(a,b)$

Overfit and underfit



cv error → use train + cv

training error → only training ds

Train error ↑, val error ↑ } underfit

Train error ↓, val error ↑ } overfit

K → Best value of (K) chosen with least error previously

Cross val – KNN

dataset → dtrain, dcv, dtest

→ for (xi, yi) in dcv:

use dtrain and find f(xi)

if f(xi) = yi → correct

else → wrong

K-fold CV

dtrain, dtest

d-train

1	2	3	4

split dtr into parts

→ pick any (1) part as dcv.

Ex :-

	train	CV	acc
K=1	1 2 3	4	0.9
	1 3 4	2	0.8
	1 2 4	3	0.9

acc for (K=1) = avg ( )

Regression using KNN

→ majority vote → classification

i/p (x_i) → Find n-nearest neighbours → mean $(y_1, y_2 \dots y_n)$

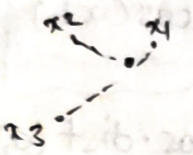
$(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$

or

median $(y_1, y_2 \dots y_n)$

Weighted KNN

give more preference to more nearer points



3NN →

	class	distance	weight
x_1	+	0.1	$\frac{1}{dist}$
x_2	−	0.9	
x_3	−	0.10	

TC (KNN) = O(n)

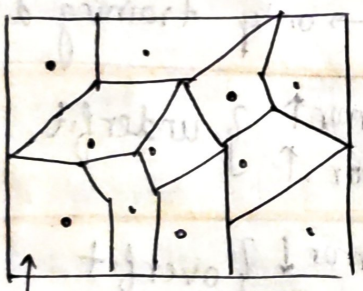
using kd tree

→ O(log n)

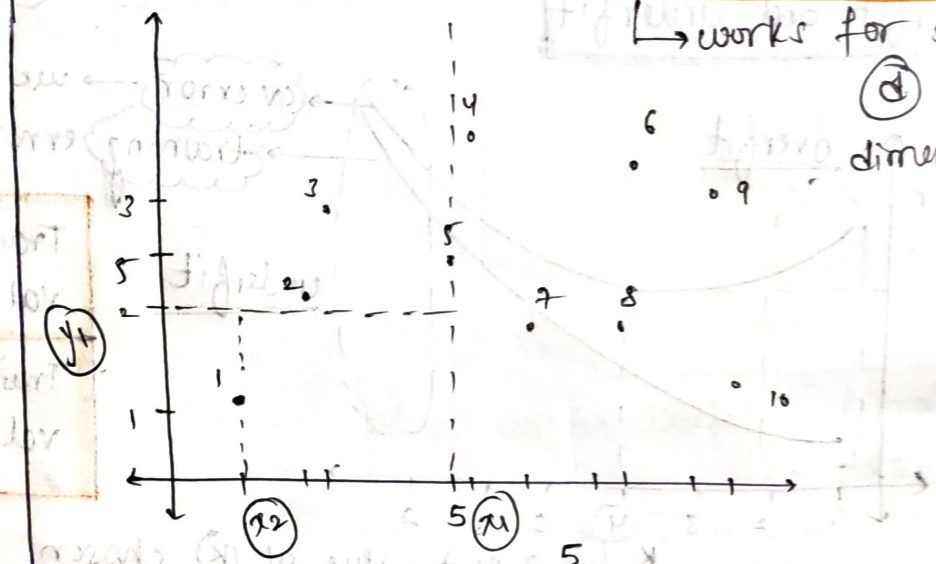
Kd tree

→ works for small (d) dimension

Voronoi diagram



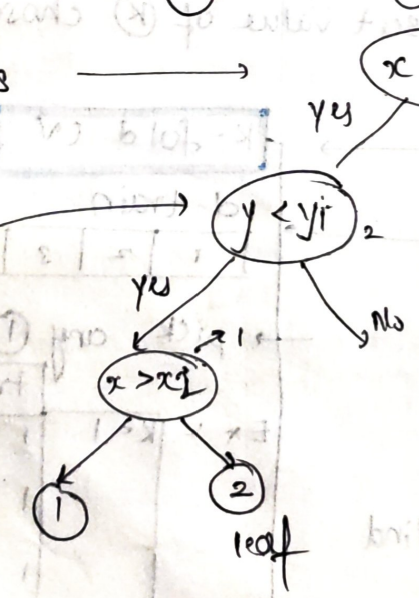
any pt in this region will have ① as first NN.



① project lines at (x) axis → $x_i = 5$ → median

② pick y axis → $y_i = 2$

③ pick x axis → $x = 1$



Kdd

Best case − O(log n)

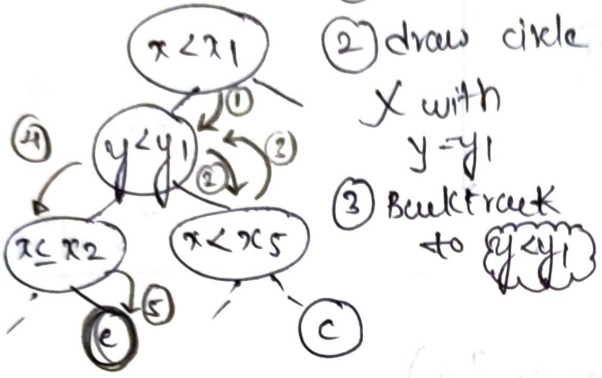
worst case − O(n)

Find nn using Kd tree

(1) use kd tree to find nearest neighbor

(2) draw a circle around that pt and check for any other nearest pt.

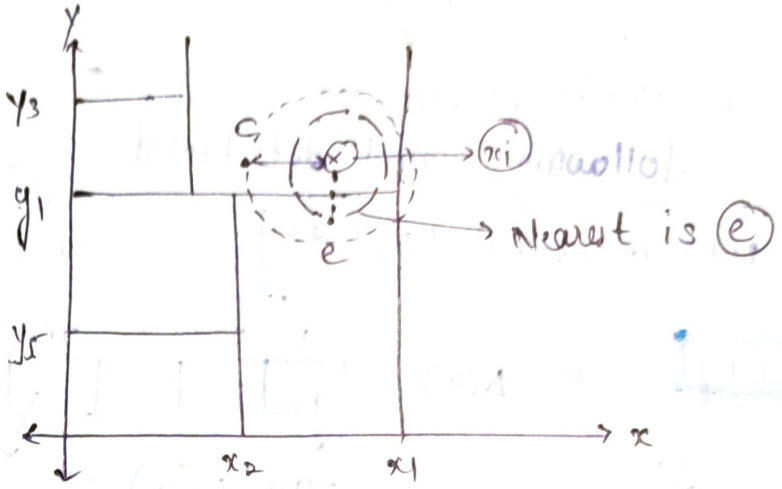
Ex



(1) $x_i \in$ (c)

(2) draw circle X with $y = y_1$

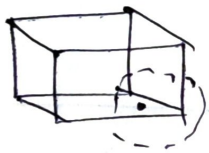
(3) Backtrack to (y<y1)



Limitations of Kd tree

→ check in 4 cells

→ 3d → 8 cells



→ 8 cells

for (d) dimensions,

check for (2^d) cells

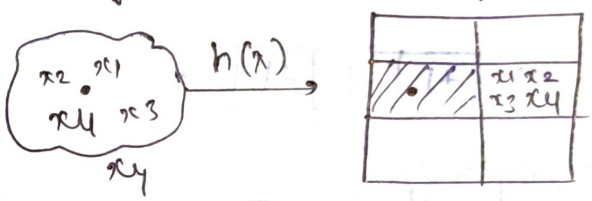
when d is large,

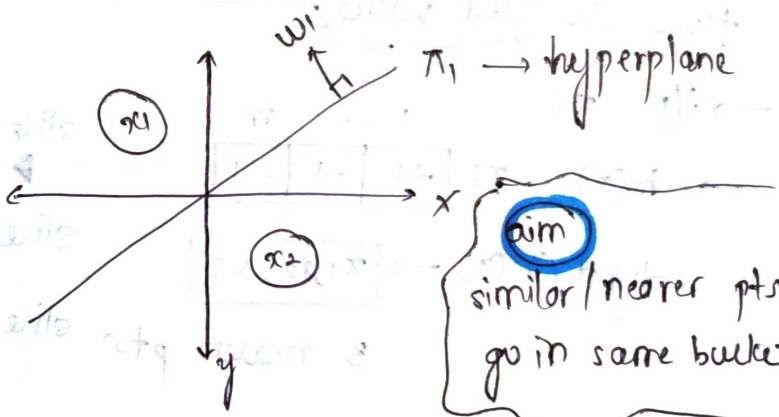
TC → $O(2^d \log n)$

=

$2^d \log n \approx n \log n$

Locality Sensitive Hashing LSH





w_1 — normal to π_1 plane

$w_1^T \cdot x_1 \geq 0$ same direction

$w_1^T \cdot x_2 \leq 0$ other side

Eqn of plane passing through origin $\{w^T \cdot x = 0$

else $\{w^T \cdot x + b = 0$

LSH for cos-sim

m → no of hyperplanes

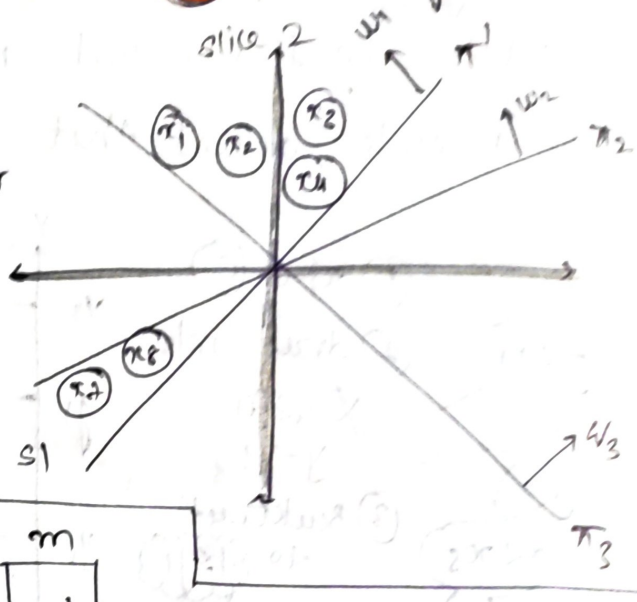
① Generate random hyperplane

→ w → [| | | |] d-dim vector

1 -- d

Each value is random no following normal distributed

np.random.normal(0,1,d)



Hashing → h(x) = [1 | 2 | -- | m]

sign($w_1^T x$) sign($w_m^T x$)

Ex:

slice (S1)

(x_7, x_8) → [+1 | -1 | -1] (1 2 3)

slice (2)

$\begin{pmatrix} x_1 x_2 x_3 \\ x_4 \end{pmatrix}$ → [+1 | +1 | +1]

space :- O(nd)

Time : O(m·dn)

Query: O(md + rd') → O(md)

How to find value?

→ i/p x_i

→ h(x) = x_i [+1 | -1 | -1] (1 -- m)

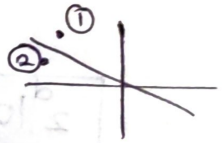
→ digest(h(x)) → [x_1, x_2, x_3]

3 nearest pts

Hash table

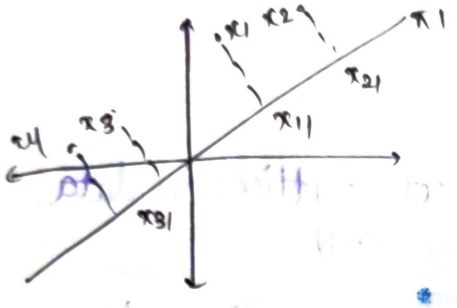
slice 1 [1 | 1 | 1]	{x_2, x_1}
slice 2 [-1 | +1 | 1]	{x_1, x_3, x_3, x_4}
slice 3 [-1 | -1 | -1]	{x_1, x_2}

problems → you could miss nearest neighbor due to cosine-similarity

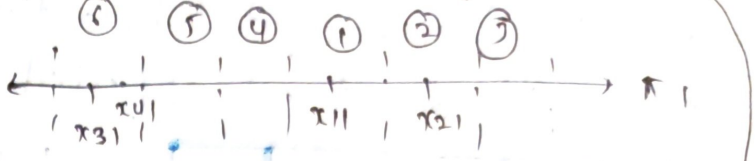


soln → have different hashtable with different hyperplanes

LSH for Euclidean values



divide π into (a) regions



$h(\pi) =$ 

region in which pt falls in on π_1 (for π_1)

$h(x_{11}) \rightarrow$ | 1 | - - |

$h(x_{41}) \rightarrow$ | 6 | - - |

x_{11} → projection of x_1 on π_1

* pts falling in same bucket are closer

KNN probabilistic class label

5-NN (+,+,+,+,+) → +ve
(+++,-,-) → +ve
} not fair → $P(x_i = +) = 1$, $P(x_i = +) = 3/5$

CLASIFICATION METRICS

Imbalanced dataset issue:

① Undersampling

D < 100 (+ve), 900 (-ve)

↓

D' < 100, 100

Eliminate 800 randomly from (-ve) class

② Oversampling

D < 100, 900 (repeat again)

↓

D' < 900, 900

① repeat

② create synthetics pts - extrapolation

③ class weights

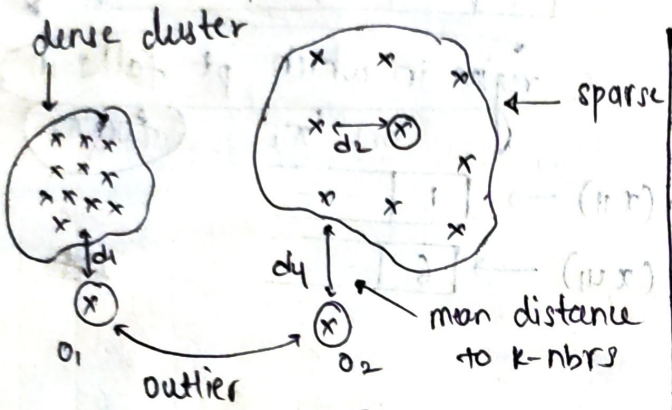
a) minory class ↑weight

b) majority class ↓weight

Impact of outliers

* smaller (K) more affected by outliers
* larger (K) doesnt get affected

local outlier Factor | LOF → technique to detect outliers in data inspired by KNN



Terms

* **K-distance (x_i)** → dist b/w (x_i) and farthest neighbor (ie K)

  distance of (x_i) from K^{th} nearest neighbor of x_i

* **Neighborhood (x_i)**

  $N_{k=5}(x_i)$ = {5 nearest nbrs}

* **Reachability distance**

$$rd(x_i, x_j) = \max(\text{K-distance}(x_j), dist(x_i, x_j))$$

(K-distance(x_j) → farthest K-nbrs)

Ex:- K=5

if $x_i \in N_{k=5}(x_j)$

reach_dist = 5-dist(x_i)

else

reach_dist = $d(x_i, x_j)$

Simple approach → (X) fails

1. for each pt (X) find K-NN
2. calculate mean distance (X) → KNN
3. sort all to mean dist

   large mean dist = outlier

fails : o_2 detected as outlier → d_4 ↑

o_1 !detected

as $d_1 \leq d_2$, because of sparse cluster

NAIVE SOLN

LOF → code

sklearn.nbrs.LocalOutlierFactor(

n_nbrs = n, ---,

contamination = ct))

→ def = 1%

↓ % of outliers you expect

→ 0.05 (5%)

LOF Flow

1. Calculate LOF ∀ $x_i \in$ Dataset
2. pick top-N (highest) LOF pts → outliers

Local reachability density | LRD

density → # per unit area

$$LRD(x_i) = \frac{1}{\sum_{x_j \in N(x_i)} \frac{reach\text{-}dist(x_i, x_j)}{|N(x_i)|}}$$

→ average reach-dist of x_i from its neighbors

$|N(x_i)|$ → size of set ≈ K

$\frac{1}{\text{avg reach-dist}}$

$$LRD(x_i) = \frac{|N(x_i)|}{\sum reach\text{-}d(x_i, x_j)} \left(\frac{\text{No of pts}}{\text{distance}}\right)$$

LOF

LOF ↓ → inlier

LOF ↑ → outlier | a↑ b↓

$$LOF(x_i) = \frac{\sum_{x_j \in N(x_i)} LRD(x_j)}{|N(x_i)|} * \frac{1}{LRD(x_i)}$$

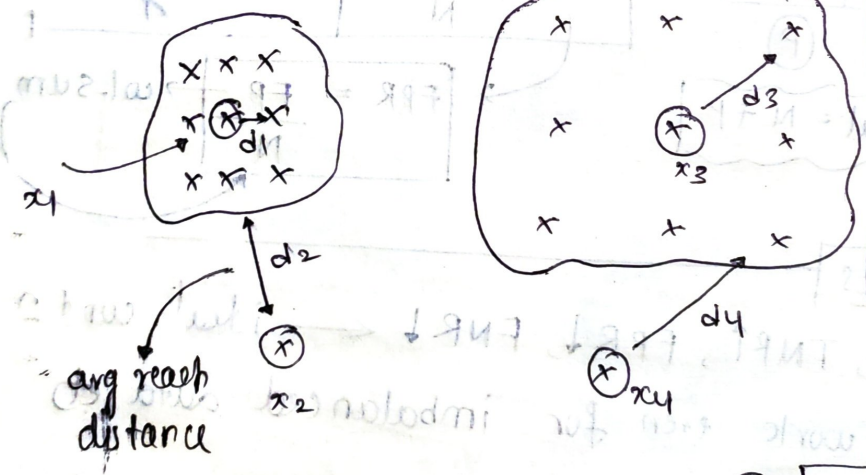
a: density in nbrs of x_i

b: avg LRD of pts in $N(x_i)$

$\frac{1}{LRD(x_i)}$ → density around x_i

LRD (x_i) is small compared to nbrs

Example



avg reach distance

approximate

* avg reach-dist : $d_1 < d_2 < d_3 < d_4$
* LRD → 1/avg(r-d)

$\frac{1}{d_1} > \frac{1}{d_2} > \frac{1}{d_3} > \frac{1}{d_4}$

↓

density(x_i) large

density $\propto \frac{1}{\text{distance}}$

① x_1

$LRD(x_i) \approx LRD(x_i \in N(x_i))$

→ LOF small

② x_2

$LRD(x_2) << LRD(x_i \in N(x_2))$

→ large LOF

③ x_3 → both are small

$LRD(x_3) \approx$ LRD(nbrs of x_3)

→ LOF small

④ x_4

$LRD(x_4) << LRD(\text{nbrs of } x_4)$

→ LOF is large

Class ⚖ metrics

Accuracy → problems

$$acc = \frac{\#\ correct\ preds}{total\ preds}$$

① Imbalanced data

let D ← 80% (+ve), 20% (-ve)

* assume dumb model ie op is ⊕/⊖

if ⊕ $acc = \frac{80}{100} = 80\%$ → misleading.

problem ②

it cant deal with prob scores

y	M_1	M_2
0	0.9	0.6
0	0.8	0.56
1	0.1	0.3
1	0.2	0.4

→ p(y==1)

→ it cant deal with prob → only way → round off

M_1 →	1	1	0	0
M_2 →	1	1	0	0

acc to acc $M_1 = M_2$

but m_1 is better than m_2 in reality

Confusion matrix

pred \ actual	0	1
0	TN	FN
1	FP	TP

was 0, predicted as 1

	0	1
0	TN	FN
1	FP	TP
	(N)	(P)

total = N + P

TP rate = $\frac{TP}{P}$	TNR = $\frac{TN}{N}$
FPR = $\frac{FP}{N}$	FNR = $\frac{FN}{P}$

$FPR = \frac{FP}{N}$ → col-sum

F P

(F) were u correct

(P) what was the prediction

Ex: Cancer

very high TPR

very low FNR

uses

TPR↑, TNR↑, FPR↓, FNR↓ ← ideal cond ⚖

can work even for imbalanced data set

issues

* you need to check ④ values

FP — Type I error

FN — Type II error

Precision

$$\text{Precision} = \frac{TP}{TP+FP}$$

* of all +ve predictions, how many were actually +ve

Recall

$$\text{Recall} = TPR = \frac{TP}{P}$$

* of all the actually +ve points, how many were predicted properly

F1 score

$$\text{F1-score} = 2 * \frac{\text{Precision} * \text{recall}}{\text{Precision} + \text{recal}} \approx \left[\text{avg}\left(\text{recall}^{-1}, \text{precision}^{-1}\right)\right]^{-1}$$

ideal → F1↑, Precision↑, Recall↑

Binary classifier ← **ROC curve** Receiver operating characteristic curve

can be applied to binary class.

Step 1) sort acc to prob class-label (decreasing)

x	y	$\hat{y}$
x_4	1	0.95
x_2	1	0.8
x_3	0	0.7
x_4	0	0.6

②

$\hat{y}$	$\tilde{y}_{z_1=0.95}$	$\tilde{y}_{z_2=0.8}$
1	1	1
1	0	1
0	0	0
0	0	0

sklearn chooses z_i by itself

Step 2) Thresholding (z_i)

if $\hat{y} > z_i$ → +ve

else → -ve

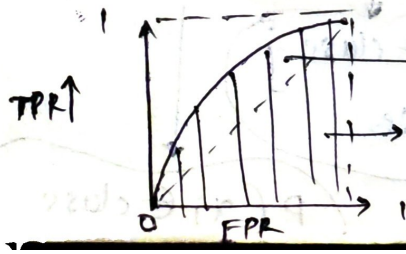
① For each threshold calculate accuracy

② select threshold with max accuracy.

Step 4) Calculate FPR and TPR for each $z_1, z_2 \dots z_n$

to calculate FPR/TPR u need 0,1 got by thresholding

Step 4) plot



→ only good for binary class

(area under ROC curve)

1 → best

0 → worst

AUC properties

① prone to imbalanced datasets

② only depends on order of $\hat{y}$ and not actual value of $\hat{y}$

③ AUC of random model = 0.5 → model which gives 0 or 1 without logic randm

④ AUC < 0.5 : worse than random

Log-Loss → uses probability scores

$0 \rightarrow \infty$

log-loss

lesser = better

x	y	$\hat{y}$	
x_1	1	0.9	$-\log(0.9) = 0.0457$ → closer to ①
x_2	1	0.6	$-\log(0.6) = 0.22$ → far from ①
x_3	0	0.1	$-\log(0.9) = 0.0457$ → closer to ⓪
x_4	0	0.4	$-\log(0.6) = 0.22$

$$\text{log-loss} = -\frac{1}{n}\sum_{i=1}^{n}\left(\log(P_i) * y_i\right) + (1-y_i)\left(\log(1-P_i)\right)$$

activated when class = +ve/1 / class = -ve/0

English defn: log-loss = average of negative of log(P(correct class label))

= Avg of neg of log (P (correct class))

multiclass

x, y (0–n), $\hat{y}$ → P(0) -- P(C)

c classes

$$\text{log-loss} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij} * \log(P_{ij})$$

activate only for class it belongs to

1 : $x_i \in$ class j

0 : else

$P(x_i \in \text{class } j)$

R^2 score only for regression

(error) $SS_{residual} = \sum_{i=1}^{n} (y_i - \hat{y})^2 \rightarrow (y_i - \hat{y})^2$

(dif frm mean) $SS_{total} = \sum_{i=1}^{n} (y_i - \bar{y})^2$

$$R^2 = \frac{1 - SS_{res}}{SS_{tot}} = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

Case 1 : No error → $SS_{res} = 0$ → $R^2 = 1 - 0 = 1$ ← best

Case 2 : $SS_{res} < SS_{tot}$ → $R^2 = 0$ to 1

Case 3 : $SS_{res} = SS_{tot}$ → $R^2 = 0$ (our model is simple mean model)

Case 4 : $SS_{res} > SS_{tot}$ → $R^2 < 0$ (your model is worse than mean model)

issues → cant handle outliers Ex:- $R^2 \propto SS_{res} \propto \sum error_i$ { if any one error is huge SS_{res} gets effected

MAD → median absolute deviation

mean	median
std	mad

→ not prone to outliers

$$MAD = \frac{1}{n} \sqrt{|x_i - median(e_i)|}$$

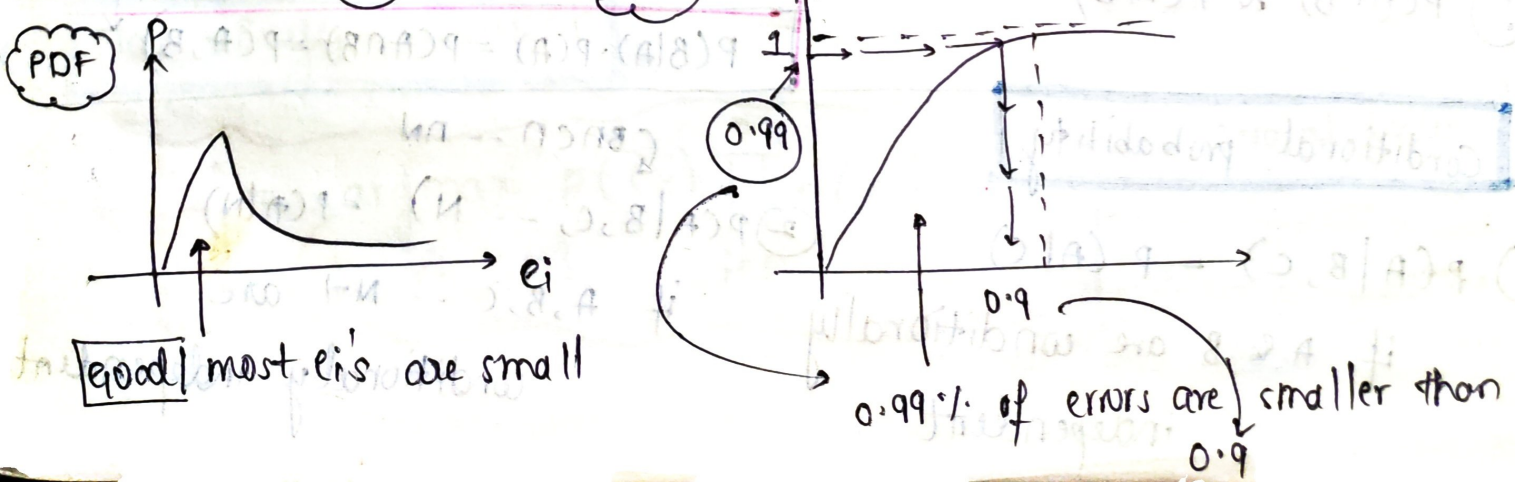
$$= median(|e_i - median(e_i)|)$$

MAD → lower the better

median ⇒ errors are small assuming only few outliers have large error

distribution of errors

Errors → e_i



Naive Bayes

(Hyper param = α)

* conditional prob → $P(A|B) = \frac{P(A \cap B)}{P(B)}$; $P(B) \neq 0$

probability of ownrence of (A) given (B) already occured

Events

* Independent event

A, B : events

$P(A|B) = P(A)$

$P(B|A) = P(B)$

* mutually exclusive

$P(A|B) = P(B|A) = 0$ → also $P(A \cap B) = 0$

Ex; A: 3 on Dice 1

B: 2 on Dice 1

$\therefore$ A, B r mutually exclusive events

$P(A|B)$ → 3 has occured on (B)

(B) → it cant be (2) as already (3)

Bayes Theorem

Derivation

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$P(A|B)$ → posterior prob

$P(B|A)$ → likelihood prob

$P(A)$ → prior prob

$P(B)$ → evidence

(!) $P(A \cap B) \approx P(A, B)$

→ $P(A|B) = \frac{P(A \cap B)}{P(B)}$ → (1)

→ $P(A \cap B) = P(B \cap A)$ → (3)

→ $P(B|A) = \frac{P(B \cap A)}{P(A)}$

→ $P(B \cap A) = P(B|A)\,P(A)$ → (2)

sub (2) in (1) using prop (3)

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

**** $P(B|A) \cdot P(A) = P(A \cap B) = P(A, B)$

Conditional probability

(1) $P(A|B, C) = P(A|C)$

if A & B are conditionally independent

$B \cap C$ -- NN

(2) $P(A|B, C, \ldots N) = P(A|N)$

if A, B, C -- N-1 are conditionaly independent

Naive assumption:- all features are conditionally independent

* point to be clasified : $X = \langle x_1, x_2 \dots x_n \rangle$ → features
* No of class → (K)
* C_k — class K

! condit⁻ probability

$P(A)\,P(B|A) = P(A,B)$

$$P(C_k|x) = \frac{P(C_k)\,p(x|C_k)}{P(x)}$$

$P(x)$ → doesnt affect much as it is same for $K=1-n$ → scaling term

$$P(C_k|x) \propto p(x, C_k)$$

chain rule → $p(a,b) = p(b) \cdot p(a|b)$

$$p(C_k, x_1, x_2 \dots x_n) = p(x_1, x_2 \dots x_n, C_k)$$

$$= p(x_1 | x_2, x_3 \dots x_n, C_k) \cdot p(x_2, x_3 C_k)$$

$$= p(x_1 | x_2 \dots C_k) \cdot p(x_2 | x_3, x_4 \dots x_n, C_k) \cdot p(x_3, x_4 \dots x_n, C_k)$$

$$= p(x_1 | x_2, \dots x_n, C_k) \dots p(x_{n-1} | x_n, C_k)\, p(C_k) \rightarrow (1)$$

asumption $x_1, x_2 \dots x_n$ are conditionaly independent

$$p(x_1 | x_2 \dots x_n, C_k) = p(x_1 | C_k)$$

(1) $\Rightarrow p(x_1|C_k)\, p(x_2|C_k) \dots p(x_{n-1}|C_k) \cdot p(C_k)$

$$\therefore \boxed{p(C_k | x_1, x_2 \dots x_n) \propto \left(\prod_{i=1}^{n} p(x_i | C_k)\right) \cdot P(C_k)}$$

→ product

Naive-Bayes-classifier

$$\hat{y} = \underset{k \in \{1 \dots K\}}{\operatorname{argmax}}\; P(C_k) \prod_{i=1}^{n} p(x_i | C_k)$$

→ No of features

→ max value

Naive-Bayes with text

x_i: sentences $\xrightarrow[\text{tokenize}\ \text{stemming}\ \text{stop-word removal}]{\text{process}}$ Bag of words $\longrightarrow w_1, w_2 \dots w_n$

words	class
– – – –	0
– – –	0
– – – –	0
– – – –	!
– – – –	!

$y_i \rightarrow$ class Ex: 0-1

steps

→ to find $P(yes \mid \langle w_1, w_2 \dots w_n \rangle = P(yes) \cdot P(w_1 \mid yes) \cdot P(w_2 \mid yes) \dots$

→ $P(w_i \mid yes) = \dfrac{\#\text{ points which have class = yes and have } (w_i) \text{ word}}{\#\text{ points which have class = yes}}$

Laplace Smoothing

test_data = $\langle w_1, w_2, w_3, w' \rangle$ $w' \rightarrow$ word which was never seen in training data

→ $P(yes \mid td) = P(yes) \cdot P(w_1) \cdots (P(w'))$ → you cant calculate as

solution: ways

- ignore ie $p=1$ → if you ignore implies $(p=1)$ false
- $p=0$ whole $P(yes \mid t) = 0$.

smoothing

$P(w' \mid yes) = \dfrac{\#\text{ with } w'}{\#\text{ pts } y = yes}$

$= \dfrac{0}{n_{yes}}$ → cant be 0

$\alpha \rightarrow$ randomly choosen

as $\alpha \rightarrow \infty$

$P(w' \mid yes) \rightarrow 1/2$

∴ if $(\alpha\uparrow) \rightarrow$ uniform probability

$$P(w' \mid yes) = \frac{0 + \alpha}{n_{yes} + \alpha K}$$

(K) no of distinct values = (2)

either word can be there or not

for run-test data → no of values it can take Ex: outlook — K= (3) rainy windy sunny

$$\therefore P(w_i \mid y=1) = \frac{\#\text{pts with } w_i \text{ \& } y=1 \; + \alpha}{\#\text{pts with } y \; y==1 \; + \alpha K}$$

→ No of distinct values → ② for text

Log-probabilities → provide numerical stability

log ab = log a + log b

log a^b = b log a

pblm :- $P(yes \mid \langle x \rangle) = P(yes) \cdot P(w_1 \mid yes) \cdot P(w_2 \mid yes)$

↓ small nos Ex :- (0.0001) (0.002)

→ product can quickly become too small

→ Ex :- 0.0 ---0009

after -16 digits after ⊙ python starts rounding off

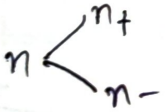
soln :- $\log(P) = \log(P(yes) \cdot P(w_i \mid yes) \cdots)$

$$= \log(P(yes)) + \sum_{i=1}^{n} \log(P(w_i \mid yes))$$

② impacts more fr minority class

* Feature Importance :- w_i having highest $P(w_i \mid y = class)$ is more imp

* Imbalanced dataset



$n_+ >> n_-$ ⟹ $P(class = +) >> P(class = -ve)$ will cause problem

majority class is favored by laplace smoothing.

soln :- ① Ignore P(yes), P(No) term ② choose diff value of α for each class Ex: α_+, α_-

③ upsampling, downsampling

Numerical features

Gaussian NB

$x_{i1} = (f_1 \text{ col of } x_i) \in R^d$

~~$P(x_i \mid yes)$~~ let D^+ : data with yes

D^- : data with No

$$P(yes \mid x_i) = P(yes) \prod_{i=1}^{n} P(x_i \mid yes)$$

→ ?

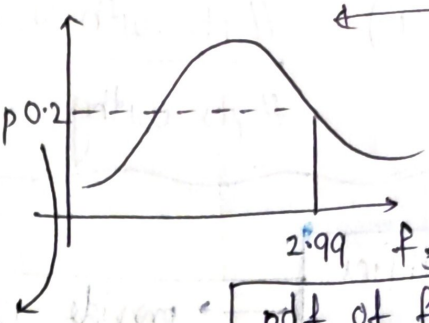
solution :-

x_i

$Ex: p(x_{i3} | yes)$

→ plot pdf of f_3 for +ve/-ve →

→ $x_{i3} = 2.99$ $p(x_{i3} | yes) = 0.2$



assume gaussian/normal (22)

pdf of f_3 on D^{+ve}

steps

① plot pdf of f_j on $D^{class=val}$

② assume pdf is gaussian / Normal distr → $N(\mu, \sigma)$ and find p

③ $P(x_{ij} | yes)$ = pdf (x_i) from pdf of f_j

---

* NB cant work directly with distance matrix

$w^T x_i = w^T X$

$= w \cdot X$ → dot product

$\sigma(w^T x) = p(y=1|x)$

predict point using Log R

x_q → find $\hat{y}$ by $w^T \cdot x_q$ → $\sigma(\hat{y})$ → yes

Probability interpret of LR

cost f^n : for minimization

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} -y_i \log(p_i) - (1-y_i) \log(1-p_i)$$

$p_i = \sigma(w^T x_i)$

activated if class = 1 ie $y_i = 1$

activated for class = 0

(x_i, y_i)

$P(y_i = 1 | x_i) = \hat{y}_i = \sigma(z)$

$= \frac{1}{1+e^{-z}}$

$z = x_i^T w = w^T w$

Logistic Regression

used for classification.

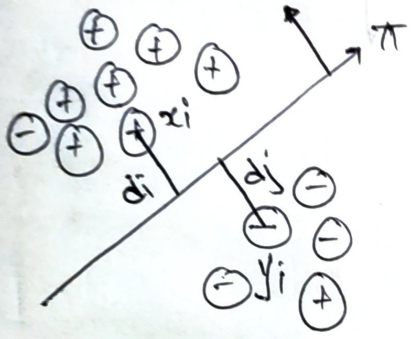
Assumption :- pts are linearly or almost linearly seperable

Task :- to find plane $\pi(w,b)$ which best seperates the points

where eqn of plane $w^T \cdot x + b = 0$ (intercept; normal to plane π)

Overall Idea

* let $\langle x_i, y_i \rangle$ be training dataset and let $y_i \to +1 \to$ positive, $\to -1 \to$ negative



→ assume π pass through origin

$\therefore w^T x = 0$ as π is unit vector $\|w\| = 1$

→ $d_i = \frac{w_T x}{\|w\|}$ $\therefore$ $d_i = w_T x_i$, $d_j = w_T x_j$

→ $d_i > 0$; if x_i lies same side of π

$d_j < 0$; if x_i lies on other side

classifier :- if $w_T x_i > 0 \to (+)$ point, $w_T x_j < 0 \to (-)$ point

$x_i, x_j \to$ test point

take term, predict :- $y_i \cdot w_T x_i \equiv y_i \cdot \hat{y}_i \to$ prediction → actual class

case 1 :- $y_i = (+1)$, $w_T x_i$ +ve

$(w_T x_i)(y_i) > 0$ } correct classify

case 2 :- $y_i = (-1)$, $w_T x_i < 0$

$(w_T x_i)(y_i) > 0$

case 3 :- $y_i = (+1) \cdot w_T x_i < 0$

$(y_i \cdot w_T x_i) < 0$

case 4 :- $y_i = (-1)$ $w_T x_i > 0$

$(y_i \cdot w_T x_i) < 0$

$\therefore$ general idea is to have more correct classifications, ie $(w_T x_i)(y_i)$ must be as +ve as posible

General aim → $\max_{w} \sum_{i=1}^{n} (y_i w_T x_i)$

x_i, y_i → constant

w^T : variable

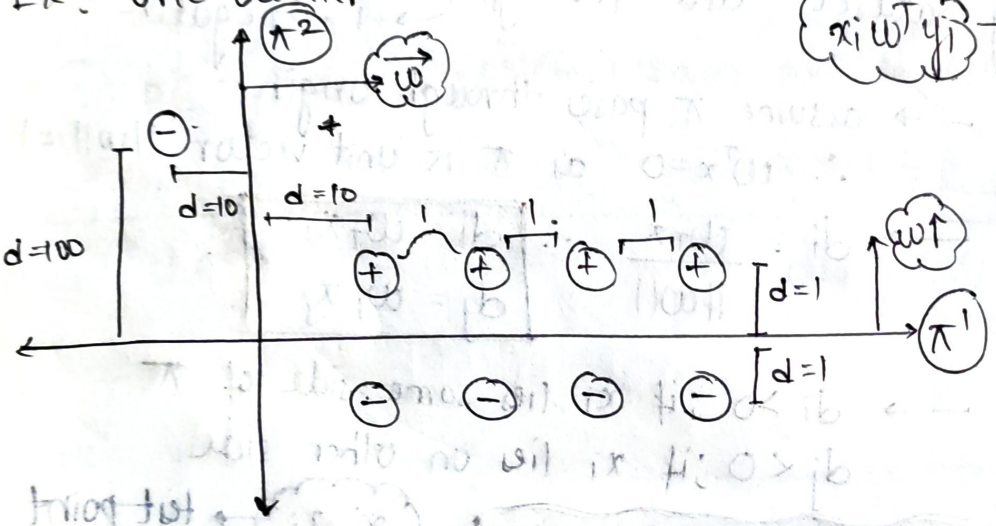
$$w^* = \arg\max_{w} \left(\sum_{i=1}^{n} y_i w^T x_i \right)$$

optimization problem → maxm

optimal w

Not stable as one outlier can distort optimization

Ex:- one outlier



$x_i w^T y_i$ → called as signed distance

Case 1 π_1

$\sum w_T y_i x_i = 1+1+1+1+1+1+1+1+1+1$

(+ve)(−1)(−1) ← $-100 = -92$

$\sum w_T y_i x_i = -92$

Case 2 π_2

$\sum x_i y_i w^T = 1+1+1+1-1-1-1-1$

$+ (-100)(-1)$ (+ve)

$= 100 + \boxed{10+11+12+13}$

$\boxed{-10-11-12-13}$ (−)

$\boxed{\sum x_i y_i w_T = 100}$

Acc to optimization, π_1 is better than π_2 as $(100 >> -92)$

but logically π_1 is better

$\therefore$ We need to modify $\boxed{\max \sum w_i^T x_i y_i}$

accuracy$(\pi_1) = \frac{8}{9}$

accuracy$(\pi_2) = \frac{5}{9}$

$acc(\pi_1) >> acc(\pi_2)$

Sigmoid Squashing

(squash) $0, \infty \longrightarrow [0, 1]$

* we found that if $d_i = w^T x_i$ was very large, it impacted model

Solution:

① use $f(w^T \cdot x_i y_i)$

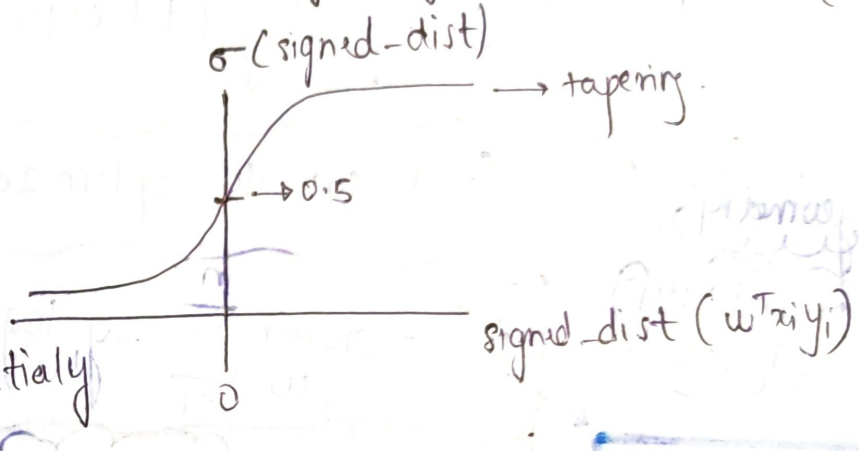
↓

sigmoid fn ($\sigma(x)$)

why ① tapering

- doesn't ↑ exponentially for d↑

② probabilistic



if $d = 0 \equiv w^T \cdot x_i = 0$ which states that pt lies on (π)

then $\sigma(w^T x_i y_i) = \sigma(0) = 0.5$ → if it lies on (π) it can belong to $y = 1$ / 0

$[0, 1]$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

New optimization,

$$w^* = \underset{w}{\text{argmax}} \sum_{i=1}^{n} \left[\sigma(y_i \cdot w^T \cdot x_i)\right]$$

* $f(x)$ is monotonous fn

① if $f(x) \uparrow$ as $x \uparrow$

② $a < b$; $f(a) < f(b)$

→ if $g(x)$ is monotonous function

then $\underset{x}{\text{argmax}}[g(f(x))] = \underset{x}{\text{argmax}}[f(x)]$

$$w^* = \underset{w}{\text{argmax}} \sum_{i=1}^{n} \frac{1}{1+e^{y_i w^T \cdot x_i}}$$

$$w^* = \underset{w}{\text{argmax}} \sum \frac{1}{1+\exp(-y_i w^T \cdot x_i)}$$

lets take $g(x) = \log x$ → monotonous function

$$w^* = \underset{w}{\text{argmax}} \sum \log\left(\frac{1}{1+\exp(-y_i w^T x_i)}\right) \longrightarrow ①$$

σ → not convex fn

log(σ) is convex function

***

wkt $\log\left(\frac{1}{x}\right) = \log(x^{-1}) = -\log x$

$$\text{argmax}[f(x)] = \text{argmin}[-f(x)]$$

① becomes, $w^* = \arg\max_w \sum -\log(1+\exp(-y_i w^T x_i))$

$$\rightarrow w^* = \arg\min_w \sum_{i=1}^{n} \log(1+\exp(-y_i \cdot w^T \cdot x_i)) \quad \left| \; y_i = \begin{matrix} +1 \\ -1 \end{matrix} \right. \; ***$$

geometry ↑ final eqn to optimize which is not prone to outliers

probability based

$$w^* = \arg\min_w \sum_{i=1}^{n} -y_i \log p_i - (1-y_i)\log(1-p_i) \quad \left| \; y_i = \begin{matrix} 1 \\ 0 \end{matrix} \right.$$

weight vector

$p_i = \sigma(w^T x_i)$ | $w^T w = 1$ | $\vec{w}$ is unit vector

$* w = \langle w_1, w_2, w_3 \cdots w_d \rangle$ (d = no of feature)

→ let $z_i = +y_i w^T x_i$

$$w^* = \arg\min_w \sum \log(1+\exp(-z)) \geq 0$$

as $\exp(-z) \rightarrow \infty$

$\log(1+e^{-z}) \rightarrow 0$

always ≥ 0 → because $\log(1) = 0$; $\log(1+x) \geq 0$

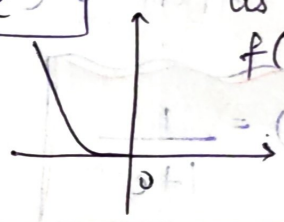
minimum value it can take is 0

↑ it occurs when $z_i \rightarrow \infty \; \forall i$

$\log(1) = 0$

e^{-x}

as $x \rightarrow \infty$, $f(x) \rightarrow 0$



$z = x_i w^T y_i$ (w^T varies; constant from Data)

→ we need to pick w s.t.

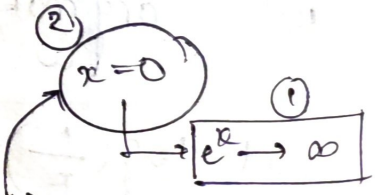
① all pts are correctly classified i.e. $w > 0$

② $z_i \rightarrow \infty$

→ if $z_i \rightarrow \infty$ then $w \rightarrow \infty$ (+ve class)

$w \rightarrow -\infty$ (-ve class)

as $w \rightarrow +\infty / -\infty$ we can say that $w \uparrow$ and cause **overfit**



Imp) ideally we need loss = 0; for loss = 0 we need $\log(1+x) = \log(1) = 0$

$\therefore$ if $e^{-x} \rightarrow \infty$, $\log(x+$

L2 regularization add to optimize pblm to prevent $w \uparrow \infty$

$\rightarrow w^* = \text{argmin} \sum \log(1+\exp(-z)) + \{\lambda w^T w\} \rightarrow w^T w = \sum_{i=1}^{d} w_j^2$

minimize both

$$w^* = \underset{w}{\text{argmin}} \left\{ \sum \log(1+e^{-z}) + \lambda \|w\|_2^2 \right\}$$

① ②

w^T w $\rightarrow \sum_{i=1}^{d} w_i^2$

tug of war b/w ① and ②

* to min ① $w \rightarrow \infty$, but ② becomes very very large (∞^2)

① → loss term
② → regularizn term

hyperparam = λ

if $\lambda = 0$; minimize only ① causes **overfit**

if $\lambda \uparrow$; min depends on ② which doesn't depend on y_i **underfit**

$w^* = \text{argmin}$ (logistic loss + regun)

$\|w\|_2^2 = \sum w_j^2$

$\|w\|_1 = \sum |w_i|$

L1 regularization

$$w^* = \underset{w}{\text{argmin}} \left[\sum \log(1+\exp(-z)) + \lambda \|w\|_1 \right]$$

Sparsity:- w/ soln to LR is sparse if many w_i's in $(w = \langle w_i \rangle)$ are 0's | if $w_i = 0$, then that f_i is less important

∴ L2 → w_i for unimp f's is small but not 0
L1 → $w_i = 0$; for f_i is unimportant

Elastic-Net

$$w^* = \underset{w}{\text{argmin}} \left[\sum \log(1+\exp(-z)) + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2 \right]$$

→ as $x \rightarrow \infty$, $e^{-x} \rightarrow 0$ ∴ $\log(1+0) = \log(1) = 0$ ← ideal loss needed

loss - minimization derivation

$f(x) = w^T x_i$

$w^* = \underset{w}{\operatorname{argmin}}$ (no. of incorrectly classified points)

0-1 loss fn

+1 : incorrect class

0 : correct class

~~$y \cdot f(x) < 0$~~ $y \cdot f(x) > 0$

$y \cdot f(x) < 0$

0-1 loss

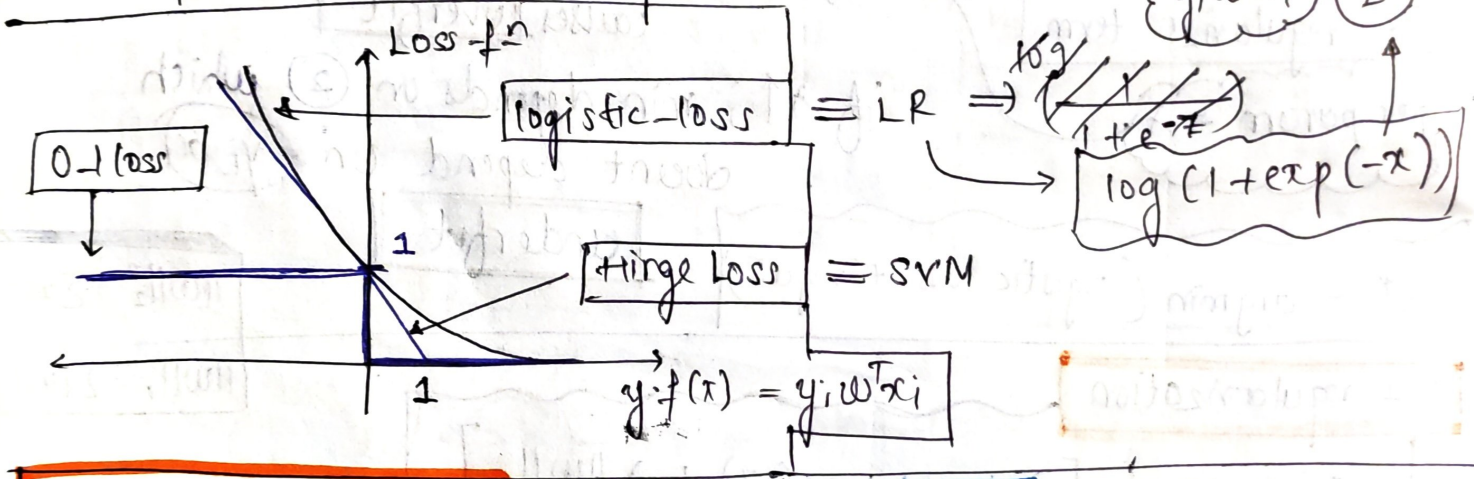
+1 : incorrect $\rightarrow y \cdot f(x) < 0$

0 : correct $\rightarrow y f(x) > 0$

← minimize this

* 0-1 loss is not continous → cant differentiate → cant optimize

→ optimize 0-1 loss ⟹ approximate 0-1 loss

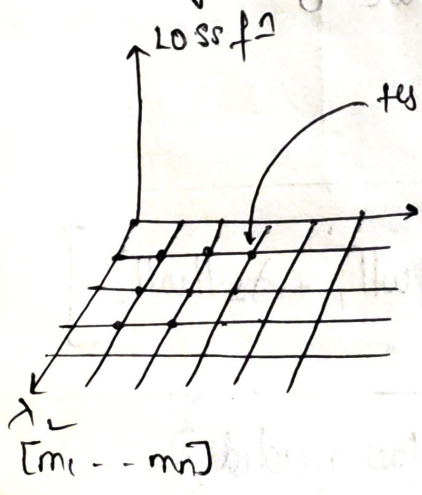


Hyper-param tuning

GridSearch

Ex:- Elasticnet

hyper params - λ_1, λ_2



Random Search

* test values randomly picking

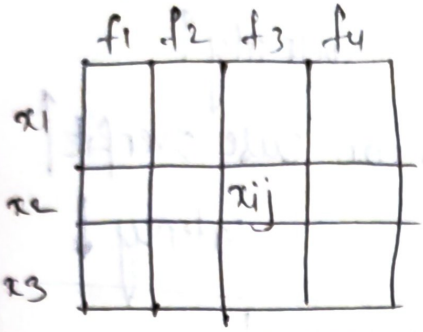
λ_1 - [m, -- mn]

λ_2 - [m, -- mn]

problem

As no. of hyper params increase no. of pts increase exponentially

* Logistic regressions needs **standardization**



$$x_{ij} = \frac{(x_{ij} - \mu_j)}{\sigma_j}$$

→ centering

→ scaling [0-1]

Feature Importance : more $|w_j|$ is better → affects $(w_j x_i)$

↳ only when features are not-collinear

* colinear → $f_1 = \alpha f_2 + \beta$ [f_1, f_2 r colinear]
* multicolinear → $f_1 = \alpha_1 + \alpha_2 f_2 + \alpha_3 f_3$ [$f_1 f_2 f_3$ are colinear]

Example

$D = \langle x_i, y_i \rangle_1^n$ ← Query point

$w^* = \langle 1, 2, 3 \rangle$; $x_q = \langle x_{q1}, x_{q2}, x_{q3} \rangle$

→ $w^T x_q = x_{q1} + 2x_{q2} + 3x_{q3}$

let asume ; f_1 and f_2 r colinear

$f_2 = 1.5 f_1$

→ $w^T x_q = x_{q1} + 3q_1 + 3q_x$

$= 4x_{q1} + 3x_{q3} = \langle 4, 0, 3 \rangle$ → gives same classifier as $w^* = \langle 1, 2, 3 \rangle$ as colinearity exists

check colinearity - **pertubate**

① calculate $w^* = \langle \text{---} \rangle$

② add small noise $N(0, 0.1)$

$x_{ij} + \epsilon$

③ calculate new $\tilde{w} = \langle \text{---} \rangle$

if ($\tilde{w}$ and w^*) differ drastically then colinear

Time Complexity

* Train LR : optimize w^* using SGD → $O(nd)$

* run-time complexity

① space - $O(d)$ [u need to store only $w^* = \langle w_1, w_2 \dots w_d \rangle$

② time - $O(d)$ ← d multiplications + d addition ← $w^T \cdot x_q$ ← you need to find only $>0 \rightarrow +$, $<0 \rightarrow -$

* low latency
* memory efficient

(30) what if d is large?

* if d = 1000,

run-time tc = 1000 multiplication + 1000 addition → sol?

(1) L1 regularization

(2) as λ↑ sparsity↑.
↳ this can cause overfit↑
latency↓

Non-linearly seperated data with LR

use feature Engineering

predict

→ let 2 features $<x_{i1}, x_{i2}>$

$x_q <f_1, f_2>$

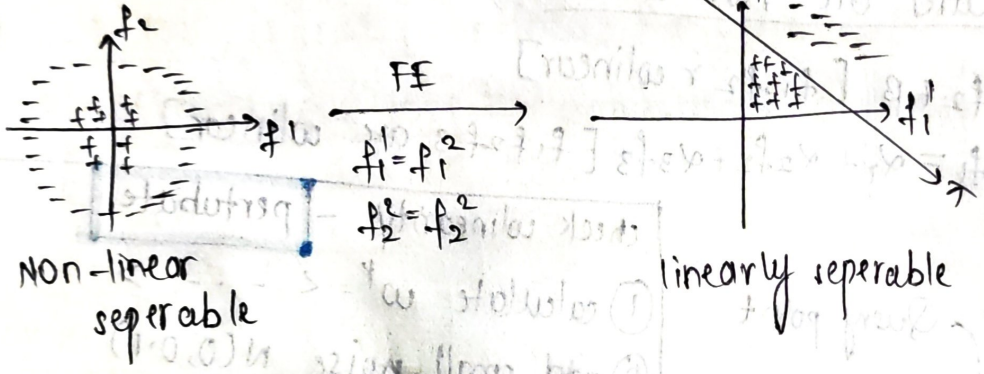
↓

$x_q' <f_1^2, f_2^2>$

↓

classify

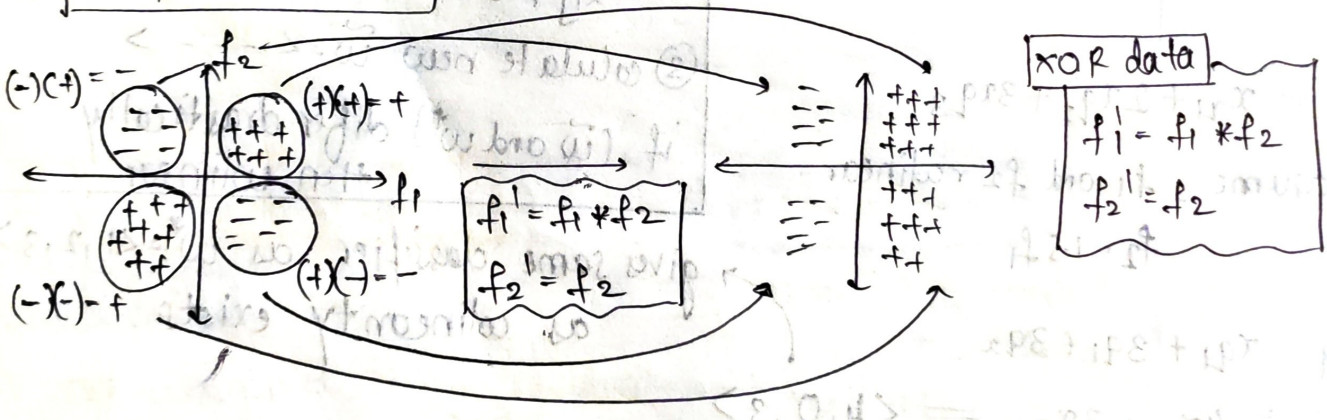
(1)



Non-linear seperable

linearly seperable

(2) LR for XOR data

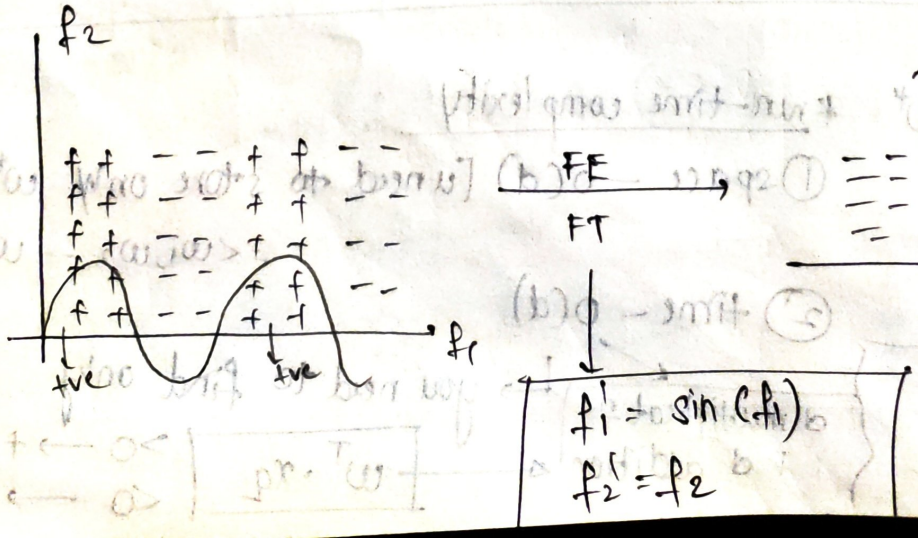


XOR data

$f_1' = f_1 * f_2$

$f_2' = f_2$

(3)



Typical transforms NLS — LS

① $f_1 \cdot f_2, f_1^2, f_2^2, f_2^3 \ldots$ → polynomial feature

② $\sin(f_1), \cos(f_1)$ → trignometric / (trig + poly)
$\sin(f_1) * \cos(f_2)$

③ OR, AND, XOR → Boolean

④ other → $\log(f_1), e(f_1), \sigma(f_1)$

Lonear Regression

apply → L2, L1 regularation
→ elastic net

Aim:- find best-fit line/plane (≥3d)

loss :- $y_i - \hat{y}_i$ → predicted value (actual value)

loss fn:- $\sum_{i=1}^{n} (y_i - \hat{y})^2$

→ find best fit plane $(w^T x + w_0 = 0)$ which best fits (least loss) data

if 2d data → $w = \begin{bmatrix} x \\ y \end{bmatrix}$ $w^T x$ → $w_1 x + w_0 = 0$
$ax + b = 0$

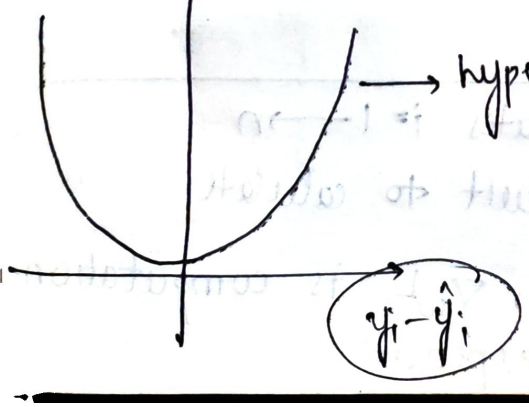
$$w^*, w_0^* = \underset{w, w_0}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \underset{w, w_0}{\operatorname{argmin}} \sum (y_i - (w^T \cdot x_i + x_0))^2$$

w^*, w_0^* → intercept; w → vector; w_0 → scalar; $\hat{y}_i$ → $f(x) = w^T x_i + x_0$

loss minimizat⁼ way

Loss fn → $(y_i - \hat{y})^2$

→ hyperbola ($y = x^2$)



feature Importance

→ same as logistic regresion
→ same regularization

Outlier → cant handle directly

① DTrain
③ remove pts very far away
④ $D_{Train} - =$ outliers
↓
loop ⓝ times

Gradient Descent

→ finding maxima/minima is tough → $\frac{df}{dx}=0$, $\nabla_x f=0$
so we use gd (iterative algorithm)

consider

$$L(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

assumptions:
① NO regularization
② No intercept: $w^T x_i + w_0 = 0$

$\therefore \nabla_w L = \sum_{i=1}^{n} 2(y_i - w^T x_i)(-x_i)$ | $\langle x_i, y_i \rangle$ are constants

$$\nabla_w L = \sum_{i=1}^{n} -2(x_i)(y_i - w^T x_i) = 0$$ → find sol^n

$D_m = \frac{2}{N}\sum (y_i - \hat{y}_i) x_i$

$D_c = -\frac{2}{N}\sum (y_i - \hat{y})$

gradient descent

① pick random vector $w_0 = \langle w_1, w_2 \dots w_n \rangle$

② $w_1 = w_0 - r \cdot \nabla_w L$

update fn

$$w_j = w_{j-1} - r \cdot \sum_{i=1}^{n} -2x_i (y_i - w_{j-1}^T \cdot x_i)$$

$$w_1 = w_0 - r \cdot \sum_{i=1}^{n} -2x_i (y_i - w_0^T \cdot x_i)$$

$$w_2 = w_1 - r \sum -2x_i (y_i - w_1^T x_1)$$

③ if $w_k - w_{k-1}$ is a very small vector

then optimal $w^* = w_{k-1}$

problem → as it calculates $i=1 \to n$ very difficult to calculate

$$\nabla_w L = \sum_{i=1}^{n} -2x_i (y_i - w_0^T x_i)$$

$\therefore w_1 = w_0 - r \nabla_w L$ is computationally expensive

GD for linear regression in 2d plane

$$\text{loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$

$\hat{y}_i = mx_i + c$ → coeff, slope

$\{w^T x_i + w_0\}$ → (m) 2d

$$L = \frac{1}{n}\sum (y_i - (mx_i + c))^2$$

$$\frac{dL}{dm} = \frac{1}{n}\sum 2(y_i - (mx_i + c))(-x_i)$$

$$\frac{dL}{dc} = \frac{1}{n}\sum 2(y_i - (mx_i + c))(-1)$$

$$D_m = \frac{-2}{N}\sum (y_i - \hat{y}_i)^2 \quad (x_i)$$

$$D_c = \frac{-2}{N}\sum (y_i - \hat{y})$$

∴ step (2) in GD

$$m_{i+1} = m_i - L \cdot Dm$$

$$C_{i+1} = C_i - L \cdot Dc$$

L → learning rate

$$\longrightarrow \boxed{\begin{aligned} m_{i+1} &= m_i - L \cdot Dm \\ C_{i+1} &= C_i - L \cdot Dc \end{aligned}}$$

Ridge Regression (L2): $L = \sum (y - \hat{y})^2 + \|w\|_2$

Lasso regression (L1): $L = \sum (y - \hat{y})^2 + \|w\|$

Stochastic Gradient Descend

stochastic = probabilistic

idea: Instead of all (n) points, pick any (K) no. of random points in each iteration.

$$w_j = w_{j-1} - r \cdot \sum_{i=1}^{K} -2x_i \,(y_i - w_{j-1}^T x_i)$$

$w^*_{SGD} = w^*_{GD}$

(K) → called batch-size

$1 << K << n$

Cases

→ if n = (1 million) takes 100 iterations to converge to w^*

then, K = 100 may take 500 it?

K = 10 may take 1000 it?

if (K == 1) SGD

if $1 << K << n$ Batch stochastic gradient

BGD

(34) **PCA constrained optimization**

$\|u\| = 1$

PCA optimization pblm : $\max_{u} \frac{1}{n}\sum_{i=1}^{n} (u^T \cdot x_i)^2$ such that $u^T \cdot u = 1$

$\frac{1}{n}\sum u^T \cdot x_i \Rightarrow \sum u^T S u$ s.t $u^T u = 1$ and $S = \frac{1}{n}\sum x_i x^T$ covariance matrix of $X = XX^T$

$\therefore \max \sum_{i=1}^{n} u^T S u$ wrt $u^T \cdot u = 1$

$L(u, \lambda) = u^T S u - \lambda(u^T u - 1)$ → Lagranges multiplier

$\frac{\partial L}{\partial u} = 0 \Rightarrow \frac{\partial}{\partial u}(u^T S u - \lambda(u^T u - 1)) = \frac{\partial}{\partial u}(u^T S u - \lambda u^T u - \lambda)$

$= Su - \lambda u = 0$ $\quad Su = \lambda u$

↓

Eigen Value/Vector

---

logistic regresion with regularization

$w^* = \underset{w}{\text{argmin}} \; (\text{logistic\_loss}) + \lambda(w^T w)$ → regularization

similar

→ $w^* = \underset{w}{\text{min}} \; (\text{loss})$ such that $\vec{w} \perp \pi$, $\boxed{\vec{w} \cdot w = 1}$

$w^* = \underset{w}{\text{argmin}} \; (\text{log\_r\_loss}) - \lambda(1 - w^T w)$ → lagrangian mult

$\boxed{w^* = \min \; (LR\_loss) - \lambda + \lambda w^T w}$

∴ Regularization ≡ equality constraint on optimization pblm

NLP Basics

corpus → set of all documents

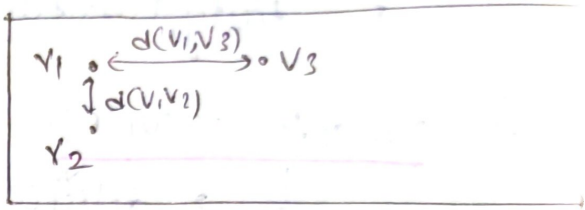
Text to vector
- BOW
- TF-IDF
- Word2Vec

* text → d-dim vector → fit a plane

consider reviews r_1, r_2, r_3 whose d-dim vector forms are v_1, v_2, v_3

→ similarity (r_1, r_2) > sim (r_1, r_3)

then, dist (r_1, r_2) < dist (r_1, r_3)



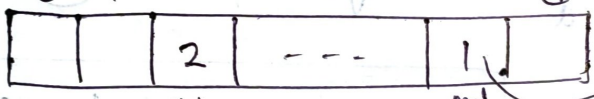
(ie) more similar ≡ more closer in plane

Bag of words (1)

step (1) :- construct a dictionary of size - (d) where 'd' is number of unique words in corpus.

step (2) :- for each document - Ex:- r_1 = this is good this

v_i =



# of ocurences of word

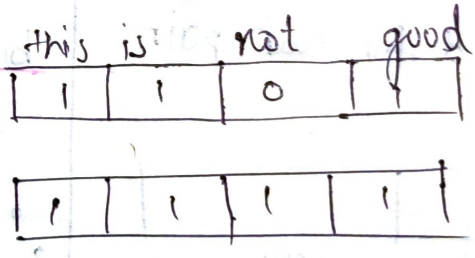
dict[d] = dict[i]

i = 0 → d

sparse vector

*Calculate distance

$v_1 - v_2 = \|v_1 - v_2\|$



Ex:- This is good

This is not good

not good

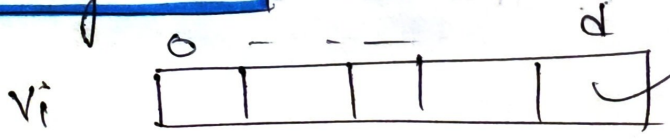
$\|v_1 - v_2\|_n = \sqrt{(1-1)^2 + (1-1)^2 + (0-1)^2 + (1-1)^2}$

$= \sqrt{1} = 1$

score is near

but (1) word **not** makes difference in real world

Binary BOW



0: if word !present

1: present 1 or more time

(36) words → vector flow

(1) Removing stop words – (this, is, as ..)

(2) convert to lowercase

(3) stemming < porter stem / snowball stem

Ex: tasty, tasteful, taste : taste (stem word)

(4) semantic meanings – word2vec

→ delicious ≡ tasty

(5) Lemmetization : breaking up sentences into parts

[This] [is] [New York]

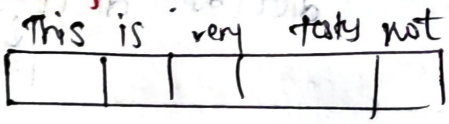
uni / bi / n grams BOW

r_1 : This is very tasty

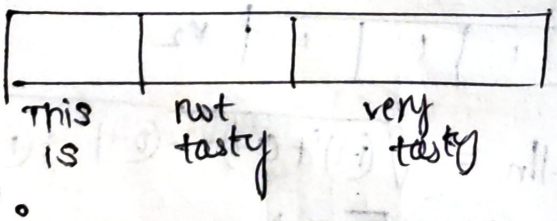
r_2 : This is not tasty

by default stop words : not, very

unigram – 1-d for each word



bi-gram → 1 d for each pair



n-grams

unigram + BOW : destroys sequential info

bigram + BOW : retains some info (very tasty)

∴ uni < bi < tri

as uni → bi → tri → n

d → increases

∴ increases dimensionality

② TF - IDF — Term freq - Inverse doc freq

→ used in information retrieval

let
r_1 — $w_1\ w_2\ w_3 \cdots w_m$
r_2 — $w_1\ w_2 \cdots$
r_3 —
⋮
r_n —

r_i = records

D_c = corpus = all records $\{r_1 \rightarrow r_n\}$

IDF

	TF	IDF
frequent word	High	Low
rare word	Low	High

$TF = TF(w_i, r_j)$ = $\frac{\#\ w_i \text{ occurs in } r_j}{\text{total \# words in } r_j}$ → It can be interpreted as prob of word (w_j) present in (r_j)

$TF \rightarrow \boxed{0 \Leftrightarrow 1}$ as $dr \geq hr$

$IDF = IDF(w_i, D_c) = \log\left(\frac{N}{n_i}\right)$ ← #docs / #docs which have w_i

$\boxed{IDF = \log\left(\frac{N}{n_i}\right)}$ $N \geq n_i$ → IDF ≥ 0

∴ n_i → no of records in which w_i occurs

where
N → no of docs/records
n_i → no of docs which contain word (w_i)

as $n_i \uparrow$ IDF ↓
$n_i \downarrow$ IDF ↑

ie rare words: high IDF
freq words: low IDF

TF - IDF format

$r_1 \rightarrow$ | w_1 | w_2 | w_2 | ... | w_m |

$TF(w_2, r_1) * IDF(w_2, D_c)$

∴ $r_i \rightarrow$ each w_j cell = $TF(w_j, r_i) * IDF(w_j, D_c)$

drawback :- doesn't focus on semantic meaning.

prioritizes frequent words in record

prioritize rare words in Corpus

(38) **Word 2 Vec** deep learning

large corpus text → [w2v] → each word is converted to vector of d-dimension, which is dense

d-dim

cases

① **similarity**

if w_1, w_2 r similar
v_1, v_2 r similar
↓
nearby

② **relationship**

$(V_{queen} - V_{king}) \,||\, (V_{man} - V_{woman})$

$V_q \rightarrow V_k$
$V_m \rightarrow V_{woman}$

Avg w2v

w2v: word → vector

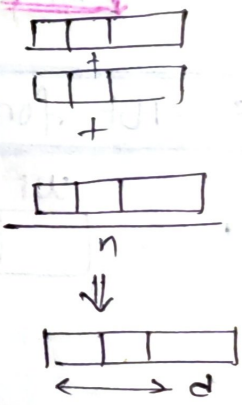
aim: sentence(r_i) → value ?

How to convert sentence to vector using w2v?

word → vec(d-dim)

→ r_i: $w_1\ w_2\ w_3 \cdots w_{n_1}$

$$r_i = \frac{1}{n_1}\ \underbrace{w2v(w_1) + w2v(w_2) + \cdots + w2v(w_{n_1})}_{\text{d-dim}}$$



→ tf-idf(r_i) = | w_1 | w_2 | w_3 | w_4 | w_5 | ... |
| t_1 | t_2 | t_3 | | | ... |

tf-weighted-w2v

→ tf-idf value of w_i in r_i

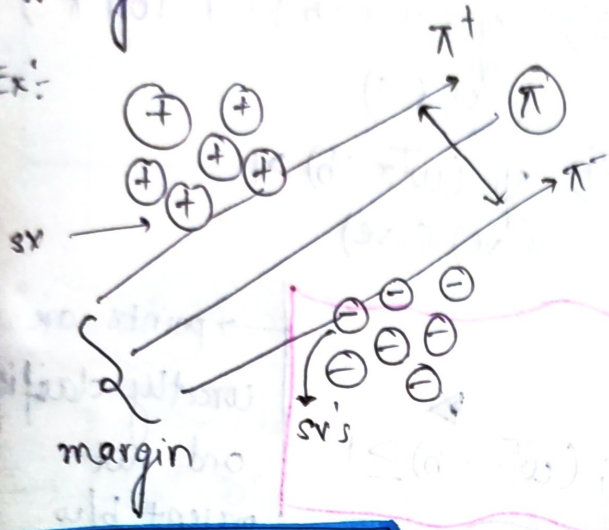
$$tf\text{-}idf\ w2v(r_i) = \frac{\sum_{i:words} (t_i * w2v(w_i))}{\sum_{i:words} t_i}$$

if ($t_i = 1$)
tf-idf-w2v = avg w2v

SVM

aim → construct optimal hyperplane which maximizes margin

Ex:

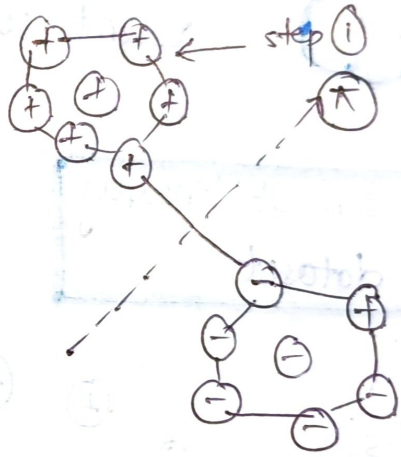


aim :- find π such that $d(\pi^+, \pi^-)$ is maximized

* points on π^+ and π^- are called support vectors (sv)
* $\pi^+ \parallel \pi \parallel \pi^-$
* while making predictions only π is considered

Alternate Intuition

1. Construct convex hull around classes of points
2. Pick 2 points one from each class when joined, connect $\oplus_c$ and $\ominus_c$ and shortest distance
3. Bisect the line to give π



Mathematical formulation

π : margin maximizing hyperplane

let π : $w^T x + b = 0$

$\pi^+ : w^T x + b = 1$

$\pi^- : w^T x + b = -1$

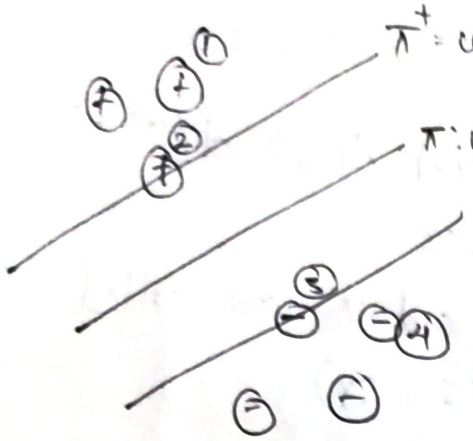
(normal w is ⊥ to all π, π^+ & π^-)

→ geometrically we can get

$$d(\pi^+, \pi^-) = \frac{2}{\|w\|}$$

$$\pi^+ - \pi^- = w^T x_1 + b = 1$$
$$- w^T x_2 + b = -1$$
$$w^T (x_1 - x_2) = 2$$

$$x_1 - x_2 = \frac{2}{w^T}$$



* pt ① : $y_i(w^Tx+b) > 1$

② : $y_i(w^Tx+b) = 1$ (on π^+)

③ : $y_i(w^Tx+b) = 1$ (on π^-)
(-ve) (-1)

④ $y_i(w^Tx+b) > 1$
(-ve) (-ve)

maximize width

$$\therefore \quad w^*, b^* = \underset{w,b}{\operatorname{argmax}} \frac{2}{\|w\|}$$
such that $\forall x_i,\ y_i(w^Tx+b) \geq 1$

→ points are correctly classified and not present b/w π^+ and π^-

Hard-margin → doesn't work for almost linearly separable data.

optm for almost linearly seperable dataset

$y(w^Tx+b) = 1.5$
$1.5 - (0)$
→ ξ

ξ → zeta

$\pi^+ (w^Tx+b=+1)$

SOFT MARGIN SVM

$y(w^Tx+b) = -1.5$
$1 - (2.5)$
→ ξ

π

$\pi^- (w^Tx+b=-1)$

$y(w^Tx+b) = 0.5$
$1 - (0.5)$
→ ξ

+1 -1.5 (-> +)

$y_i(w^Tx+b) = -1.5$
$= 1 - (2.5)$
→ ξ

$y(w^Tx+b) = -0.5$
$= 1 - (1.5)$
→ ξ

+ ← <-1

from using variable (ξ) to convert values in the form 1 - (ξ) we get

$\forall x_i$, if $\xi = 0$; $y(w^T x + b) \geq 1$ (Correctly classified)

if $\xi > 0$; point is some units distance away from the correct hyperplane in incorrect direction

$\therefore w^* b^* = \arg\max \frac{2}{\|w\|} \longrightarrow \arg\min \frac{\|w\|}{2}$

hyper param

$$(w^*, b^*) = \arg\min_{w,b} \left\{ \frac{\|w\|}{2} + C\left(\frac{1}{n}\sum_{i=1}^{n} \xi_i\right) \right\}$$

loss fn

regularizer

such that : $y_i(w^T x_i + b) \geq 1 - \xi_i$ & $\xi \geq 0$

less $\Sigma(\xi_i)$ lesser misclassified

Note : in sukit learn

$C = \frac{1}{\lambda}$ → hp of logistic regression

$C \uparrow$ = overfit

$C \downarrow$ = underfit

why $\pi^+ : +1$ and $\pi^- : -1$

geometric way

$\pi^+ : w^T x + b = +K$

$\pi^- : w^T x + b = -K$

why same ± (K)

→ π^+, π^- are equally far from π but in opposite direction

and for ease $K = 1$

algebraic intr

$w^T x + b = K$

$\left(\frac{w^T}{K}\right) x + \left(\frac{b}{K}\right) = 1$

$w' x + b' = 1$

we don't say that $\|w\| = 1$

if $i \geq 0$ $\sum_{i=1}^{n} = (n)$?

(42) **dual form**

Linear svm $x_i^T x_j$
kernel svm : $K(x_i, x_j)$

primal form

$$\min \frac{1}{2}\|w\| + C\sum_{i=1}^{n} \xi_i$$

st, $y_i(w^T x_i + b) \geq 1 - \xi_i$

$\xi_i \geq 0$

$\xrightarrow{\text{dual form}}$

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

st, $\alpha_i \geq 0$

$\sum \alpha_i y_i = 0$

Note

① x_i is replaced by α_i

② x_i only occur in pairs $\longrightarrow$ $x_i^T x_j \equiv x_i \cdot x_j \equiv \cos\text{-sim}(x_i, x_j)$ given $\|x_i\| = \|x_j\| = 1$

③ primal : $f(x) = (w^T x_q + b)$ ← predict fn

dual = $f(x_q) = \sum_{i=1}^{n} \alpha_i y_i x_i^T x_q + b$ — IMP

→ only depends on support vectors as $\alpha_i > 0$

④ $\alpha_i > 0$ → support vectors

$\alpha_i = 0$ → non-support vectors

⑤ $x_i \cdot x_j$ or $x_i^T x_j \equiv$ cos-sim

∴ $x_i \cdot x_j$ can be replaced by any similarity

→ $K(x_i, x_j) \equiv$ Kernel function

dual form

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} (\alpha_i \alpha_j)(y_i y_j)(x_i^T x_j)$$

→ $K(x_i, x_j)$

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i (x_i^T x_q) + b = \sum_{i=1}^{n} K(x_i, x_j)\alpha_i y_i$$

Kernel SVM used for non-linearly separable dataset

→ $K(x_i, x_j) = x_i^T x_j$ → $K(x_i, x_j)$: similarity b/w (x_i, x_j)

Polynomial Kernel

$$K(x_1, x_2) = (x_1^T x_2 + c)^d$$

Quadratic kernel $c=1, d=2$

$$K(x_1, x_2) = (1 + x_1^T x_2)^2$$

let $x_1 = \langle x_{11}, x_{12} \rangle$
$x_2 = \langle x_{21}, x_{22} \rangle$ → aply Quad-kernel → 2d

poly ≡ feature transform

$$K(x_1, x_2) = (1 + x_{11}x_{21} + x_{12}x_{22})^2$$

$$= 1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{21}x_{12}x_{22}$$

↳ this can be written / rearranged as

$$(1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}) \rightarrow x_1' \rightarrow 6d$$

$$* (1, x_{21}^2, x_{22}^2, \sqrt{2}x_{22}, \sqrt{2}x_{22}, \sqrt{2}x_{21}x_{22}) \rightarrow x_2'$$

$$(x_1')^T (x_2') \quad \text{or} \quad x_1' \cdot x_2'$$

Feature transform

x_1	x_2	
↓	↓	$d \rightarrow d'$
x_1'	x_2'	$d' > d$

∴ Kernalizat n ≈ feature transfrm

(f_1, f_2) → kernel trick / kernel fn → (f_1', f_2')

d-dim → d'-dim

Mercers theorem

d → (non-linearly sep) → Kernel trick → d' $(d' > d)$ → (linearly separable)

(44) **RBF Kernel** Radial Basis function kernel

$\gamma = \frac{1}{2\sigma^2}$

RBF = KNN

$$K(x_1, x_2) = \exp\left(-\frac{d_{12}^2}{2\sigma^2}\right)$$ where,

$d_{12} = \|x_1 - x_2\| \leftarrow dist(x_1, x_2)$

σ = hyper parameter

Obs :- (i) as $d \uparrow$, $K \downarrow$

ie as distance ↑, similarity decreases.

RBF + SVM margin

2 hyperparam

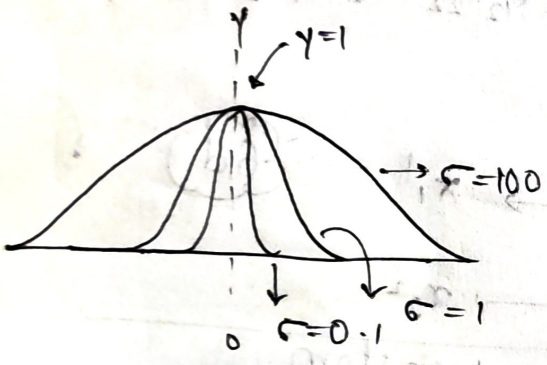
(i) σ (ii) C

(ii) affect of σ

plot → $e^{\frac{-x^2}{2\sigma}}$ → $\sigma = 0.1$; (*******) → k>0

$\sigma = 1$; (*******)

$\sigma = 100$ (*** | * * *)



∴ as $\sigma \uparrow$, no of pts considered for similarity increases

σ is similar to (K) in KNN

Time complexity

Train

i) SMO

ii) sequential minimal optimization (SMO)

→ $O(n^2)$: kernel svm

→ $O(nd^2)$: if $d < n$

∴ if n is very large, dont prefer SVM

Runtime $f(x_q) = \sum \alpha_i y_i K(\,) + b$

$O(kd)$ → k : no of suport vectors

Train → $O(n^2 \cdot d)$

as $\sigma \downarrow$,

only consider nearest points

$\exp\left(\frac{-d^2}{2\sigma^2}\right)$ $\left(\gamma = \frac{1}{2\sigma^2}\right)$

$\equiv \exp(-\gamma \cdot d^2)$

nu-SVM

1) default svm – C-SVM

hyperp = C ($C \geq 0$)

→ hp: nu → $0 \leq nu \leq 1$

nu ≥ fract^n of errors

nu ≤ fract^n of svectors

usage:

nu – max error which can hpn

(1) want error < 1%

nu = 0.01

Ex: if nu = 0.01

% errors ≤ 1% → upper bound

support vectors ≥ 1% → lower bound

nu-svm can control errors

but cant control sv's → Preds depend on only sv's (ie $\alpha_i > 0$)

Support Vector Regression (SVR)

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

s.t,

$$y_i - \hat{y}_i \leq \epsilon$$

$$\hat{y}_i - y_i \leq \epsilon$$

$\equiv |\hat{y} - y| \leq \epsilon$

→ ϵ is similar to minimum error

ie $\epsilon = 0.1$

max error is → 0.1

: $y - \hat{y} \leq 0.1$; $\hat{y} - y \leq 0.1$

$$\hat{y}_i = w^T x_i + b$$

steps

→ data (x_i, y_i)

→ train

find optimal w,b

→ predict

$\hat{y}_i = w^T \cdot x_i + b$

↳ predictions

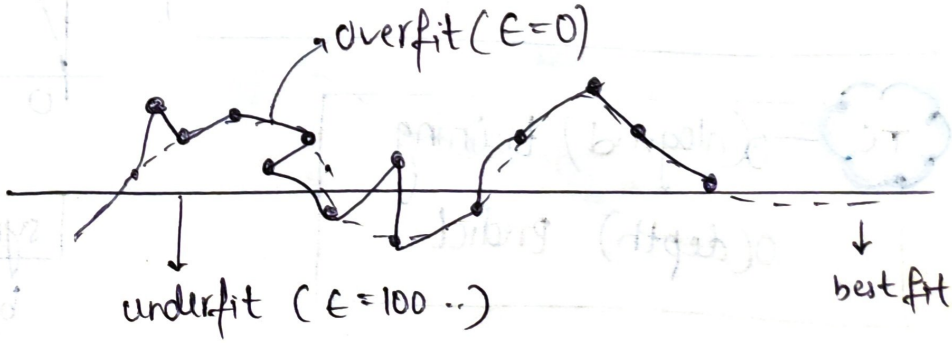
cases

if $\epsilon = 0$;

overfit chances → minimizes error to 0

if $\epsilon = 100$;

underfit → error to minimize is 100



46

SVM - key conclusions

- Best — u have kernel
- Worst — n is large; sv are large

* Outliers — little impact as only support vectors matter
  - Exception :- RBF kernel with small $K \approx 5$

* Bias - variance = C↑ - overfit - var↑
  C↓ - underfit - bias↑

* large dimensions → good for svm as it tries to $d \rightarrow d'$ $(d' > d)$

Decision Tree

Entrophy → degree of randomness

$\log_2 = \lg$

$$\text{Entrophy}(Y) = -\sum_{i=1}^{K} P(y_i) \log_2 (P(y_i))$$

K : no of classes
binary — yes/no | K=2

$D(14) < {}^{+(9)}_{-(5)}$ $\quad H(Y) = -\frac{9}{14}\lg\left(\frac{9}{14}\right) - \frac{5}{14}\lg\left(\frac{9}{14}\right)$

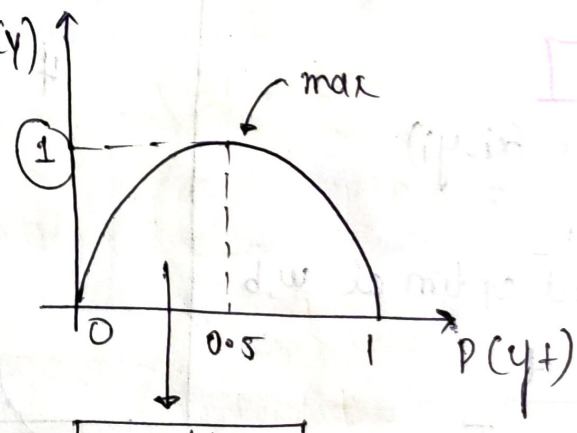
consider 2 state var-y

① $D(10) < {}^{5(+)}_{5(-)}$ $\quad H(Y) = 1$ (maximum)

② $D_{10} < {}^{0(+)}_{10(-)}$ $\quad H(Y) = 0$ → $0 \cdot 0 + \log 1 = 0$

Results

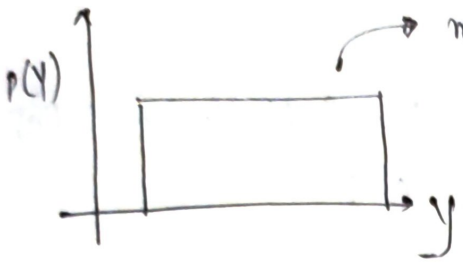
(i) $H(Y) = 1$; if equiprobable
(ii) if one class dominates fully → $H(Y) = 0$



symetric
because $P(y+) = 1 - (P(y-))$

TC — $O(n \log n \cdot d)$ training
$O(\text{depth})$ predict

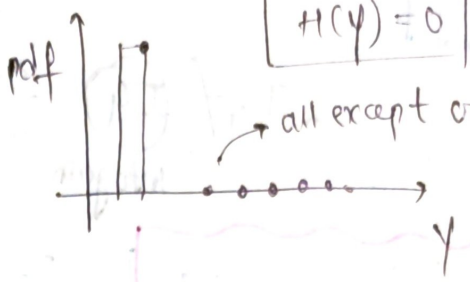
PDF based Interpretation

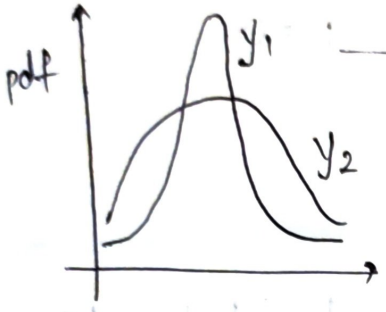


maximum entropy

discrete

$H(y) = 0$





$\rightarrow H(y_2) < H(y_2)$

$\rightarrow y_2$ is more spread ie leads to equiprobable

y_1 is peaked, ie one value might be most probable

$\therefore H(y_2) > H(y_1)$

peak ↑ entropy ↓

iq = Ent(parent) − weighted_entropy (child)

Info-gain

categories

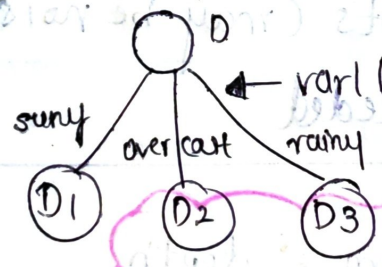
$Y \xrightarrow{\text{var}} y_1, y_2 \cdots y_K$

$$iq(Y, var) = -\left(\sum_{i=1}^{K} \frac{|D_i|}{|D|} \times H_{D_i}(y)\right) + H_0(y)$$

$H_0(y)$ → entropy before splitting

$\sum \frac{|D_i|}{|D|} \times H_{D_i}(y)$ → weighted entropy after splitting

Ex:-



iq = −weighted_entropy + entropy (previous)

$$iq(D, var1) = \left[\frac{|D_1|}{|D|} H_{D_1}(Y) + \cdots\right] - H_D(Y)$$

* iq ↑ better
* weighted ent ↓ → iq ↑

Ae

Gini Impurity → $y < \begin{matrix} y^+ \\ y^- \end{matrix}$

$y \rightarrow y_1, y_2 - y_k$ (k - categories) → $I_G \neq$ (Eq) Integam

$$I_G(y) = 1 - \sum_{i=1}^{K} (p(y_i))^2$$

gini impurity
- → equiprobable → max-val = 0.5
- → one dominate → 0

case 1

$y < \begin{matrix} 5 \\ 5 \end{matrix}$ $I_G(y) = 1 - (0.25 + 0.25) = 0.5$

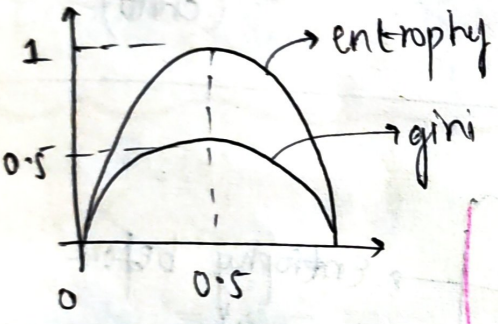
$I_G(y) = 0.5$

case 2

$y < \begin{matrix} 10 \\ 0 \end{matrix}$ $I_G(y) = 1 - 1 = 0$

$I_G = 0$

Entrophy vs Gini → gini is same as entrophy but only less scaled ie max value is 0.5



why gini >> entrophy?
→ no log term to calculate

Initial split

(H(y+, y-)

① calculate $H_D(y)$

② find weighted-entrophy for each ~~cate~~ variable

$I_G(y, \text{outlook}|\text{sunny}| \cdots)$

③ for first split

↓

pick lowest weight-entr or highest IG

when to stop

① pure node - all pts belong to 1 class

② very few pts (may be noisy)

③ depth exceeded

hyperparam ≡ depth

d ↑ → overfit

d ↓ → underfit

for IG u can use entrophy / gini impurity

Numerical features

form

$f1 - (0 \longleftrightarrow 10000)$

then

$f1 < \tau_1$
$f2 < \tau_2$
$\vdots$
$f_1 < \tau_n$

} pick best information gain yielding threshold

$\tau_i \rightarrow$ thresholds

steps

$f_1 - (f_{11}, f_{12}, f_{13} \dots f_{1n})$

(1) sort f_1

(2) pick thresholds $(\tau_i = f_{1i})$ and split

(3) τ_i which maximizes (1)(4)

Categorical ft with large categories

Mean Encoding

Ex:- pincode — may have 1000's of pincodes

→ (solution) convert to numerical features

consider — class = +/− (0,1)

→ pincode = $\{P_1, P_2, P_3 \dots P_n\}$

$$tp = P_i = P(y=1 \mid P_i)$$

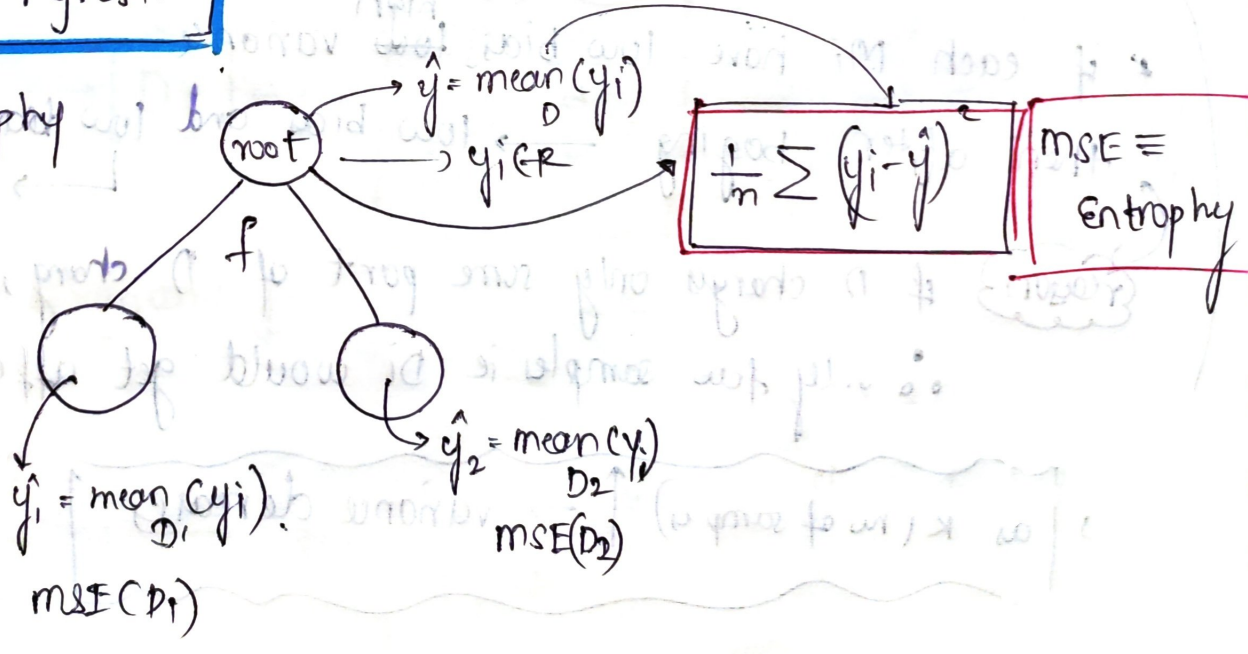
→ probability that $(y==1)$ given $(\text{pincode} = P_i)$

$$P_2 = P(y=1 \mid P_2) = \frac{\#(y_i==1 \ \& \ P_j = P_2)}{\#P_2}$$

no of P_2 with $y=1$ / total

Decision Tree regression

*MSE ⇔ Entropy



50

∴ IG = MSE(D) − weighted MSE(D1, D2)

(MSE(D): before split; weighted MSE(D1, D2): after split)

variance ↑ ≡ overfit

classifn	reduce entropy — ideal = 0
regresn	reduce MSE / MAD — ideal = 0

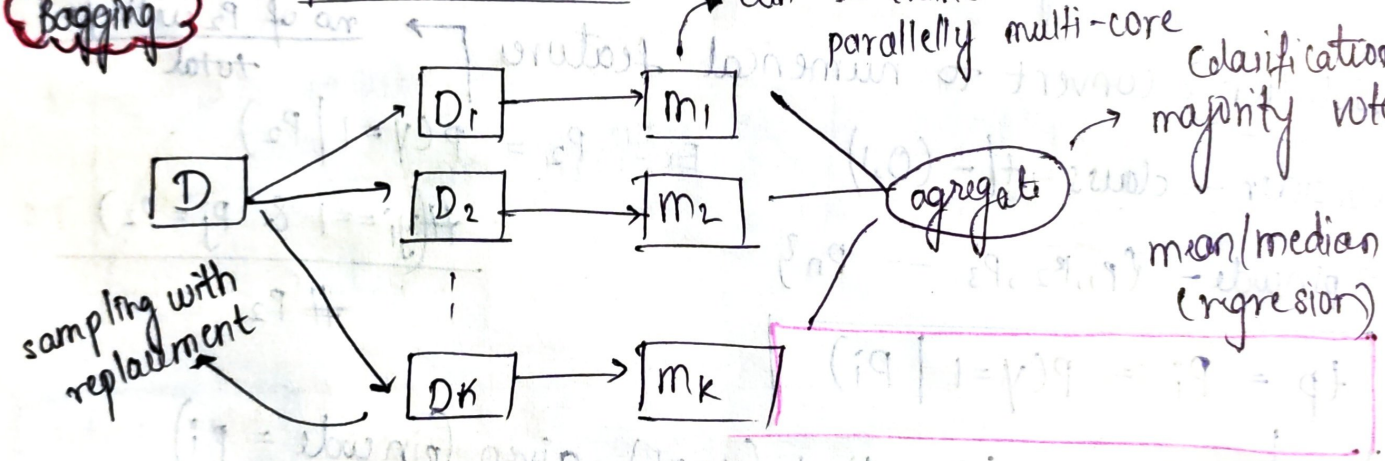
Ensemble Techniques

- Bagging
- Boosting
- Stacking
- cascading

Random Forest — based on Bagging (bootstrap aggregation)

→ reduce variance

Bagging



variance :- how much model changes when data changes

→ lower the better

* if each Mi have low bias, ~~low~~ high variance

then after bagging → low bias and low variance

→ low or high!

reason if D changes only some part of D changes

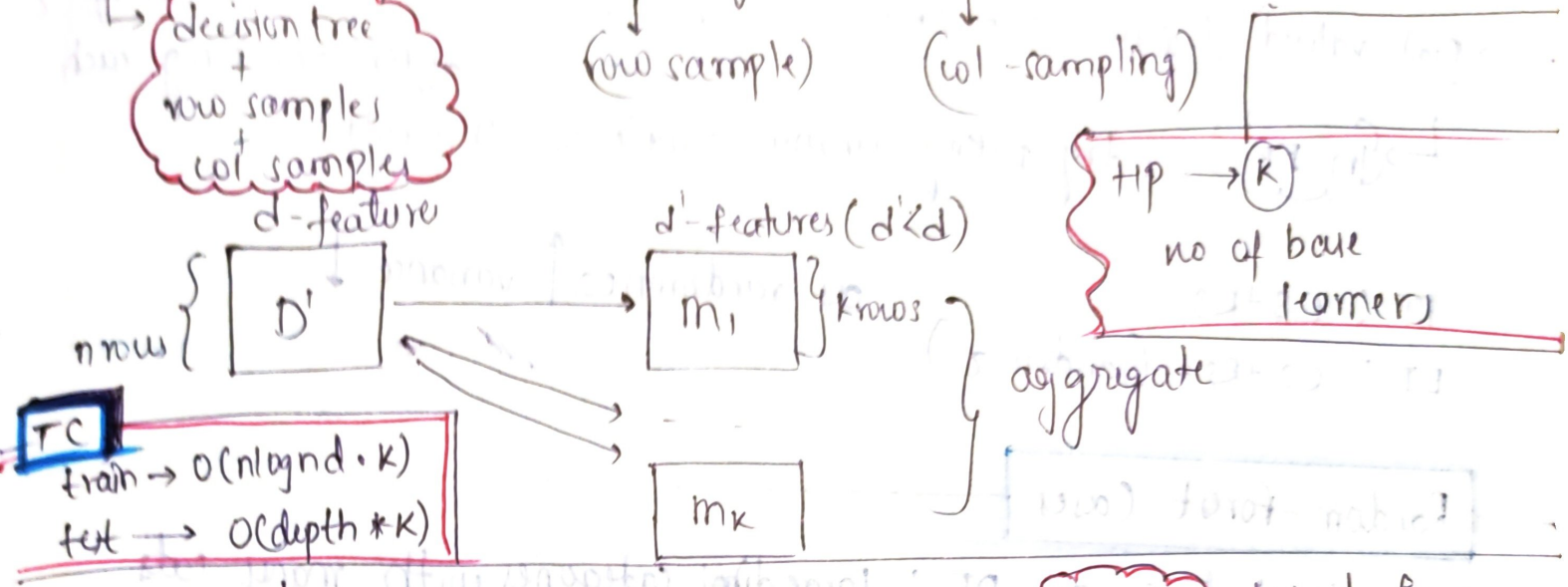
∴ only few samples ie Di would get affected

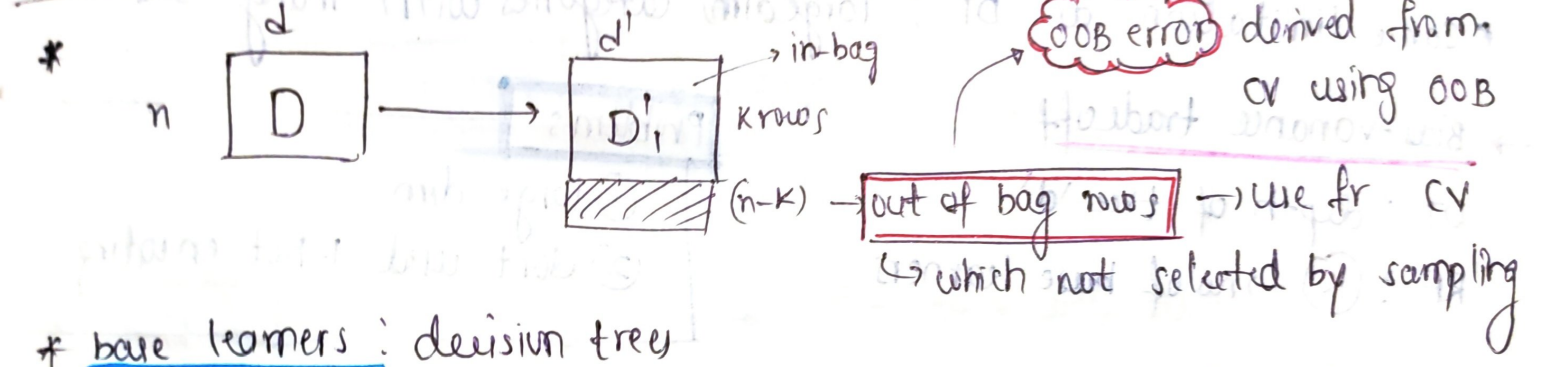
as K (no of samples) ↑ → variance decreases

Random forest : DT + Bagging + feature sampling

↳ (decision tree + row samples + col samples)

(row sample) (col-sampling)

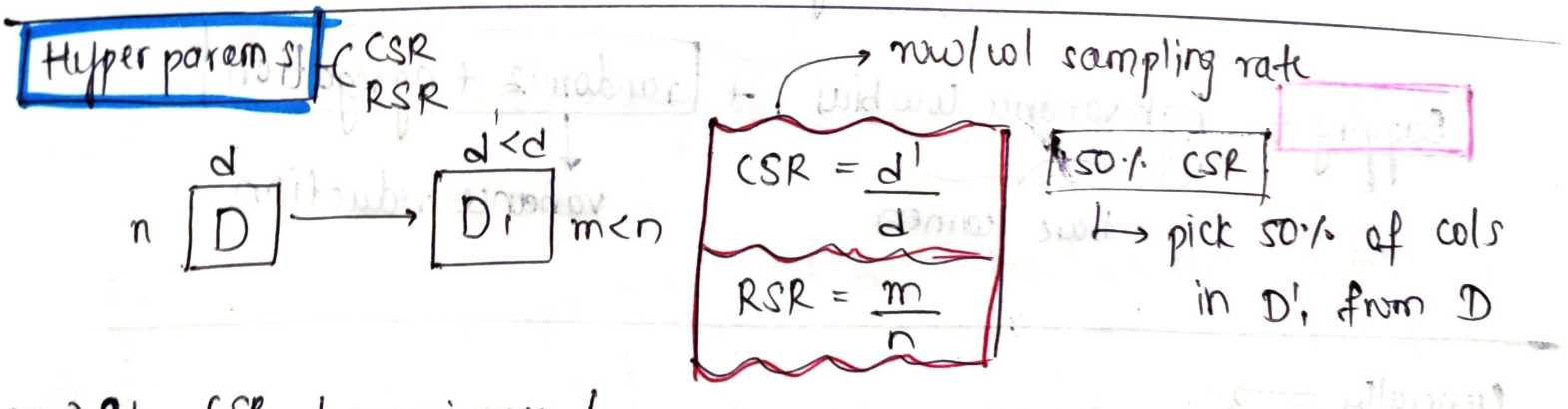




* base learners : decision trees

→ DT have large depth bcoz if depth↑ - variance↑ and we can reduce variance using bagging.

Hyperparams : CSR, RSR



→ as CSR, RSR ↓ ≡ variance ↓

General way

↳ fix CSR and RSR → try best value of (K) → choose best K using CV

(52) Extremely Random Tree

→ Real valued features : RT/DT → sort values → pick best value of ? after checking each

↳ In ERT → try picking random sample of thresholds

RT : CS + RS

ET : CS + RS + (random z)

as randomness ↑ variance ↓

Random-forest Cases

* same limitation as DT : large dim categories with many cats

→ Bias-variance tradeoff

DT : depth of tree 'd'

RF : (K) → no of base learners

Problems

① large dim

② dont used 1-hot encoding

→ Feature Importance

DT : more imp to feature which reduces Entrophy at level

RF : ↳ which reduces entrophy in all (K) model.

Bagging = high variance, low bias + randomiz + aggregation

base learners

variance reduction

Generally →

$$\text{Error} = \text{bias}^2 + \text{var} + \epsilon$$

irreducable error

Boosting

- Gradient Boosting (GBDT)
- AdaBoost (adaptive boost)

Boosting = low-var, high-bias + additive combine
(base model) (reduce bias)

low-bias (bi) high var = DT with dT

low var high bias → shallow DT (d↓)

Core Algo

Base model

step 0

* $D_{train}\ (x_i, y_i)_1^n \longrightarrow M_0 \quad y = h_0(x)$ → will have high error as D_{train} is low var high bias

$error_i = y_i - h_0(x_i)$ $\quad y_i - \hat{y}_i$ = AE

$\{x_i, y_i, error_i\}_1^n$

step 1 → it tries to predict $error_i = y_i - h_0(x_i)$

$\{x_i, error_i\} \longrightarrow M_1 \quad y = h_1(x)$

$\Rightarrow F_1(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x)$ } model at end of stage 1

step 2

$\{x_i, err_i\}_1^n \longrightarrow m_2 \quad y = h_2(x) \Rightarrow F_2 = \alpha_0 h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x)$

$\hookrightarrow y_i - F_1(x_i)$

step K → trained to fit residual error @ prev step

$\Rightarrow F_K(x) = \sum_{i=0}^{K} \alpha_i h_i(x)$ → $h_i(x)$ is trained on $\{x_i, err_i\}$

additive weighted model

error at end of stage $(i-1)$

as residual error ↓ ⇒ train error ↓ ⇒ bias ↓

$err_i = y_i - F_{i-1}(x_i)$

(54) **Residual**

* residual at stage (K) = $err_i = y_i - F_K(i)$ —used in→ building model F_{K+1}

loss minimization

let $L(y_i, F_K(x_i)) = (y_i - F_K(x))^2$ → squared Loss

let $F_K(x_i) = z_i$

$$\frac{dL}{dz_i} = 2(y_i - z_i) - 1 = -2(y_i - z_i)$$

$$\Rightarrow -\frac{\partial L}{\partial z_i} = 2(y_i - z_i)$$

$$\therefore -\frac{\partial L}{\partial z_i} \approx (y_i - F_K(x_i))$$

(residual; negative gradient called pseudo-residual)

idea

* (x_i, err_i) is used for training

$err_i \rightarrow$ residual$_i$

$err_i \rightarrow$ pseudo-residual $= -\frac{\partial L}{\partial F_{i-1}(x)}$

stage i

$(x_i, err_i) \rightarrow m_i$

↳ pseudo residual { not residual: $y_i - F_K(x_i)$

can have any Loss function, only constraint it should be differentiable

(L)

∴ Gradient Boosting uses pseudo-residuals

Hence, you can use any Loss function (L)

ie minimize any diff^{ble} loss function

residuals keep on getting smaller

$\arg + (\alpha_1 x_1) + (\alpha_2 x_2)$ res

Gradient Boosting

Input : $\{x_i, y_i\}_1^n$, $L(y, f(x))$: differentiable loss fn , M → no of base learners

Algo :

① Initialize model ← find γ which minimizes

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum L(y_i, \gamma)$$

→ Here if L is squared-loss

$$\min_{\gamma} (y_i - F_k(x_i))^2 \rightarrow \boxed{\gamma = \text{mean}}$$

↑ this what you get after minimizing

② for $m = 1$ to M

① Compute pseudo-res

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x) = F_{m-1}(x)}$$

② fit learner $h_m(x)$ with training data $\{x_i, r_{im}\}$ ← pseudo residual

③ Compute constant γ_m

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum \left[L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))\right] \quad = \alpha_i$$

easy (why?)

only γ is unknown

④ update

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

③ output → $F_m(x) = F_0(x) + \sum_{i=1}^{m} \gamma_i h_i(x)$

$$F_M(x) = h_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x)$$

$$\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

average

(56) Regularization and shrinkage

hyper-params → M, V
- cross validation
- grid search
- random grid search

* Bias-var tradeoff ≈ K/M

M ↑ — overfit
M ↓ — underfit

shrinkage → modifying update rule

$$F_m(x) = F_{m-1}(x) + \gamma_m \cdot h_m(x) \quad (\cdot \nu)$$

→ more (ν) → more importance given to $h_m(x)$

ν → learning rate / shrinkage coefficient

Time complexity → Not easy to parallelize

* Train — O(n lg(n) * M) space :- O(store each tree + γ_m)
* run — O(depth * M)
  - small ⇒ we use shallow trees (good for low latency applications)

XGBoost — best implementation of GBDT
- GBDT + row sampling + column sampling

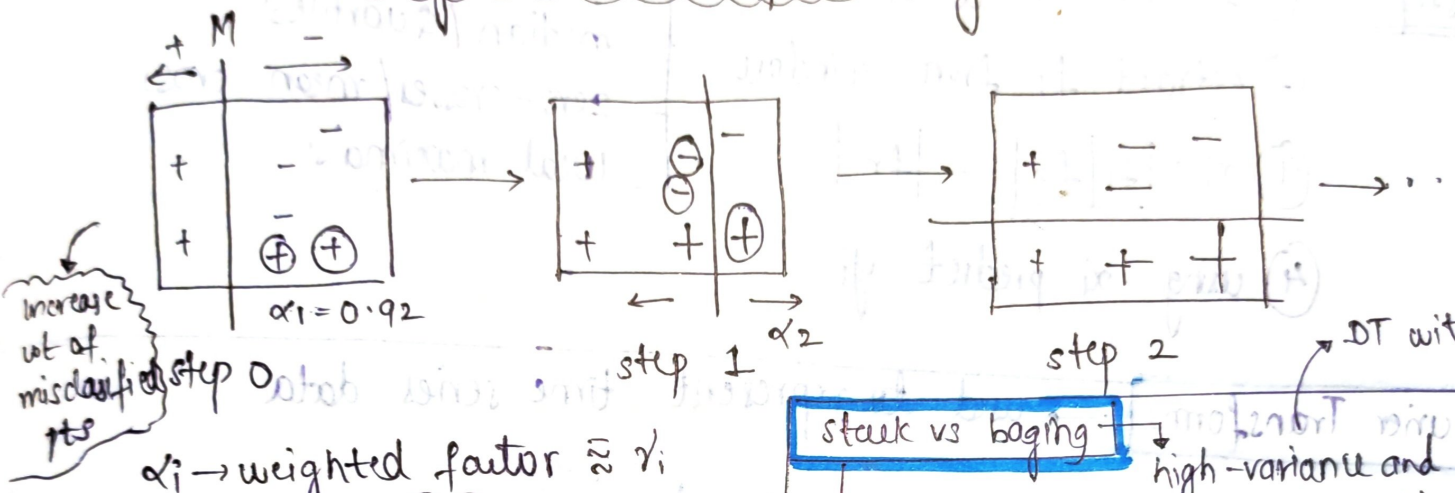
added params
* colsample_bytree
* colsample_level
* reg-alpha → L1 regularization ⇒ $\alpha L_1 + \lambda L_2$
* reg-gamma lambda → L2 regularization

sklearn.GradBoost()
- loss = 'deviance' → GBDT (same to log-loss)
- loss = "exponential" → Adaboost

Adaboost used for face detection

main idea : at each step keep on changing weights assigned to pts to incorporate errors.

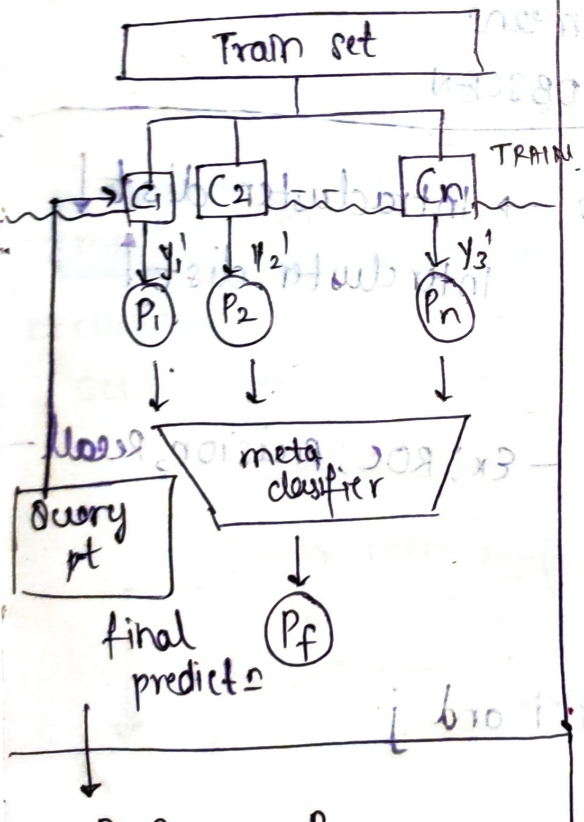
* weights increase exponentially



$\alpha_i \rightarrow$ weighted factor $\approx \gamma_i$

stack vs bagging
- well def bias-variance
- high-variance and low-bias

stacking extend



$P_1, P_2 \dots P_n$ may be ① class labels ② probabilities

algo

i/p : $D - (x_i, y_i)_1^n$ o/p : H

step 1 : learn 'T' base classifiers

for i ← 1 to T

train model $h_i(x)$ on D

step 2 : new dataset D'

for i ← 1 to n

$D' = \{x_i', y_i'\} \equiv x_i' = \{h_1(x), h_2(x) \dots h_m(x)\}$

step 3 : meta-classifier

return $H(x)$ trained on $\{x_i', y_i\}$

mlxtend

Predict

→ x_q

→ find $h_1(x_q), h_2(x_q) \dots$

→ meta clf

$\langle h_i(x_q) \rangle \rightarrow y$

Cases

① classifiers must be as different as possible

② useful in Kaggle compt^n s

Featurization → page (18) book-2

moving window

steps: ① fix window width
② extract fi from window
④ xi : | f1 | f2 | -- | fn |
④ using xi predict yi

- max-min
- avg
- mean/sd
- median / Quartiles
- zero-crosses / mean cross
- local maxima's

Fourier Transforms → used to represent time series data

time-based region → frequency-based region

CLUSTERING

- hierarchical clustering
- Kmeans
- DBSCAN

Applications

*group similar customers
*image segmentation

good clusters → Intracluster dist ↓
Intercluster dist ↑

metrics — there is no (yi) to check error — Ex: ROC, Precision, Recall

Dunn Index [DUNN INDEX]

$$D = \frac{\min_{i,j} d(i,j)}{\max_{K} d'(K)}$$

$d(i,j)$ → dist b/w cluster i and j

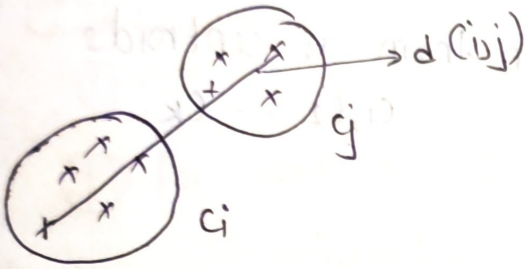
$d'(K)$ → Intra cluster distance

no of clusters = K

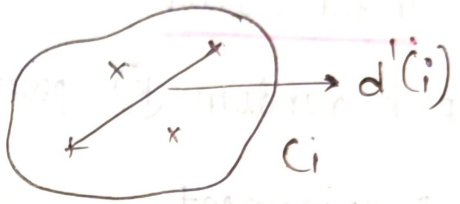
D ↑ — better
↳ max inter & min intra

calculating d and d'

$d(i,j)$ → inter
↳ distance b/w 2 farthest pts in C_i and C_j



$d'(i)$ → intra
↳ farthest dist b/w 2 pts in cluster i



K-means → centroid based clustering
→ hyper param - K : no of clusters

Centroid :- geometrically a mean

Ex :-

$$C_i = \frac{1}{n}\sum_{j=1}^{n} x_j$$

$n \rightarrow$ (size of S_i) ↓ sets

mathematical formulation

K-centroids — $C_1, C_2 \dots C_K$
sets — $S_1, S_2 \dots S_K$

$\forall x_i \in S_j$.
$\forall i,j,\ S_i \cap S_j = \phi$

NP-hard pblm

$$\arg\min_{C_1,C_2\dots C_n} \sum_{i=1}^{K} \left(\sum_{x \in S_i} \|x_i - C_i\|^2 \right)$$

(dist b/w points in set and its centroid)
→ Intracluster dist

such that,
$x \in S_i$
$S_i \cap S_j = \phi$

as FC ↑
↓
approximation
(Lyods algorithm)

Lloyds algorithm

$D = \{x_i\}_i^n$, K - clusters

step 1 : Initialization

→ pick random (K) points and asign them as centroids $c_1, c_2 \dots c_K$

step 2 : assignment

→ for each pt (x_i) in D :

→ select nearest (c_j) :

$$\min \left(d(x_i, c_j)\right) \quad \forall j = 1 \text{ to } K$$

→ add x_i to nearest c_j set (s_j)

step 3 :- recompute centroids

→ for each centroid c_j $\forall j = 1 \rightarrow K$

$$c_j = \frac{1}{|s_j|} \sum_{x_i \in s_j} x_i$$

step 4 :- repeat (2) and (3) until convergence

new updated centroids and older ~~olde~~ ones dont change much

ie $|C_{old} - C_{new}| \rightarrow 0$

Initializing centroids → how do you assign (K) pnts as centroid

→ (1) try randomly
(2) K-means++

→ **Kmeans++ Init** → $[C_1, C_2 \dots C_k]$ (probabilistic)

(1) pick C_1 randomly

(2) for each $x_i \in D$,

calculate $d^2(x_i, \text{Nearest Centroid})$ → $(|x_i - C_{nearest}|)^2$

x_1	d_1
x_2	d_2
x_n	d_n

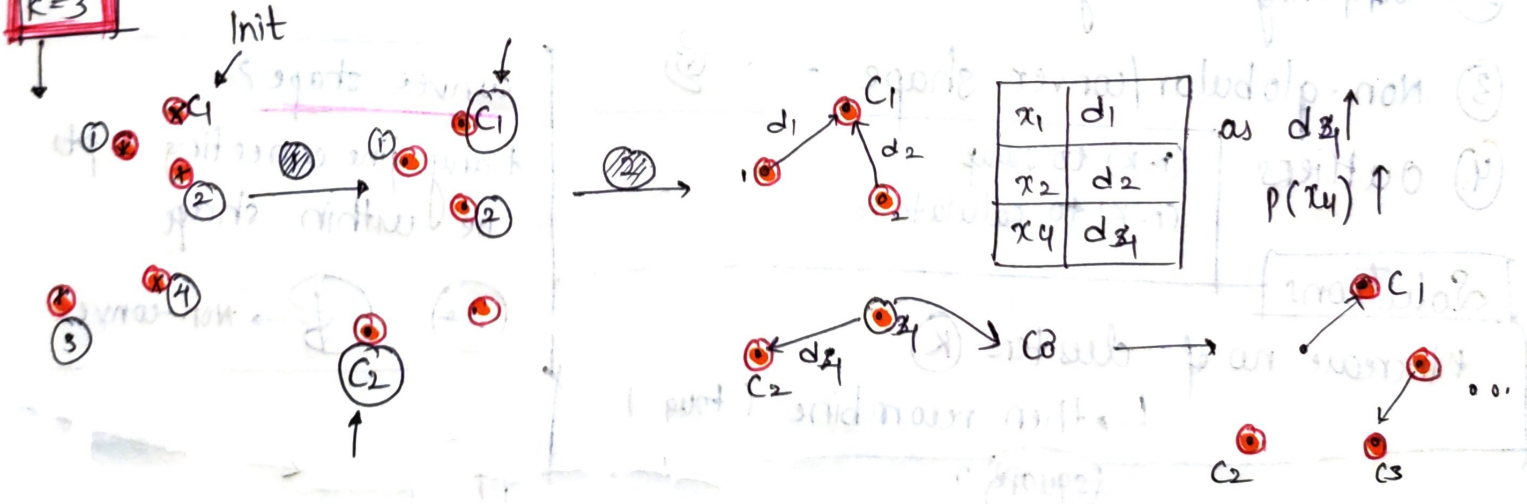
first case → only 1 centroid → $d(x_i, C_1)$

(3) → pick a point from $D - \{\text{already assigned}\}$ → already assigned centroids as centroid with probability $\approx d_i$ → proportional

$P(x_i \text{ chosen as centroid}) \approx d$ ∴ ↑d → prob↑

K=3

Init



x_1	d_1
x_2	d_2
x_4	d_4

as d_4↑ $P(x_4)$↑

why prob(d) and not directly max(di) ?

→ outliers will always become centroids

62

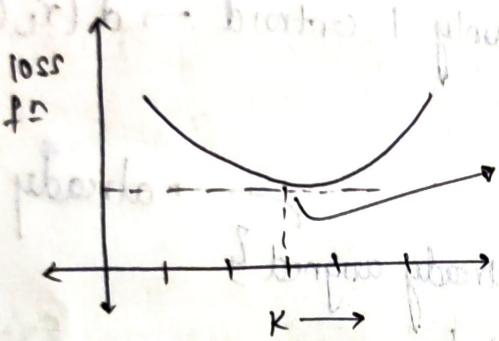
Best K

① Domain knowledge : you know exactly how many class

Ex:- (K=2) → diseased pred yes/No
↳ spam — yes/No

② Knee / Elbow method

→ loss fn : $\sum_{i=1}^{K}\left(\sum_{x \in S_i} |x - C_i|^2\right)$ [Train model on different values of K]



elbow point → (pick best K ≡ lower loss)

Limitations

① Different sized clusters :- Kmeans ideally creates same sized clus

NY Taxi

② Differing density :- it tends to stretch them out

③ Non-globular/convex shape →

④ Outliers

(n-K) to swap
(n-K) to calculate loss

Solutions

* Increase no of clusters (K)
↳ then recombine (tough)

convex shape?

* any line connecting 2 pt lie within shape

→ Non-convex

square ?

TC

calculate distance each (medoid) swap by each non-medoid

non-meidiods (n-K)

$O(K*(n-K)^2) \equiv O(n^2)$

K meidio

Kmeans → Lyod
K-mediods → PAM

K-mediods — PAM (partitioning around mediods)

→ problem with centroids: Each centroid is actually a datapoint

*Not interpretable : $C_i \notin D$, ie centroid may not belong to D, u can get info about it

PAM

Initialization : - K-means++ (probabilistic)

$O(n^2)$

Assignment : same as K-means

$\forall x_i \in C_j$, if for x_i the closet mediod is C_j

***update : (a) swap each mediod with non-mediod → (n-k) swaps

(b) if loss decreases keep swap else undo swap.

PAM

$$\sum_{i=1}^{K} \sum_{x_i \in S_i} \|x - m_j\|^2 \longrightarrow \boxed{dist(x_i, m_j)}$$

can be kernalized [Kernel matrix]

⟹ repeat until converge

uses

*At last each mediod is a datapoint

⟹ $\forall C_i \in D$ so easily interpretable.

* $\sum\sum d(x_i, m_j)$ → can be replaced by Kernel trick.

less prone to outliers.

Time Complexity

LINEAR TC → LLYODS

$O(nkdi)$

n – #pts
K – no of clusters
d → no of dimensions
i → no of iterations

space

$O(nd + kd)$

nd: store n pts
kd: store K centroids

(64)

Heirarchical clustering

- Aglomerative — bottom up
- ~~Divisive~~ Divisive — top-down

Aglomerative algorithm

No need to give (K) as hyper-param

↳ assume all pts are clusters → keep on reduce no of clusters

* you need distance b/w pts as well as clusters

Dendogram: a tree which depicts sequence of merges/splits

Algorithm

→ dist matrix

① Compute proximity matrix

② Initialy all pts are clusters

③ Repeat

   ⓐ merge the 2 closest clusters

   ⓑ update proximity matrix

④ until only one cluster

	P_1	P_2	P_3	...
P_1	0			→ lowest dist
P_2		0		
P_3			0	

↓

	$P_1 \cup P_3$	P_2	P_4
$P_1 \cup P_3$	0		
P_2		0	
P_4			0

needs update

main issue → how to update prox-mat?
how to define similarity b/w clusters?

Def Intercluster distance

pick lowest value of each | min (max i) ...

→ **MIN** $sim(C_i, C_j) = \min_{p_i \in C_i,\ p_j \in C_j} sim(P_i, P_j)$ [pick smallest line formed by (P_i, P_j) where $p_i \in C_i$, $p_j \in C_j$]

→ **MAX** pick 2 farthest pts in C_i, C_j

→ **GROUP AVG** $sim(C_1, C_2) = \frac{\sum sim(P_i, P_j)}{|C_1| * |C_2|}$

↳ less prone to noise

↳ biased towards globular

wardi → (^2)

same as group_avg but $\frac{\sum (d(P_i, P_j))^2}{(C_1 | C_2)}$

→ Centroid distance — calculate d(centroid(Ci), centroid(Cj)) (63)

↳ cant generalize easily? → because Ci, Cj may not be in D

∴ we dont have d(Ci, Cj)

pros/cons

* MIN — works for noneliptical shapes
  - ⊗ sensitive to noise and outliers
* MAX — less sensitive to noise
  - ⊗ Breaks large clusters into smaller ones
  - ⊗ biased towards globular shapes

MIN — single link
MAX — complete link

Time complexity — v.v large

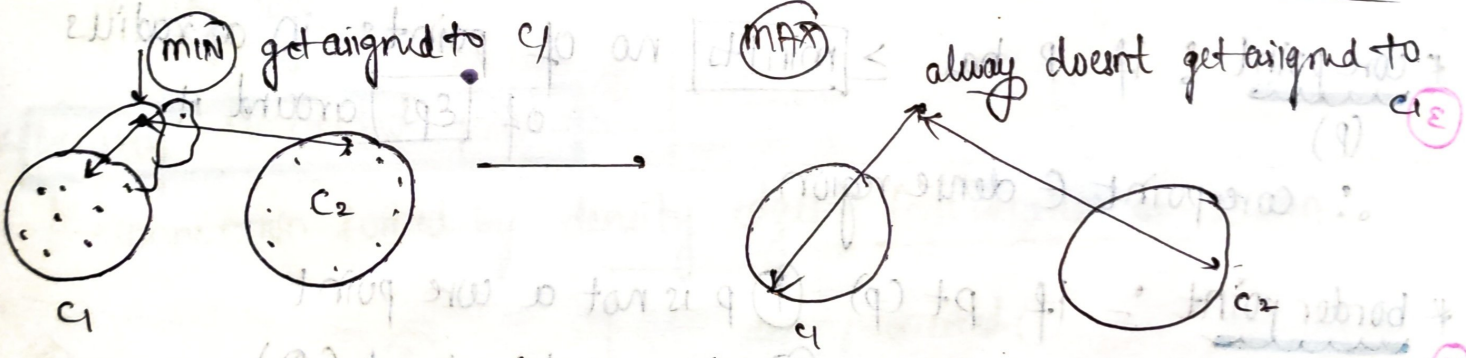
space :- $O(n^2)$ ($n \times n$) → proximity matrix

time :- $O(n^3)$ ⟹ (n) iterations

↳ In each update proximity matrix which is (n^2)

$O(n * n^2) = n^3$

MAX vs MIN in presence of noise



∴ min accumulates more and more noise

DBScan

Density Based clustering

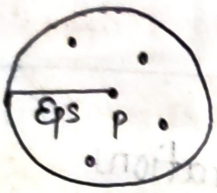
main aim → get sporse regions → noise
dense regions → clusters

measuring density → min Points, Eps } hyperparameters

(1)

① density at point P :- No of points in hypersphere formed by radius = ε around point P.

(2)



d(p) = 4

② dense region :- a hypersphere with radius = Eps that contains atleast MinPts no of points

(2)

D = {xi}, minPts, Eps ← given input

(3)

* core point : if p has ≥ minPts no of points in a radius of Eps around it (p)

∴ core point ∈ dense region

(4)

* border point :- if pt (p) (i) p is not a core point
(ii) p ∈ Neighborhood (Q)
Q → core point
dist(P,Q) ≤ Eps

Neither core Nor border ⇒ Noise pts

(5)

Ex: let Eps - 1.
minpts - 4

x_1 = core
x_2 = border
x_3 = noise

pts around circle - 3 < 4

but, $d(x_2, x_1) \leq Eps$ and x_1 is core pt

range Query

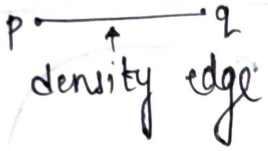
x_3 → O(log n) using K-d tree

Si → range-q (x_i, D, Eps)
↳ all points within Eps radius of (x_i)

* Density Edge (6)

① p, q are core points
② dist (p, q) ≤ Eps

then



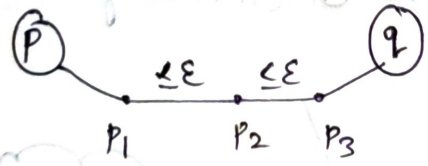
Time complexity

O(n · log n) → range query using Kd-tree

space
O(n)

* Density Connected points (7)

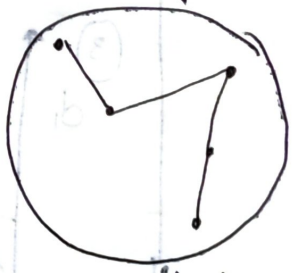
↳ connection formed by density edge



p_1, p_2, p_3 → corepts
p_1 — p_2 : density edge

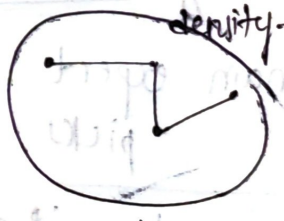
first iteration of DBScan

density-connected (1)



cluster 1

density-conn (2)



cluster 2

(68) **Algorithm** DBSCAN

rangeQuery ⇒ implemented Kd-tree

Input : $\{x_i\} = D$, MinPts, Eps

step 1 : $\forall x_i \in D$, label as core pt, border pt, noise pt

$s_i \rightarrow$ rangeQuery (x_i, D, Eps) = returns all pts within Eps radius of x_i

step 2 : Remove all noise points

step 3 : for each core point (P) not assigned to cluster,

iteration ← (i) create new cluster with p

(ii) add all pts that are density connected to p into this new cluster

step 4 : $\forall$ border point, asign it to nearest core pts cluster

Hyperparameters minPts, Eps

① **minpts** → dim

* usually $mp > d+1$ and $mp = 2*d$

* for noisy data pick larger MinPts

* domain expert picks

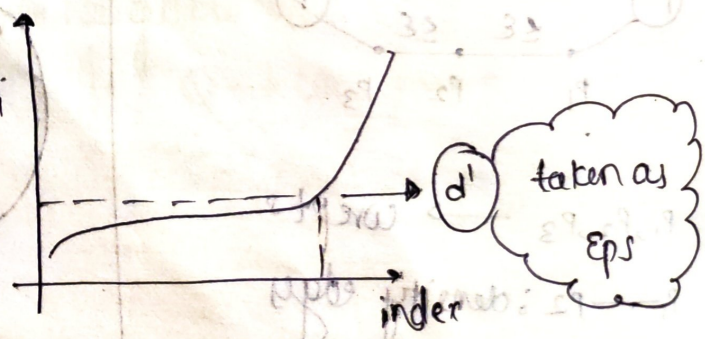
② **Eps**

Ex: minPts = 4

① $\forall x_i \rightarrow$ (d_i): dist from x_i to 4th nearest neighbour of x_i

② sort x_i acc to d_i (increasing order)

③



$d_i \uparrow \rightarrow$ noisy pt if $d(x_i, 4^{th}) \uparrow$ then to remaining pts will be even mo

Advantages

- ✓ resistance to noise
- * you dont need to specify no of clusters at start
- * 2 Hp — mp, Eps
- * suitable for databases
  R*-tree for retrieval
- * cluster shape may vary

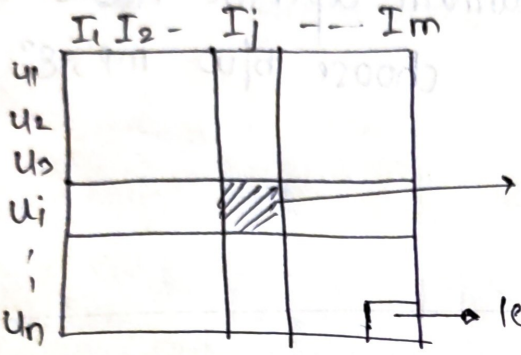
Disadvantages

- * data with varying densities
- * high-dimensional data.
- * v.v. sensitive to hyper parameters
- * domain expertise needed to choose b/w mp, ε°

Recommender Systems

Dataset

let ds be matrix (A) ⟶ n: no of users
m: no of items



A(i,j) ⟶ score given (Netflix)
watched/not (youtube)

leave blank if not watched/rated

sparse matrix

sparsity = # non-empty cells / # of total cells

Task

given user (ui) and I, Im
suggest In which (ui) will like

Types

- content based — classf°, regresion
- collaborative filtering (CF)

Core-assumption for CF

⟶ users who agreed/liked in past will also agree in feature

content based

⟶ doesnt use Aij
⟶ use meta data to represent ui, Ij
Ex: Ij – genre, autor, director, decade, year
ui → favorite genre, likes, dislikes

Similarities

- item-item sim
- user-user sim

usr-usr

u_i^T = [| | | - - |] (I1 I2 --- Im) ⟹ user vector → sparse
same as BoW

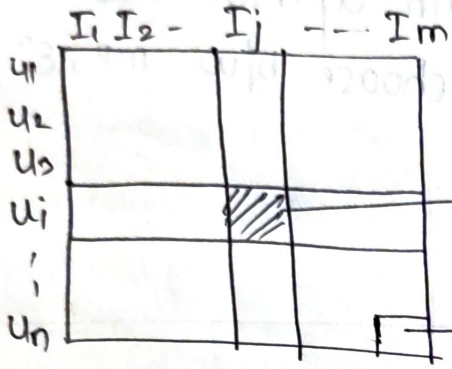
$sim(u_i, u_j) = cosine(u_i, u_j) = \frac{u_i^T u_j}{\|u_i\| \|u_j\|}$

(S^u_{ij})

Recommender Systems

Dataset

let ds be matrix (A) ⟶ n: no of users
m: no of items



sparse matrix

A(i,j) ⟶ score given (Netflix)
watched/not (youtube)

⟶ leave blank if not watched/rated

sparsity = # non-empty cells / # of total cells

Task

given user (ui) and I, Im
suggest In which (ui) will like

Types

- content based — classfn, regression
- collaborative filtering (CF)

Core-assumption for CF

→ users who agreed (liked in past will also agree in feature

content based

→ doesn't use Aij
→ use meta data to represent ui, Ij
Ex: Ij – genre, actor, director, decade, year
ui → favorite genre, likes, dislikes

Similarities

- item-item sim
- user-user sim

usr-usr

u_i^T = [I1 I2 --- Im] ⟹ user vector → sparse
same as BoW

$sim(u_i, u_j) = cosine(u_i, u_j) = \frac{u_i^T u_j}{\|u_i\| \|u_j\|}$

(S_{ij}^u)

S^u → user similarity matrix ⇒ S_{ij} (similarity b/w user i, j) (91)

Ex:- recomend movies to u_n

(1) find users u_i -- u_m similar to (u_1)

(2) recomend movies watched by u_i -- u_m to (u_1)

problem with user-user sim

* user preferences change over time.

if (user > items)
use Item-Item sim sys

Item-Item based

→ why? rating generaly dont change significantly after initial time

S^I → Item similarity matrix

Matrix Factorization Decomposition

PCA

→ centered data → covariance matrix $S_{d \times d} = \frac{X \cdot X^T}{n-1}$

$S_{d \times d} = \underset{d \times n}{X^T} \cdot \underset{n \times d}{X}$

find top eigen values — $\lambda_1, \lambda_2 \dots \lambda_n$ and eigen vectors — $w_1, w_2 \dots w_n$ such that

$\lambda_1 w_1 = S w_1$,

$\lambda_2 w_2 = S w_2$ ...

PCA as mat decomp? → Eigen decomposition PCD

$$S_{d \times d} = W_{d \times d} \cdot \Lambda_{d \times d} \cdot W^T_{d \times d}$$

S = covariance(X)

where,

$W_{d \times d} = [\uparrow w_1 \ \uparrow w_2 \ \dots \ \uparrow w_n]$ ⟹ matrix of eigen vectors (eigen vectors)

$\Lambda_{d \times d} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \lambda_3 & \\ 0 & & & \ddots \ \lambda_n \end{bmatrix}$ ⟹ diagonal matrix of eigen values

(72) **Singular Value decomposition** SVD

- SVD → can word when X is rectangular
- PCA → $S = cov(X) = X^T \cdot X$ is square matrix & symetric

$$X_{n\times d} = U_{n\times n} \cdot \Sigma_{n\times d} \cdot V^T_{d\times d}$$

$U = [u_1\ u_2\ \dots\ u_n]$ same for V

left singular vector of $X_{n\times d}$ — right singular vector of $X_{n\times d}$

$\Sigma_{n\times d}$ = diagonal matrix

s_i : singular values of X

$s_1 \geq s_2 \geq s_3 \geq \dots \geq s_d$

SVD ↔ PCA

$\frac{s_i^2}{n-1} = \lambda_i$ → Eigen values

u_i = i^{th} eigen vector of $X_{n\times d} X^T_{d\times n}$

v_i = i^{th} eigen vector of $X^T_{d\times n} \cdot X_{n\times d}$

→ Not S $(n\times d \cdot d\times n = n\times n)$

→ This is S $(d\times d)$

NMF (Non-negative MF)

SVD: $X_{n\times m} = U_{n\times k}\ \Sigma_{k\times k}\ V^T_{k\times m}$ — select return only k

$A_{n\times m} = B_{n\times d} (C^T)_{d\times m}$

s.t. $B_{ij} \geq 0 \quad \forall i,j$

$C_{ij} \geq 0$

NMF

Each cell in B and C is non-negative

AMIO

not sure

$A_{ij} = B_i^T \cdot C_j$

or

$A_{ij} = B_j^T \cdot C_i$?

MF for collaborative filtering

$A_{n \times m}$ — n-users, m-items, sparse matrix → A_{ij}

$\hat{r}_{ui} = b_i^T c_u$

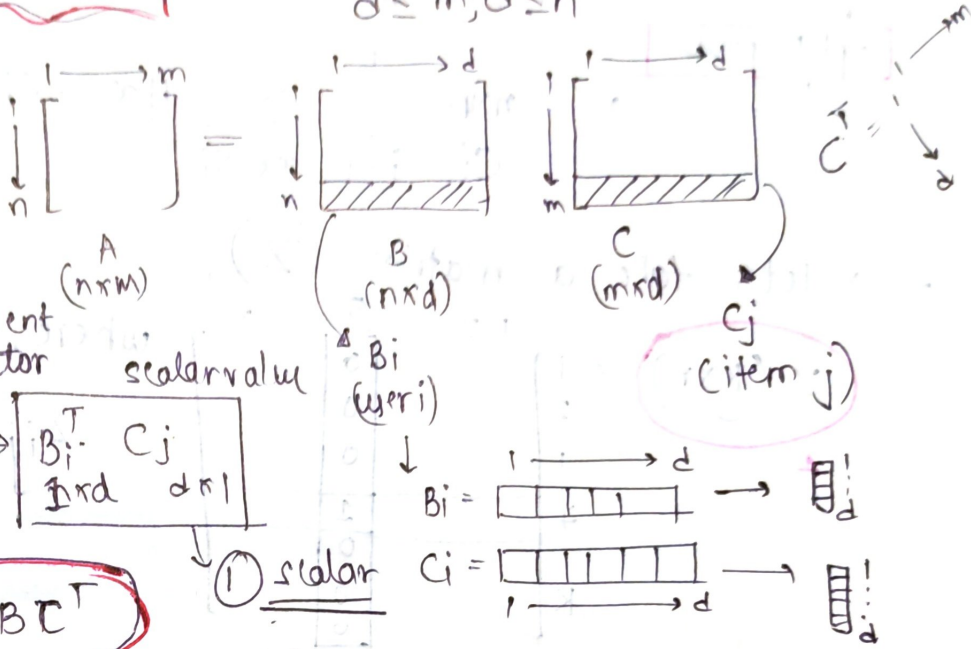
let $A_{n \times m} = B_{n \times d} (C^T)_{d \times m}$ where, $d \geq 0$, $d \leq m, d \leq n$

$A_{ij} = B_i * C_j$ — B_i represented as col vector, C_j represent as col vector

$d \times 1$

i^{th} row of B, j^{th} row of C

$A_{ij} = B_i \cdot C_j = B_j^T \cdot C_j$ → $B_i^T C_j$ ($1 \times d$, $d \times 1$) → scalar value



① scalar

$A_{ij} = B_i^T \cdot C_j \rightarrow A = BC^T$

optimization problem

$(y_i - \hat{y}_i)^2$

∴ optimization prob = $\underset{B,C}{\text{argmin}} \sum_{ij \in (A_{ij} \text{ is non-empty})} (A_{ij} - B_j^T C_i)^2$ → if empty you can't optimize

Bi and Cj r now represented as colum vectors (d×1)

Steps

dont use empty cells

step 1

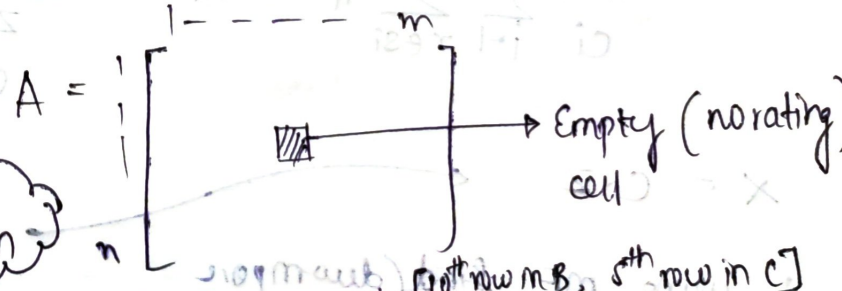
solve $\underset{B,C}{\text{argmin}} \sum (A_{ij} - B_i^T \cdot C_j)^2$

step 2

∴ filling empty cell = recomendation

u get values of B and C (n×d, m×d)

step 3 matrix completion



[10^{th} row in B, 5^{th} row in C]

$A_{10,5} = B_{10}^T * C_5$ → fill all gaps

predictive value

∴ now, if u_i had not rated I_i it would be empty

now we have predicted rating (B_{10}^T, C_5)

(14) **K-means as MF**

K-means : $D = \{x_i\}_1^n \longrightarrow \begin{pmatrix} S_1, S_2 \cdots S_K \\ c_1, c_2 \cdots c_K \end{pmatrix}$ S_i = sets of cluster i, c_i = cluster i.

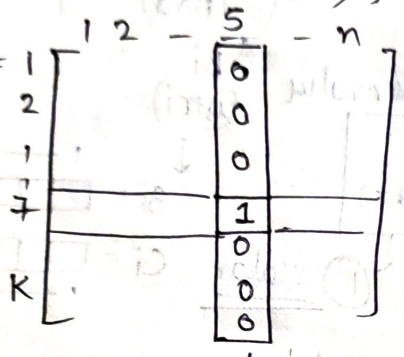
↳ n : no of pts
K : no of centroids

opt^n prob

$$\min_{c_i} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2$$

→ lets take a matrix (Z),

$Z_{(K \times n)}$ (1)



where,

$$z_{ij} = \begin{cases} 1, \text{ if } x_j \in S_i \\ 0, \text{ else} \end{cases}$$

→ $x_5 \in S_j$ and all other ele in col = 0 (each pt ∈ only 1 cluster)

→ assume matrices C and X

$X_{d \times n}$ (2) $= \begin{bmatrix} \uparrow & & \uparrow \\ x_1 & \cdots & x_n \\ \downarrow & & \downarrow \end{bmatrix}$ $C_{d \times K}$ (3) $= \begin{bmatrix} \uparrow & & \uparrow \\ c_1 & \cdots & c_K \\ \downarrow & & \downarrow \end{bmatrix}$

Now,

$$\min_{c_i} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2 \quad \text{?} \quad \min_{z_{ij}, c_i} \sum_{i=1}^{K} \sum_{j=1}^{n} z_{ij} \|x_j - c_i\|^2$$

$$\Downarrow$$

$$\min_{C,Z} \|(X - CZ)\|^2$$

$X = C \cdot Z$ ← (using MF find decompose)

∴ using MF find decompose

$X = C \cdot Z$

where, $z_{ij} = 0/1$

$\sum$(column of Z)$_i = 1$

due to this it is NP-hard

new, optimization

$$X_{d \times n} = C_{d \times K} \cdot Z_{K \times n}$$

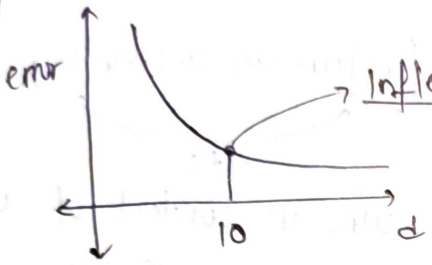
Hyperparam tuning → (d)

$$A_{n \times m} = B_{n \times d} \cdot C^T_{d \times m}$$

$$\text{error} = \sum_{i,j} \left(A_{ij} - B_i^T C_j\right)^2$$

hyper-param = (d) — (1) problem-specific

(2) systematic way : (as d↑ error ↓)



Netflix prize

input : r_{ui} → rating by user (u) on movie (i)

q_i → item vector

P_u → user vector

i = no of items/movies

u = no of users

$r_{ui} \equiv A_{ij}$

opt^n pblm : $\min_{q,p} \sum_{(u,i)} \left(r_{ui} - q_i^T P_u\right)^2 + \lambda\left(\|p\|^2 + \|q\|^2\right)$

(loss fn)

(L2-reg)

cold-start problem

what happens when new user joins? u dont have its data

(sol) use metadata abt user

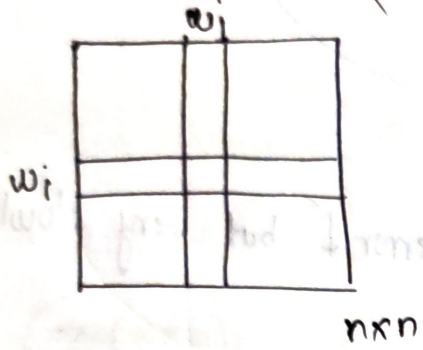
→ ip, browser, os

Word vectors as MF → Co-ocurence matrix + truncated SVD

$D = \{reviews_i\}_1^n$

n = # of words in corpus

Co-ocurence matrix (X)



X_{ij} = # of times w_j ocurs in context of w_i

v.v large [so instead of all (n) word, use (K) most important words]

$K << n$

context: neighborhood (ex = neig = 3)

$r_2 =$ w_j w_2 $[w_i]$ w_3 w_2 w_3 w_4 w_j

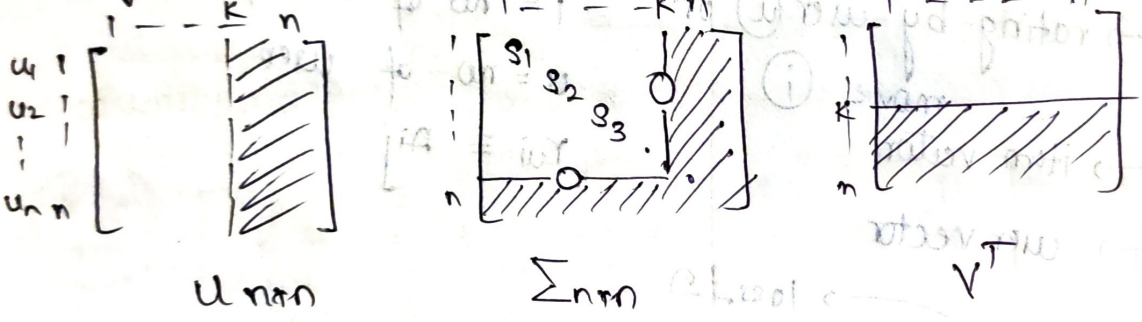
3 , 4+2

$\therefore$ w_j ocurs in context of w_i (1) time in r_2

MF SVD

$X_{n \times n} = U_{n \times n} \cdot \Sigma_{n \times n} \cdot V^T_{n \times n}$ (using SVD)

S_i → singular values



$U_{n \times n}$ $\Sigma_{n \times n}$ V^T

Truncated SVD

*** → discard last columns → $\Sigma_{n \times n}$ pick top (K) singular value

Ex: $U_{n \times n} \equiv \hat{U}_{n \times K}$

$\Sigma_{n \times n} \equiv \hat{\Sigma}_{K \times K}$

$V^T \equiv \hat{\Sigma}_{K \times n}$

(PCA) – pick top 'K' eigen values

$1 \leq K \leq n$

take $\hat{u}$ after truncated SVD → new vectors

$\hat{u} = u_1, u_2 \; - - \; u_n$

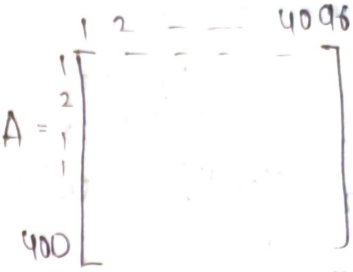
u_i = word vector for word i

final word vector

Eigen faces → PCA on Images

* Dataset → 400 images, each 64 × 64

$① = [r_1 | r_2 | \dots | r_{64}] \Rightarrow A =$



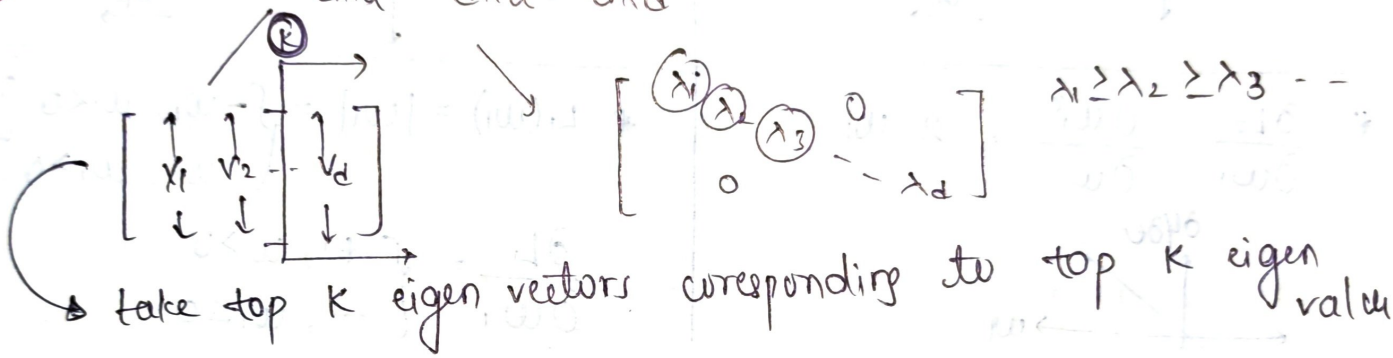
$n \times d$ (4096)

64 × 64 / 4096

Step 1 compute covariance matrix

$A_{n \times d} \longrightarrow Cov(A) = S_{d \times d}$

② $S_{d \times d} = W_{d \times d} \Lambda_{d \times d} W^T_{d \times d}$



③ $\hat{W}_{d \times K} = [v_1 \; v_2 \; \dots \; v_K]$ $K \ll d$

$A_{n \times d} \cdot \hat{W}_{d \times K} = A_{n \times K}$ → K-dim vector for each image

$\therefore$ d-dim → K-dim (PCA)

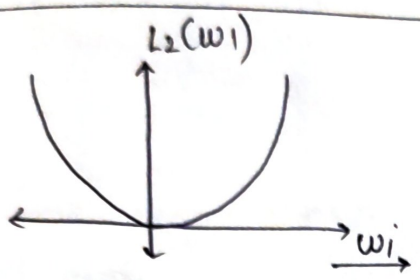
Note as $K \downarrow$ – clarity of photos decreases

why L_1 reg creates sparsity

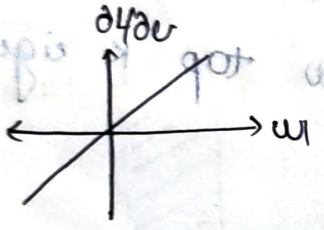
L_2-reg

* $L_2 = \|w\|_2^2 = w^T \cdot w$

$$= \sum_{j=1}^{n} w_j$$

* 

* $\frac{\partial L_2}{\partial w_i} = \frac{\partial w_i^2}{\partial w} = 2 \cdot w_i$



* $w_{new} = w_{old} - r \cdot (2w_i)$

∴ observations

as $w_i < 0$, the gradient / $\frac{\partial L_2}{\partial w}$ also decreases gradually

∴ $w_{new} = w_{old} - (\Delta)$ — decreases constantly

w_{new} keeps ↓ slowly,

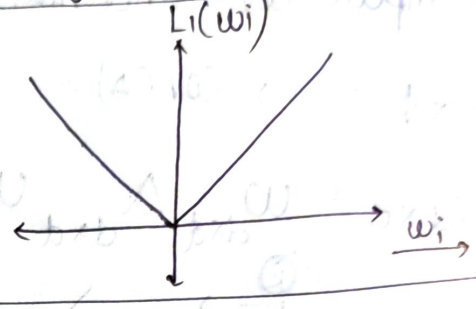
because as w_i ↓, ΔL gradu

Sparsity ie ≈0

L_1-reg

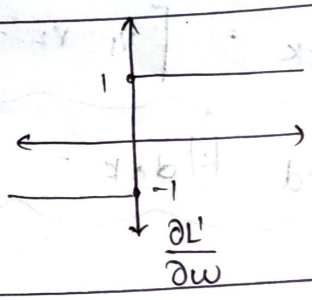
* $L_1 = \|w\|_1$

$$= \sum_{j=1}^{n} |w_j|$$ → absolute value

* 

* $L_1(w_i) = |w_i| = \begin{cases} -w_i & w_i \le 0 \\ w_i & w_i > 0 \end{cases}$

$$\frac{\partial L_1}{\partial w_i} = \begin{cases} +1, & w_i > 0 \\ -1, & w_i \le 0 \end{cases}$$



* $w_{new} = w_{old} - r(w_i)$ → +1 / -1

observations

as $w_i < 0$, the grad does not decreases, it remains constant

∴ $w_{new} = w_{old} - \boxed{\Delta}$

↓ constant

∴ w_{new} keeps ↓ because Δ is constant (i)

SGD with L2 normalization

Logistic regression

Loss fn

$$L = \sum -y_i(\log(\sigma(w^T x+b)) - (1-y_i)(\log(1-\sigma(w^T x+b)) + \frac{\lambda}{2}\|w\|^2$$

wkt,

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z)) \quad \& \quad 1+\sigma(z) = \sigma(-z)$$

$$\nabla_w L = \nabla_w -y\log(\sigma(w^T x+b)) - (1-y)\log(\sigma(-w^T x-b)) + \frac{\lambda}{2}\|w\|^2$$

$$= -y\cdot\frac{\sigma(w^T x+b)(1-\sigma(w^T x+b))}{\sigma(w^T x+b)}\cdot x + (1-y)\frac{\sigma(-w^T x-b)(1-\sigma(-w^T x-b))}{\sigma(-w^T x-b)}\cdot(-x) + \frac{\lambda}{2}\cdot 2\|w\|$$

$$= -y(1-\sigma(w^T x+b))\cdot x + (1-y)(1-\sigma(-w^T x+b))\cdot(-x) + \frac{\lambda}{2}\|w\|$$

$$1-\sigma(-(w^T x+b)) = \sigma(-(-w^T x-b))$$

$$= \sigma(w^T x+b)$$

$$= -xy + xy\,\sigma(w^T x+b) + (x-xy)\,\sigma(w^T x+b) + \lambda w$$

$$= -xy + xy\,\sigma(w^T x+b) + x\,\sigma(w^T x+b) - xy\,\sigma(w^T x+b) + \lambda w$$

$$= -xy + x\,\sigma(w^T x+b) + \lambda w = -x(y-\sigma(w^T x+b)) + \lambda w$$

$$\therefore \nabla_w L = \sum_{i=1}^{n} -x_i(y_i - \sigma(w^T x_i+b)) + \lambda w$$

$$\nabla_w L = -\sum_{i=1}^{n} x_i(y_i - \sigma(w^T x_i+b)) - \lambda w$$

(20) $\nabla_b L = -y\left(\log(\sigma(w^Tx+b))\right) - (1+y)\log(\sigma(-w^Tx-b)) + \frac{\lambda\|w\|}{2}$

$= -y \cdot \frac{\sigma(w^Tx+b)(1-\sigma(w^Tx+b))(1)}{\sigma(w^Tx+b)} - (1+y)\frac{\sigma(-w^Tx-b)(1-\sigma(-w^Tx-b))(-1)}{\sigma(-w^Tx-b)}$

$= -y(1-\sigma(w^Tx+b)) + (1-y)\sigma(w^Tx+b)$

$= -y + y\sigma(w^Tx+b) + \sigma(w^Tx+b) - y\sigma(w^Tx+b)$

$= -y + \sigma(w^Tx+b)$

$= -(y - \sigma(w^Tx+b))$

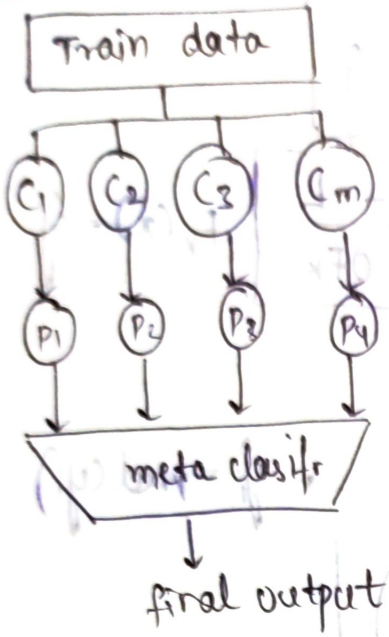
$$\nabla_b L = \sum_{i=1}^{n} y_i - \sigma(w^Tx_i+b)$$

∴ final update eqn,

$$b^{t+1} = b^t + \alpha\left(-\sum y_i - \sigma(w^Tx_i+b)^t\right)$$

$$\Rightarrow \begin{cases} b^{t+1} = b^t + \alpha\left[\sum y - \sigma(w^Tx+b)\right]_{w=w^t,\, b=b^t} \\ w^{t+1} = w^t + \alpha\left[\sum x(y-\sigma(w^Tx+b)) - \lambda w\right]_{w=w^t,\, b=b^t} \end{cases}$$

stacking



Train algorithm :- $\{x_i, y_i\}_1^n$

Step 1 train (m) models (Base classifiers)

using $\{x, y\}_1^n$ train C_1, C_m

Step 2 New dataset (∀) each data point x_i, pass it through each classifier

$D' = \{x', y\}$

$x' = \{h_1(x), h_2(x), \ldots h_m(x)\}$

step 3

train meta classifier using D'

Prediction : x_q

① pass x_q through all base models

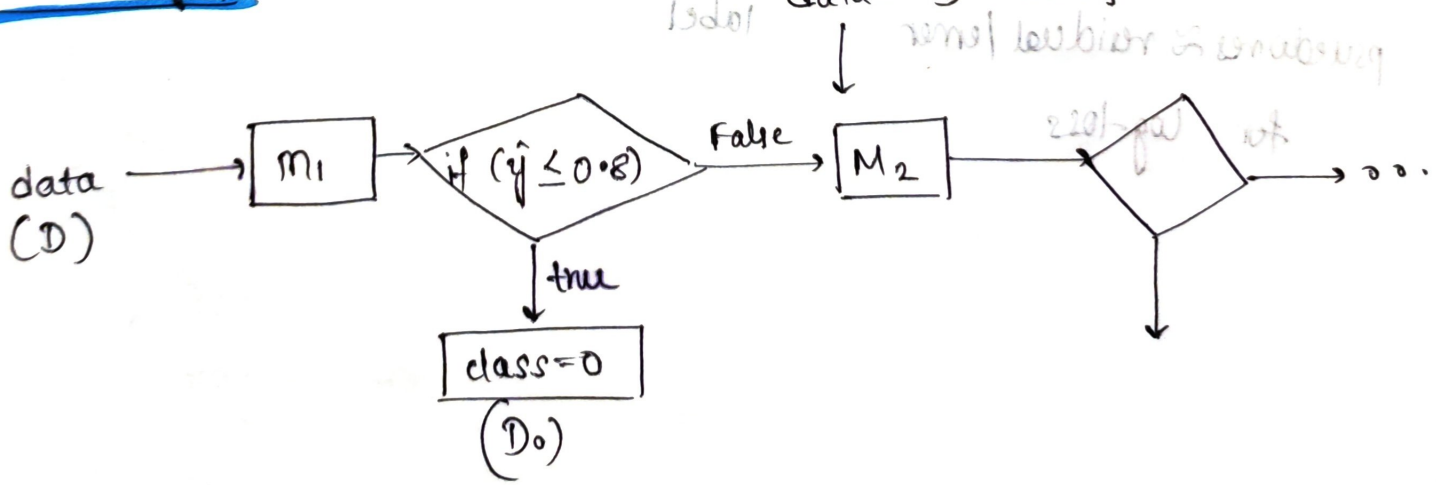
$x_q \xrightarrow{C_1, C_m} \langle C_1(x_q), C_2(x_q), C_3, x_q) \cdots (h_m(x_q)\rangle \longrightarrow$ meta model $\longrightarrow$ res = ŷ

TC

train : tc to train (m) base models + 1 meta model

prediction :- tc to run / predict of (m) base models + meta model

Cascading



(22) **psuedo-res ≡ residual ?**

(1) Squared Loss [refer page (54)]

$$\text{residual} = y - \hat{y}$$

yes psuedo res ≈ residual

$$-\frac{\partial L}{\partial F_k(x_i)} = 2(y_i - F_k) \approx (y_i - F_k)$$

$$\text{psuedo-res} = \frac{-\partial L}{\partial F_k} \qquad F_k(x) = \hat{y}$$

(2) Log-loss

$$L = y_i \cdot \log(p_i) + (1-y_i)\log(1-p_i)$$

$\hat{y}_i = \text{pred}(y_i)$

$$L = y_i \log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1-y_i)\log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

$$\frac{\partial L}{\partial \hat{y}_i} = \frac{y_i - (1-y_i)e^{\hat{y}_i}}{1+e^{\hat{y}_i}}$$ multiply nr and dr by $e^{-\hat{y}_i}$ we get $= p_i - y_i$

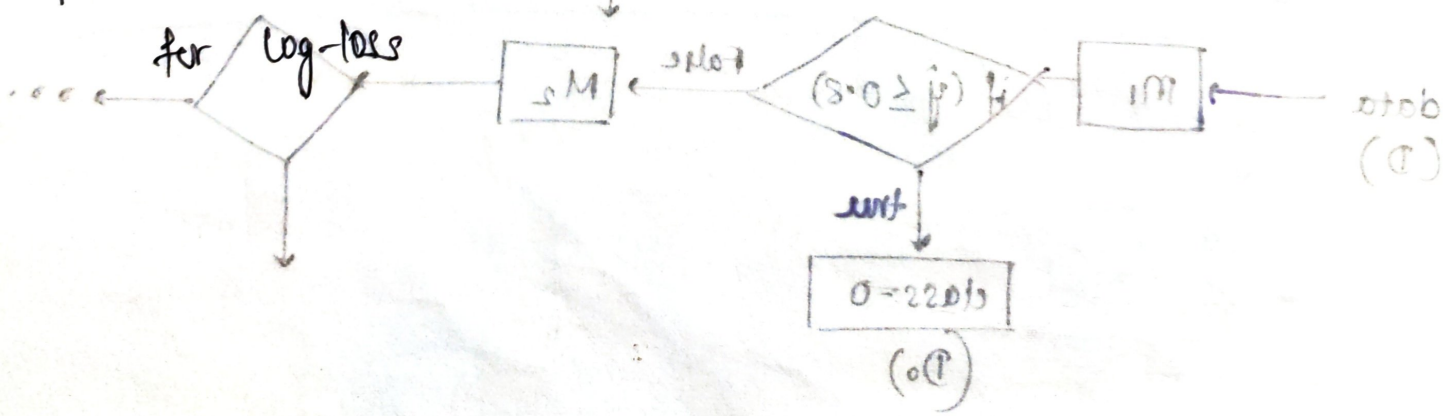
$\hat{y}_i = F_k(x_i)$

$$\Rightarrow \boxed{-\frac{\partial L}{\partial F_k(x)} = p_i - y_i} = \left(\frac{1}{1+\exp(-F_k(x))} - y_i\right) \approx \text{residual}$$

↳ error ↓ actual class label

∴ for bin class, psuedores ≈ residual / error

predicted label



Decision Tree Extras

* It doesn't need standardization as it is not distance based.

d → no of features

TC : Train :- $O(n \log n \cdot d)$ → n: #pts, d: dimensions

sorting for numerical feature

∴ DT ∝ dim

run-TC :- max-depth of decision tree

→ very low ∴ low-latency

Best when
- large data
- small dim
- low-latency

Cases

* Imbalanced data can't be used or handled directly
* as d↑ → train-tc↑ ∴ avoid one-hot-encoding

why? f_1 = state —one→ $f_1 = \langle f_{11}, f_{12}, f_{13} \cdots f_{1n} \rangle$

d↑ → train-tc↑

multi-class

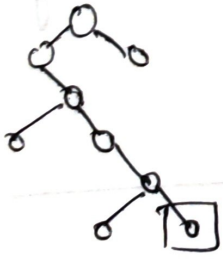
- one vs rest
- Entrophy handles multi-class → at leaf node do majority vote

outliers

→ as depth↑ – outlier impact more

unstable tree

why?



may we have 2 points and they maybe outliers.

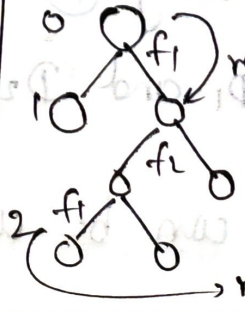
Interpretibility

→ yes = if-else conditions

Feature Importance

FI for (f_i) = reduction in H or Ig

FI(f_i) = normalized sum of reductions in H/Ig due to f_i

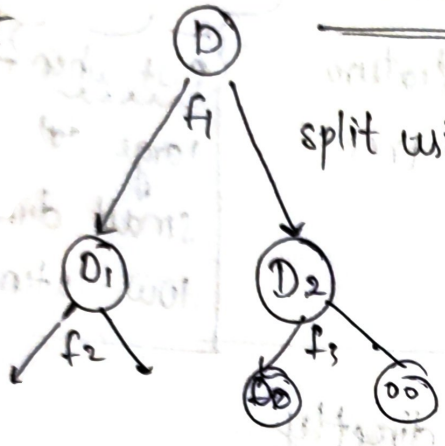


$fI(f_1)$ = Σ reduction at lvl 0, 2

Decision Tree Regression

→ Impurity = MSE

Ex:



$$\Longrightarrow \quad MSE(D) = \frac{1}{n}\sum_{i=1}^{n}(y - y_i)^2$$

split using (f1)

Split using f_1 = MSE(D1) + MSE(D2)

f_2 = MSE(D1) + MSE(D2)

∴ which reduces MSE(D) most is selected as split

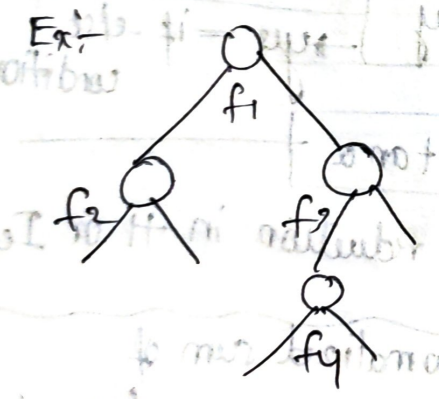
How to predict

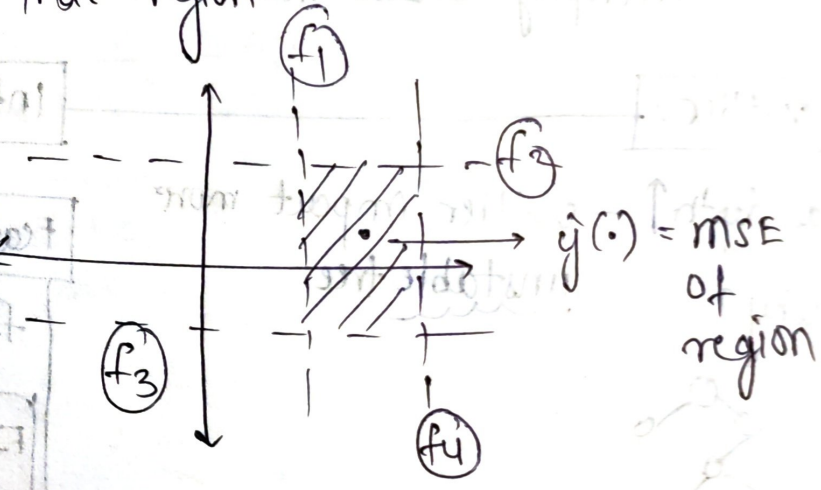
→ given point x_q → follow path and reach ~~not~~ leaf node

at leaf node $\left(\frac{1}{n}\sum_{i=1}^{n}(y-\bar{y})^2\right)_{n=\text{no of nodes in leaf}} = \hat{y}$

∴ prediction ≡ MSE score of that region

Ex:





Important

→ asume D is split using (f1) into D_1 and D_2

→ while splitting D_1 and D_2 we again consider (f1)

∴ one feature can be used for spliting given it reduces entropy

Kernel svm

Normal svm

→ Train phase and get optimal $\vec{w}$

→ Test sign $(\vec{w}^T x + b)$ if +, + class
−, −ve class

Log-R

train : w

test : $p = \sigma(w^T x_q)$
$\Rightarrow 0.5 = c_0$ | $\hookrightarrow c_0$

Kernel svm

Train find best (w)

Test decision fn $= \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_q)$

$\Rightarrow$ sign $(\sum \alpha_i y_i K(x_i, x_q)) =$ class label

$\rightarrow$ Query/test point

$\rightarrow$ Kernel (RBF)

$\sigma(-\gamma \cdot$ euclidean dist$)$

$i = 1 \to n$ is set of all x_i's

sklearn :- (x_i, y_i) dual-coefs

x_i = support-vectors

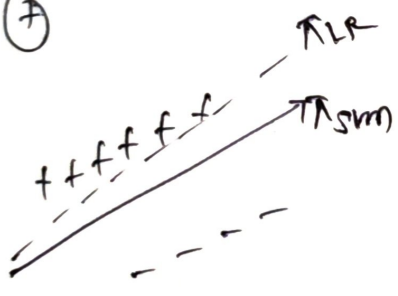
x_q = test point

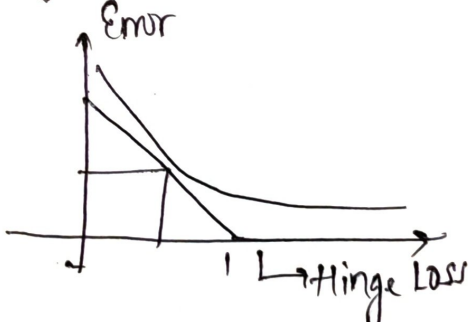
why LR is more impacted by outliers and svm is less

Ex: you have one correctly clasified pt but is very far from π.

In case of LR, π_{LR} changes but π_{svm} doesn't change. why?

(7)





→ In hinge loss, if point is correctly clasified ie $y_i w^T x_i \geq 1$ then error = 0 no matter if $> \infty$

→ but in log-loss, although error ≈ 0 but not exactly $= 0$

Conf matrix terms

* Sensitivity :(TPR) $= \frac{TP}{TP+FN}$ [proportional of positives correctly identified as +ve

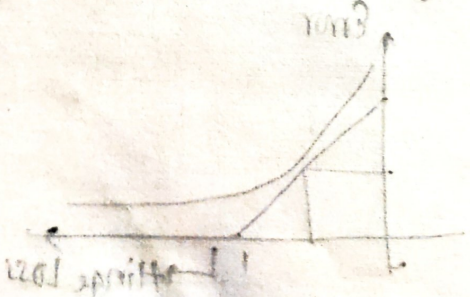
* specificity :- (TNR) = proportional of negative of correctly identified as (-ve)
ie out of all (-ve) how many (-ve) are correctly classified

sensitivity = recal = TPR

specificity = 1 − FPR

* $TPR = \frac{TP}{P} = \frac{TP}{TP+FN} = 1 - FNR$

* $TNR = \frac{TN}{N} = \frac{TN}{TN+FP} = 1 - FPR$



ALGO DEEP DIVE

KNN

Train : No train

Test : $O(nd)$
`K` is small

space : $O(nd)$
store all train dataset

Limitations
* you need large space complexity
* TC is also very huge ≡ High Latency

Feature - Importance
* Forward feature selection

KD-Tree → Helps to speed up nearest neighbor finding

No of comparisons : Best − $O(\log n) \cdot K$
worst − $O(n) \cdot K$

→ check all adjacent boxes

→ problem : for dimension 2 → 4 checks [generally (d) is small size
* d-dimensions → 2^d checks

TC : $O(2^d \cdot \log n) \equiv O(n \log n)$
→ due to adjacent checks

* data must be uniformly distributed
else TC ≠ $O(\log n)$.

LSH

* Train TC : $O(m \cdot dn)$ [m - no of hyperplanes]
you need to take each point and check ∀ hyperplane on which side it lies → | 0 | 1 | 0 | 1 |
π_1 π_2 .. π_m

* Query : $O(md + n'd)$

ip : (q)
you need to find
$\text{sign}(w^T x q)$ ∀ $w \in \pi_m$

if the hash-value / bucket has n' elements, you need to traverse each to compare (maybe find k-nearest).

36 Naive-Bayes

* Bias-variance in Laplace smooth

$\alpha = 0$: you give prob to even rare words → overfit

$\alpha \uparrow$: $p \to 1/2$ → random model

∴ underfit

* outliers

① test
new word — laplace smooth

② train
if word$_i$ occurs less than a certain threshold (z) ignore
→ laplace smooth with high α value

Cases

① large dimensionality text
→ can handle text with high dim
→ use log-probabilities

② good fr high-dim text data + categorical features
↳ used as baseline model

③ Not used much fr real data

④ Prone to overfit when laplace smoothing is not used

⑤ Highly interpretable

Assumptions
* conditional dependence of features

Time complexity

Train → $O(n)$
Test → $O(nK)$ (K → no of classes)
space → low

logistic-regression

2 1/2

* we used distance :. standardization is mandatory
* if multi-colinearity → perturbation test
  - cant use |wi| as feature importance

Time complexity

* Train :- sklD — O(nd)
* Test/run time → only w is needed
  - space — O(d)
  - time ← O(d) → $\sum_{i=1}^{d} w_i x_i$

* Very good for low latency apln
  - → Internet companies

perturbation test

* D_T D_t → add small ∈ to each x_i
  - → Train
    - $w \rightarrow \langle w_1, w_2 \ldots w_n \rangle$
    - $w' \rightarrow \langle w_1', w_2' \ldots w_n' \rangle$
    - if w_i and w_i' differ significantly ⟹ they r colinear or multicolinearity

* if d is large (≈1000)
  - → use L1 regularization which would make non-useful features = 0

Asumptions

1. Linearly seperable
2. ! colinearity

Best Case

* linearly separable data
* low-latency required
* low TC

Outliers

→ reduces impact

Handling?

* fit hyperplane
  - ↓
  - remove very far pts
  - ↓
  - fit again

large d

↑ more chances of finding optimal π^* but as d↑ the latency also increases.

LR = log-loss + λ||w||2

- λ↑ → underfit
- λ↓ → overfit
- ← λ↑ → sparsity↑ → lat↓ (L1 reg)

(25) **SVM** — Support vector machines

Time Complexity → $O(n^2)$ is large

Train → SMO
Sequential minimal optimization

* Train $\approx O(n^2)$ for kernel SVM
* optimized, $O(nd^2)$ if $d < n$

Run → $O(kd)$
↳ no of support vectors

* not used when (n) is large

Outliers
* very less impact — depend only on

worst case
* large (n) → high latency
↳ no of support vectors > large

Problems
* No Interpretability esp Kernel SVM

Decision Tree

* Scaling is not needed (it is not distance based method)

Categorical with many cat. → mean encoding / pick-top

cat → numerical feature

$P(y_i == 1 | P_j)$

P_i	y_i
P_1	0
P_2	0
P_3	1

$P(y=1|P_i)$ = # times you have P_i and $y=1$ / # of P_i

Overfit / underfit

depth ↑ → overfit (you can grow tree to fit even noisy points)

How predict?

At each leaf you can have → only one point — pure node

→ multiple nodes — majority vote

Time complexity

Train : $O(n \log n \cdot d)$ — d – at each node you need to check for all feature for info gain

$n \log n$ — mostly when numerical attributes

space :- traindt → nested if else — $O(\#\text{internal nodes} + \#\text{leaf nodes})$

runtime : $x_q \rightarrow y_q$

$O(\text{max-depth of any leaf node})$ generally depth <

Cases

good when ← low latency, large data, low dimensions

40 Heirarchical Clustering

space :- $O(n^2)$

time :- $O(n^3)$ → n iterations
- at each iter you may need to join 2 clusters
- update similarity matrix

Limitations

* No clear objective fn / optimization pblm
* MIN – outlier pblm

  MAX – breaks large clusters, can't accomodate different sized clusters
* space and time complexity

DBSCAN

Pros
* Resistant to noise
* handle diff cluster size, shape
* no need to specify (K) value

Cons
* not entirely deterministic
* depends on distance measures
* cannot cluster datasets with large difference in densities