Topics:

- clustering
- Dive deep into K-means + variations

Ques

Naive Bayes

(1) $x_{qv} = w_1\ w_2 \cdots w_k$ ($O(k)$) — # words in x_{qv} NOT train data

Run time compl = $O(k)$

Precomputed from Dtrain:

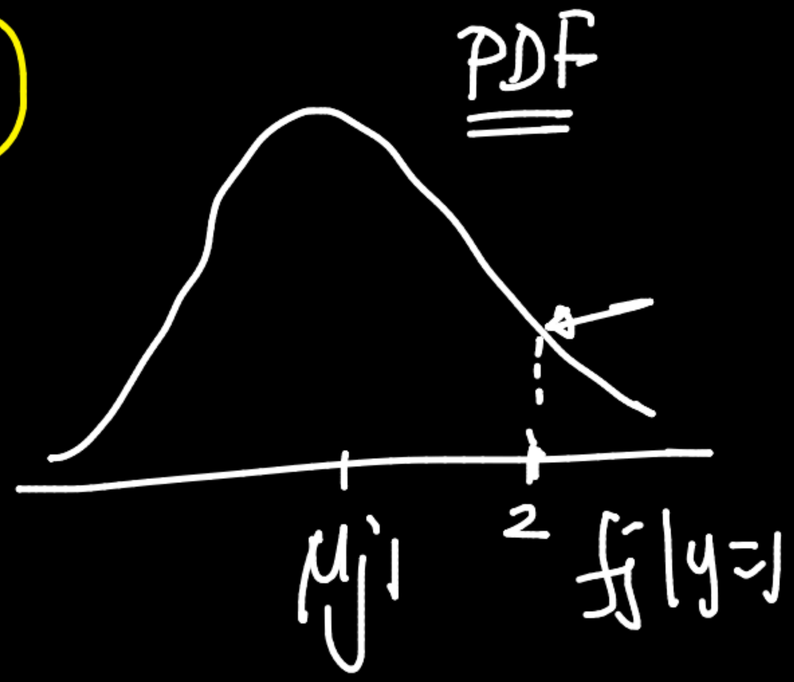
- $P(y=1)$; $P(y=0)$
- $P(w_i|y=1)$; $P(w_i|y=0)$

key = $i,1$ val = ?

② Gaussian NB:-

$P(f_i = 2 \mid y = 1)$ → PDF

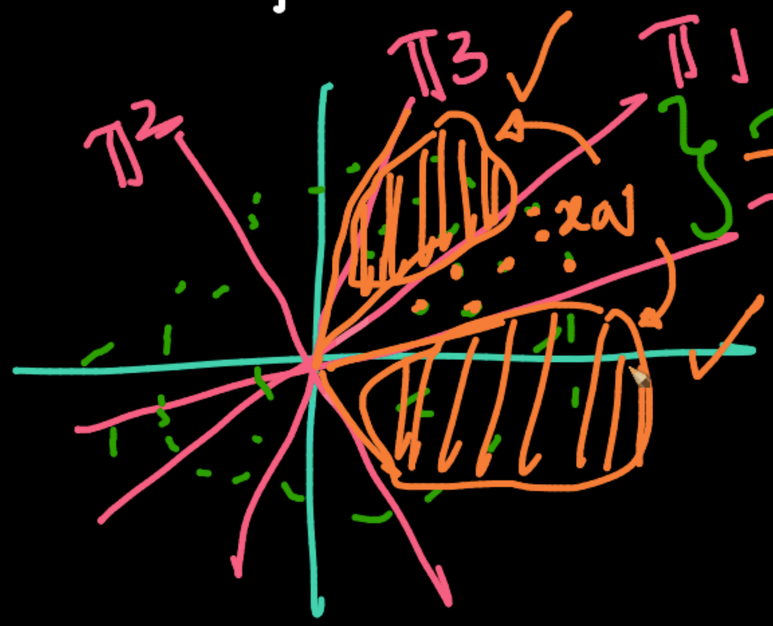
↳ $N(\mu_{ji}, \sigma_{ji})$



(Q)

LSH: Cosine-Similarity (intuition)

K > #points in a Segment



K=7

[-1, +1, -1, +1]

Clustering

Classification

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

Regression

$y_i \in \mathbb{R}$

Supervised

y_i exists

e.g.: 100 MM Amazon Customers

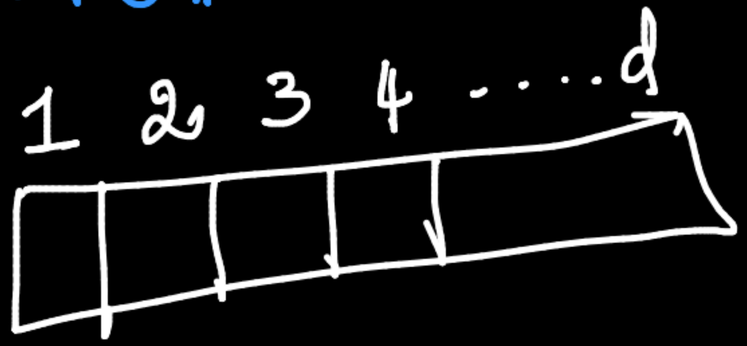
10M — C_1

8M — C_2

⋮

$$\mathcal{D} = \{ x_i \}_{i=1}^{n}$$

$x_i \in \mathbb{R}^d$



no y_i

'groups' similar Customers

e.g: 100k english words → Cluster similar words

1M newspaper articles → group similar articles

{ KNN & cosine-similarity ← k-Means

✓ [Unsupervised Learning :- no y_i's

↳ Clustering

PCA : dim-reduction ✓

$x_q \rightarrow \hat{y}_q$

✓ { Semi-supervised Learning : } Dtrain → some pts have y_i's, many do NOT have y_i's

Self-Supervised Learning : → Images (CV & DL)

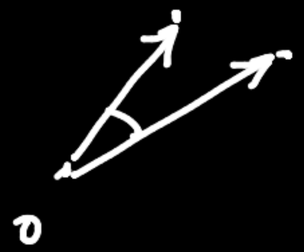
Practical

Rule-of-Thumb

Euclidean - dist → low-dim

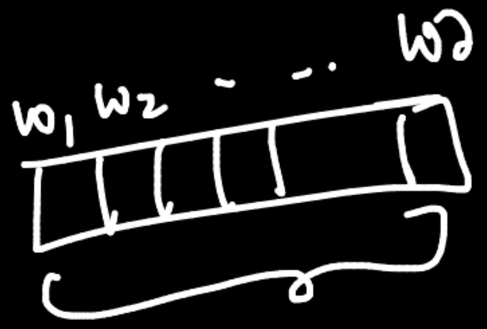
Manhattan dist → low-medium dim

Cosine sim → high-dim

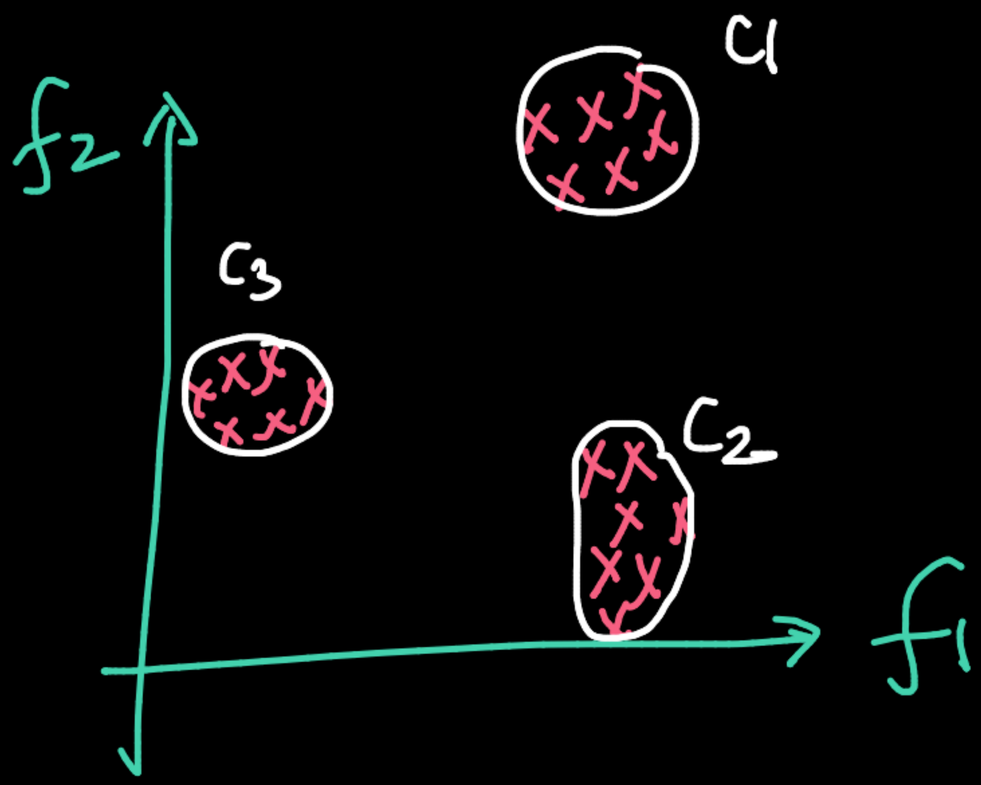


↳ Text-data BoW

sparse



Practically: Distance-metric is a hyper-param

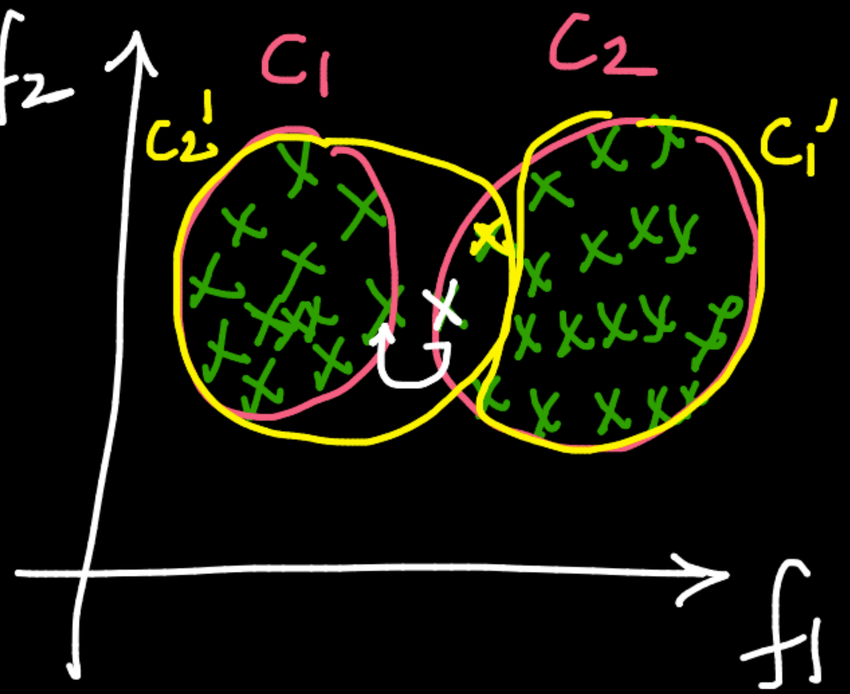


ideal-case

INTUITION

- points in a cluster are close to each other
- points in different Clusters are far from each other

exceptional cases



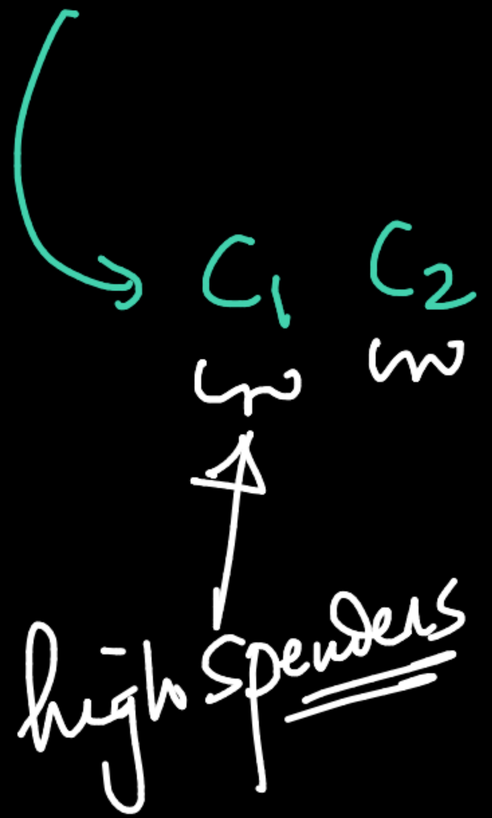
- different algorithms
- no ground truth

cluster output must make biz sense

IDOMM Amazon

$x_i \in \mathbb{R}^2$

Special Deals



Input: $\{(x_i)_{i=1}^{n} ; x_i \in \mathbb{R}^2\}$

output: $C_1, C_2 \cdots C_K$

$\{ \} \quad \{ \} \quad \{ \}$

x_{new} → which cluster does x_{new} belong to?

Personas e-commerce

n x_i's

→ biz-sense

↳ Supervised

Euc. dist; Manhattan dist; Cosine-sim

↳ clustering

metrics → common-sense

↳ DUNN-Index

Metrics

Classfn → AUC; Pr, Re, f1 ...

Reg → R^2

Metric

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

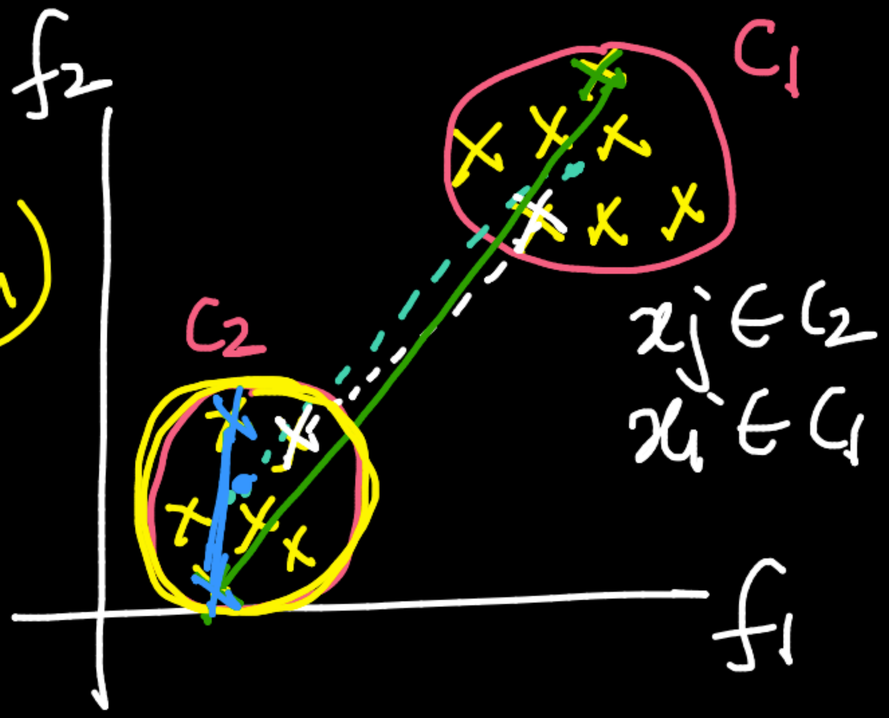
dist b/w cluster centres

inter-cluster dist (↑ high)

- min dist
- max dist

intra-cluster dist (↓ low)

- ↳ avg dist from center
- ↳ max dist



$$\min d(x_i, x_j)$$
$$\text{s.t } x_i \in C_1$$
$$x_j \in C_2$$

k-clusters

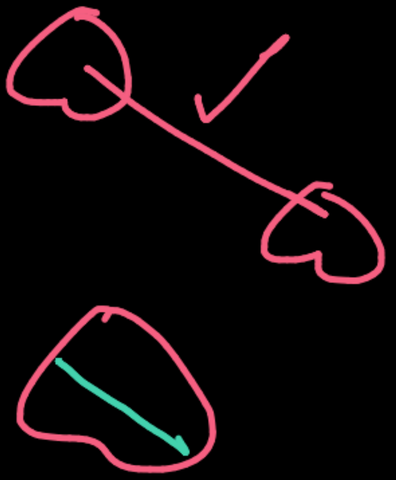
Metric

$$\text{DUNN-Index} = D = \frac{\min_{i,j} d(i,j)}{\max_{k} d'(k)}$$

$d(i,j)$ → inter-cluster dist

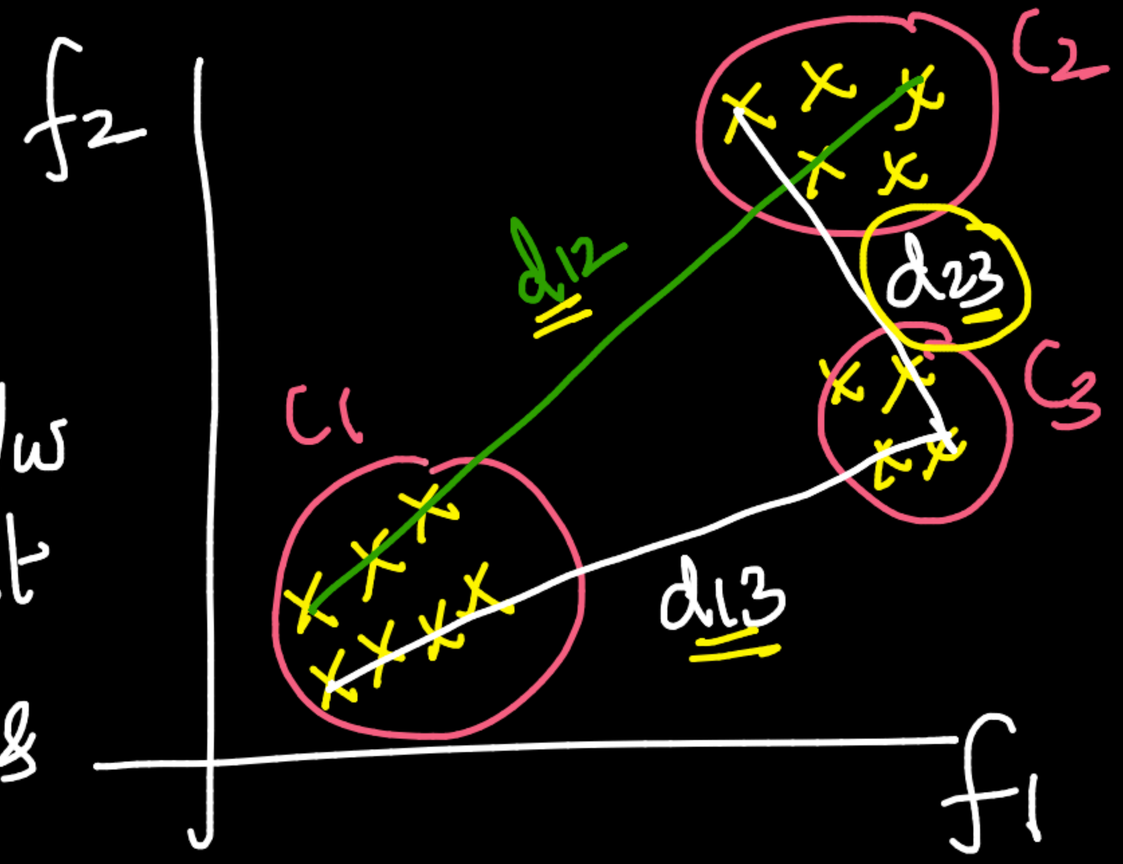
$d'(k)$ → intra cluster dist

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



Inter-cluster distance

$d(i,j)$ = distance b/w the farthest pts in C_i & C_j



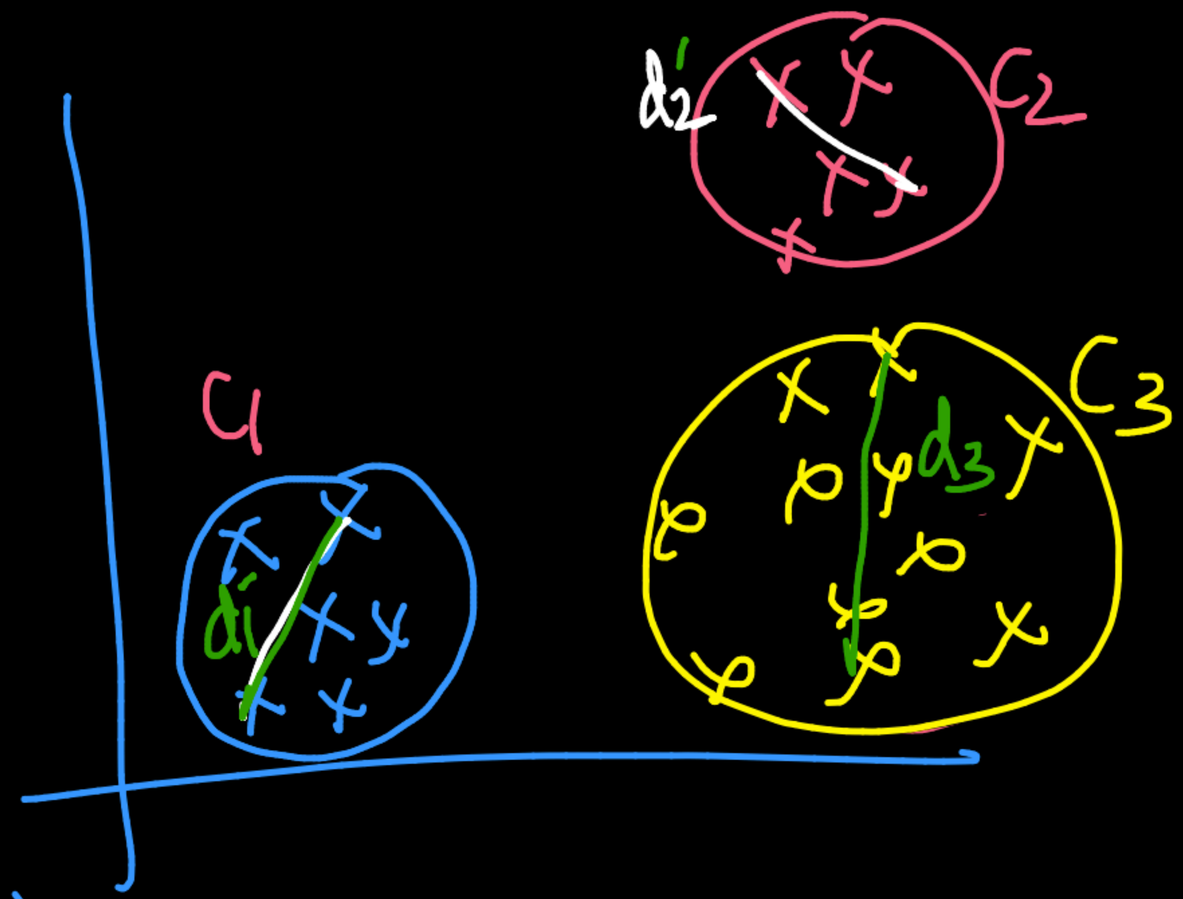
{ k-clusters
k_{C_2} distances

Min (d_{12}, d_{13}, d_{23})

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:59

Intra-clust dist

$\max(d_1', d_2', d_3')$

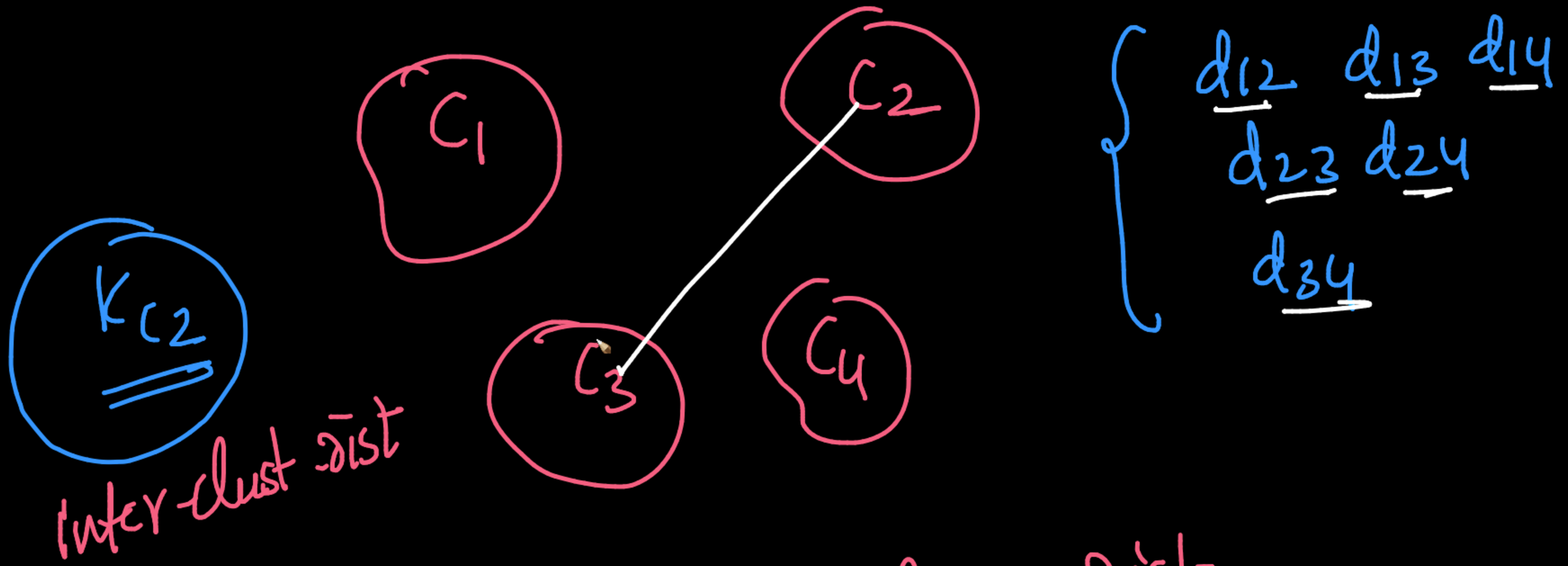


$d'(i)$ = max dist b/w 2 points in C_i

21 / 21

Design your metric

$$\frac{\text{intra-clust dist}}{\text{inter clust dist}}$$

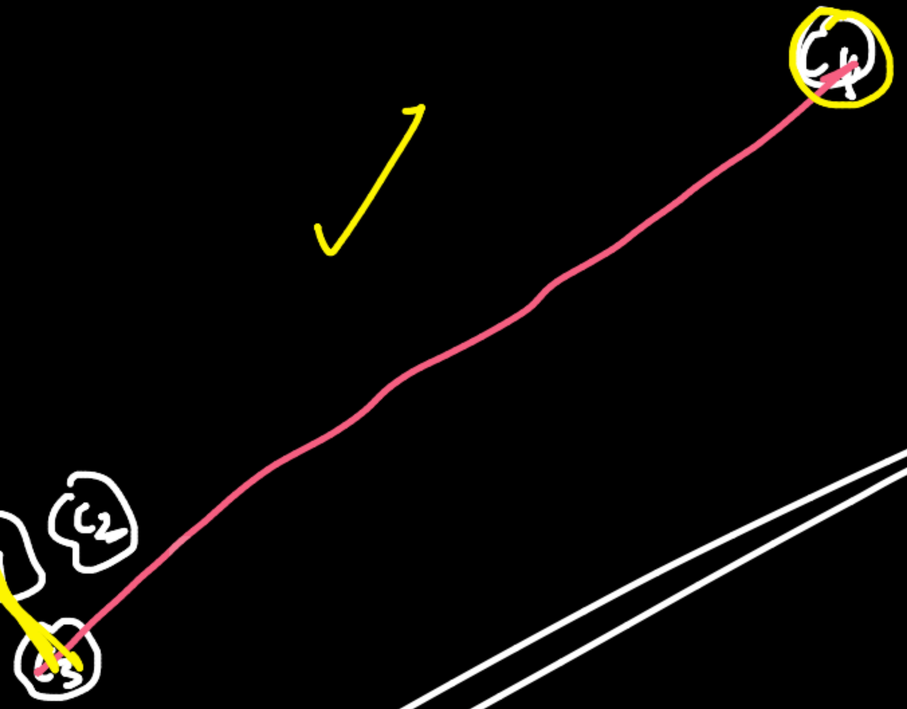


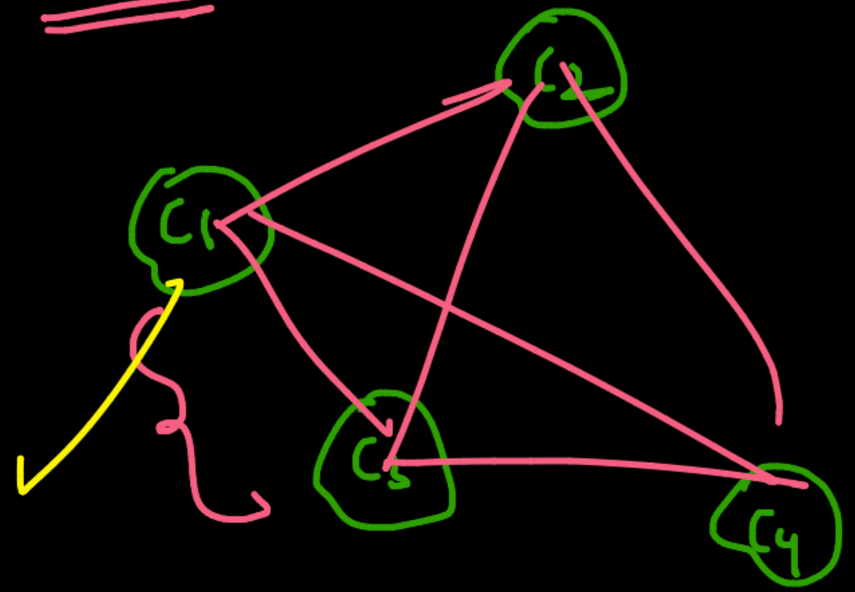
Case 1:

Max dij

Case 2:

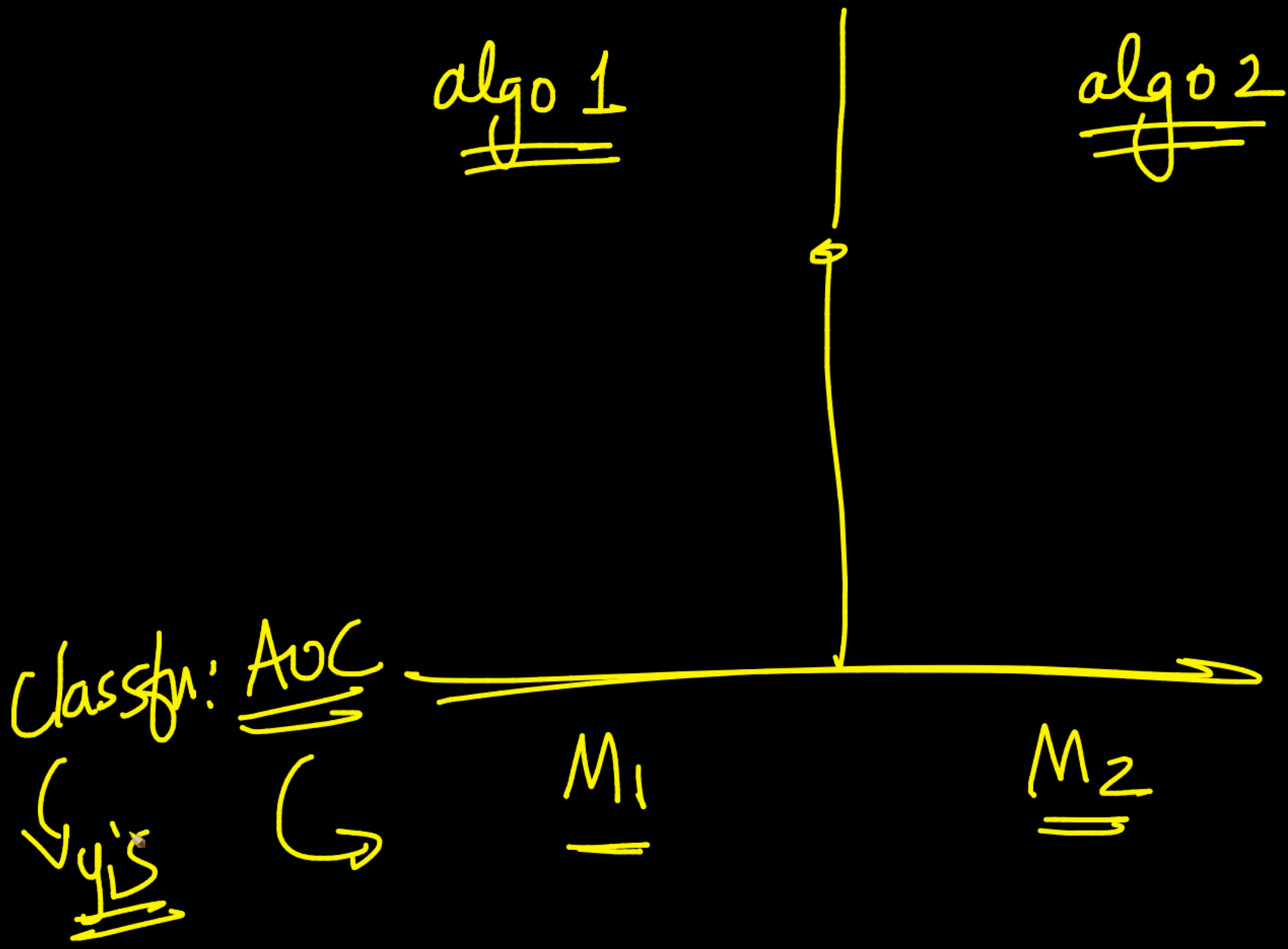
Min dij

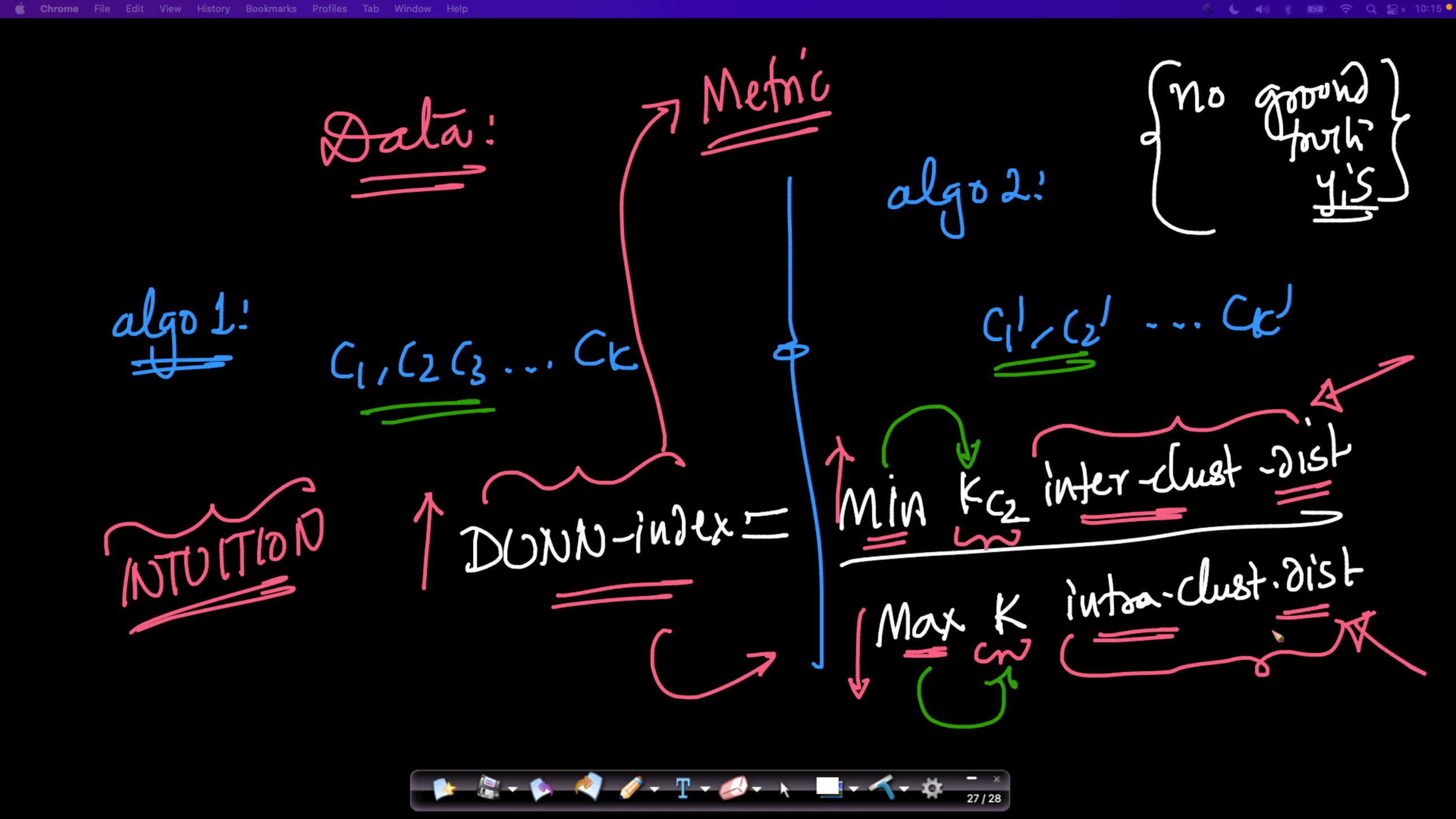




if metric :

$$\frac{\min_{i,j} d(i,j)}{\max_{k} d'(k)}$$




Data:
Metric
algo 1:
algo 2:
{no ground truth y's}
$C_1, C_2 C_3 \ldots C_k$
$C_1', C_2' \ldots C_k'$
INTUITION
DUNN-index =
Min $_{k_{C_2}}$ inter-clust-dist
Max $_{k}$ intra-clust.dist
27 / 28

DUNN - Index → Metric

→ not an optimization problem

Q

$1 < DUNN\text{-}Index = \frac{\min \partial_{ij}}{\max d'(k)}$ → $+\infty$

$\mathcal{D} = \{x_i\}_{i=1}^{n}$

$C_1 C_2 \cdots C_k$ (one-clustering)

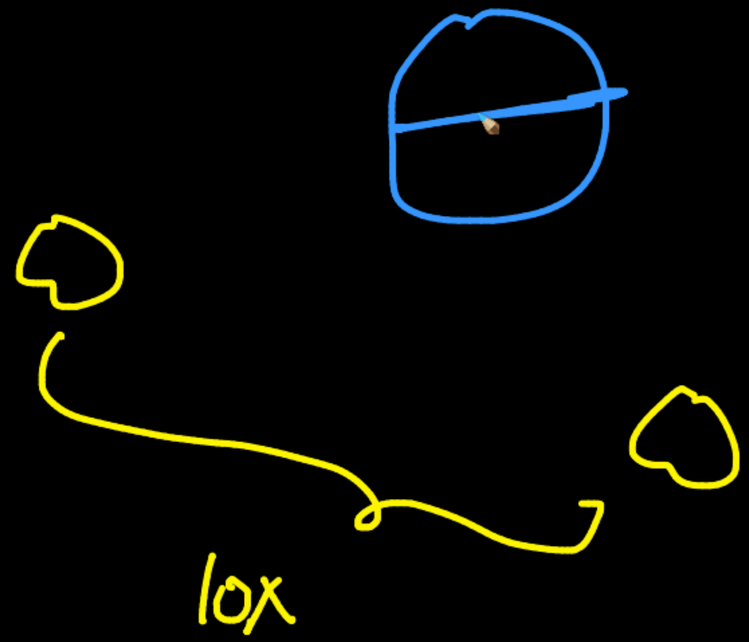
val 1 = 10 (let)

HINT: AUC; — M_1: 0.7

random: 0.5

ideal - 1

$$\frac{\text{Min inter-clust dist}}{\text{Max intra clust dist}}$$



Random clustering $\mathcal{D} = \{x_i\}_{i=1}^{n}$

baseline clustering
(K-means)

C_1 C_2 ... C_k

↓

DUNN-Index

Clustering :-

Input : n d-dim pts,
#clusters = K

Output : K-clusters
$C_1, C_2 \ldots C_K$

x_{new} → NN-like strategies

Chrome File Edit View History Bookmarks Profiles Tab Window Help

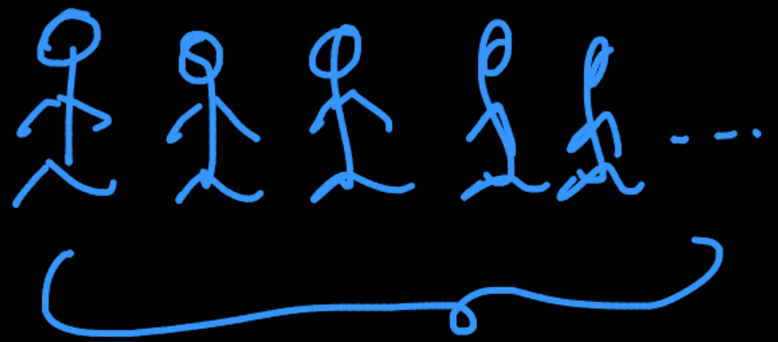
Clustering:

unlabelled

10 Million images

each image → 2 dim vec

label → 10 classes



100 - clusters

50k

1000 pls

Mens-jeans

deploy (later)

Smart-label

K-means

- Simple; popular; baseline
- basic-variant

#clusters

Input:- n d-dim pts, K

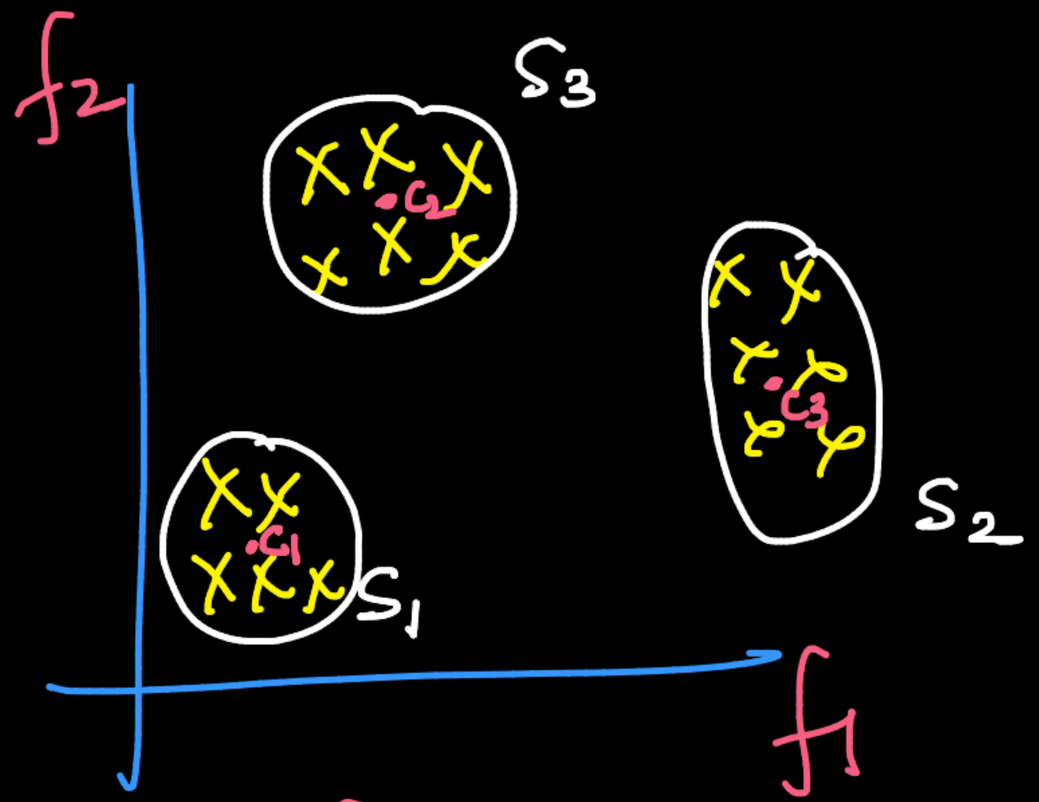
- Geom
- optimzation
- Lloyd's algo
- Kmeans ++
- k Means ..

$S_1 \cup S_2 \cup S_3 = \mathcal{D}$

$s_i \rightarrow c_i$

✓ $s_i \cap s_j = \phi$

each $x_i \in$ one s_j



means
$\in S_1$

$$\text{Centroid}_1 = \frac{x_1 + x_2 + \cdots + x_m}{m}$$

Centroid $\{ c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ = mean vector

$|S_i|$ → size/cardinality of set

k-means: Centroid-based approach

(later) Alt: hierarchy → Agglom...
density → DBSCAN

find $s_1, s_2 \ldots s_k$

$c_1, c_2 \ldots c_k$

Chrome File Edit View History Bookmarks Profiles Tab Window Help

optimization

$$\mathcal{D} = \{x_1, x_2, \ldots x_i \ldots x_n\} \quad \text{each } x_i \in \mathbb{R}^d$$

Find :- $S_1, S_2 \ldots S_k$ s.t

$c_1 \; c_2 \ldots c_k$

(max inter clust dist ← between)

and

(min intra clust dist ← within)

✓

38 / 39

GD

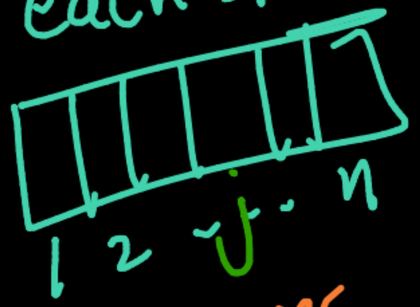
min $S_1, S_2 \dots S_K$

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

loss

min sum of within Cluster distances

each S_i



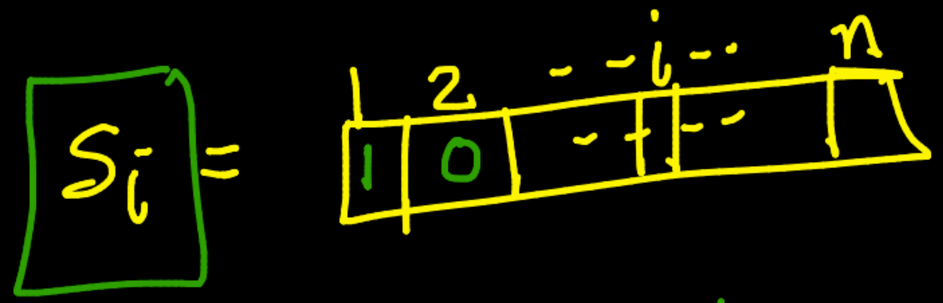
binary-vec

s.t each $x_j \in S_i \rightarrow \sum_{i=1}^{K} S_{ij} = 1 \quad \forall j : 1 \to n$

$S_i \cap S_j = \phi$

$i \neq j \rightarrow S_i^T S_j = 0 \quad \forall i \neq j$

Centroid $\{ c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$



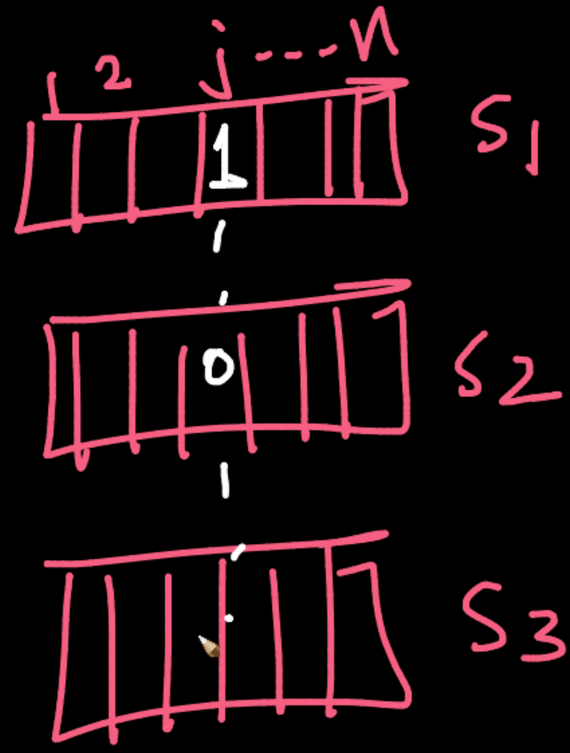
if $x_1 \in S_i$
$x_2 \notin S_i$

0/1 binary vector of size - n

$s_i \cap s_j = \phi \Rightarrow s_i^T s_j = 0 \quad \forall i,j$

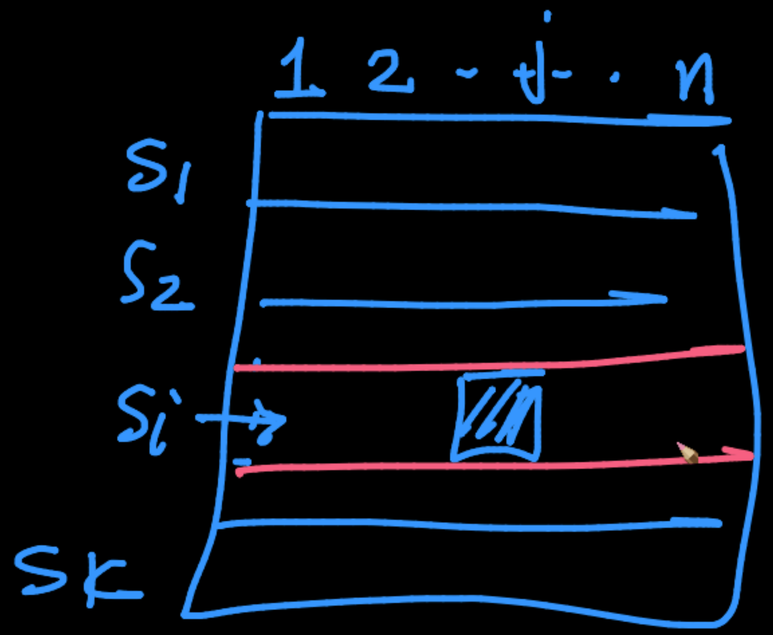
some

each $x_j \in s_i$



$$\sum s_i = \boxed{1 \mid 1 \mid 1 \mid \cdots \mid 1}$$

$$\left\{ \sum_{i=1}^{k} s_{ij} = 1 \quad \forall j : 1 \to n \right.$$



Sij

opt!

$$\mathcal{L} = \widetilde{Loss} + \lambda_1 \square_{Si's} + \lambda_2 \square_{Si's} \cdots$$

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial s_{ij}} \neq 0; & (s_{ij})_{new} = (s_{ij})_{old} - \eta \dfrac{\partial L}{\partial s_{ij}} \end{cases}$$

Can we solve K-means using GD

- very clean formulation

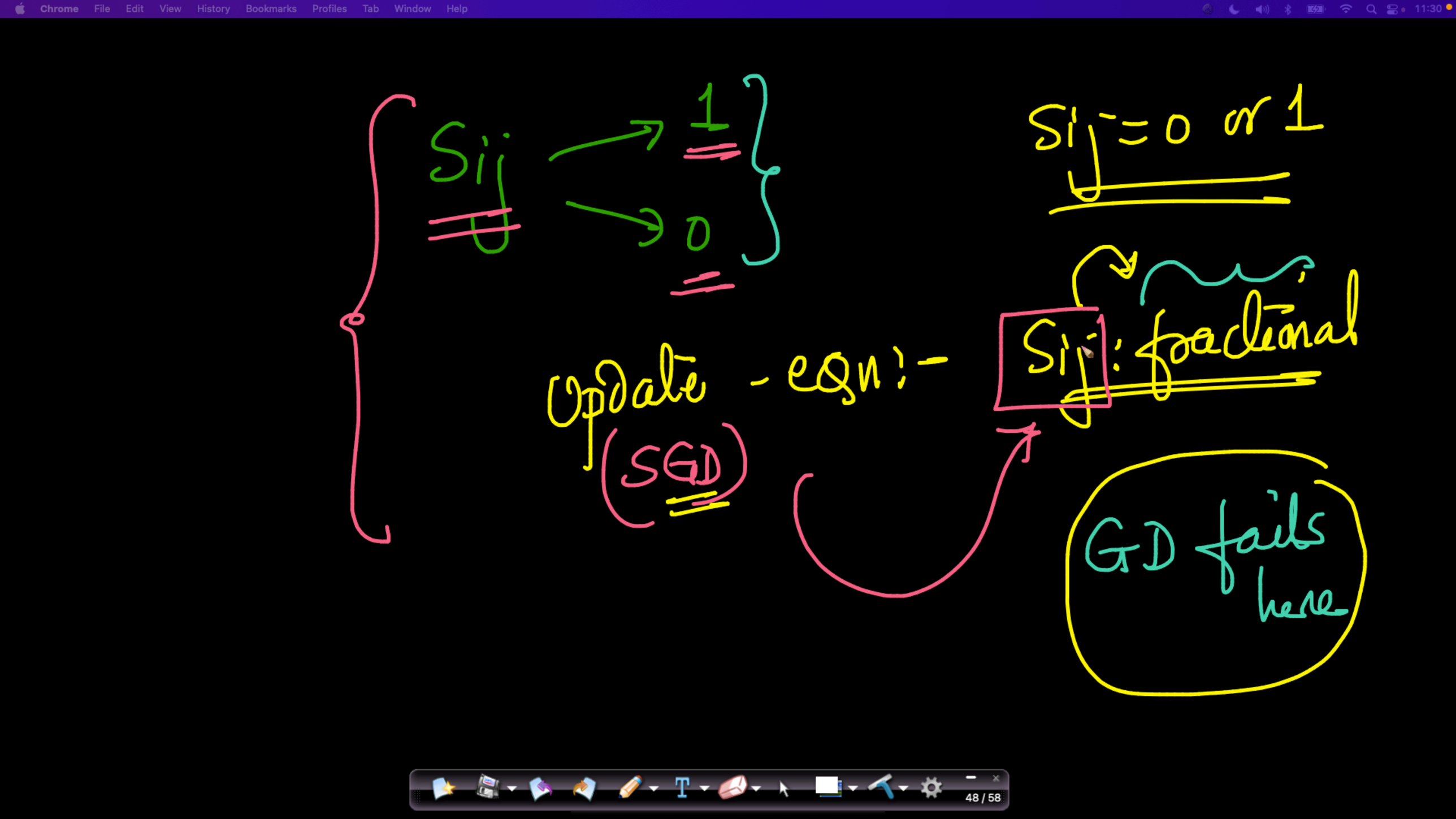
Yes, No

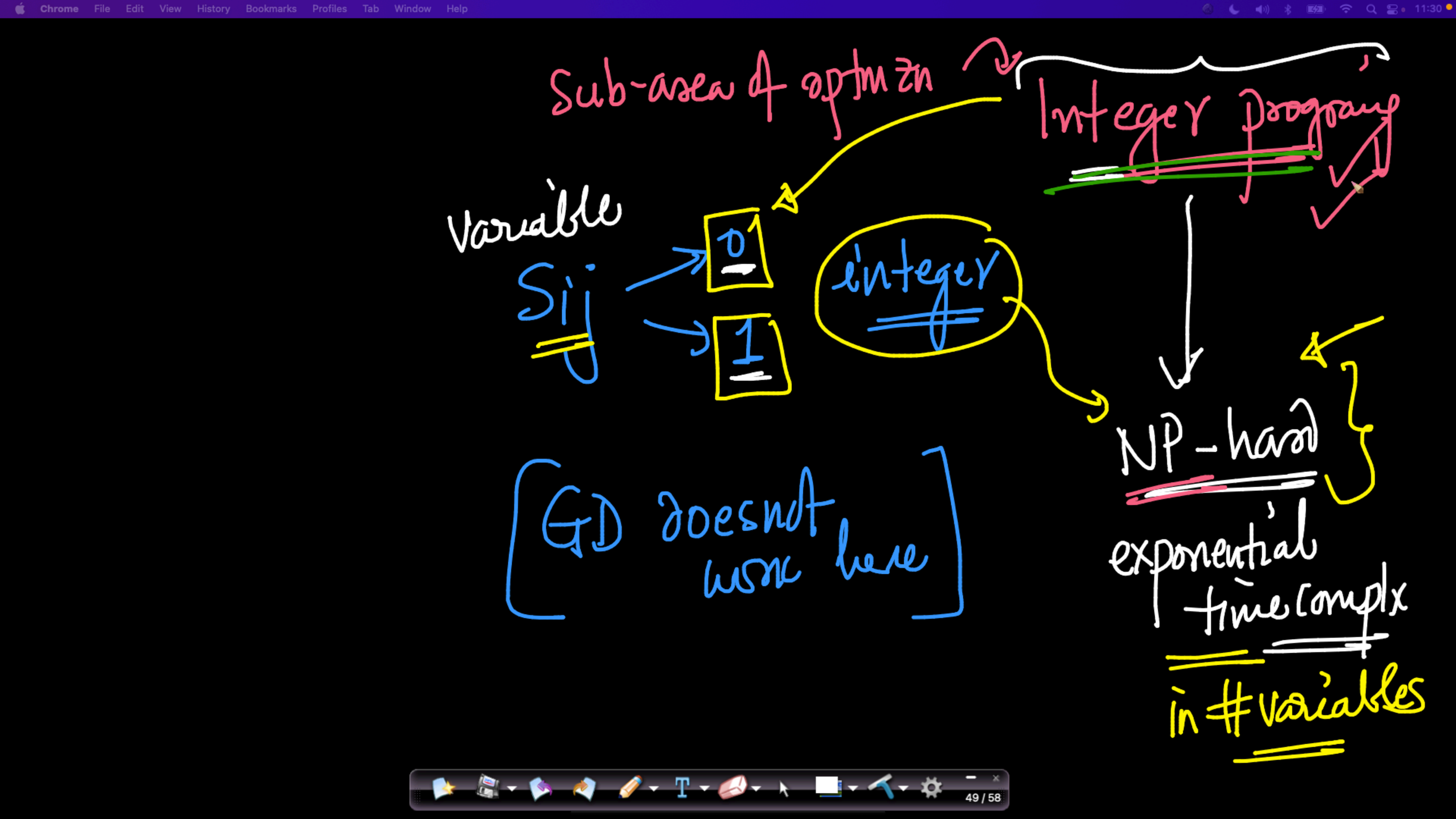
✓ PCA: where was y_i's → GD

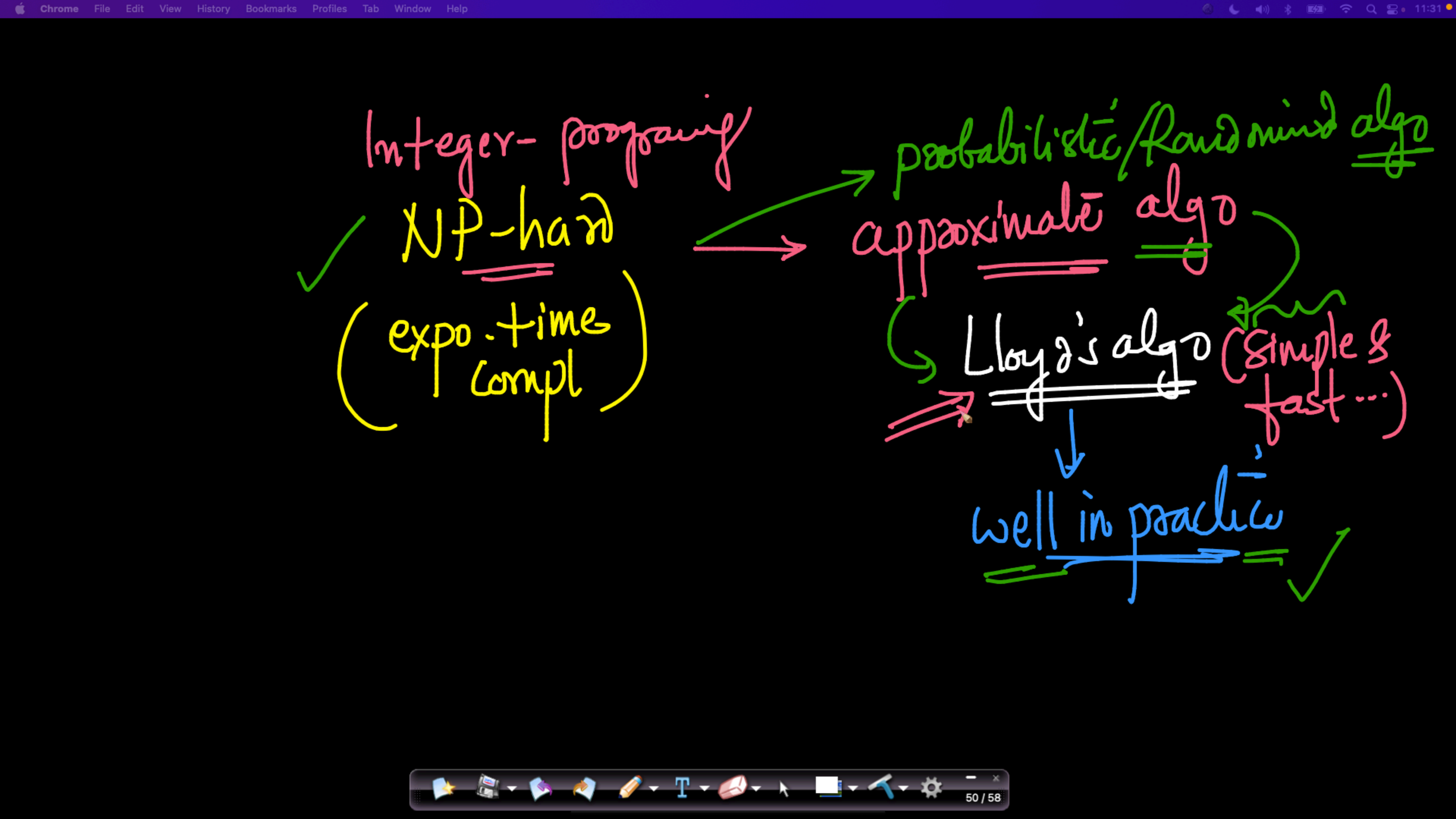
✓ when to stop: $\left|\frac{\partial L}{\partial s_{ij}}\right|$: v-small $\forall i, j$

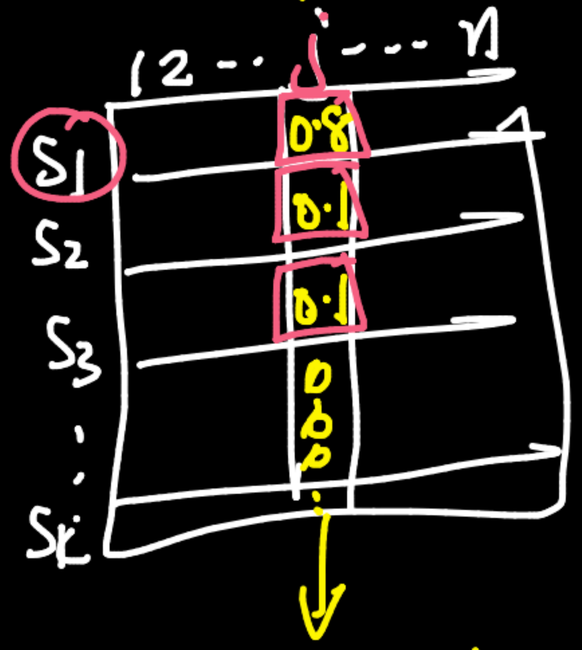
$$\checkmark\ s_{ij} = \begin{cases} 1 & \text{if } x_j \in s_i \\ 0 & \text{otherwise} \end{cases}$$

$\checkmark$ centroid $c_i \rightarrow s_i$


Sij
1
0
Sij = 0 or 1
Update - eqn :-
(SGD)
Sij : fractional
GD fails here


Sub-area of optmzn
Integer Programg
Variable
Sij
0
1
integer
NP-hard
exponential time complx
in # variables
[GD doesnt work here]


Integer-programming
NP-hard
(expo.-time compl)
probabilistic/Randomized algo
approximate algo
Lloyd's algo
(simple & fast...)
well in practice



fractional belonging to a cluster

Optimization problem

solve it GD

harder to solve than Lloyd's algo

Fuzzy clustering

soft-clustering

Lloyd's algo

distance-metric

Input :- $\{x_1, x_2 \dots x_n\} = D$

K = #clusters

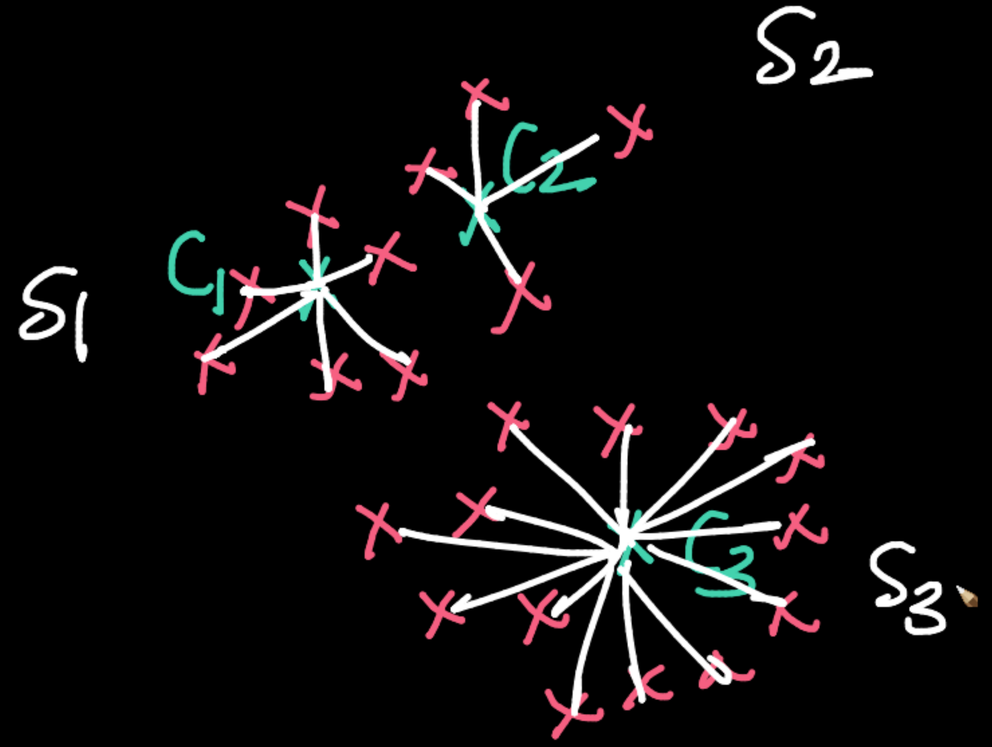
(1) Initialization :- pick K-pts randomly from the n-pts

$(c_1, c_2 \dots c_k)$ → initial centroids

(2) Assignment

for each x_i in D

- Select the nearest centroid: $\tilde{c}_j$ (let) → dist-metric
- add x_i to S_j



(3) Recompute / update the Centroids

for j in 1 to k

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

means ⇒ prone to outliers

mean-pt

Lloyd's algo

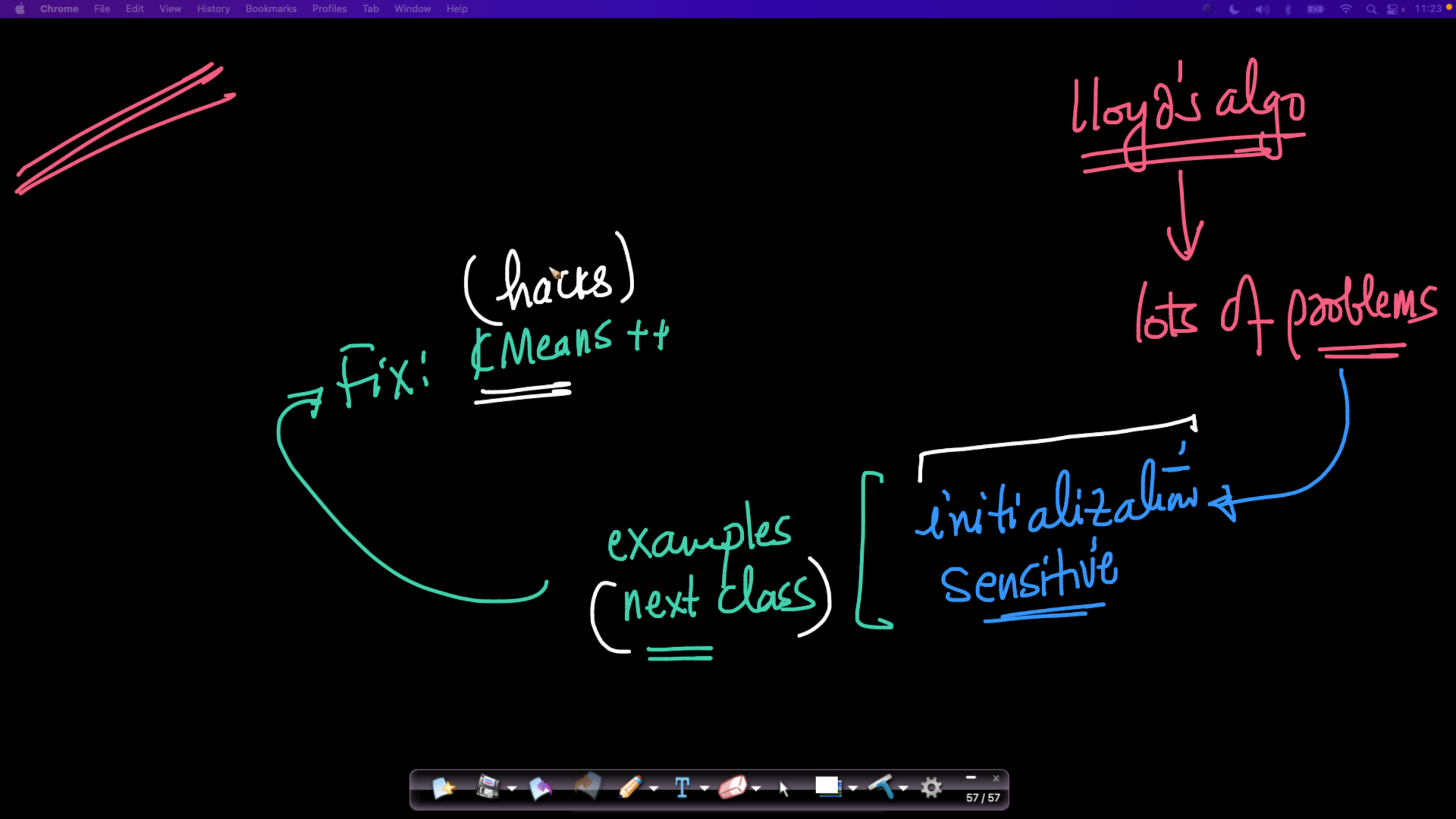
↓

lots of problems

Fix: kMeans ++

examples next class

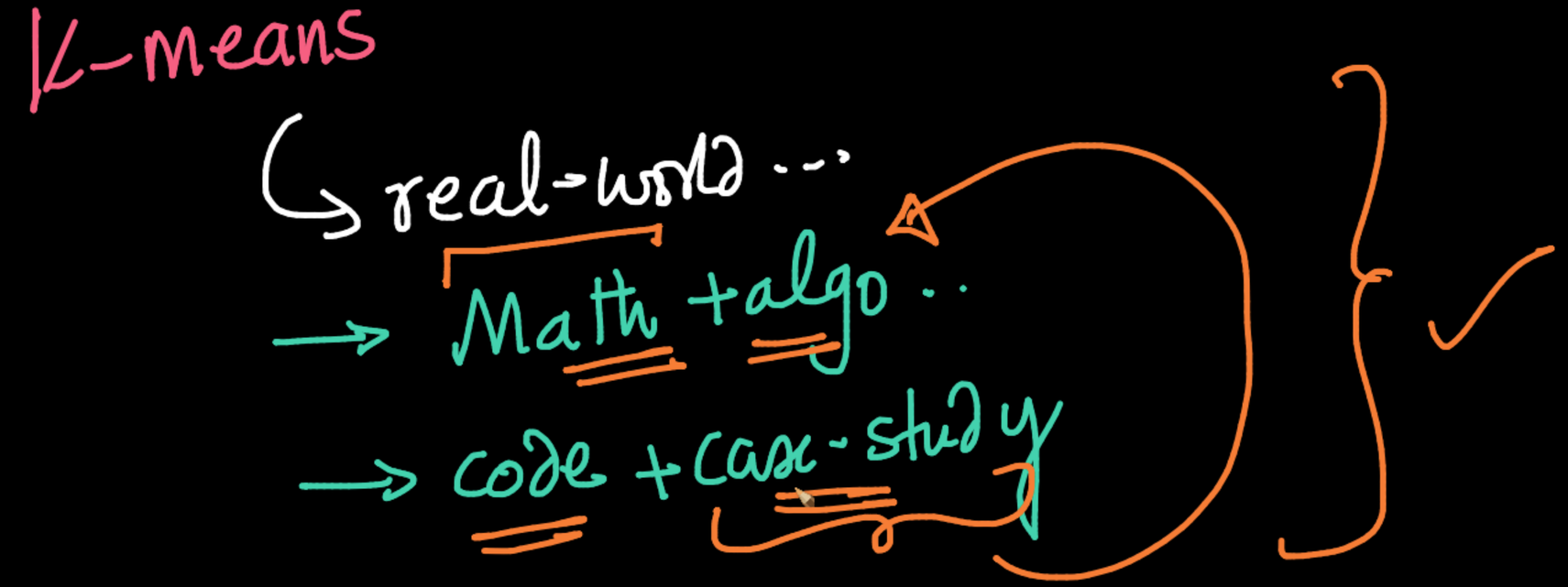
initialization sensitive


Lloyd's algo
lots of problems
initialization sensitive
examples (next class)
Fix: (hacks) KMeans++

Next - class:

- Init. Sensitivity + K means ++
- Best K
- Limitations K-means -- (Geom/Math/ real-world)
- K-medoids - PAM
- Code + Case study

K-means



Chrome File Edit View History Bookmarks Profiles Tab Window Help

Applications in different domains

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# as dim -> INF; (max_dist - min_dist)/min_dist -> 0

max_min_ratio=[]
for dim in range(1,1000):
  # dimensionality
  d=dim

  # generate "n" uniformly random points in d-dim space in the unit hypercuboid of each dimension=1
  n = 1000
  X = np.random.rand(1000,d) # random numbers in the range [0,1] => unit hypercuboid
  q = np.full([1,d],0.5) # [0.5,0.5,0.5,....] is the central point in the unit hypercuboid.

  dist=[]
  # compute the pair-wise distances from each other
  for i in range(n):
    dist_iq = np.linalg.norm(X[i]-q) # euclidean distance
    dist.append(dist_iq

  max_min_ratio.append( (max(dist)-min(dist))/min(dist) )
```

61 / 61