K means Clustering

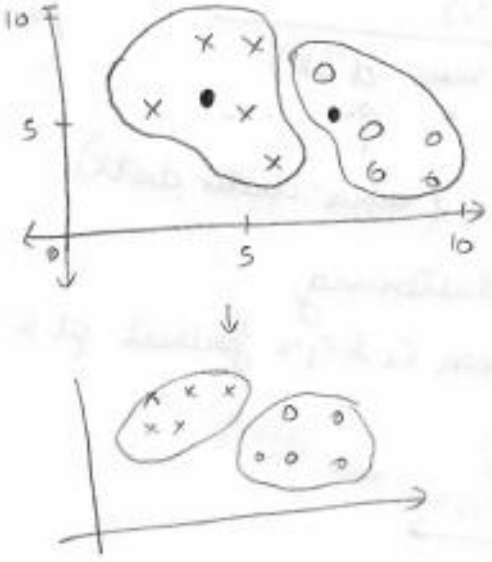
The output of K means is K clusters

Types of data in clustering analysis

1.) P- dimensional points (vectors) in Euclidean space
2) Nominal variables
3) Strings, trees, graphs and other objects.
4) Data of mixed types.

Algorithm :

1) Partition objects into k non empty subsets (clusters)
2) Compute center for each clusters
3) Assign each object to the cluster with the nearest center.
4) Go back to step 2; terminate when there is no new assignment.



• → Center of each cluster.

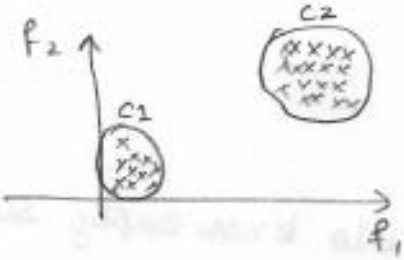
Metrics for clustering:

$\mathcal{D} = \{x_i\}$; no. y_i's

Classification & regression y_i's ← class labels
← regression values
{ground truth}

$f(x) = y$

geom: what is a clustering result (good)



K-Clusters
2-Clusters

Scatter plot

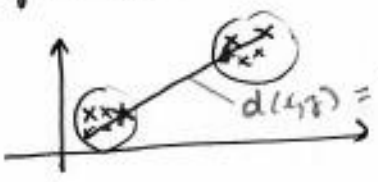
? Basis of how good our cluster is ?
intra cluster → means within a cluster → this is small
inter cluster → across between clusters → this is kept large

✧ Dunn-index: $D = \dfrac{\min\limits_{i,j} d(i,j)}{\max\limits_{k} d'(k)}$ → distance between C_i & C_j (maximum inter cluster dist)

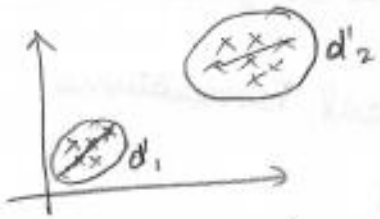
($d'(k)$ ↑ intra-cluster dist)

D is high ⇒ good clustering

$d(i,j)$ = dist between C_i & C_j's farthest pts.



$\max d'\{K\}$.

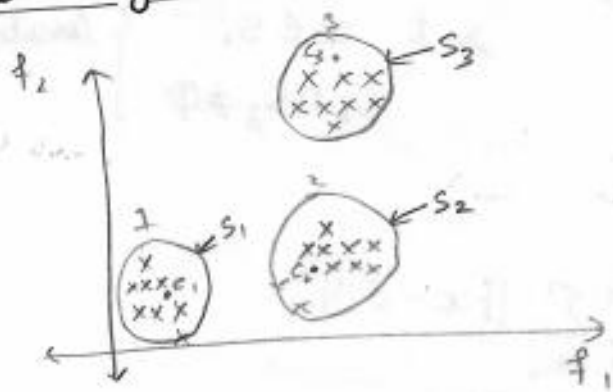


$\max\{d'_1, d'_2\}$

Many other measures / metrics for measuring clustering

ideal → we should have high inter distance
we should have low intra distance

K-means: Geometric intuitions, Centroids



K-means (K=3)

C_1, C_2, C_3 : Centroids

$S_1 \cup S_2 \cup S_3 = D$

$S_1 \cap S_2 \neq \emptyset$
$S_2 \cap S_3 \neq \emptyset$
$S_1 \cap S_3 \neq \emptyset$

K : # Clusters
↑ hyper parameter → CV ideas like

K-Clusters ≃ K-centroids → $C_1, C_2, C_3, \cdots, C_K$
K-Sets → $S_1, S_2, S_3, \cdots, S_K$

$$C_i = \frac{1}{n} \sum_{x_j \in S_i} x_i$$ ← mean-point to S_i

K-means → Centroid based clustering scheme.

Big Challange :- How to find K-Centroid

K-means :- Mathematical Formulations

$$D = \{x_1, x_2, \ldots, x_n\}$$

Task

K centroid :- $c_1, c_2, \ldots, c_k$

sets :- $S_1, S_2, \ldots, S_k$

$$\underset{c_1, c_2, \ldots c_k}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2$$

← intra cluster distance is minimized

proximity → $(S_1, S_2, \ldots S_k)$

s.t $x \in S_i$

$S_i \cap S_j \neq \phi$

} constraints

$$\underset{c_1, c_2, \ldots c_k}{\arg\min} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2$$

$c_1, c_2, \ldots c_k$: all clusters

$\sum_{x \in S_i} \|x - c_i\|^2$: Sum of squared distance from centroid in cluster i

Very hard problem

↓

NP hard problem

exponential time complexity to solve problem

Approximation algorithm → As hard to solve so introduced approximation algorithm

K-means: Lloyd's algorithm

① Initialization :

→ randomly pick K points from D and call them our centroids $c_1, c_2, \ldots, c_k$.

② Assignment :

for each point x_i in D

Loop → select the nearest c_j - dist (x_i, c_j) $\forall$ $j = 1, 2, \ldots, k$

→ add x_i to set S_j.

③ Recompute centroid / update

$x_i \rightarrow S_j \quad j = 1, 2, \ldots, k$

→ recalculate / update centroids c_j's as follows

$$c_j = \underbrace{\frac{1}{|S_j|} \sum_{x_i \in S_j} x_i}_{\text{mean-pt}}$$

④ Repeat step 2 & 3 until convergence

↙ assignment ↘ update

Convergence → means when centroids don't change much.

(old centroid ≈ new centroids)

$(c_1' - c_1, c_2' - c_2, c_3' - c_3, \ldots, c_k' - c_k)$

Small values

How to initialize K-means:

Lloyd's algorithm : initialization stage

↓

random ← random : pick K-pts randomly from D
int

↓

$C_1, C_2, \ldots, C_K$

Problem:-

initialization sensitivity -

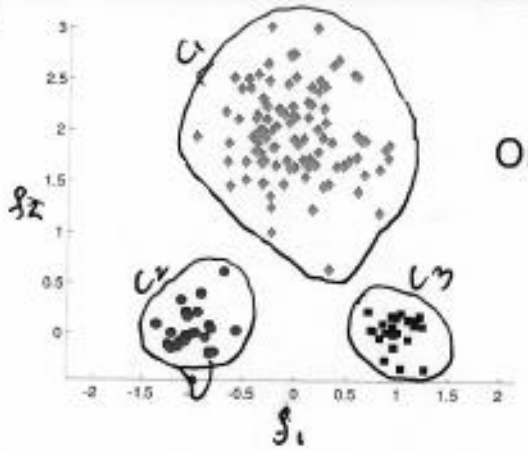
eg (s): Toy datasets

final clusters & centroids depends on how you initialize

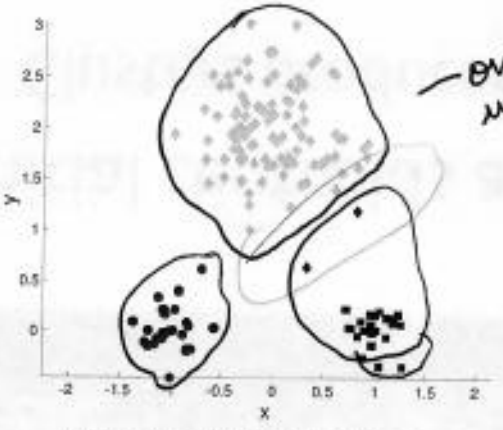
K-means Algorithm – Initialization

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.

Two different K-means Clusterings

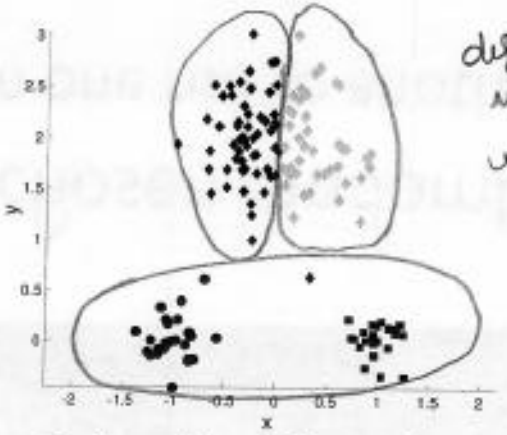


Original Points – human labels



– one initialization

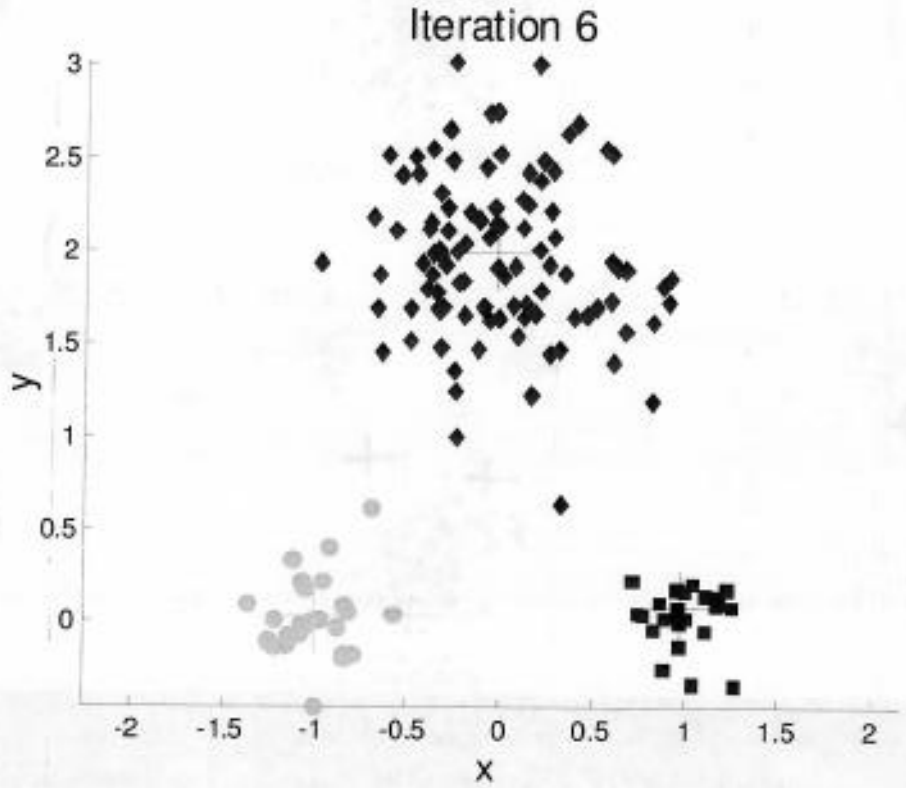
Optimal Clustering



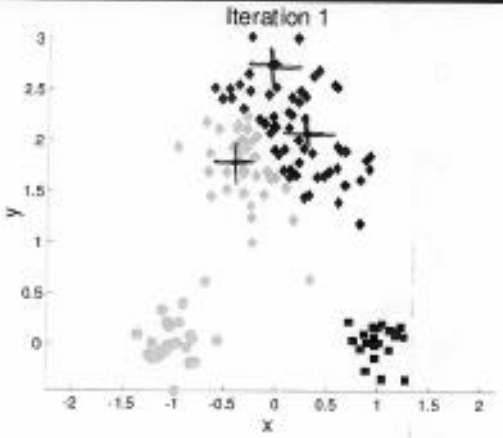
different initialization undesirable result

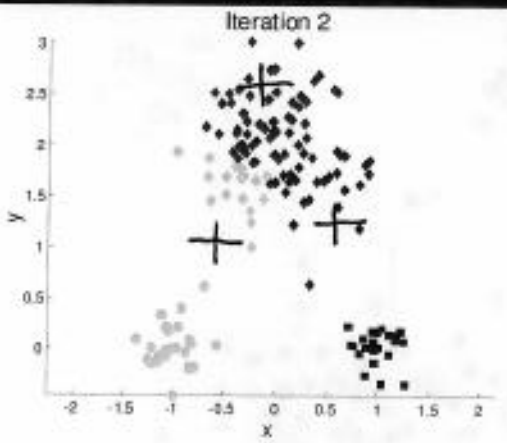
Sub-optimal Clustering

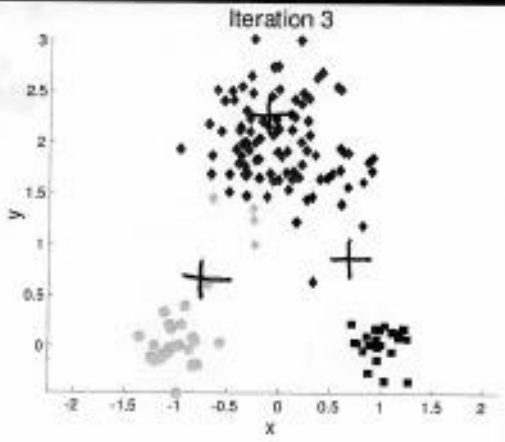
Importance of Choosing Initial Centroids

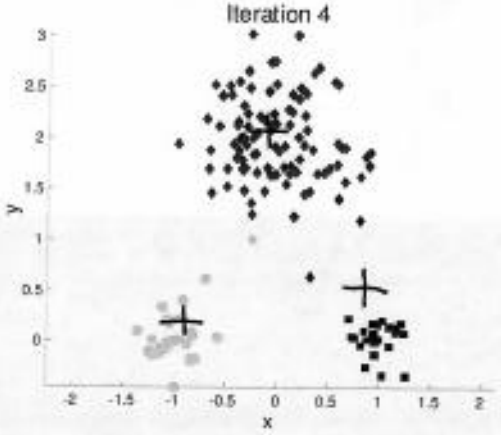


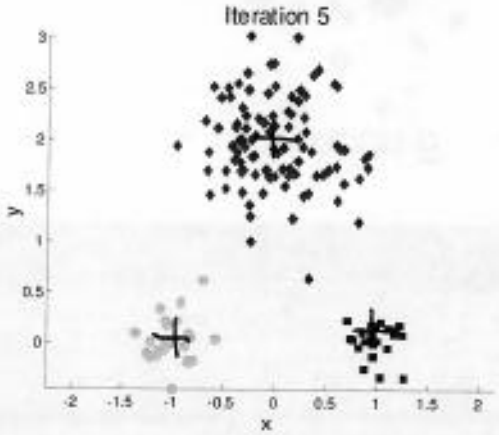
Importance of Choosing Initial Centroids

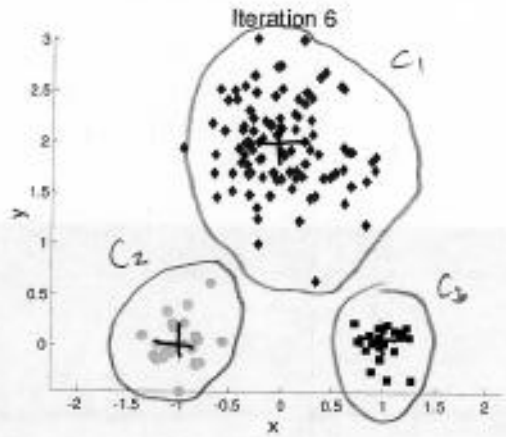




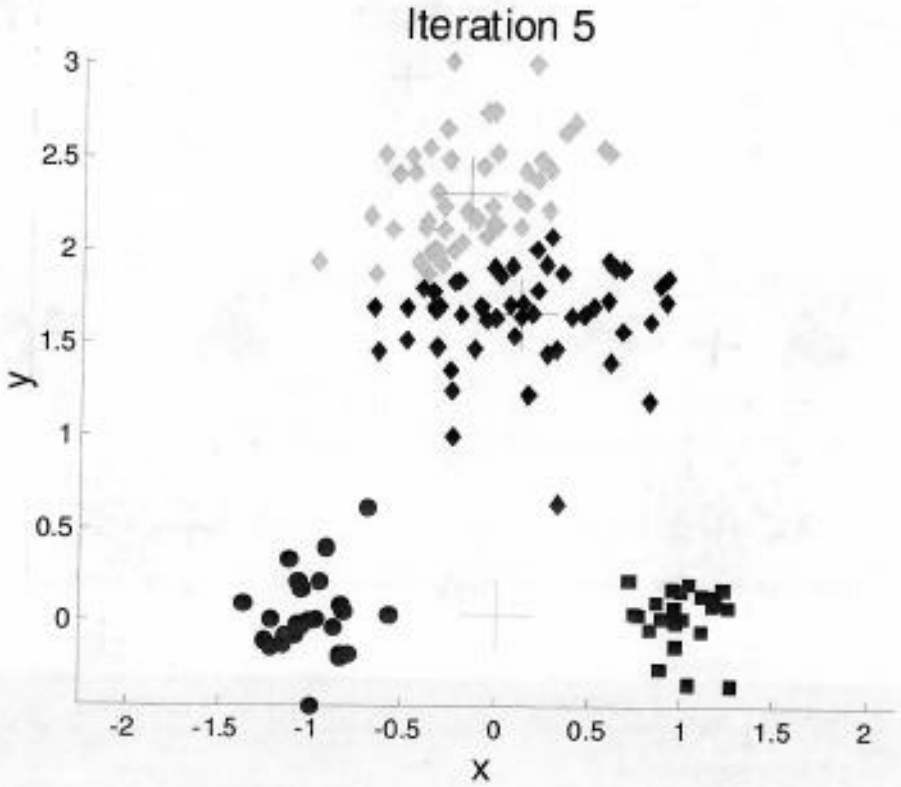






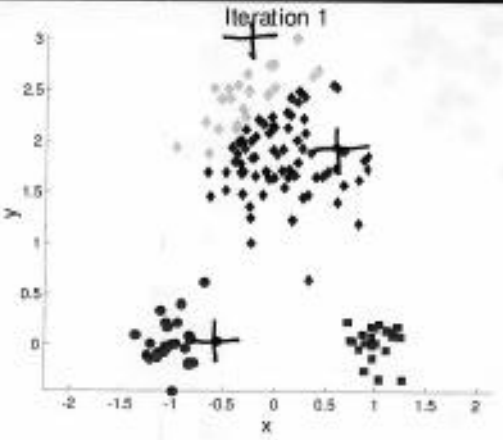


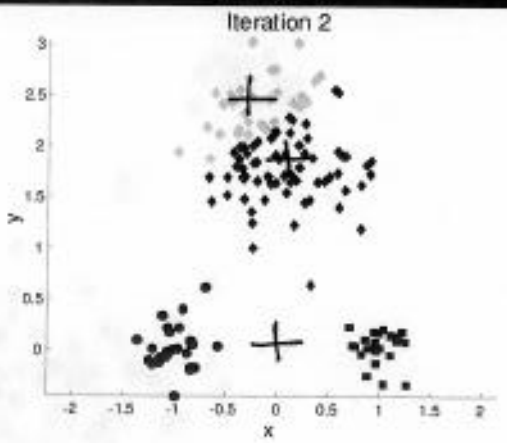
Importance of Choosing Initial Centroids

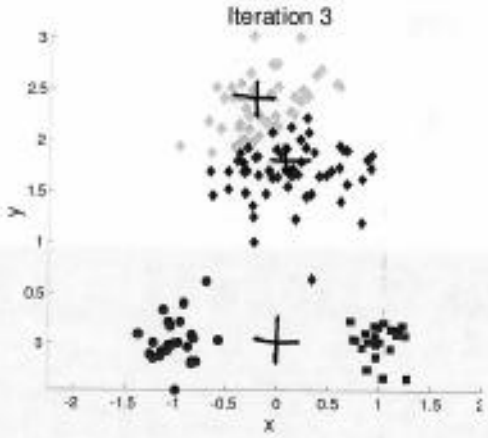


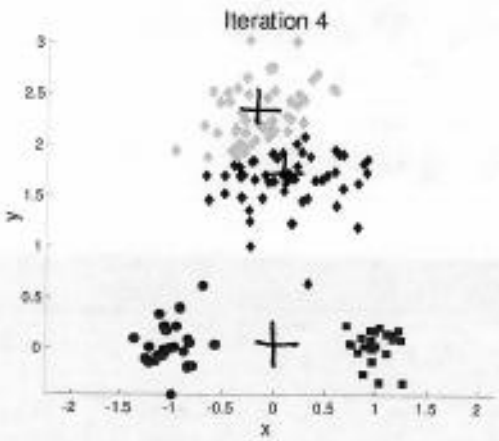
Importance of Choosing Initial Centroids ...

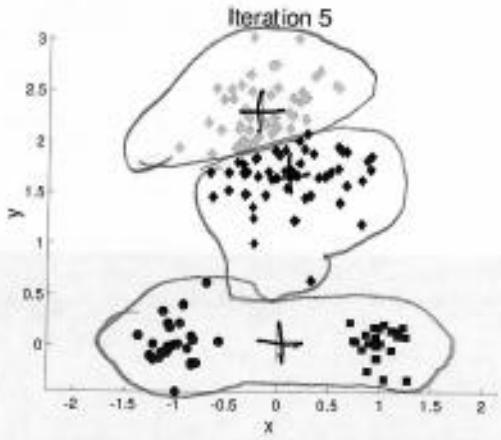
Choosing different cluster centroid











Dealing with Initialization

- Do multiple runs and select the clustering with the smallest error
- Select original set of points by methods other than random . E.g., pick the most distant (from each other) points as cluster centers (K-means++ algorithm)

K-means Algorithm – Centroids

- The centroid depends on the distance function
  - The minimizer for the distance function
- 'Closeness' is measured by Euclidean distance (SSE), cosine similarity, correlation, etc.
- Centroid:
  - The mean of the points in the cluster for SSE, and cosine similarity
  - The median for Manhattan distance.
- Finding the centroid is not always easy
  - It can be an NP-hard problem for some distance functions
    - E.g., median form multiple dimensions

It can be improved

→ repeat K-means multiple times with different initializations
↳ pick the clustering based on
{ smaller intra - cluster }
{ max-imum inter-cluster }

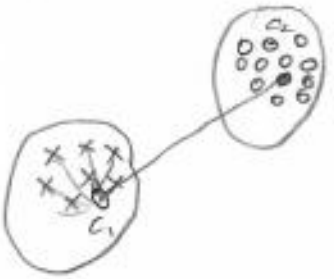
② K-means ++
↳ random - init → smart init

initialization in K-means: (Task) pick $c_1, c_2, \dots, c_k$

① pick the first centroid randomly → c_1 from D.

② $\forall\ x_i \in D$ create a distribution:

$x_i \rightarrow$ distance^2(x_i, nearest centroid)

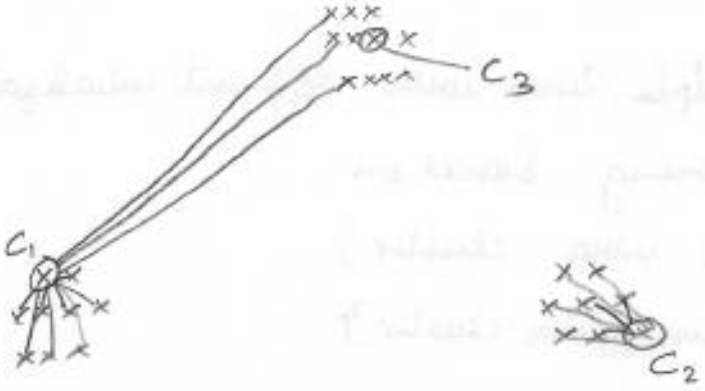


{ probabilistic approach }

x_1	d_1
x_2	d_2
x_3	d_3
⋮	⋮
x_n	d_n

$\|(x_i - c_1)\|^2$

pick a point from $D - \{c_1\}$ with a prob. proportional to d_i



In initialization we are tryeng to pick points as (centroid) that are as far as possible from other centroids.

Q Why do we this probabilistically?

Ans pick a point which has the highest value of $d^2(x_i, \text{nearest centroid})$

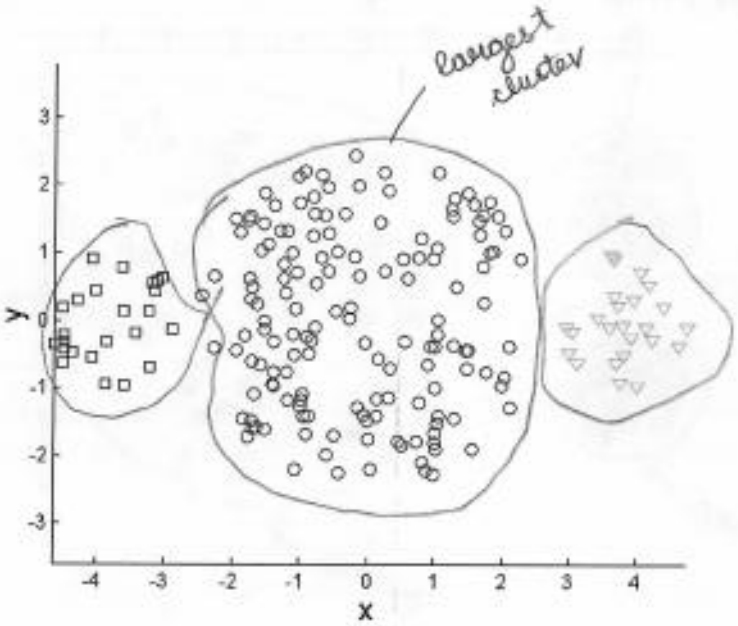
Because the point farther can be outlier. which we donot want

K++ gets effected by outlier ~~but~~ so we us probalulity for this.

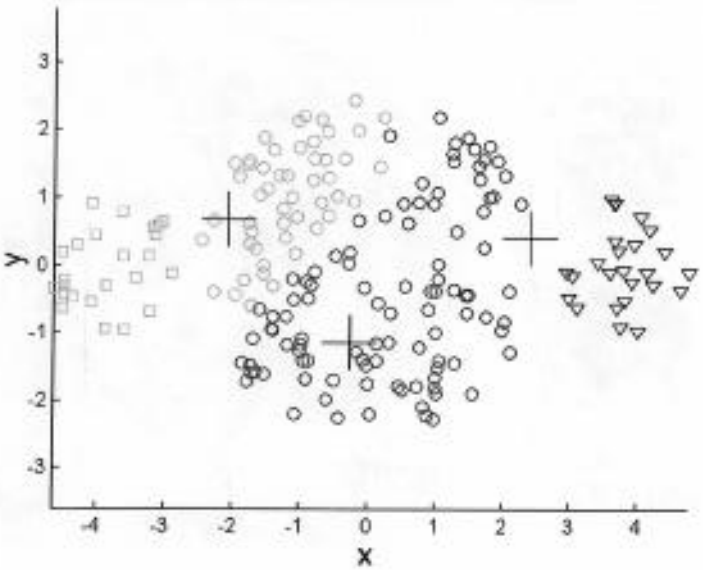
Limitations of K-means

1.) clusters of different sizes
2.) clusters of different densities
3.) cluster of Non-globular shape (non-convex shape)
4.) Kmeans has problems when the data contains outliers

Limitations of K-means: Differing Sizes



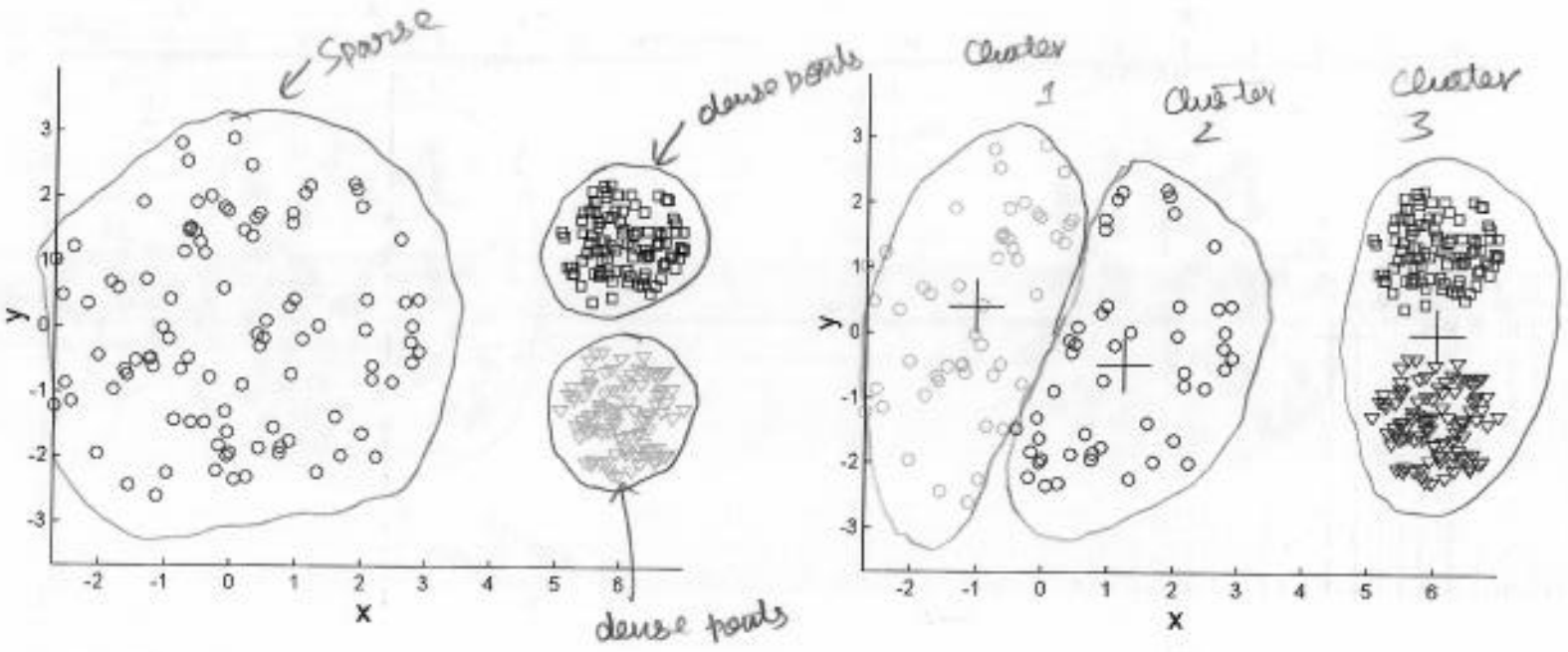
Original Points



K-means (3 Clusters)

clusters of same size always formed that is problem.

Limitations of K-means: Differing Density

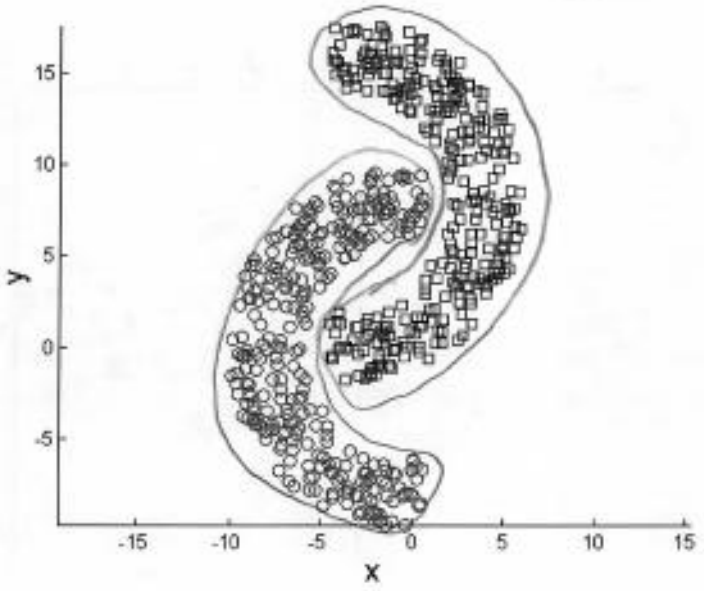


Original Points

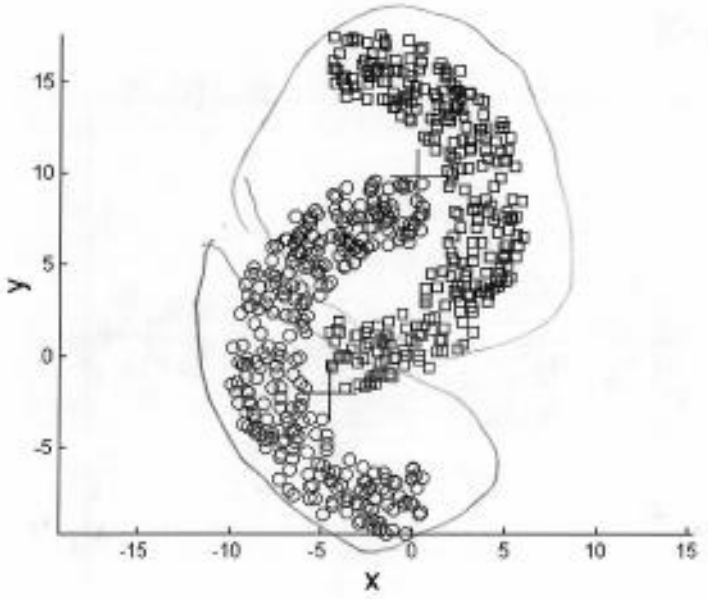
K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

non convex data / Nonglobular shapes

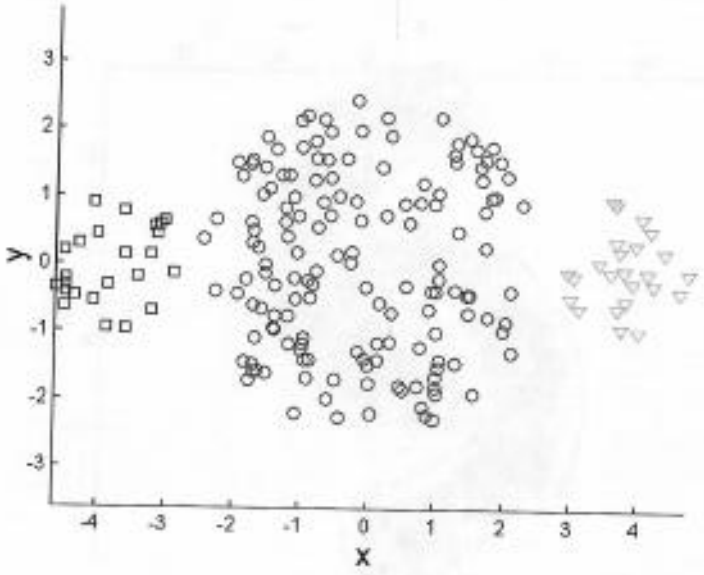


Original Points

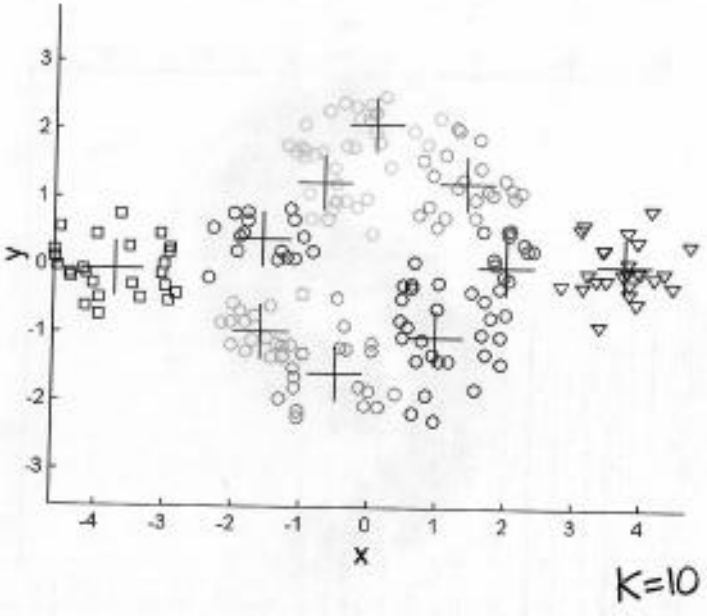


K-means (2 Clusters)

Overcoming K-means Limitations



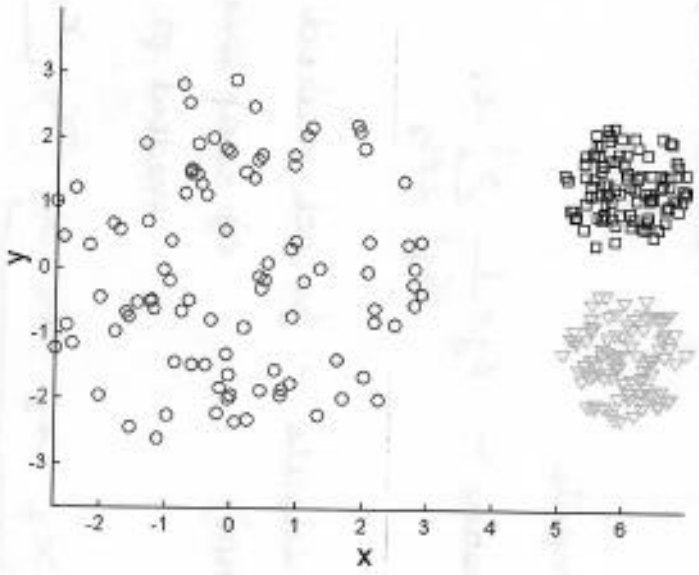
Original Points



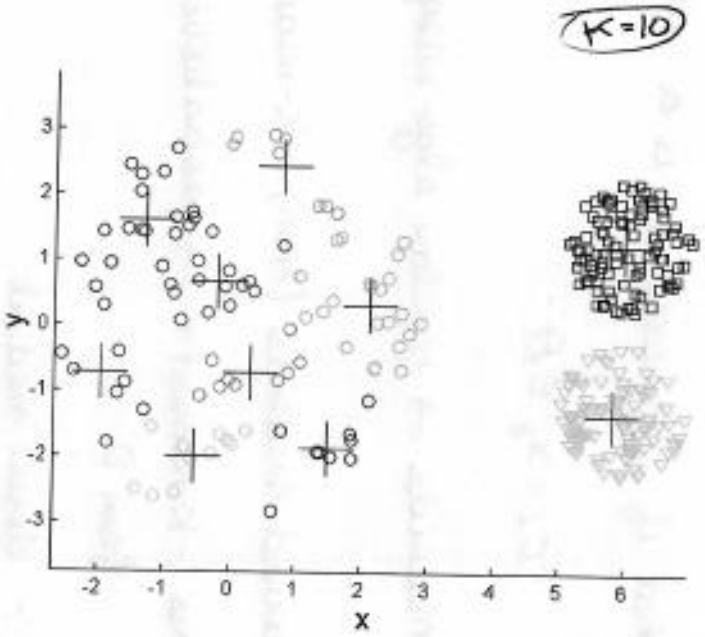
K-means Clusters

→ One solution is to use many clusters.
Find parts of clusters, but need to put together.
We need to increase the value of K

Overcoming K-means Limitations



Original Points



K-means Clusters

K-Medoids

problem K-means: $c_1, c_2, \ldots, c_k$. → centroid may not be interpretable

$$D = \{x_1, x_2, \ldots, x_n\}.$$

→ Big-idea: what if each centroid is a datapoint in D.

$c_i = x_j \in D$

K-Medoids → popular algo interpret centroids

Partioning around medoids (PAM): K-medoids

1.) initializations :- K-means++ → probabilistic methods pick K pts from D.

2.) Assignment :- closest medoid

$x_i \in C_j$ if medoid j is the closest medoid to x_i

3.) Update / Recompute :

→ K-means :- $c_j = \frac{1}{|S_j|} \sum_{x \in S_j} x_i$ ✗

→ K-medoids :- for each medoid

(PAM)

a) swap each medoid with a non-medoid pt.

M_1 M_2

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}

non medoid non medoid

b) if loss decreases keep the swap else undo the ~~loss~~ swap

loss in K-means

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

m_z ↑ medoid$_z$

Determining the right 'K'

↑ hyperparameter

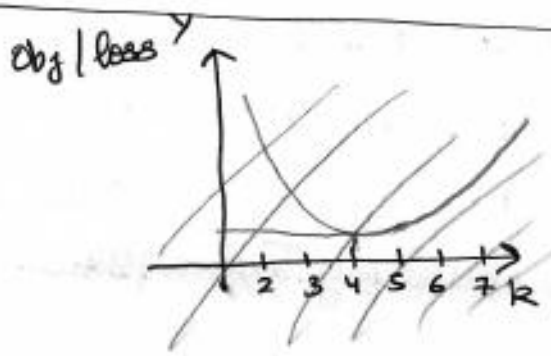
① domain-knowledge: Food reviews

↳ +ve & -ve

K=2

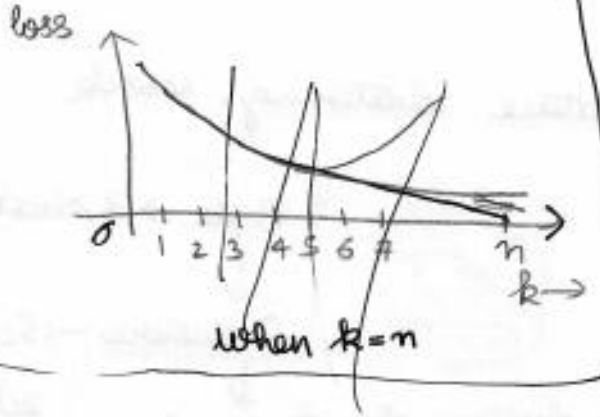
- We use elbow method or Knee method

loss / Objective function: $\sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2 \rightarrow$ minimize



best k=4

minimum value of obj is k=4



when k=n

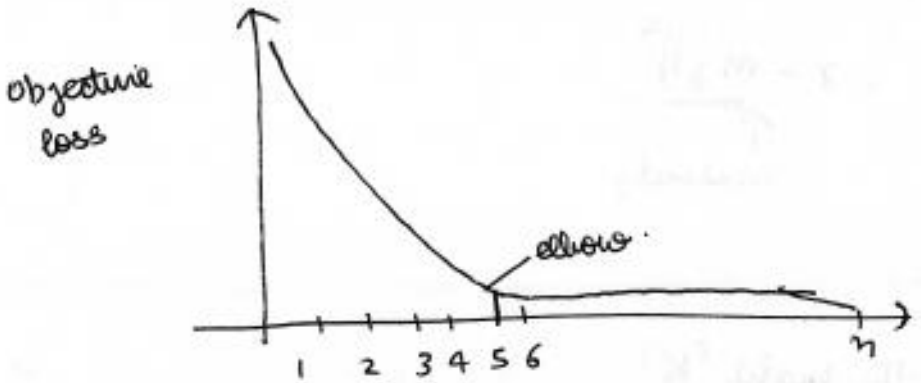
NOTE

K = n = # point

↳ 1000 cluster

⇓

every point is a cluster



$k=5$ is best as it is forming an elbow.

Time and space complexity

K-means $O(n K d i)$ # iterations

- n → #pts
- K → # clusters
- d → dim

$K \leq 10$ Typically

$i \leq 300$

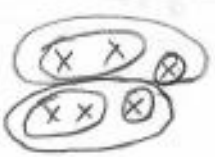
Space $O(nd + Kd)$

$\approx O(nd)$ ← linear

Hierachical clustering Technique

☆ Hierachical Clustering

~~Agglomerative~~ 1.) Agglomerative clustering work.



13 pts → 13 clusters

↓

9 clusters → 5 cluster

↓

2 cluster

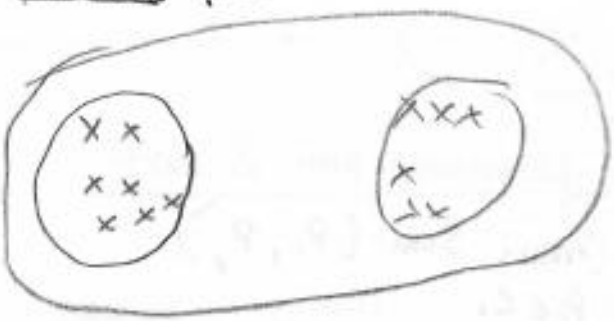
↓

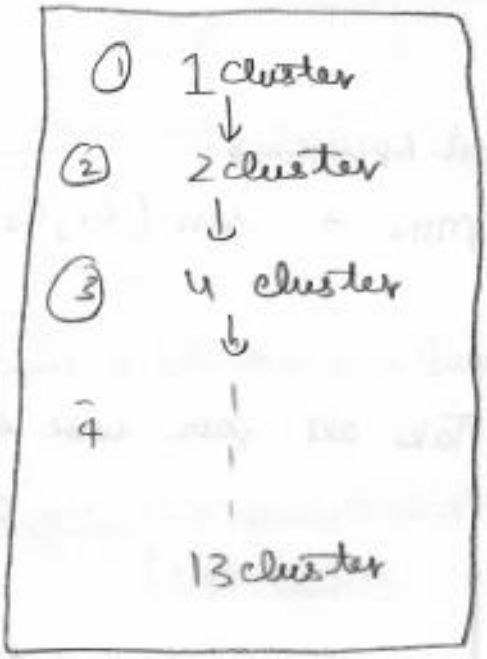
1 Clusters

→ Incrementally combine clusters close to each other

Agglomerative cluster → combine clusters close to each other.

2.) Divisive :—





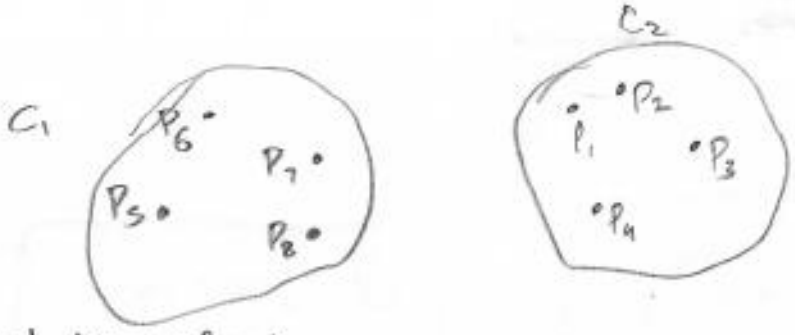
Agglomerative is more used in comparision to divisive

Key ingredient → (similarity or distance between clusters)

NOTE → It is called Hierachical because it forms trees which group datapoints into clusters. This tree is called dendogram.

Proximity methods:
Advantages and Limitations

☆ How to Define Inter Cluster Similarity



Different Approaches:

1) MIN → $Sim(C_1, C_2) = \min_{P_i \in C_1, P_j \in C_2} Sim(P_i, P_j)$

Take all pair wise similarity of cluster and pair wise similarity of points.

2) MAX → Farthest or dissimilar point

$$MAX: Sim(C_1, C_2) = \max_{P_i \in C_1, P_j \in C_2} sim(P_i, P_j)$$

3) Group Average :→

$$AVG: Sim(C_1, C_2) = \frac{\sum_{P_i \in C_1, P_j \in C_2} Sim(P_i, P_j)}{|C_1| * |C_2|}$$

$|C_1|$ ← Size of C_1 ; $|C_2|$ → Size of C_2

NOTE

Kernel Trick can be applied for all 3 approaches
⇒ Ker (sim (P_i, P_j))

4) Distance between Centroids:

Define centroid and compute distance

Space & Time complexity for hierarchical clustering

Space: $O(n^2)$ ⟶ Sim matrix

n # data points ⟶ a lot when n is large

Time:- $O(n^3)$:- atmost n iteration → group 2 cluster
↳ 1 cluster.

↓

update similarity matrix $O(n^2)$

Drawback: not very useful with large data points.

Limitations of Hierarchical clustering

① No objective (math) function that we are directly solving :- (K means - clear math obj)

↳ neat algo soln.

② MIN → problem with outliers
MAX → breaks large clusters can't accomodate different sized clusters
Group average have problem too as it is combination of min and max

3.) MOST IMPORTANT LIMITATION:-

Space & time Complexity

$O(n^2)$

$\hookrightarrow O(n^2 \lg n)$

$O(n^3)$

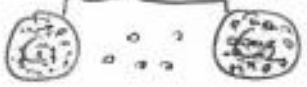
K-means have nice complexity compared to it.

DBSCAN: Density Based Clustering Technique

- Centroid-based :- K-means
- Hierachical based :- Agglomerative

$D = \{x_i\}$

Sparse region



density-region → clusters
Sparse regions → noise

Measuring density: Min Pts, EPS

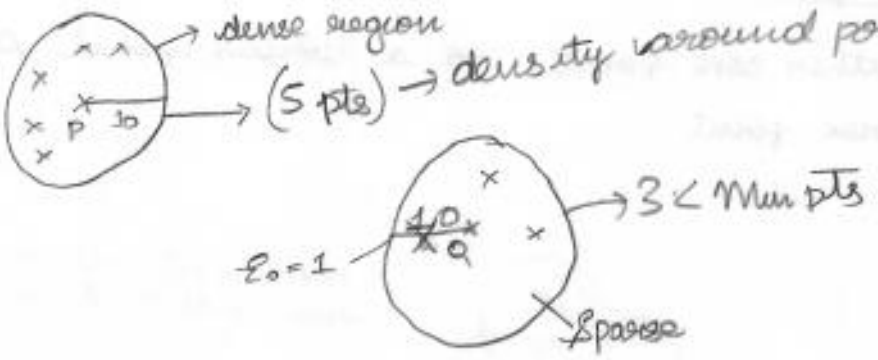
↳ Hyperparameter of DBSCAN

(1) density at a P: # pts within a hypersphere of radius ε_0 around P.

↳ Circle 2D
↳ Sphere 3D

ε_0 usually = 1.0

(2) dense region :- a hypersphere / circle of radius ε_0 (1.0) that contains atleast Min Pts (4) points



If number of points in radius is greater than threshold than it is dense region

☆ Core, Border and Noise points

$D = \{x_i\}$ Min Pts · ε_0

① Core point (P) : if P has ≥ Min Pts in an ε_0 radius around it.

Every core pt. belongs to dense region.

② Border point (P)

① P is not a core point & has < Minpt points ε_0 radius

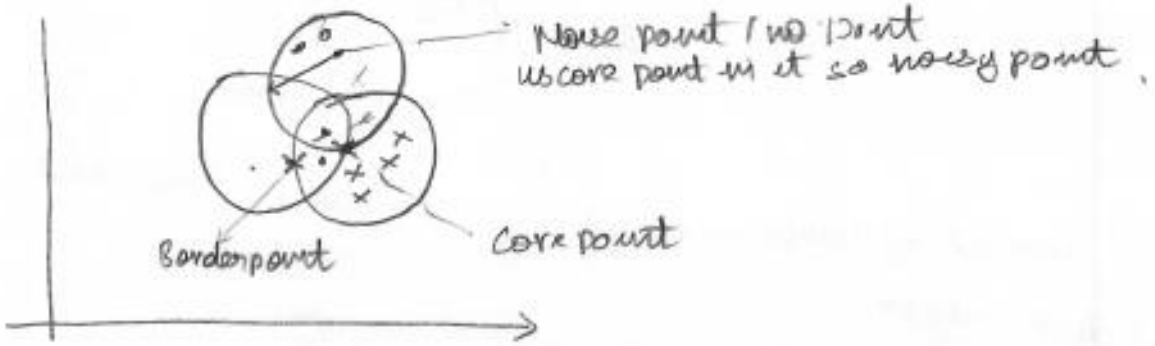
② P ∈ Neighbourhood (Q)

Q : core point

dist (P, Q) ∈ ε_0

③ Noise point :-

neither core point nor a border point are called noise point.



☆ Density edge & density connected points

① density edge → connection

↳ P, q :- core point

② density connected points : P & q : core points