> **Subject**: ML slides pack - clustering with and without kmeans by Claudio Sartori
>
> **Course:** Artificial Intelligence - LM
>
> **Department**: DISI (Department of Computer Science and Engineering) - University of Bologna, Italy
>
> **Author**: Roberto Zanolli
>
> **TIPS**
>
> - w.r.t means "with respect to"

Clustering

What is clustering

Given: A set of N objects x_i, each described by D values x_{id}

Task: Find a natural partitioning into K clusters and possibly identify noise objects

Result: A clustering scheme - a function mapping each data object to $\{1 \ldots K\}$ (or to noise)

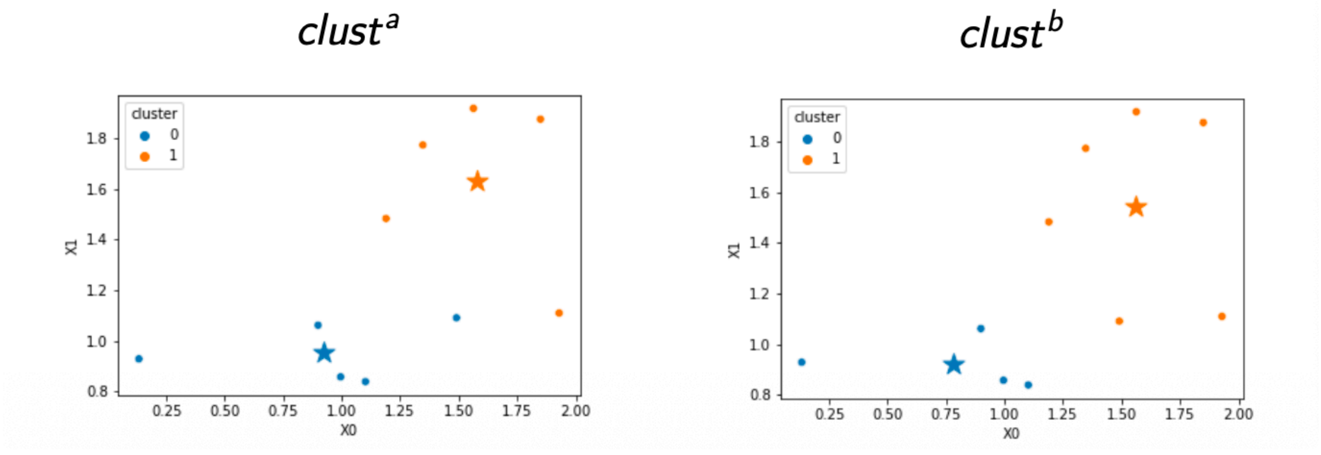
Desired cluster property:

- Objects in same cluster are similar
- Look for clustering scheme that **maximizes intra-cluster similarity**

Formal definition for clustering function

> Find a function $clust()$ from X to $\{1..K\}$ such that:
>
> 1. $\forall x_1, x_2 \in X$, $clust(x_1) = clust(x_2)$ when x_1 and x_2 are similar
> 2. $\forall x_1, x_2 \in X$, $clust(x_1) \neq clust(x_2)$ when x_1 and x_2 are not similar

How do we choose a measure to determine clusters



Centroid

Definition: A point whose coordinates are the average coordinates of all points in the cluster

Calculation: For each cluster k and dimension d, the d-th coordinate of the centroid is:

$$centroid_d^k = \frac{1}{|x_i : clust(x_i) = k|} \sum_{x_i : clust(x_i)=k} x_{id}$$

Taxonomy of clustering method

> The ones with * means that has been seen during lab

Partitioning:

- **K-means (MacQueen 67)** *
- Expectation maximization (Lauritzen 95)
- CLARANS (Ng and Han 94)

Hierarchic:

- **Agglomerative/divisive*** *
- BIRCH (Zhang et al 96)
- CURE (Guha et al 98)
- Based on linkage

Based on density:

- **DBSCAN (Ester et al 96)** *
- DENCLUE (Hinnenburg and Keim 98)

Statistics:

- IBM-IM demographic clustering
- COBWEB (Fisher 87)
- Autoclass (Cheeseman 96)

K-MEANS CLUSTERING

The challenge: Given data that appears to form distinct groups (like a five-component gaussian mixture), how do we automatically discover these groups in high-dimensional space?

Core idea: View clustering as a lossy compression problem. If we must transmit point coordinates using very few bits, we need to group similar points and represent them by a common "code" (centroid).

Loss function: We measure quality by the sum of squared errors (SSE) between actual points and their representative centroids.

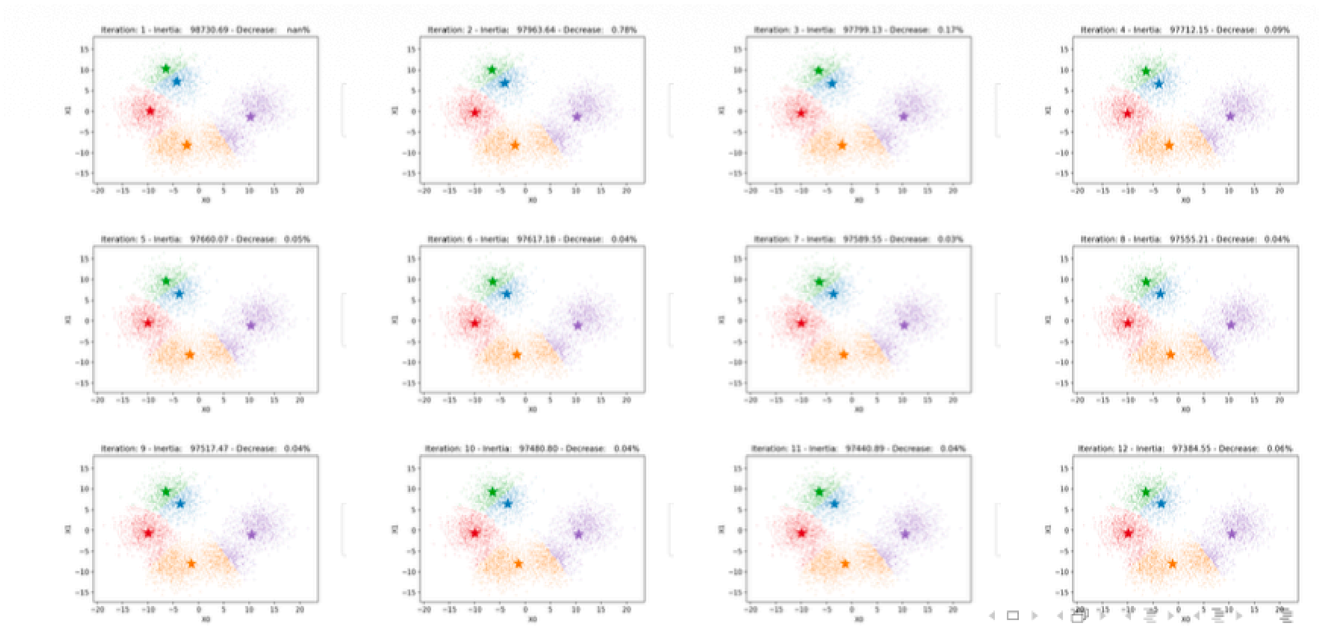
Algorithm intuition: Instead of a fixed grid, let the data itself determine optimal group representatives:

1. Start with random guesses for group centers
2. Assign each point to its nearest center
3. Update centers to be the centroid (mean) of their assigned points
4. Repeat until assignments stabilize

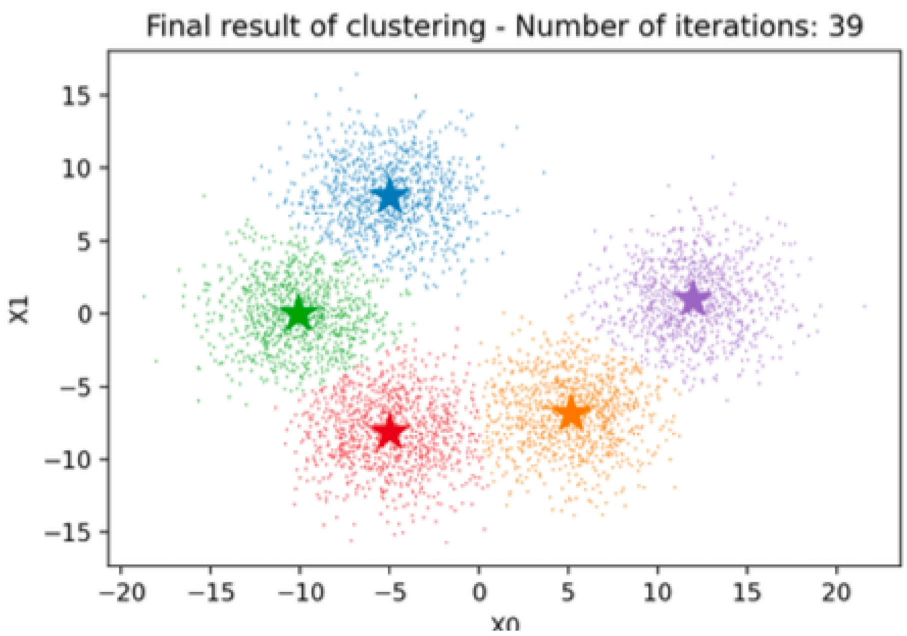
Key questions that remain:

1. What are we trying to optimize?

2. Is termination guaranteed?
3. Are we sure that the best clustering scheme is found? (And what defines "best"?)
4. How should we initialize the centers?
5. How can we determine the optimal number of clusters?



From numerous iterations we get here:



Question 1: Distortion (Inertia)

Definition: Given:

- Dataset $\{x_i, i = 1...N\}$
- Encoding function $Encode : \mathbb{R}^D \to \{1..K\}$
- Decoding function $Decode : \{1..K\} \to \mathbb{R}^D$

$$Distortion = \sum_{i=1}^{N} (x_i - Decode(Encode(x_i)))^2$$

Let $Decode(k) = c_k$, then:

$$Distortion = \sum_{i=1}^{N} (x_i - c_{Encode(x_i)})^2$$

Minimal distortion

Question: What properties do $c_1, \ldots, c_K$ need for minimal distortion?

> **Property 1:** x_i must be ***encoded with the nearest center***
>
> - Otherwise distortion could be reduced by choosing the nearest center $c_{Encode(x_i)} = \arg\min_{c_j \in \{c_1,\ldots,c_K\}} (x_i - c_j)^2$

> **Property 2:** Partial derivative of distortion w.r.t. each center must be zero
>
> - Indicates maximum or minimum of function

Mathematical derivation

$$Distortion = \sum_{i=1}^{N} (x_i - c_{Encode(x_i)})^2 = \sum_{j=1}^{K} \sum_{i \in OwnedBy(c_j)} (x_i - c_j)^2$$

$$\frac{\partial Distortion}{\partial c_j} = -2 \sum_{i \in OwnedBy(c_j)} (x_i - c_j) = 0 \text{ at minimum}$$

Solution:

$$c_j = \frac{1}{|OwnedBy(c_j)|} \sum_{i \in OwnedBy(c_j)} x_i$$

Question 2: Algorithm termination

Proof: Finite number of states ensures termination

- Finite ways to partition N objects into K groups
- Finite configurations where centers are centroids of their points
- Each iteration reduces distortion → new state never visited before
- Eventually no new states reachable → algorithm stops

Question 3: Local vs global minimum

Problem: Ending state not necessarily global optimum

- Multiple local minima possible
- Final clustering depends on initialization

Question 4: Finding good starting points

Strategies:

1. Random first point, subsequent points far from previous ones

2. Run algorithm multiple times with different random starts
3. Choose best result based on distortion

Question 5: Choosing number of clusters K

Challenge: No easy automatic method

Approach:

- Try various K values
- Use quantitative evaluation metrics
- Balance intra-cluster compactness vs inter-cluster separation

Summary of application

- Try with different starting points with same K and choose the best starting point to ensure you get to global minimum (at least we try)
- After you choose

Proximity function

Default: Euclidean distance (works well for vector spaces)
Alternatives: Various distance metrics for specific data types

Sum of Squared Errors (SSE)

Formula:

$$SSE = \sum_{j=1}^{K} \sum_{i \in OwnedBy(c_j)} (x_i - c_j)^2$$

Properties:

- $SSE_j = 0$ only if all points coincide with centroid
- SSE decreases with increasing K, reaches 0 when $K = N$
- Cannot minimize SSE to choose K (always prefers more clusters)

Outliers

Impact: High distance from centroid → high contribution to SSE

- Bad influence on clustering results
- Sometimes beneficial to remove (domain-dependent)

Common uses of K-means

1. **Data exploration:** Quick initial clustering
2. **1D discretization:** Create non-uniform bins
3. **Vector quantization:** Signal processing, compression
4. **Color quantization:** GIF compression, color palette selection

Time complexity

> **Time:** $O(TKND)$
>
> *Where*:
>
> - T = number of iterations
> - K = number of clusters
> - N = number of data points
> - D = number of dimensions

Pros and cons

PROS:

- Fairly efficient (nearly linear in N)
- $T, K, D \ll N$ in practice

CONS:

- Requires space where centroid computable (works with Euclidean, some other distances)
- Cannot handle nominal data
- Requires K parameter (though can be searched)
- Sensitive to outliers
- No noise handling
- Poor with non-convex clusters