Third Summer School on ML for Electron Microscopy

May 19 – 23 2025

Clustering on Imaging and Spectroscopic Data

- Kamyar Barakati

THE UNIVERSITY OF TENNESSEE KNOXVILLE

What is clustering?

process of **grouping a set of items** so that

- Maximize inter cluster **similarity**
- Minimize intra cluster **similarity**

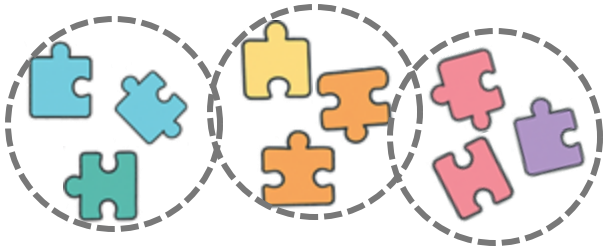
What is similarity?

- Euclidean distance
- Angle like cosine similarity

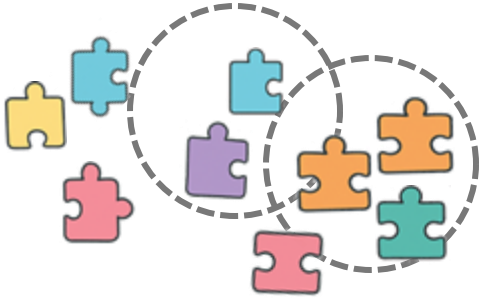
In microscopy:

Morphology → Similar shapes

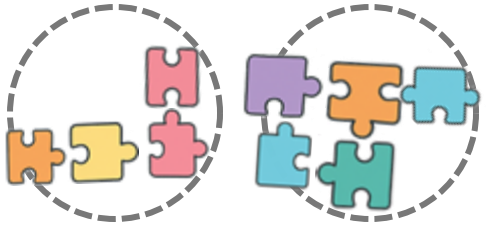
Intensity → Similar electron density?



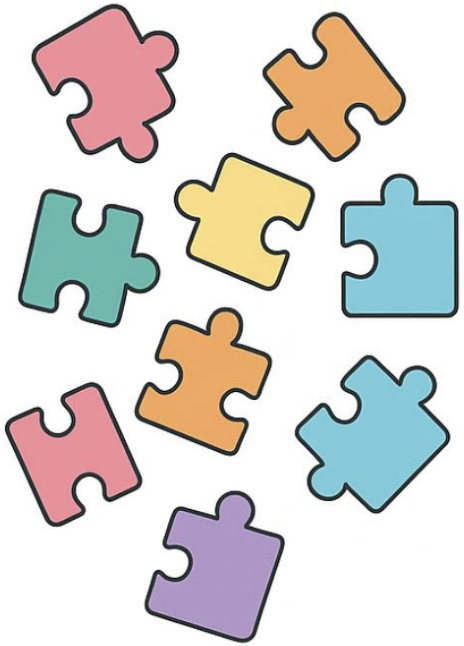
Color



Shape



Pattern



No final image

to guide you

- **No supervision**
- **no labels**

Just structure hidden in the data.

Clustering → Data structure

Clustering is just grouping by "**similarity**," but you have to decide how you represent your data

- **Hierarchical clustering**
- **K-means clustering (2D)**
- **Gaussian Mixture Models (2D)**
- **Density-based clustering**
- **Spectral clustering**

A **dissimilarity matrix** is simply a way of representing all pairwise "how different?" **scores** between your **n** objects in a single **n*n** table

Data matrix (two modes)

Feature

Data point

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Dissimilarity matrix (one mode)

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

What distances can tell us in clustering?

Distance measure on pairs of objects: $\boldsymbol{d(x, y)}$

Single linkage:

$$\boldsymbol{d_{single}} = \min_{\boldsymbol{x} \in \boldsymbol{A}, \acute{\boldsymbol{x}} \in \boldsymbol{B}} \boldsymbol{d(x,\ \acute{x})}$$

Complete linkage:

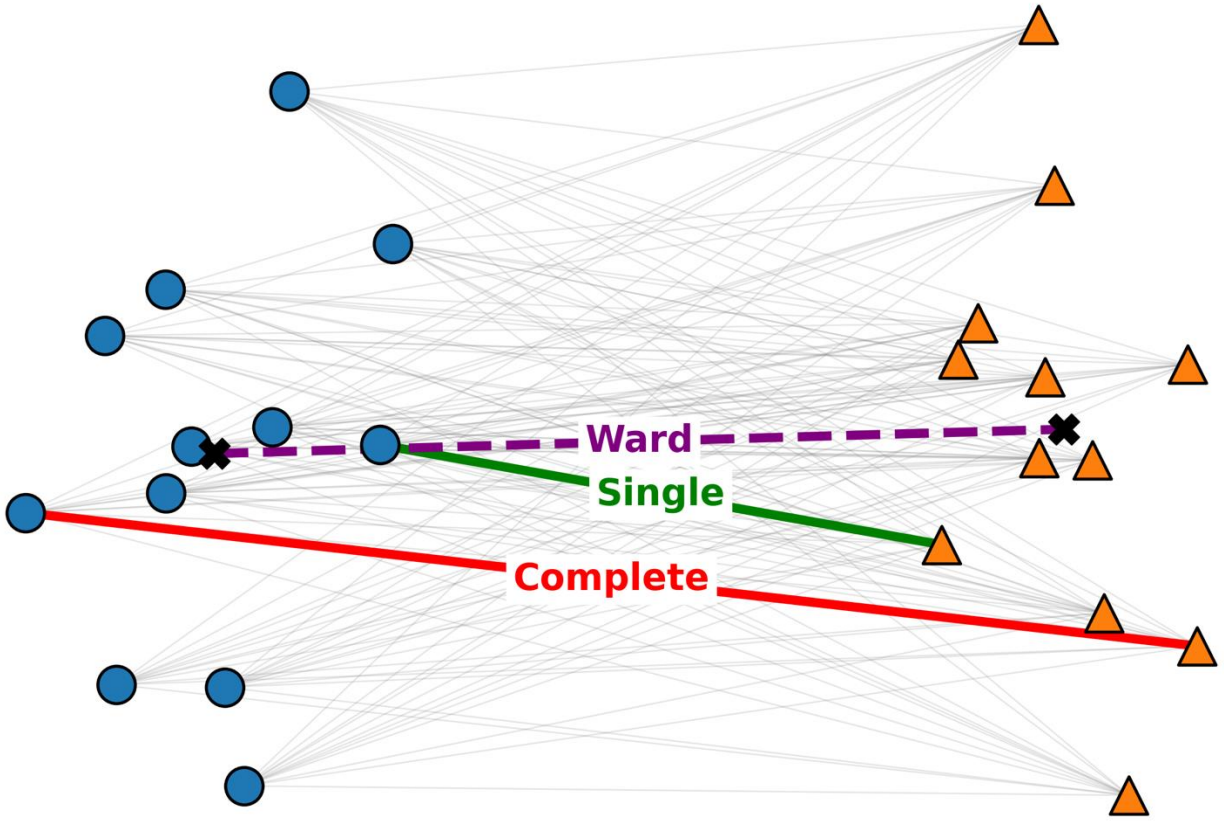
$$\boldsymbol{d_{complete}} = \max_{\boldsymbol{x} \in \boldsymbol{A}, \acute{\boldsymbol{x}} \in \boldsymbol{B}} \boldsymbol{d(x,\ \acute{x})}$$

Average linkage:

$$\boldsymbol{d_{average}} = \operatorname*{average}_{\boldsymbol{x} \in \boldsymbol{A}, \acute{\boldsymbol{x}} \in \boldsymbol{B}} \boldsymbol{d(x,\ \acute{x})}$$

Wards method:

$$\boldsymbol{d_{ward}} = \frac{|\mathrm{A}|+|\mathrm{B}|}{|\mathrm{A}||\mathrm{B}|} \, ||\overline{\boldsymbol{x}}_A - \overline{\boldsymbol{x}}_B||^2$$

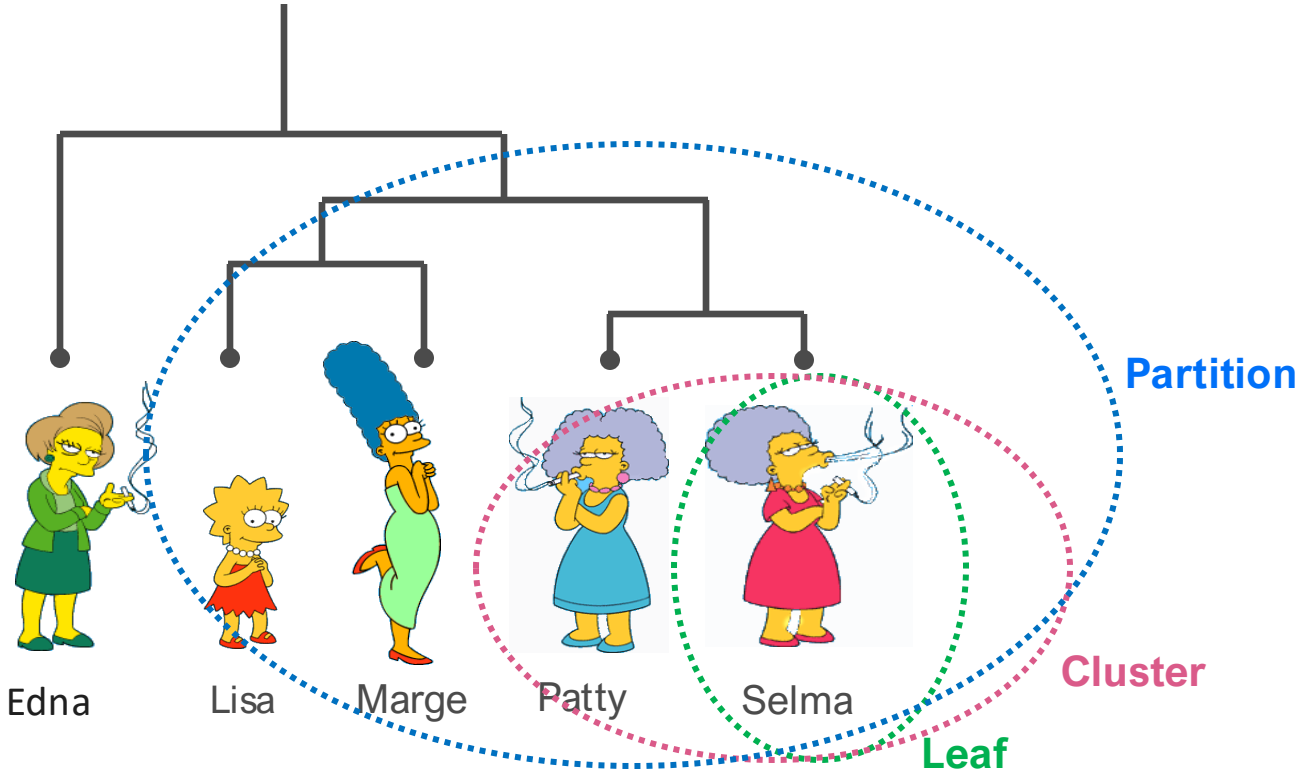


Hierarchical Clustering

It is all about building a **tree of groups** (a dendrogram) that shows how data can be merged or split step by step.

$$\#\text{dendrograms with } n \text{ leaves} = \frac{(2n-3)!}{2^{n-2}(n-2)!}$$

Number of Leaves	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	…
10	34,459,425



Heuristic

Bottom-Up (Agglomerative)

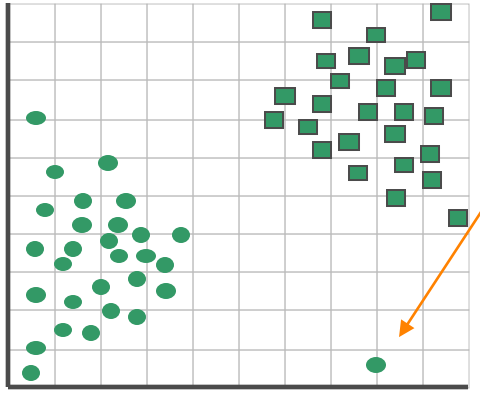
- **Start**: Every point is its own tiny cluster.
- **Step**: Find the **closest** two clusters and **merge** them into one.
- **Repeat**: Keep merging the nearest pair until you end up with one big cluster.

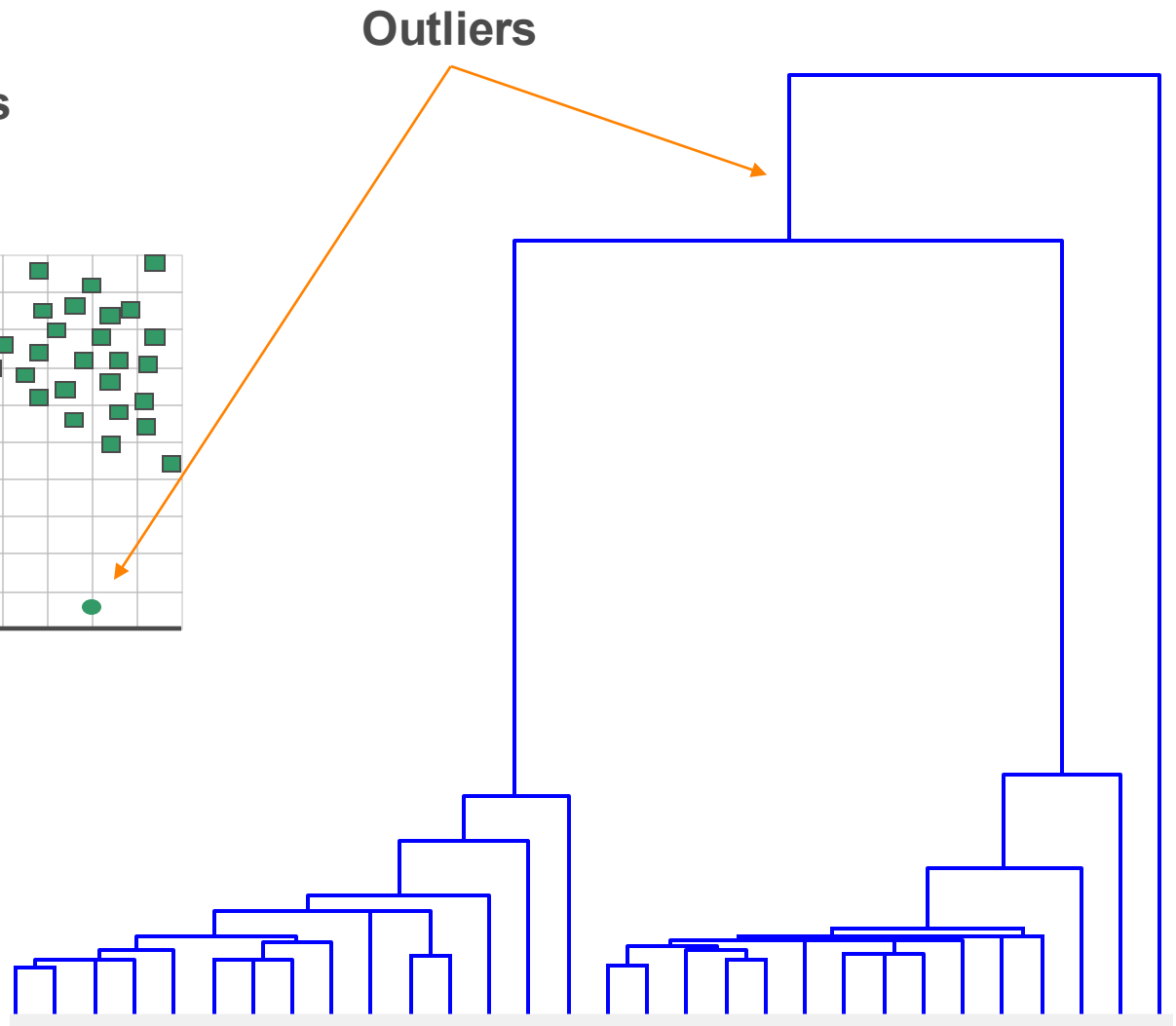
Top-Down (Divisive)

- **Start**: All points are in one big cluster.
- **Step**: Find the **best split** that breaks this cluster into two groups.
- **Repeat**: Apply the same "best split" rule to each subgroup until every point is alone.

Hierarchical Clustering

Practical use of a dendrogram: **spotting outliers**





Low horizontal joins → very **similar** groups

High horizontal joins → **quite different** and only merged once everything else was already joined.

Hierarchical Clustering

Key Takeaways

→ Strengths

No preset K required: You don't have to decide the number of clusters ahead of time.

Nested structure: Produces a full tree of groupings, revealing natural, multi-level patterns.

Intuitive visualization: Dendrograms make it easy to see how clusters form and split.

→ Limitations

Scalability: Its time and memory needs grow quadratically, so it doesn't work well on large data.

Greedy merges: Each step commits to a single merge, which can get stuck in suboptimal configurations.

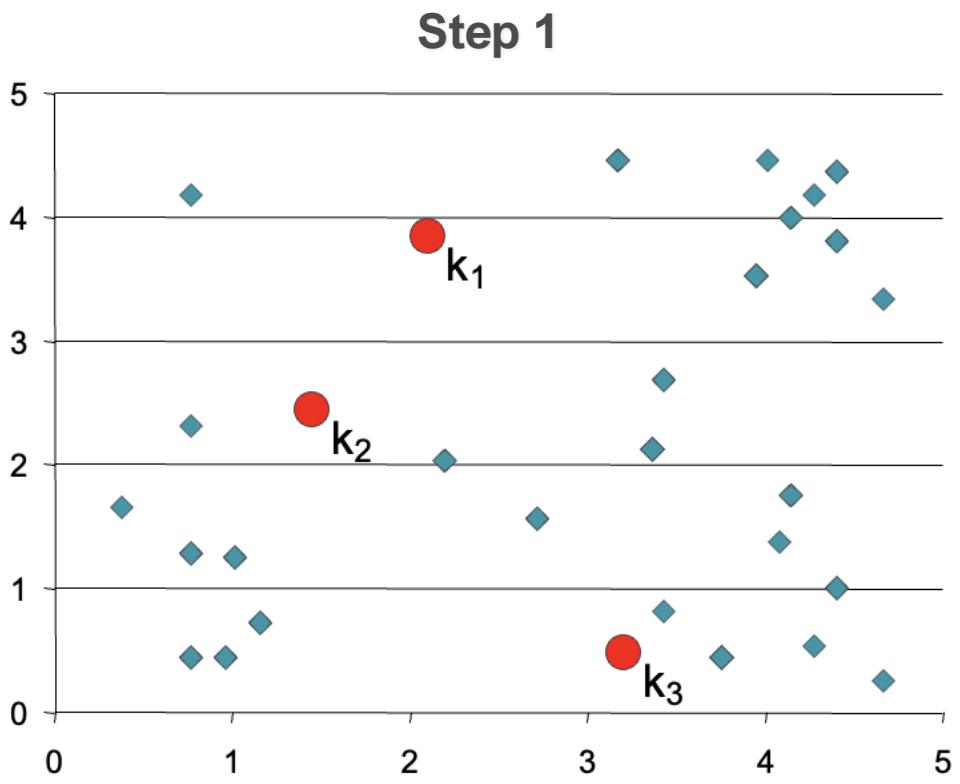
Subjective interpretation: Choosing where to "cut" the tree and what constitutes a meaningful cluster often relies on human judgment.

Partition Algorithm 1: K-Means

Algorithm Steps

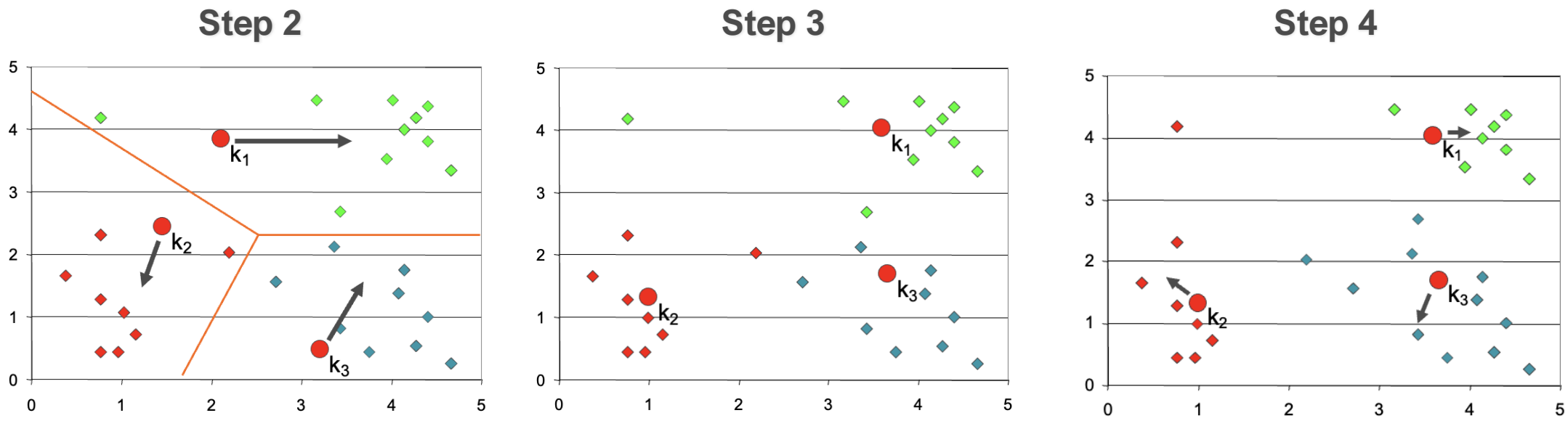
1. **Choose** the number of clusters (***k***).
2. **Initialize** ***k*** centroids (e.g., randomly).
3. **Assign** each data point to its **nearest** centroid.
4. **Recompute** centroids based on those assignments.
5. **Repeat** steps 3–4 **until** no point changes its cluster.

Distance Metric: **Euclidean Distance**



Partition Algorithm 1: K-Means

Distance Metric: **Euclidean Distance**



Partition Algorithm 1: K-Means

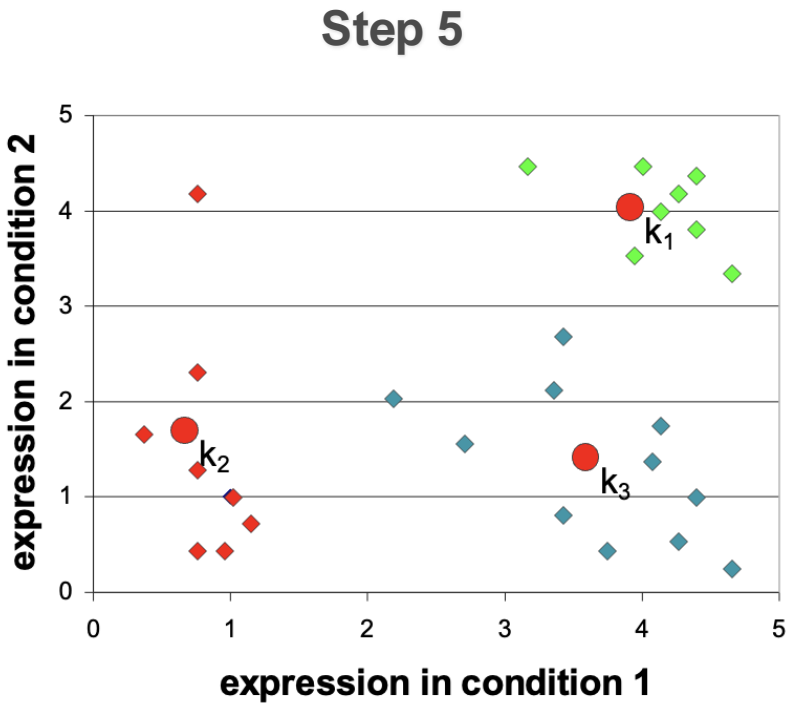
→ Strengths

Fast & scalable: Runtime is roughly O(n · k · t):

n = #points, k = clusters, t = iterations. Normally, $k, t << n$

Simple to implement & understand: Produces a full tree of groupings, revealing natural, multi-level patterns.

Deterministic convergence: Often terminates at a local optimum



→ Limitations

Only for numeric data: Requires a well-defined mean, so what about categorical data?

Must pick k in advance, Only finds convex, equally sized clusters, and

Vulnerable to outliers & noise

Mixture of Gaussians

K-means algorithm

- Assigned each example to exactly one cluster
- What if clusters are overlapping?
  - Hard to tell which cluster is right
  - Maybe we should try to remain uncertain
- Used Euclidean distance
- What if cluster has a non-circular shape?

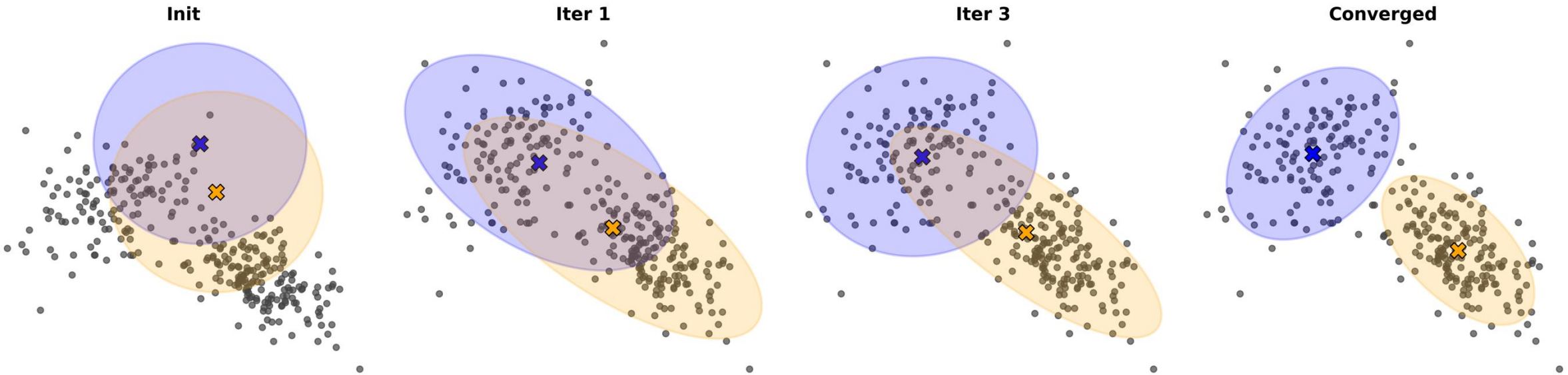
Soft membership

Points near a component's center get almost full membership in that component.

In overlap regions, **membership** is split (e.g., 70% A and 30% B)

Gaussian mixture models

- Clusters modeled as Gaussian distributions
- Each component j has its weight, mean, and covariance
- We seek the parameters that maximize the data likelihood
- **EM algorithm**: assign data to cluster with some probability



For each component x_i:

$$\{\pi_j, \mu_j, \Sigma_j\}_{j=1}^{K}$$

E-Step

$$\gamma_{i,j} = P(z_i = j \mid x_i) = \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(x_i \mid \mu_\ell, \Sigma_\ell)}$$

Share of point x_i that belongs to cluster **j**

M-Step

$$N_j = \sum_{i=1}^{N} \gamma_{i,j}$$

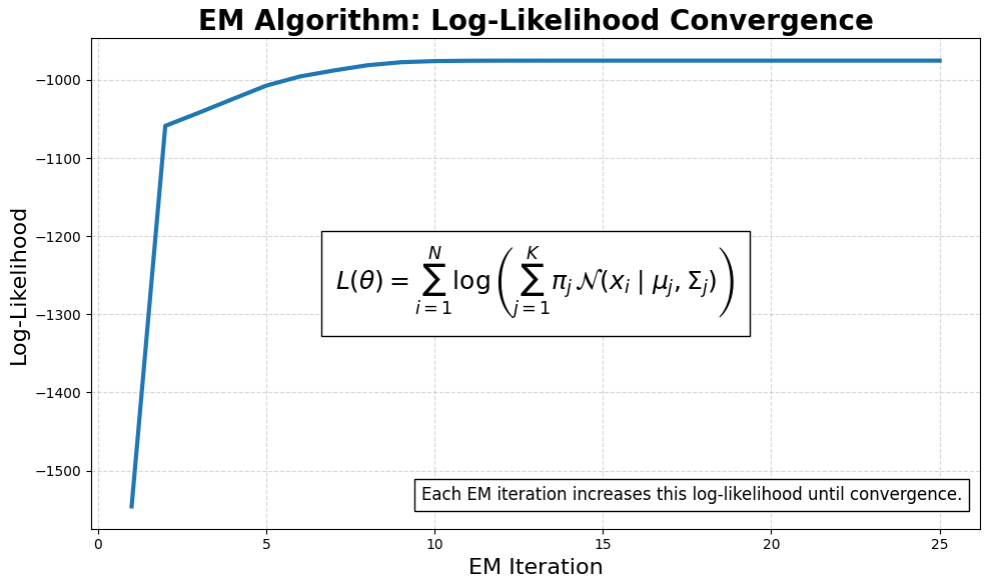
effective number of points in component **j**

New mean → μ_j moves to the center of its assigned points

Recompute → π_j , $\mu_j, \sum j$ → maximize the expected complete-data **log-likelihood**

Mixture of Gaussians

Find the mixture parameters $\boldsymbol{\theta} = \boldsymbol{\pi_j}, \boldsymbol{\mu_j}, \sum \boldsymbol{j}$ that make our data $\mathbf{X} = \{x_1, \ldots, x_N\}$ as likely as possible. Concretely, we maximize the **log-likelihood** →



→ Strengths

- **Soft assignments**: gives probabilities, not hard labels
- **Flexible shapes**: full covariances capture ellipses of any orientation
- **Probabilistic model**: provides likelihood scores and cluster weights
- **Handles overlap**

→ Limitations

- **EM is slower**: iterates over data, can be costly for large n
- **Requires K**: must choose number of components in advance
- **Init-sensitive**: poor starts can lead to bad local optima

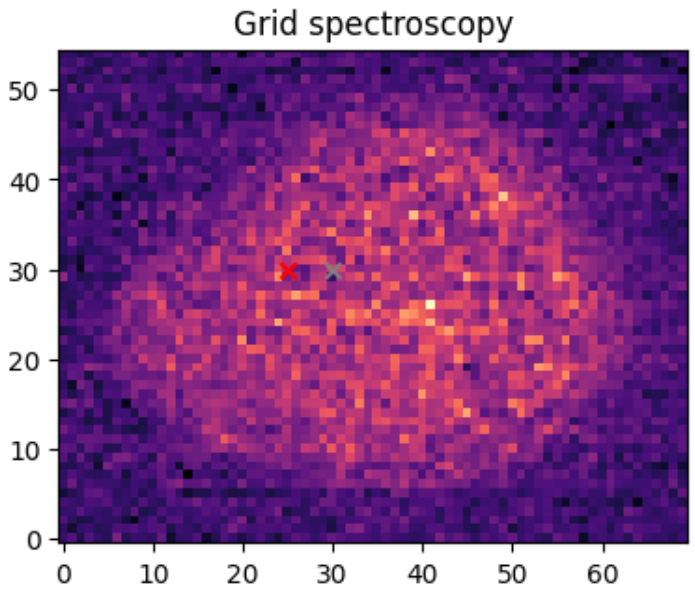
Clustering notebook

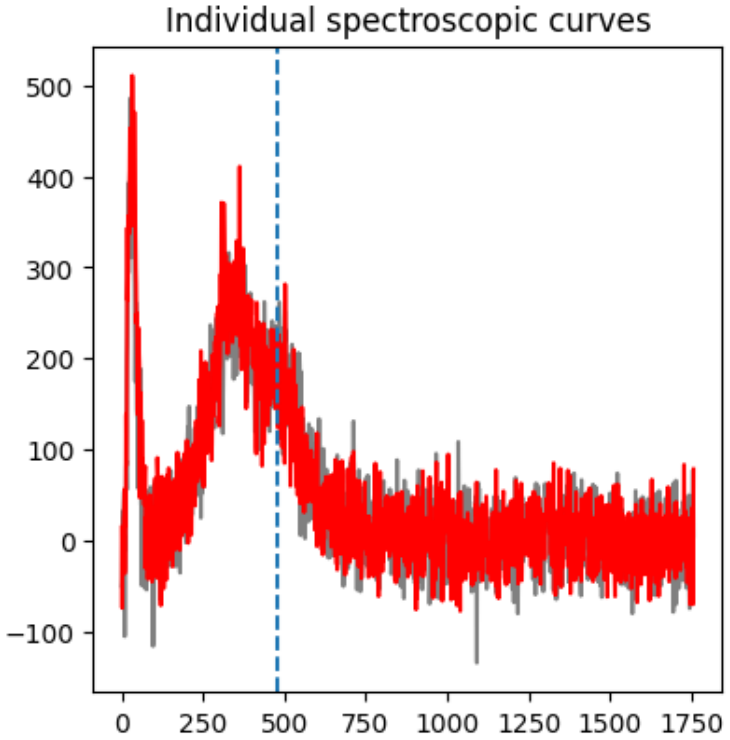
Graphene

https://colab.research.google.com/github/SergeiVKalinin/MLSTEM2024/blob/main/Day2/Day2_Clustering_graphene.ipynb#scrollTo=I_A0sdgaE4_V

Clustering of spectral data

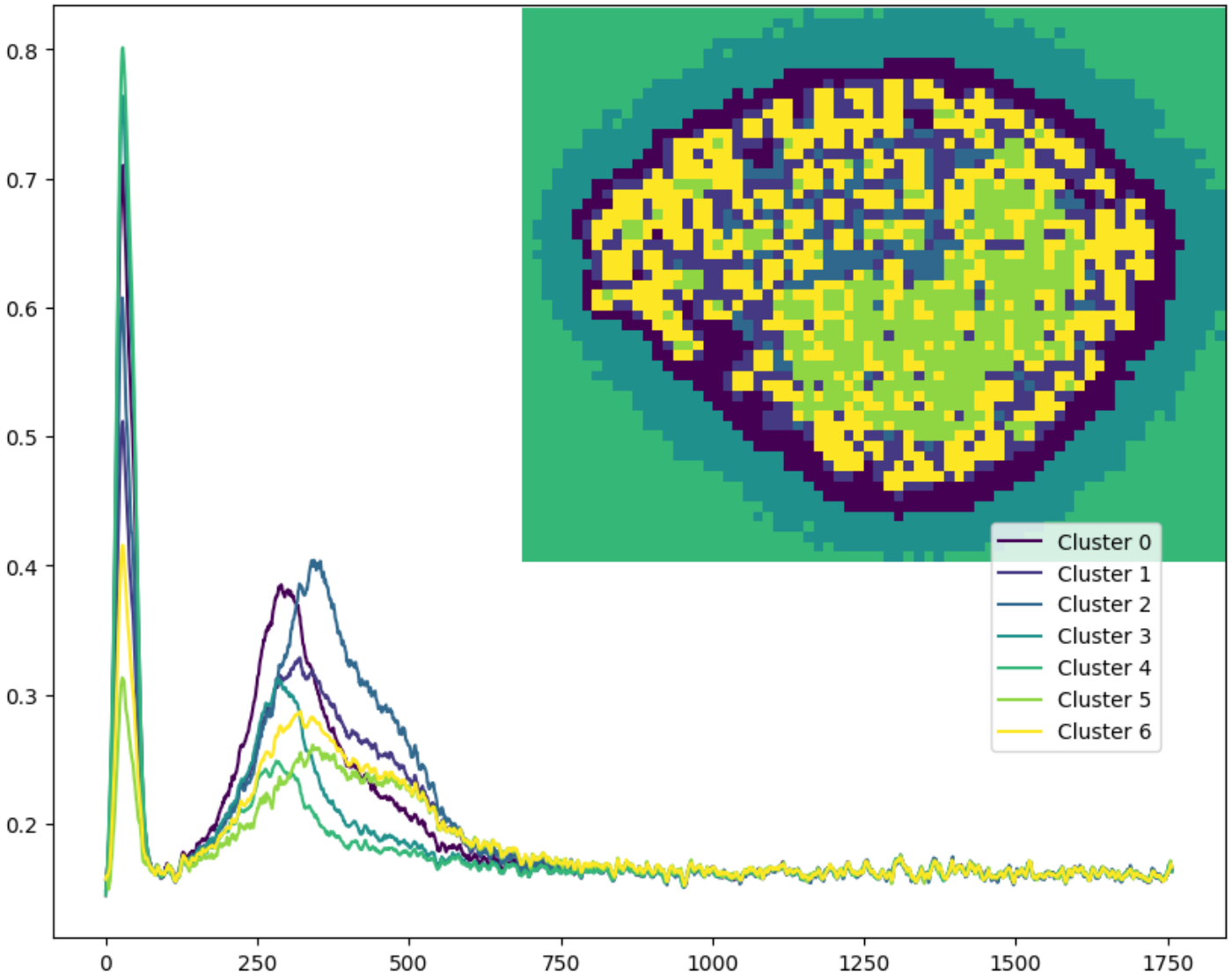
- The hyperspectral data set contains spectrum at each spatial position on the dense rectangular grid
- We use clustering to establish internal structure of this dataset





Clustering of spectral data

- Experiment with number of clusters
- Based on domain experience, explore the behavior of the components and images of class labels



Clustering notebook

EELS dataset

https://colab.research.google.com/github/SergeiVKalinin/MLSTEM2024/blob/main/Day3/Day3_Clustering_EELS.ipynb#scrollTo=0rOeVg5CQe3w