MA477: Data Science

Lesson 34 Outline — 21 April 2026

United States Military Academy, West Point
Instructor: MAJ Patrick Kuiper

---

1 Administrative

- Lesson Review
- Mentimeter
- Dim Reduction Lecture
- Coding Exercise

2 Clustering Lesson Objectives

- Understand the k-means and hierarchical clustering algorithms.
- Use functions in sklearn to perform clustering, such as DBSCAN.
- Understand practical issues in clustering.

3 Clustering: Motivation and Overview

Clustering is an **unsupervised learning** task in which we attempt to group observations into clusters so that observations in the same cluster are similar, while observations in different clusters are dissimilar. Unlike supervised learning, there is no response variable telling us the correct grouping. As a result, clustering is often used for:

- exploratory data analysis,
- customer or population segmentation,
- anomaly detection,
- preprocessing or feature engineering,
- identifying hidden structure in data.

The central difficulty in clustering is that there is often no single "correct" answer. Different algorithms define a cluster differently. Some methods seek compact spherical groups, while others seek groups connected by density or proximity. Thus, one of the most important practical questions in clustering is not only *how* to cluster, but also *what kind of structure* we believe is present in the data.

In this lecture we study three important clustering approaches:

1. **K-means clustering**,
2. **hierarchical clustering**,
3. **DBSCAN** (density-based clustering).

We also discuss practical issues such as scaling, distance choice, choosing hyperparameters, sensitivity to outliers, and evaluating results.

4 K-Means Clustering

4.1 Main Idea

K-means clustering attempts to partition the data into K clusters, where K is chosen in advance. Each cluster is represented by its **centroid**, which is the mean of the observations assigned to that cluster. The algorithm seeks clusters that are as compact as possible.

Formally, if the clusters are $C_1, \ldots, C_K$, then k-means attempts to minimize the **within-cluster sum of squares**:

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2,$$

where μ_k is the mean of the points in cluster C_k.

Thus, k-means prefers clusters in which points lie close to their assigned centroid.

4.2 Algorithm

The standard k-means algorithm proceeds as follows:

1. Choose K, the number of clusters.
2. Initialize K centroids, often randomly.
3. **Assignment step:** assign each observation to the nearest centroid.
4. **Update step:** recompute each centroid as the mean of the observations assigned to it.
5. Repeat the assignment and update steps until the assignments stop changing or the objective function stops improving.

This is an iterative algorithm. Each iteration improves or leaves unchanged the objective function, so convergence is guaranteed, but only to a **local minimum**, not necessarily the global minimum.

4.3 Why K-Means is Useful

K-means is popular because it is:

- simple to understand,
- computationally efficient,
- easy to implement,
- effective when clusters are compact and roughly spherical.

It works especially well when the clusters have:

- similar sizes,
- similar variances,
- roughly convex or spherical shapes,
- little contamination from extreme outliers.

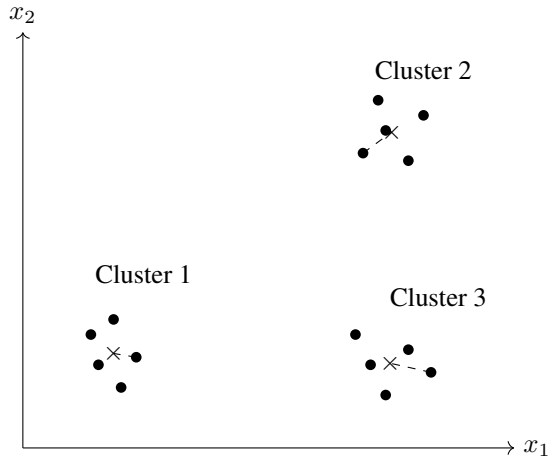
4.4 Limitations of K-Means

K-means also has important limitations:

- The number of clusters K must be chosen in advance.
- It is sensitive to initialization; different starting points can produce different final clusters.
- It is sensitive to outliers, since means can be pulled by extreme observations.
- It works poorly for non-spherical or highly unequal clusters.
- It depends strongly on the scale of the variables.

For example, if one feature is measured in dollars and another in inches, the larger-scale variable can dominate the distance calculations unless the data are standardized.

4.5 Diagram of K-Means



In this diagram, the points are grouped around three centroids, marked by $\times$. K-means repeatedly updates which centroid each point belongs to and then moves each centroid to the mean of its assigned points.

4.6 A Small Practical Note

Because k-means can converge to different local minima, it is common to run it multiple times with different random initializations and keep the best solution. In `sklearn`, this is handled automatically through multiple initializations.

5 Hierarchical Clustering

5.1 Main Idea

Hierarchical clustering builds a nested sequence of groupings rather than producing only one partition. The result is a tree-like structure called a **dendrogram**, which shows how observations or groups are merged over time.

There are two broad versions:

- **Agglomerative** (bottom-up): start with each observation as its own cluster and repeatedly merge clusters.
- **Divisive** (top-down): start with all observations in one cluster and repeatedly split.

In practice, agglomerative hierarchical clustering is much more common.

5.2 Agglomerative Hierarchical Clustering Algorithm

1. Begin with n observations, each in its own cluster.
2. Compute distances between all clusters.
3. Merge the two closest clusters.
4. Recompute distances between the new cluster and the remaining clusters.
5. Repeat until all observations have been merged into one cluster.

A major issue is how to define the distance between two clusters. Different definitions produce different hierarchical clusterings.

5.3 Linkage Methods

Common linkage methods include:

- **Single linkage:** distance between the two closest points in the clusters.
- **Complete linkage:** distance between the two farthest points in the clusters.
- **Average linkage:** average pairwise distance between points in the two clusters.
- **Ward linkage:** merge clusters to produce the smallest increase in within-cluster variance.

These linkage choices matter a great deal:

- Single linkage can produce long chains.
- Complete linkage tends to form compact clusters.
- Average linkage is often a compromise.
- Ward linkage often behaves somewhat like k-means in preferring compact groups.

5.4 Why Hierarchical Clustering is Useful

Hierarchical clustering is attractive because:

- it does not require choosing the number of clusters at the start,
- it provides a full hierarchy of possible clusterings,
- the dendrogram gives a visual summary of structure in the data.

Once the dendrogram is built, we can choose the final number of clusters by **cutting the dendrogram** at a chosen height.

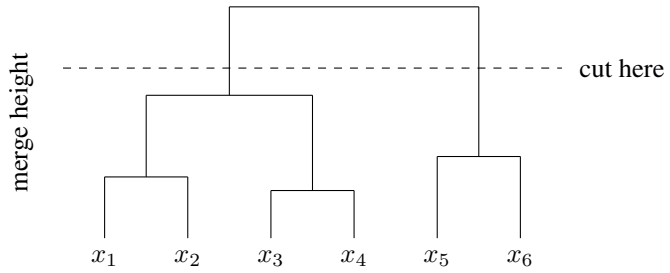
5.5 Limitations of Hierarchical Clustering

Hierarchical clustering also has drawbacks:

- It can be computationally expensive for large datasets.
- Once a merge is made, it cannot be undone.
- It is sensitive to the distance metric and linkage rule.
- It can be distorted by noise and outliers.

Thus, while hierarchical clustering is often very interpretable, it is usually better suited to small or moderate-sized datasets than extremely large ones.

5.6 Diagram of Hierarchical Clustering



The dendrogram records the sequence of merges. If we cut the tree at the dashed line, we obtain a specific number of clusters. Lower cuts yield more clusters; higher cuts yield fewer clusters.

6 DBSCAN

6.1 Main Idea

DBSCAN stands for **Density-Based Spatial Clustering of Applications with Noise**. Unlike k-means, DBSCAN does not try to partition the data into spherical groups around centroids. Instead, it looks for regions of high point density and treats sparse regions as boundaries or noise.

This makes DBSCAN especially useful when:

- clusters have irregular shapes,
- outliers are present,
- the number of clusters is not known in advance.

6.2 Core Concepts

DBSCAN depends on two main tuning parameters:

- ε (`eps`): the radius defining a neighborhood,
- `min_samples`: the minimum number of points required in that neighborhood for a point to be considered dense.

Using these, DBSCAN classifies points as:

- **Core points:** points with at least `min_samples` points in their ε-neighborhood,
- **Border points:** points near a core point but not dense enough to be core points themselves,
- **Noise points:** points not sufficiently close to any dense region.

Clusters are formed by connecting dense regions.

6.3 DBSCAN Algorithm (Density-Based Spatial Clustering)

- **Input:**
  - Dataset of points $\{x_1, x_2, \ldots, x_n\}$
  - Distance threshold ε (epsilon)
  - Minimum number of points MinPts
- **Definitions:**

  - ε-neighborhood of a point x_i: all points within distance ε of x_i
  - Core point: a point with at least MinPts points in its ε-neighborhood
  - Border point: a point within ε of a core point but not itself a core point
  - Noise point: a point that is neither a core point nor within ε of any core point
- **Algorithm:**
  - Mark all points as unvisited
  - Initialize cluster label $k = 0$
  - For each point x_i:
    - If x_i is already visited, continue to next point
    - Mark x_i as visited
    - Find all points in the ε-neighborhood of x_i
    - If the number of neighbors is less than MinPts:
      - Mark x_i as noise (temporarily)
    - Otherwise (if x_i is a core point):
      - Create a new cluster: $k = k + 1$
      - Assign x_i to cluster k
      - Initialize a list of neighbors
      - For each neighbor point:
      - If the neighbor is not visited:
      - Mark it as visited
      - Find its ε-neighborhood
      - If it has at least MinPts neighbors:
      - Add its neighbors to the list (expand the cluster)
      - If the neighbor is not yet assigned to any cluster:
      - Assign it to cluster k
- **Output:**
  - Cluster assignments for each point
  - Points not assigned to any cluster are labeled as noise

6.4 Why DBSCAN is Useful

DBSCAN has several major advantages:

- It can discover non-spherical clusters.
- It identifies noise points automatically.
- It does not require choosing the number of clusters in advance.

This is a major contrast with k-means.

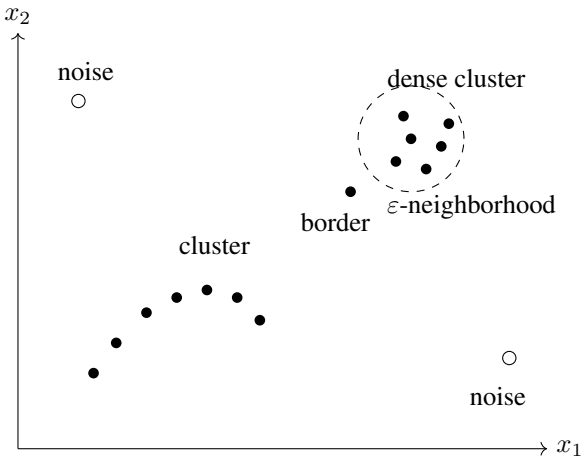
6.5 Limitations of DBSCAN

DBSCAN also has limitations:

- It can be sensitive to the choices of ε and `min_samples`.
- It struggles when clusters have very different densities.
- It can become less effective in high-dimensional settings where distances become less informative.

Thus, DBSCAN is often excellent for spatial or geometric data with clear density structure, but it is not universally superior.

6.6 Diagram of DBSCAN



This diagram illustrates the density-based viewpoint: DBSCAN can identify curved or irregularly shaped clusters, while isolated points are marked as noise rather than forced into a cluster.

7 Practical Issues in Clustering

7.1 Scaling Matters

Most clustering algorithms rely on distances. If one variable has a much larger scale than another, it can dominate the analysis. Therefore, standardizing predictors is often essential before clustering.

7.2 Distance Choice Matters

Euclidean distance is common, but it is not always the best choice. For some problems, Manhattan distance, cosine similarity, or domain-specific distance measures may be more appropriate.

7.3 There May Be No True Number of Clusters

Especially in real data, the notion of a cluster may be ambiguous. Different methods may suggest different structures, and several clusterings may all be reasonable.

7.4 Outliers Can Distort Results

Outliers can severely affect k-means and can influence hierarchical clustering. DBSCAN is often better at handling isolated points because it can classify them as noise.

7.5 High-Dimensional Data is Difficult

In high dimensions, distances become less informative, which makes clustering harder. Dimension reduction methods such as PCA are often used before clustering.

7.6 Evaluation is Hard

Since clustering is unsupervised, we usually do not know the correct labels. Thus, evaluation often relies on:

- interpretability,
- stability across runs,
- internal measures such as inertia or silhouette score,
- domain knowledge.

8 Clustering in `sklearn`

8.1 K-Means Example

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=1, n_init=10)
labels = kmeans.fit_predict(X)
```

8.2 Agglomerative Hierarchical Clustering Example

```
from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = hc.fit_predict(X)
```

8.3 DBSCAN Example

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.5, min_samples=5)
labels = db.fit_predict(X)
```

9 Comparison Table

Method	Main Idea	Strengths	Limitations
K-means	Partition data into K groups by minimizing within-cluster squared distance to centroids	Fast, simple, scalable, easy to interpret, works well for compact spherical clusters	Must choose K in advance, sensitive to initialization and outliers, poor for irregular shapes or unequal densities
Hierarchical clustering	Build a tree of nested clusters by repeatedly merging nearby clusters	No need to choose K initially, dendrogram gives rich structural view, flexible through linkage choice	Computationally expensive, sensitive to distance/linkage choice, early merges cannot be undone
DBSCAN	Find dense regions and label sparse points as noise	Finds irregular shapes, identifies outliers naturally, no need to specify number of clusters	Sensitive to `eps` and `min_samples`, struggles with varying densities, can perform poorly in high dimensions

10 Takeaway

The best clustering method depends on the kind of structure we believe exists in the data.

- Use **k-means** when you expect compact, roughly spherical clusters and need a fast method.
- Use **hierarchical clustering** when interpretability and a full cluster hierarchy are important.
- Use **DBSCAN** when clusters may have irregular shapes and when identifying noise is important.

In practice, clustering should rarely be treated as a push-button procedure. Good clustering requires careful attention to scaling, distance measures, algorithm assumptions, and interpretation of the final groups.

11 Worked Problems: Hierarchical Clustering and K-Means

11.1 Problem 1: Bottom-Up Hierarchical Clustering

Consider the following three data points, each with two features:

$$A = (1, 1), \qquad B = (2, 2), \qquad C = (5, 5).$$

Use **agglomerative hierarchical clustering** with **Euclidean distance** and **single linkage**. Compute the pairwise distances and show the order in which the clusters merge.

Solution

We begin by computing all pairwise Euclidean distances.

Distance between A and B

$$d(A, B) = \sqrt{(1-2)^2 + (1-2)^2} = \sqrt{(-1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$$

Distance between A and C

$$d(A, C) = \sqrt{(1-5)^2 + (1-5)^2} = \sqrt{(-4)^2 + (-4)^2} = \sqrt{16+16} = \sqrt{32} \approx 5.66$$

Distance between B and C

$$d(B, C) = \sqrt{(2-5)^2 + (2-5)^2} = \sqrt{(-3)^2 + (-3)^2} = \sqrt{9+9} = \sqrt{18} \approx 4.24$$

Thus, the pairwise distances are:

$$d(A, B) \approx 1.41, \qquad d(A, C) \approx 5.66, \qquad d(B, C) \approx 4.24$$

Initially, each point is its own cluster:

$$\{A\}, \qquad \{B\}, \qquad \{C\}$$

The smallest distance is between A and B, so these merge first:

$$\{A, B\}, \qquad \{C\}$$

Now compute the distance between cluster $\{A, B\}$ and cluster $\{C\}$ using **single linkage**. Single linkage defines the distance between two clusters as the *minimum* distance between any point in one cluster and any point in the other cluster.

So we compare:

$$d(A, C) \approx 5.66, \qquad d(B, C) \approx 4.24$$

Therefore,

$$d(\{A, B\}, \{C\}) = \min(5.66, 4.24) = 4.24$$

So the second merge occurs at distance 4.24:

$$\{A, B, C\}$$

Final Answer The clustering sequence is:

1. Merge A and B at distance $\sqrt{2} \approx 1.41$
2. Merge $\{A, B\}$ with C at distance $\sqrt{18} \approx 4.24$

11.2 Problem 2: K-Means Clustering

For this problem, interpret the four points as

$$A = (1, 1), \qquad B = (1, 3), \qquad C = (9, 4), \qquad D = (10, 11).$$

Use **k-means clustering** with $K = 2$. Suppose the initial cluster centers are chosen randomly as points B and A.

> Compute the distance from each point to each initial cluster center. Assign each point to the nearest center. Then compute the updated cluster centers after the first iteration.

Solution

The initial centroids are:

$$\mu_1^{(0)} = B = (1, 3), \qquad \mu_2^{(0)} = A = (1, 1)$$

We now compute the Euclidean distance from each point to each centroid.

Point $A = (1, 1)$ Distance to $\mu_1^{(0)} = (1, 3)$:

$$d(A, \mu_1^{(0)}) = \sqrt{(1-1)^2 + (1-3)^2} = \sqrt{0+4} = 2$$

Distance to $\mu_2^{(0)} = (1, 1)$:

$$d(A, \mu_2^{(0)}) = \sqrt{(1-1)^2 + (1-1)^2} = 0$$

So A is assigned to cluster 2.

Point $B = (1, 3)$ Distance to $\mu_1^{(0)} = (1, 3)$:

$$d(B, \mu_1^{(0)}) = 0$$

Distance to $\mu_2^{(0)} = (1, 1)$:

$$d(B, \mu_2^{(0)}) = \sqrt{(1-1)^2 + (3-1)^2} = \sqrt{4} = 2$$

So B is assigned to cluster 1.

Point $C = (9, 4)$ Distance to $\mu_1^{(0)} = (1, 3)$:

$$d(C, \mu_1^{(0)}) = \sqrt{(9-1)^2 + (4-3)^2} = \sqrt{8^2 + 1^2} = \sqrt{64+1} = \sqrt{65} \approx 8.06$$

Distance to $\mu_2^{(0)} = (1, 1)$:

$$d(C, \mu_2^{(0)}) = \sqrt{(9-1)^2 + (4-1)^2} = \sqrt{8^2 + 3^2} = \sqrt{64+9} = \sqrt{73} \approx 8.54$$

So C is assigned to cluster 1.

Point $D = (10, 11)$ Distance to $\mu_1^{(0)} = (1, 3)$:

$$d(D, \mu_1^{(0)}) = \sqrt{(10-1)^2 + (11-3)^2} = \sqrt{9^2 + 8^2} = \sqrt{81+64} = \sqrt{145} \approx 12.04$$

Distance to $\mu_2^{(0)} = (1, 1)$:

$$d(D, \mu_2^{(0)}) = \sqrt{(10-1)^2 + (11-1)^2} = \sqrt{9^2 + 10^2} = \sqrt{81+100} = \sqrt{181} \approx 13.45$$

So D is assigned to cluster 1.

Cluster assignments after the first assignment step Cluster 1:

$$\{B, C, D\}$$

Cluster 2:

$$\{A\}$$

Update the centroids The new centroid for cluster 1 is the mean of points B, C, and D:

$$\mu_1^{(1)} = \left(\frac{1+9+10}{3}, \frac{3+4+11}{3}\right) = \left(\frac{20}{3}, \frac{18}{3}\right) = \left(\frac{20}{3}, 6\right) \approx (6.67, 6)$$

The new centroid for cluster 2 is the mean of point A alone:

$$\mu_2^{(1)} = (1, 1)$$

Final Answer After one full iteration of k-means:

- Cluster 1 contains B, C, and D
- Cluster 2 contains A
- The updated centroids are

$$\mu_1^{(1)} = \left(\frac{20}{3}, 6\right), \qquad \mu_2^{(1)} = (1, 1)$$