Unsupervised Learning: Clustering and Dimensionality Reduction

UNIVERSITY OF OXFORD
Mathematical Institute

Lida Kanari

Mathematical Institute
University of Oxford

Introduction to Machine Learning, October 2025

Oxford Mathematics

Table of Contents

Unsupervised Learning

- Learn patterns in data without labels.
- Discover structure: clusters, manifolds, or lower-dimensional representations.

Differences from supervised learning:

- No ground-truth labels Y.
- Harder evaluation (no direct prediction error).
- Focus on similarity, structure, and representation.

Challenges:

- Requires large datasets.
- Sensitive to hyperparameters.
- Results can be hard to interpret.

Unsupervised Learning

We have no labels Y. What can we use to assess similarity between data, based only on the features X?

Problem Setup

Setup:

$$x_1, x_2, \ldots, x_n \in \mathcal{X}, \quad \text{no labels.}$$

Goal: Find structure such that:

- In clustering: assign each x_i to a cluster label $c_i \in \{1, \ldots, K\}$.
- In dimensionality reduction: map $x_i \mapsto z_i \in \mathbb{R}^d$ with $d \ll \dim(\mathcal{X})$.

Objective: Minimize within-group distances and maximize between-group distances.

Clustering: Basic Idea

Definition: Partition n data points into K groups such that points in the same group are "similar".

Key idea: we need to define a "distance" to assess similarity between data.

Main approaches:

- Centroid-based (e.g., K-means).
- Density-based (e.g., DBSCAN).
- Hierarchical clustering.

Clustering Example

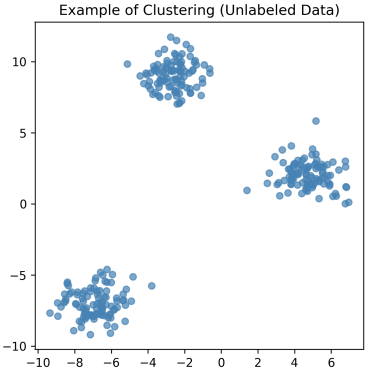


Figure: Example of 2D data points.

Clustering Example

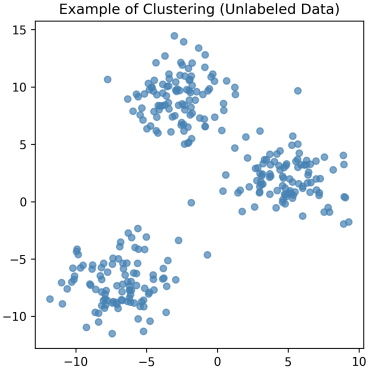


Figure: Example of 2D data points.

Clustering Example

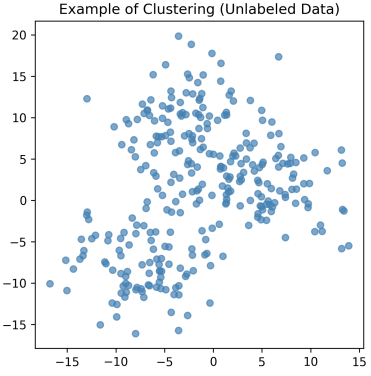


Figure: Example of 2D data points.

K-Means Algorithm

Idea: Partition data into K clusters, each represented by its centroid.

1. Randomly initialize K centroids.
2. Assign each point to the nearest centroid.
3. Update centroids as the mean of their assigned points.
4. Repeat steps 2 to 3 until the algorithm converges.

Objective:

$$\min_{c_1,\ldots,c_K} \sum_{i=1}^{n} \|x_i - \mu_c\|^2.$$

Objective: Partition n datapoints $\{x_i\}_{i=1}^n$ into K clusters so that points within a cluster are as close as possible to their cluster center.

- Each cluster C_j is represented by its centroid (mean) μ_j.
- We seek clusters that minimise within-cluster variance.
- Equivalent to minimising the sum of squared distances between each point and its assigned centroid.

K-Means Objective

We jointly optimise cluster assignments and centroids:

$$\min_{\{C_j\},\{\mu_j\}} W(C) = \sum_{j=1}^{K} \sum_{i \in C_j} \|x_i - \mu_j\|^2.$$

Note:

- $\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i$ — the mean of cluster j.
- Problem is **NP-hard** even for $K = 2$ in $\mathbb{R}^D$.
- But becomes easy if we fix either centroids or assignments.

K-Means Objective

Two simple subproblems:

- Assignment step: assign each point to a cluster.
- Update step: compute the mean values for each cluster.

Alternate between these two steps $\rightarrow$ *K-Means algorithm*.

K-Means — Initialization

Initialization: Choose K initial centroids $\mu_1, \ldots, \mu_K$.

- We can start by selecting K random data points.

K-Means — Initialization

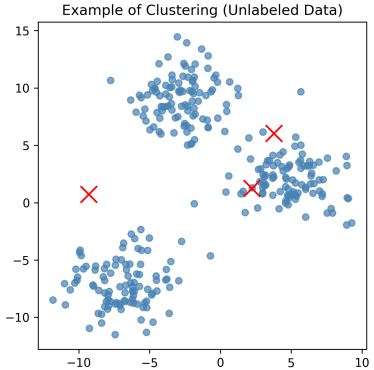


Figure: Initialization of Kmeans.

K-Means — Initialization

Initialization: Choose K initial centroids $\mu_1, \ldots, \mu_K$.

- We can start by selecting K random data points.
- Improvement: spread out initial centers to improve convergence.

K-Means — Initialization

Initialization: Choose K initial centroids $\mu_1, \ldots, \mu_K$.

- We can start by selecting K random data points.
- Improvement: spread out initial centers to improve convergence.

Good practice:

- Run the algorithm multiple times with different initializations.
- Keep the result with smallest objective $W(C)$.

K-Means — Iteration Step

Repeat until convergence:

1. **Assignment step:** Assign each point to its closest centroid:

$$C_j = \{i : j = \arg\min_{j'} \|x_i - \mu_{j'}\|^2\}.$$

2. **Update step:** Recompute centroids using current assignments:

$$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i.$$

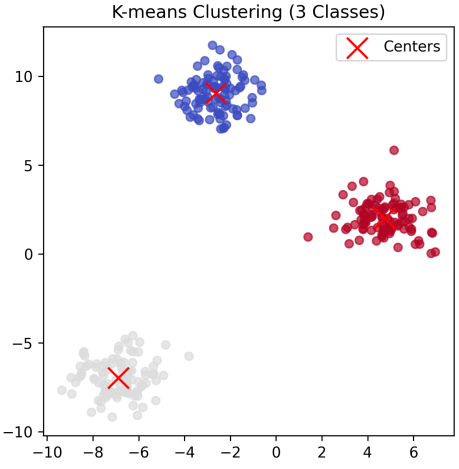
K-Means — Stopping criterion

Stopping criterion:

- Cluster assignments no longer changes.
- Objective $W(C)$ only improves by less than a threshold.

Output: Final centroids $\{\mu_j\}$ and data points assigned to classes $\{C_j\}$.

K-Means Visualization



K-Means - Converge

Does the K-Means algorithm always converge?

Does the K-Means Algorithm Always Converge?

Yes!

1. There are finitely many possible partitions of n points into K clusters.
2. $W(C)$ (within-cluster sum of squares) is either the same or **decreases** in each update step.
3. Therefore, the algorithm must stop after a finite number of iterations.

K-Means Limitations

However, there are some limitations:

- The algorithm converges to a **local minimum**, there is no guarantee for the global one.
- Convergence speed: often fast in practice, but can be slow in the a "worse-case" situation.
- Assumes clusters are roughly **spherical** (or convex).
- The algorithm is sensitive to initializationn and outliers.
- Number of clusters need to be defined in advance.

K-Means Limitations — Shape Assumptions

To improve robustness:

- Run multiple times with random initializations.
- Select the run that minimizes $W(C)$.

K-Means Alternatives

When to avoid K-Means:

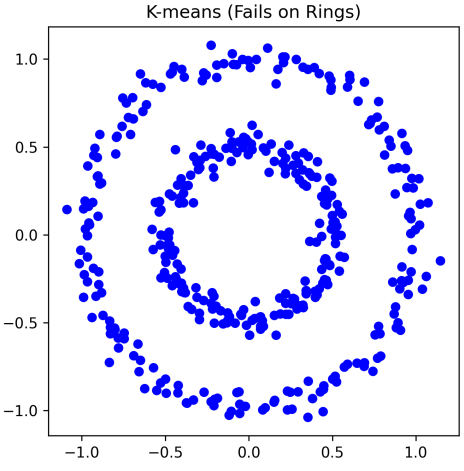
- Clusters with non-spherical shapes (e.g., concentric rings).
- Highly unbalanced cluster sizes or densities.
- Non-Euclidean or categorical feature spaces.

Alternatives:

- **Spectral clustering** — embeds data via Laplacian eigenvectors before clustering.
- **Hierarchical clustering** — Separates the data hierarchically based on distances.

Takeaway: K-Means is simple, fast, and effective for spherical clusters, but struggles on complex geometries or uneven data.

Limitations of K-Means



DBSCAN: Density-Based Clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

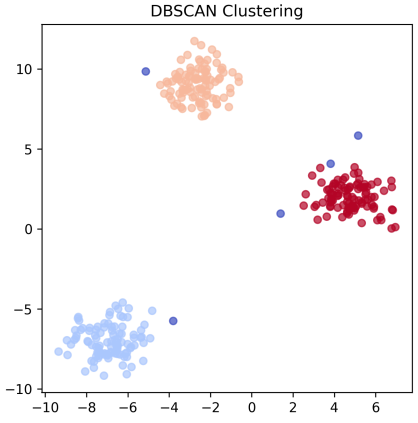
- Groups densely packed points.
- Identifies outliers as noise.
- Does not require specifying K.

Parameters:

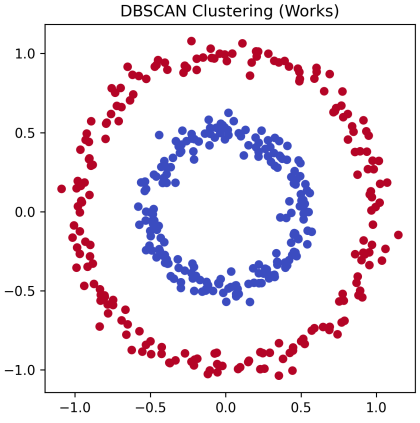
- ε: neighborhood radius.
- *MinPts*: minimum points to form a dense region.

Output: Clusters and noise points (outliers).

DBSCAN Example



DBSCAN Example



Comparing K-Means and DBSCAN

K-Means:

- Requires number of clusters K.
- Assumes spherical, similar-sized clusters.
- Sensitive to outliers.

DBSCAN:

- No need for K.
- Finds arbitrarily shaped clusters.
- Robust to outliers.
- Typically more expensive.

Choosing the Number of Clusters

Problem: Algorithms like K-means require K.
Strategies:

- Visual inspection (plots, dendrograms).
- Statistical criteria (AIC, BIC for model-based clustering).
- Heuristics: **Elbow** and **Silhouette** methods.

Elbow Method

Idea: Plot within-cluster sum of squares (WCSS) vs. K:

$$WCSS(K) = \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2.$$

The "elbow" indicates a balance between compactness and complexity.

Elbow Method

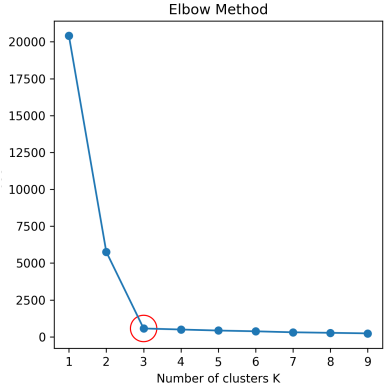


Figure: Elbow method to select optimal K.

Silhouette coefficient:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

where $a(i)$ is the average distance to same-cluster points and $b(i)$ to the nearest other cluster.

Interpretation:

- $s(i) \approx 1$: well-clustered.
- $s(i) \approx 0$: on boundary.

Silhouette Method

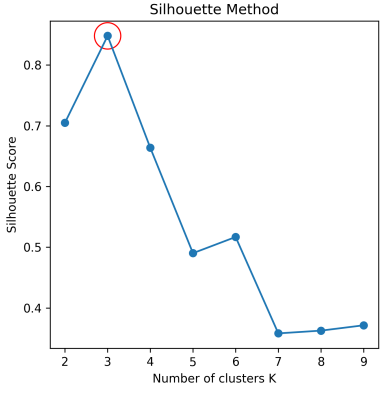


Figure: Silhouette method to select optimal K.

Hierarchical Clustering

Idea: Build a hierarchy of clusters by defining an iterative process.

- Start by computing the distances of all the points in our dataset.
- Then, based on the distances, we can distribute points into clusters iteratively.

Strategies to distribute points iteratively:

- **Agglomerative:** start with singletons, merge clusters iteratively.
- **Divisive:** start with one cluster, recursively split.

Hierarchical Clustering

Strategies to distribute points iteratively:

- **Agglomerative:** start with singletons, merge clusters iteratively.
- **Divisive:** start with one cluster, recursively split.

Linkage criteria:

- Single-link (minimum distance).
- Complete-link (maximum distance).
- Average-link (mean pairwise distance).

Hierarchical Clustering Example

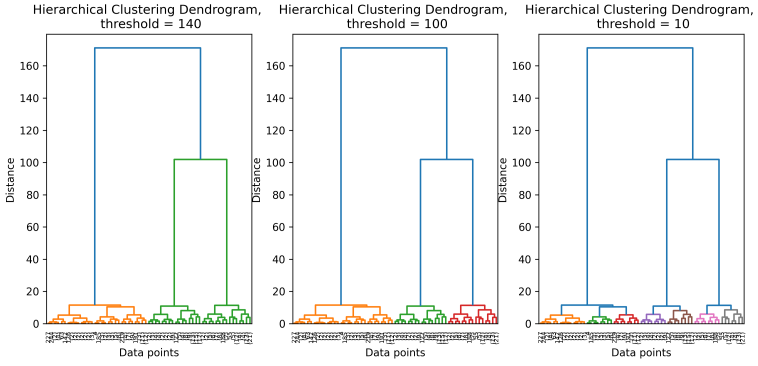


Figure: Hierarchical clustering visualized as a dendrogram.

The Curse of Dimensionality

Problem: As dimensionality d increases:

- Distances between points become less meaningful:

$$\frac{\max_i \|x_i - \mu\| - \min_i \|x_i - \mu\|}{\min_i \|x_i - \mu\|} \to 0.$$

- Nearest neighbor search becomes unreliable.
- The data requirements for statistical significance increases exponentially.

Implication: Clustering performance deteriorates in high-dimensional spaces.

Dimensionality Reduction: Motivation

- Remove noise and redundancy.
- Visualize high-dimensional data.
- Mitigate the curse of dimensionality.

Principal Component Analysis (PCA)

- Identify the directions that explain the most variance in the data.

Principal Component Analysis (PCA)

Goal: Find a linear projection that maximizes the variance in the data.

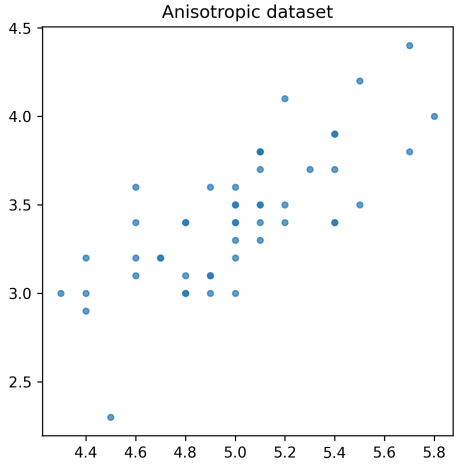
$$\max < (x_i) >_{i=1}^{N}$$

Solution: Find the eigenvalues and eigenvectors of the covariance matrix

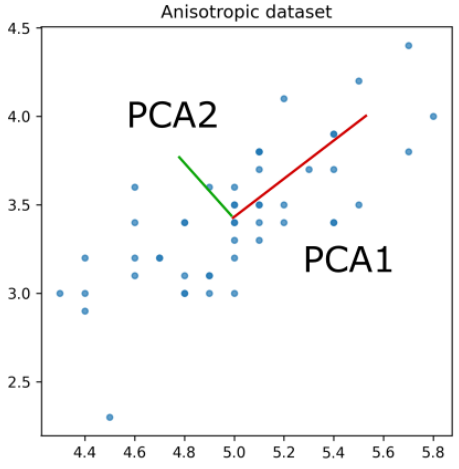
$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^\top.$$

Interpretation: PCA finds directions of maximal variance and orthogonal axes for projection.

Principal Component Analysis (PCA)



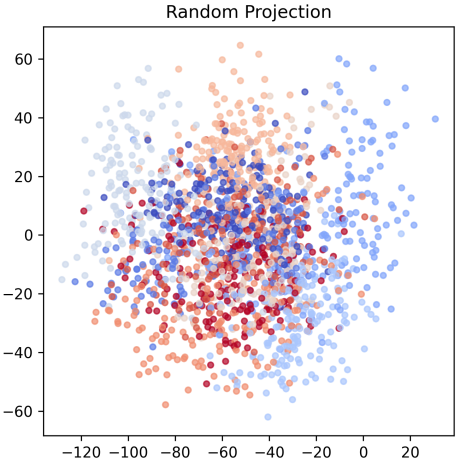
Principal Component Analysis (PCA)



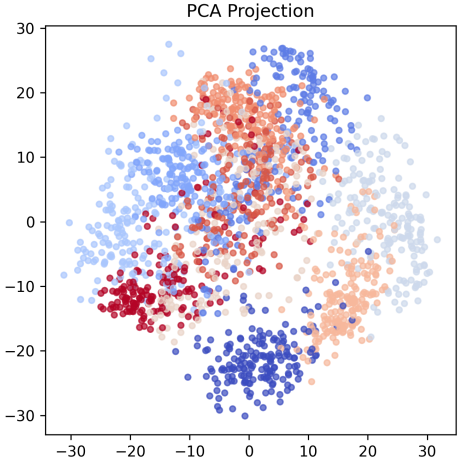
Algorithm: Find a set of orthonormal vectors $v_1, v_2, \ldots, v_k$.

- The first principal component PCA1 is the direction of largest variance v_1.
- The first principal component PCA2 is the direction v_2 of the second maximum variance, that is orthogonal to v_1.

Dimensionality reduction



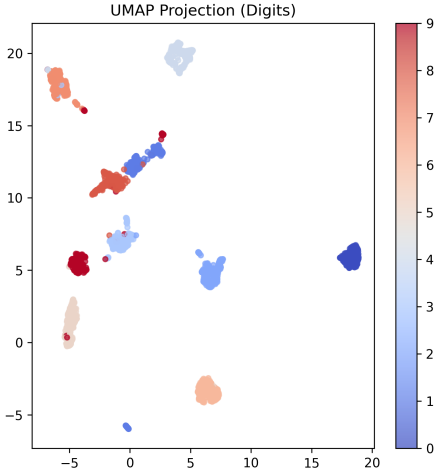
PCA



Uniform Manifold Approximation and Projection (UMAP):

- Graph-based manifold learning.
- Preserves local neighborhoods.
- Scales well to large datasets.

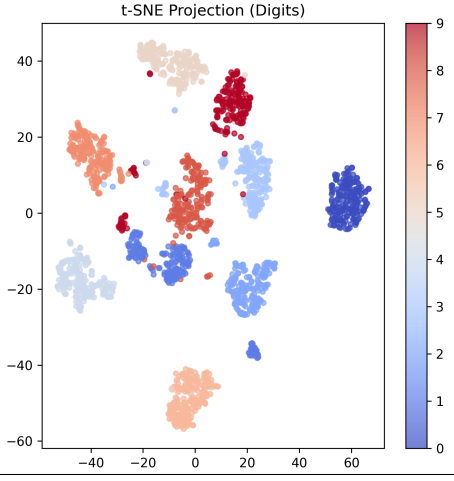
Manifold Learning: UMAP



t-Distributed Stochastic Neighbor Embedding (t-SNE):

- Preserves local similarities using probabilistic neighborhoods.
- Effective for 2D/3D visualization.
- Can distort global structure; not ideal for quantitative analysis.

Manifold Learning: t-SNE



Spectral clustering

Spectral clustering is similar to combining a dimensionality reduction method, such as PCA, and then perform Kmeans.

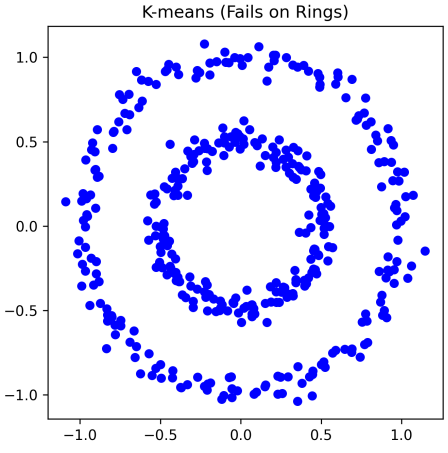
Spectral clustering algorithm:

- Construct a graph from the data using a similarity measure:

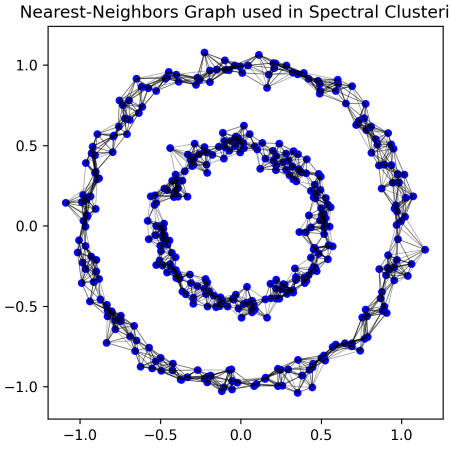
$$s_{i,j} = \exp(-|x_i - x_j|^2/\sigma)$$

- Generate the K-nearest neighbors subgraph, based on similarity $s_{i,j}$
- The weights of the edges is $s_{i,j}$
- Use the graph to partition the dataset into clusters, based on the laplcian.

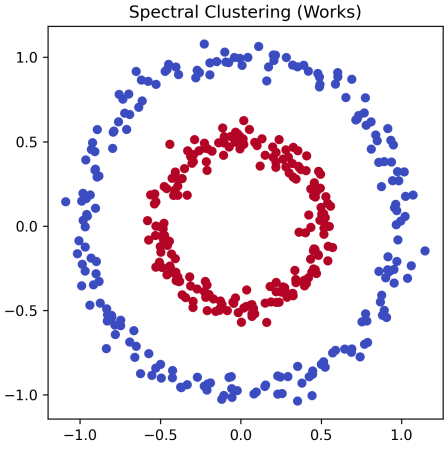
Spectral clustering for two circles



Spectral Clustering



Spectral Clustering



Closing Remarks

Unsupervised classification uses the input data $\mathcal{X}$ and their respective distances to generate clusters that minimize the distances of the data **within** a class and maximize the distance **between** classes.

- K-means is an efficient algorithm to solve simple clustering problems.
- However, K-means converges to a local minimum, it is not always fast and assumes spherical distribution of the data.
- Effective alternatives are density based algorithms and spectral algorithms.
- Combining the clustering method with a dimensionality reduction method can optimize the computational cost and result in more accurate clustering.

Thank you! Questions?