Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Unsupervised ML ⚇

[Dr. Hanjo Odendaal]

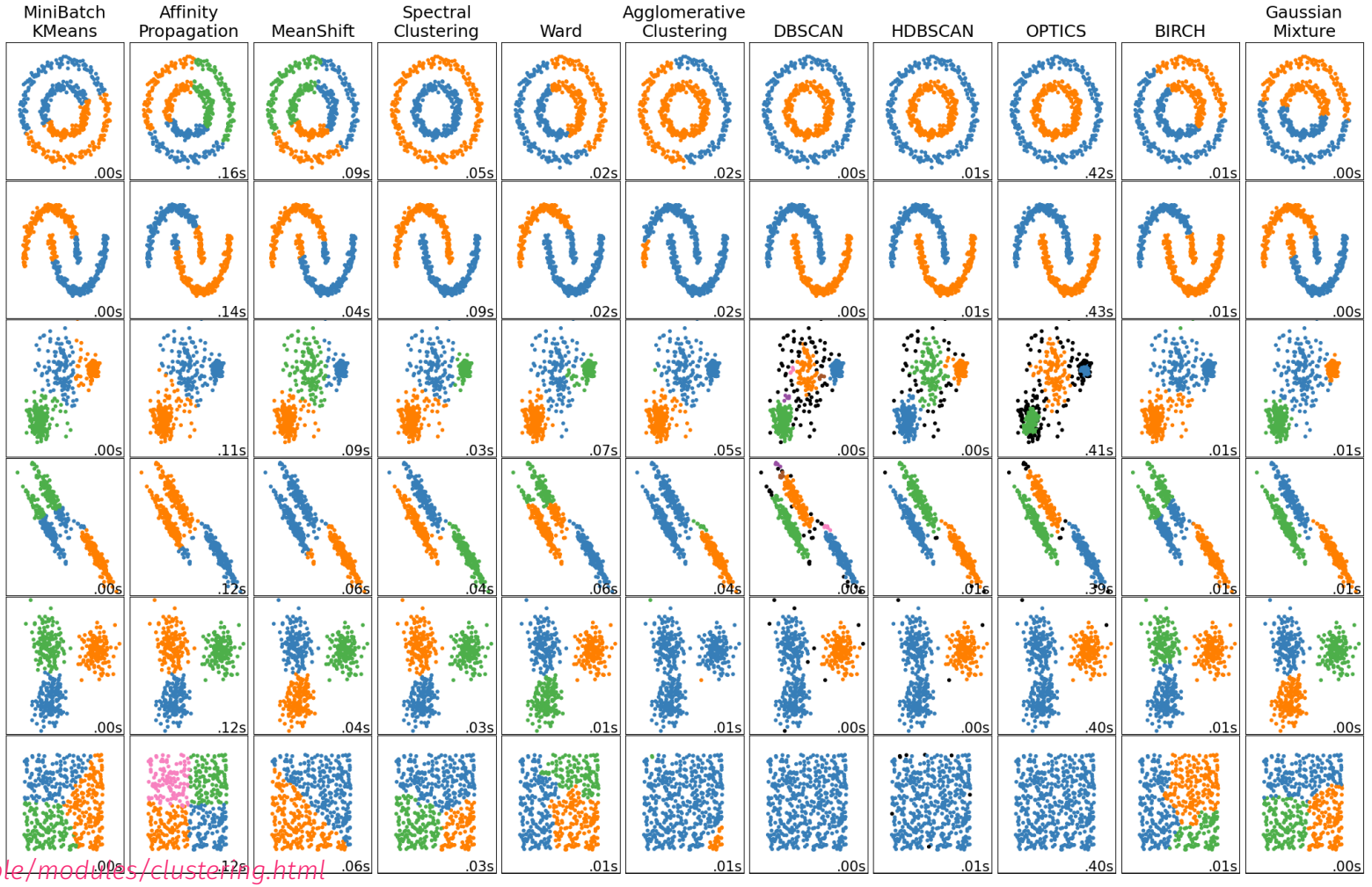
What is *unsupervised* learning?

- We only observe the **inputs** X - there are **no labels** y telling us what the structure should be.
- The **goal** is to let the model explore how the data might **fit together**, like sorting **Lego blocks**:
  - which pieces cluster together?
  - which ones are out of place?
  - do they form repeating shapes or patterns?
- Unlike supervised learning, where we follow instructions to build a model, here we **let the blocks guide us** - discovering patterns that emerge from their shape, size and colour.

> *Turn raw data into a map of itself - let the data tell its own story*

Domain	Example objective
Marketing	Customer segmentation for targeting & pricing
Finance	Detect unusual transactions (fraud / AML)
Genomics	Cluster gene-expression profiles
Manufacturing	Group sensor streams to spot fault patterns
NLP / Search	Topic modelling, word embeddings
Recommender	Cold-start grouping of new users/items

Horses for courses



https://scikit-learn.org/stable/modules/clustering.html

Clustering Families: The Landscape

Unsupervised clustering algorithms can be grouped by their *core strategy*:

- **Similarity-based**: assign points to nearest cluster centers
- **Connectivity-based**: merge or split based on distance chains
- **Model-based**: fit probabilistic distributions to the data
- **Density-based**: find dense areas of points, separate sparse regions
- **Dim. Reduction + Clustering**: learn a low-dimensional structure before grouping

Family	Prototype	Key Idea
Similarity-based	K-Means	Minimize within-cluster variance
Connectivity-based	Agglomerative	Build a tree via pairwise distances
Model-based	Gaussian Mixtures	Fit data as weighted combinations of densities
Density-based	DBSCAN, HDBSCAN	Clusters = high-density areas separated by gaps
Dimensionality-driven	PCA, t-SNE, UMAP	Reduce dimension, then cluster hidden structure

K-Means Clustering

Partition n observations into K clusters by minimizing the **within-cluster sum of squares**:

$$\min_{\{\mu_k\}} \sum_{i=1}^{n} \left\| x_i - \mu_{c(i)} \right\|^2$$

- μ_k is the centroid of cluster k, $c(i)$ is the cluster assigned to point x_i

Algorithm (Lloyds method)

1. **Initialise**: Select K initial centroids
2. **Assign**: Allocate each point to its *nearest* centroid
3. **Update**: Recalculate each centroid as the **mean of assigned points**
4. Repeat until convergence (no change in assignments or centroids)

Key Hyperparameters & Considerations

- K: Number of clusters (use elbow method, silhouette score, gap statistic to choose)
- Initialisation: Random vs. **k-means++** (more stable)
- **Scaling**: All features should be on comparable scales (standardization usually essential)
- Distance metric: Euclidean (default), but variants exist for others

> ⚠️ K-means assumes convex, equally-sized clusters and is sensitive to outliers.

K-Means Clustering

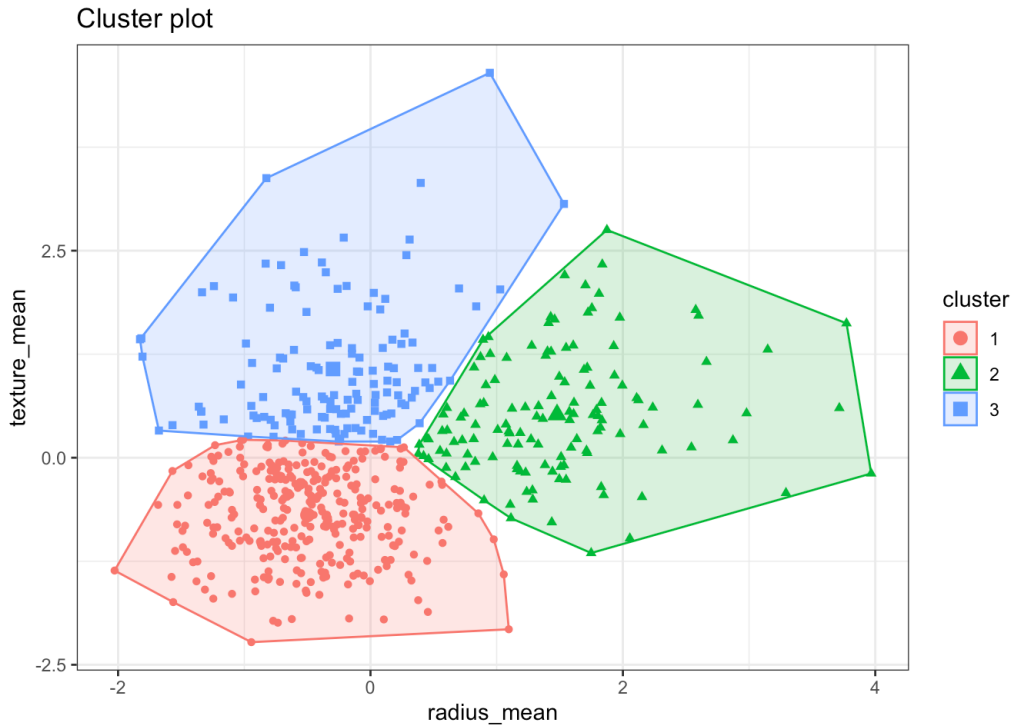
Partition n observations into K clusters by minimizing the **within-cluster sum of squares**:

$$\min_{\{\mu_k\}} \sum_{i=1}^{n} \left\| x_i - \mu_{c(i)} \right\|^2$$

- μ_k is the centroid of cluster k, $c(i)$ is the cluster assigned to point x_i

Algorithm (Lloyds method)

1. **Initialise**: Select K initial centroids
2. **Assign**: Allocate each point to its *nearest* centroid
3. **Update**: Recalculate each centroid as the **mean of assigned points**
4. Repeat until convergence (no change in assignments or centroids)



The Hard Boundary Problem

What is it?

- In hard clustering, each data point is assigned to exactly one cluster.
- There's *no overlap*, no uncertainty - the assignment is binary:

> You belong to Cluster 2, full stop.

- In **k-means**, this is due to the assignment step:
  - Each point is allocated to its *nearest centroid* (Euclidean distance).
  - Even if a point lies *equally between two clusters*, it must pick one.
  - There's no fuzzy or probabilistic interpretation.

When is it a problem?

- When **clusters overlap**, e.g., noisy or continuous domains like:
  - **Economics**
  - **Genomics**
  - **Natural language**
- When you want to model *uncertainty* in membership (e.g. in customer segments)
- When points lie *near boundaries*, risking unstable or misleading assignments

> This can distort the cluster structure, especially at the edges.

Hierarchical Clustering

Agglomerative clustering builds a hierarchy of clusters *bottom-up*

1. Start with each point in its **own cluster**
2. Iteratively **merge** the two **closest** clusters
3. Continue until all points are merged into one large cluster

The result is a **dendrogram** - a tree showing how clusters combine at various distance thresholds.

A horizontal cut through the tree yields **flat clusters**. 💡
No need to pre-specify K - choose it visually or set a height cut-off, but you have to set linkage criteria:

- Single (minimum distance)
- Complete (maximum distance)
- Average
- Ward (minimizes total within-cluster variance)

Key Properties & Considerations

- No need to define K in advance
- Works well when the goal is to explore or visualize nested relationships

Limitations:

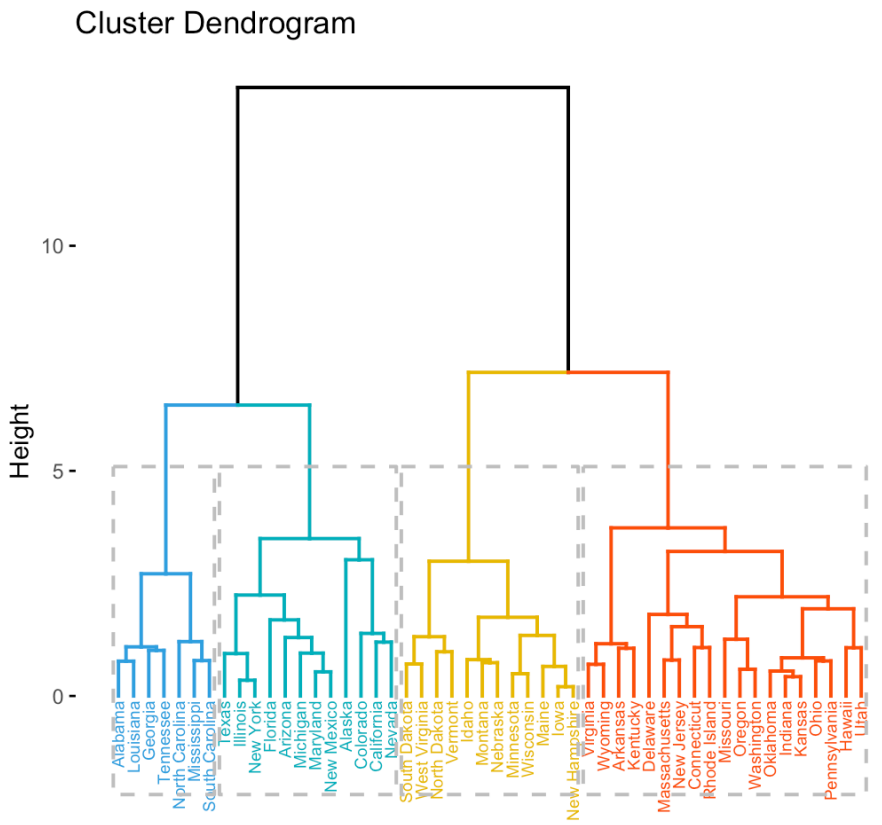
- Sensitive to **noise** and feature **scaling**
- Requires a full distance matrix: memory = $O(n^2)$

> ⚠️ Once two points are merged, they cannot be separated. Ideal when interpretability and structure are more important than scalability.

Hierarchical Clustering

This plot is a **dendrogram**, the standard visual output of hierarchical clustering.

- Each 🌿 *leaf* at the bottom is an individual data point (here: US states).
- As you move *upward*, points are merged into clusters based on proximity.
- The **height** of each merge reflects the **distance** (or dissimilarity) between merged clusters.
- The **coloured branches** show a specific number of clusters - in this case, 4 distinct groups.
- The **dashed horizontal line** (cut height) determines where to "slice" the tree to get flat clusters.



Model-Based Clustering (Gaussian Mixtures)

Model-based clustering assumes that data is generated from a **mixture of distributions** - one per cluster.

For K clusters, each point is assumed to come from a *weighted density*:

$$p(x_i) = \sum_{k=1}^{K} \pi_k \, f_k(x_i \mid \theta_k)$$

- π_k: mixing proportion, prior probability of cluster k
- f_k: probability density (often multivariate Gaussian)
- θ_k: parameters of cluster k (mean, covariance)

The model is estimated via the **Expectation-Maximization (EM)** algorithm:

1. **E-step**: compute soft assignments

$$\gamma_{ik} = \Pr(z_i = k \mid x_i)$$

2. **M-step**: update parameters π_k, θ_k

Model-Based Clustering (Gaussian Mixtures)

Imagine the data as being built from **K different Lego kits**.

- Each kit has its own shape, color and size - a different *distribution*.
- Every data point is made by **mixing pieces** from these kits (we don't know which one exactly, just the probabilities).

The **EM algorithm** is like playing a two-step Lego guessing game:

1. **E-step**: For each Lego piece (data point), guess how likely it came from each kit

   > *This block looks 70% like it came from Kit A, 30% from Kit B.*

2. **M-step**: Use those guesses to rebuild better versions of the kits

   > *Let's shift Kit A's blueprint closer to the pieces that look like it.*

Repeat until the kits explain the pieces well.

> This gives **soft clustering**: points belong to clusters in degrees, not absolutes.

Nice visualisations: https://ryanwingate.com/intro-to-machine-learning/unsupervised/gaussian-mixture-models-and-cluster-validation/

Evaluating Clusters: Internal Indices

✅ When we don't have ground truth...

Internal validation metrics help us evaluate clustering **without labels**. One of the most widely used is the **Silhouette coefficient**:

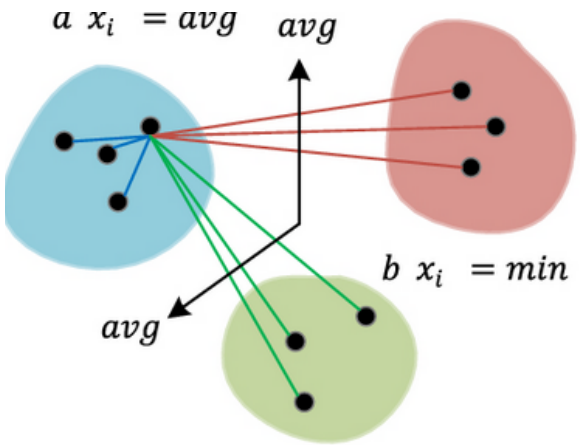
- Measures how well each point fits within its cluster. Defined for each point as: Similarity to own cluster - similarity to nearest other cluster

Ranges from **−1 to 1**:

- **Close to 1**: well-matched to its cluster
- **Near 0**: close to the decision boundary
- **Below 0**: probably in the wrong cluster

Silhouette can be averaged over:

- Each cluster
- The entire dataset (global score)



$$s\ x_i = \frac{b\ x_i - a\ x_i}{\max b\ x_i , a\ x_i}$$

(a) Silhouette coefficient

Evaluating Clusters: Internal Indices

Caveats and Cautions

- Silhouette prefers compact, spherical clusters
- Not reliable for **non-convex shapes** (e.g. rings, spirals)
- Adding more clusters (larger K) doesn't always improve silhouette
- Over-segmentation leads to clusters that are too small and tight, but poorly separated
- **DBSCAN** often gets unfairly penalized:
  - No concept of "noise" in Silhouette
  - Irregular cluster shapes lead to low scores

Alternative for DBSCAN

- Use **DBCV** (Density-Based Clustering Validation)
- It accounts for variable density and presence of noise

> Silhouette works well when your goal is to find **well-separated, round-ish groups** - not when structure is complex.

Why internal validation?

When we don't have ground-truth labels, we use **internal indices** to evaluate: How well does the clustering structure fit the data itself?

These indices use only:

- The distances between points
- The distribution of clusters
- The overall compactness and separation

They help us:

- Choose the number of clusters
- Compare different clustering algorithms
- Avoid overfitting or underfitting the structure

> *No single index is perfect - each one captures a different intuition about "good" clustering.*

Index	Range	Interpretation
Silhouette Index	$[-1, 1]$	Measures how well a point fits within its cluster vs others
Calinski-Harabasz	$[0, \infty)$	Ratio of between-cluster to within-cluster dispersion
BIC (for GMM)	$\propto$ log-likelihood	Penalised likelihood: trade-off between fit and complexity
Dunn Index	$[0, \infty)$	Ratio of min inter-cluster to max intra-cluster distance

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Dimension Reduction 

Dimension Reduction in Unsupervised Learning

Why reduce dimensions?

- In high-dimensional data, patterns become harder to detect:
  - Clusters are harder to separate
  - Distance measures become unstable
  - Visualisation becomes impossible

Dimension reduction:

- Projects data to a lower-dimensional space
- Preserves important variance or structure
- Supports clustering, anomaly detection, and interpretation

Key Techniques

- **PCA (Principal Component Analysis)**
  - Works on *continuous* variables
  - Finds linear combinations (components) that capture maximum variance
- **MCA (Multiple Correspondence Analysis)**
  - Used for *categorical* variables
  - Identifies underlying structure by analysing patterns of co-occurrence
- **MFA (Multiple Factor Analysis)**
  - Handles *mixed* data types (quant + qual blocks)
  - Balances the influence of different variable groups
  - Ideal for complex surveys, socio-economic data etc

Principal Component Analysis (PCA)

PCA finds a new set of **orthogonal axes** (principal components) that:

- Are linear combinations of the original variables
- Capture the **maximum variance** in the data

Given centered data matrix X, PCA solves:

$$\max_{\mathbf{w}} \quad \text{Var}(X\mathbf{w}) \quad \text{subject to} \quad \|\mathbf{w}\| = 1$$

The solution gives:

- Eigenvectors of the **covariance matrix** of X
- Eigenvalues = variance explained by each component

You can project the data onto the first k components:
$Z = XW_k$

Where W_k contains the top k eigenvectors.

- PCA is like **rotating the coordinate system** to align with the directions where the data spreads out the most.
- It finds **hidden axes** of variation and discards directions with minimal change.
- Think of it as:

> *Rewriting the data using fewer, more informative dimensions.*

Principal Component Analysis (PCA)

Why use it?

- Reduce dimensionality without losing much information (Think curse of dimensionality)
- Visualize high-dimensional data (e.g. in 2D)
- Pre-process for clustering or anomaly detection
- Remove multicollinearity in modeling

⚠️ PCA assumes **linearity** and is sensitive to **scaling**
Always standardize before applying it.

❓ Is PCA just OLS?

No, but they are related.

- **OLS** predicts y from X: it maximizes explained variance in the **dependent variable**
- **PCA** doesn't care about y: it finds directions that maximize variance in **X itself**

So PCA is more like *unsupervised regression* - but without a target.

Multiple Correspondence Analysis (MCA)

MCA is the counterpart of PCA for **categorical variables**.

- It operates on a **complete disjunctive table**: each category of a variable becomes a binary column.
- MCA decomposes the **inertia** (i.e., variance) in this table using **singular value decomposition (SVD)**.

Steps:

1. Convert categorical variables to 0/1 indicator matrix Z
2. Compute the **correspondence matrix** (normalized frequencies)
3. Apply SVD: $Z^* = U\Sigma V^\top$

Principal components are derived from the left singular vectors U, weighted by Σ.

The Intuition

- MCA reduces **many-category categorical data** to a few **synthetic dimensions**.
- It uncovers **underlying relationships** between levels of variables, e.g.:

> *People who prefer brand A also tend to buy product X and live in region Y.*

- Think of it as:

> *Letting the categories cluster and co-occur naturally in a low-dimensional space.*

Multiple Correspondence Analysis (MCA)

Why use it?

- Dimensionality reduction for **survey**, **demographic**, or **choice** data
- Pre-processing step for clustering with categorical inputs
- Visualize associations between **variables** and **categories**

> Like PCA, MCA is *unsupervised* and assumes linear associations in indicator space.