Clustering

CMSC 173 - Machine Learning

Noel Jeffrey Pinton

October 6, 2025

Department of Computer Science
University of the Philippines - Cebu

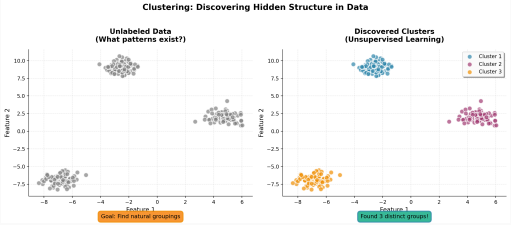
Outline

Introduction & Motivation

What is Clustering?

Definition

Clustering is the task of grouping a set of objects such that objects in the same group (cluster) are more similar to each other than to those in other groups.



Key Characteristics

- Unsupervised learning
- No labeled data required
- Discover hidden patterns
- Data-driven groupings

Goal

Find natural groupings in data without prior knowledge of group labels.

Supervised vs Unsupervised Learning

Supervised Learning

- Training data has **labels**
- Learn mapping: $f : X \to Y$
- Goal: Predict labels for new data
- Examples: Classification, regression

Example:

- Input: Email text
- Label: Spam/Not Spam
- Task: Learn to classify

Unsupervised Learning

- Training data has **no labels**
- Discover structure in X
- Goal: Find patterns, groups
- Examples: Clustering, dimensionality reduction

Example:

- Input: Customer purchase data
- No labels
- Task: Find customer segments

Clustering = Unsupervised

We discover groups without knowing what they should be in advance.

Real-World Applications

Business & Marketing

- **Customer segmentation**: Group customers by behavior
- **Market research**: Identify consumer groups
- **Recommendation systems**: Group similar items

Biology & Medicine

- **Gene expression**: Find related genes
- **Disease diagnosis**: Identify patient subgroups
- **Protein structure**: Analyze protein families

Image & Vision

- **Image segmentation**: Group pixels by similarity
- **Object recognition**: Cluster visual features
- **Color quantization**: Reduce color palette

Text & Web

- **Document clustering**: Group similar documents
- **Topic modeling**: Discover themes
- **Social network analysis**: Find communities

Common Theme

All involve finding structure in unlabeled data!

Types of Clustering

Partitional Clustering

- Divide data into **K** non-overlapping groups
- Each point belongs to exactly **one** cluster
- Flat structure
- Examples: K-Means, K-Medoids, GMM

Characteristics:

- Need to specify K
- Fast and scalable
- Sensitive to initialization

Hierarchical Clustering

- Build a **tree** of clusters (dendrogram)
- Can extract K clusters at any level
- Nested structure
- Examples: Agglomerative, Divisive

Characteristics:

- No need to specify K upfront
- More interpretable hierarchy
- Computationally expensive

This Lecture

Focus on **Partitional** (K-Means, GMM) and **Hierarchical** methods.

Foundation: Distance & Similarity

Distance Metrics

Common Distance Metrics

For $\mathbf{x} = (x_1, \ldots, x_d)$ and $\mathbf{y} = (y_1, \ldots, y_d)$:

1. **Euclidean Distance** (L2)

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

2. **Manhattan Distance** (L1)

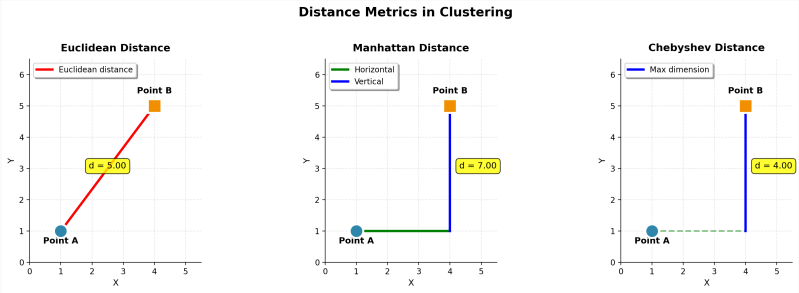
$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$$

3. **Chebyshev Distance** (L∞)

$$d(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$$

4. **Cosine Similarity**

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$



Properties of Distance Metrics

A valid distance metric must satisfy:

Metric Axioms

For all points $\mathbf{x}, \mathbf{y}, \mathbf{z}$:

1. **Non-negativity**: $d(\mathbf{x}, \mathbf{y}) \geq 0$
2. **Identity**: $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$
3. **Symmetry**: $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
4. **Triangle inequality**: $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

Choosing the Right Metric

- **Euclidean**: Most common, assumes all features equally important
- **Manhattan**: Less sensitive to outliers, good for high dimensions
- **Cosine**: Good for text/document clustering (direction matters)
- **Custom**: Domain-specific distances (e.g., edit distance for strings)

Partitional Clustering: K-Means

K-Means Algorithm: Overview

Goal

Partition n points into K clusters

Key Idea

- Each cluster has **centroid** (mean)
- Assign points to **nearest** centroid
- Update centroids iteratively
- Minimize within-cluster variance

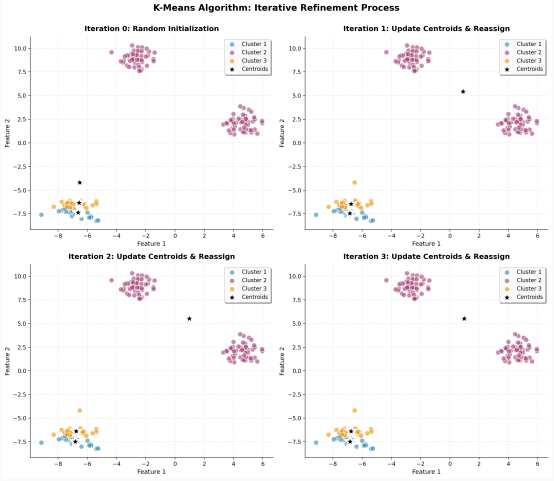
Input & Output

Input:

- Dataset X, Number K

Output:

- Assignments $\{C_1, \ldots, C_K\}$
- Centroids $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$



K-Means Objective Function

Minimize within-cluster sum of squares (WCSS):

Objective

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

where:

- C_k = set of points in cluster k
- $\boldsymbol{\mu}_k$ = centroid of cluster k
- $\|\cdot\|$ = Euclidean distance

Interpretation

- Minimize total squared distance from points to their centroids
- Encourages **compact**, **spherical** clusters
- Also called **inertia** or **distortion**
- NP-hard to minimize globally, but heuristics work well

Note

K-Means finds **local minimum**, not necessarily global!

K-Means Algorithm (Lloyd's Algorithm)

Algorithm 1 K-Means Clustering

Require: Dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, number of clusters K

Ensure: Cluster assignments and centroids

Initialize K centroids $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ randomly
repeat
  Assignment Step:
  for each data point $\mathbf{x}_i$ **do**
    Assign $\mathbf{x}_i$ to cluster $k^* = \arg\min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$
  end for
  Update Step:
  for each cluster $k = 1, \ldots, K$ **do**
    Update centroid: $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$
  end for
until centroids do not change (or max iterations reached)

Key Properties

Convergence: Guaranteed (objective always decreases). **Complexity:** $O(nKdT)$ where T = iterations

K-Means Example: Dataset

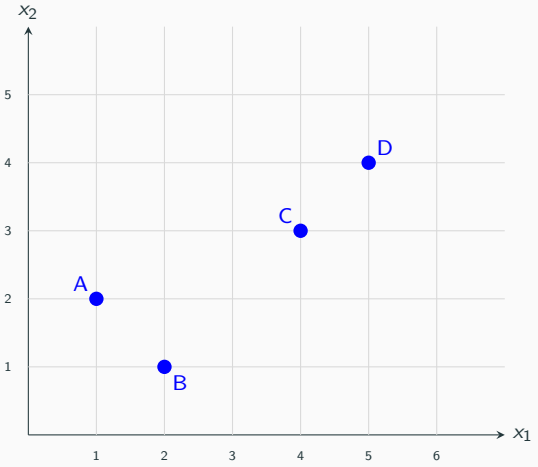
Let's apply K-Means to a toy dataset with $K = 2$

Dataset (4 points, 2D)

Point	x_1	x_2
A	1	2
B	2	1
C	4	3
D	5	4

Goal

Cluster into $K = 2$ groups



K-Means Example: Step 1 - Initialization

Step 1: Randomly initialize 2 centroids

Random Initialization

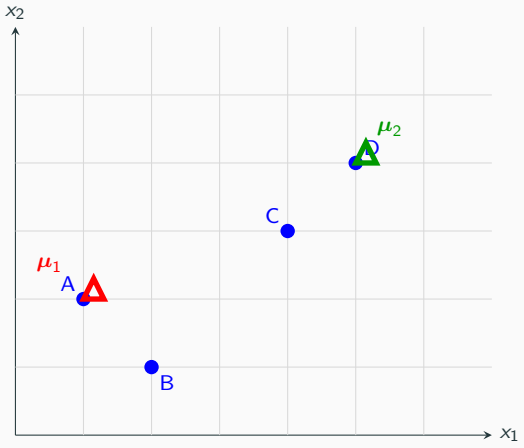
Let's choose first two points as centroids:

$\mu_1 =$ Point A $= (1, 2)$

$\mu_2 =$ Point D $= (5, 4)$

Note

In practice, use K-Means++ initialization!



K-Means Example: Step 2 - Assignment

Step 2: Assign each point to nearest centroid

Distance Calculations

For each point, compute distance to both centroids:

Point A (1,2):

- To μ_1: $\sqrt{(1-1)^2 + (2-2)^2} = \mathbf{0}$
- To μ_2: $\sqrt{(1-5)^2 + (2-4)^2} = 4.47$
- $\Rightarrow$ Assign to Cluster 1

Point B (2,1):

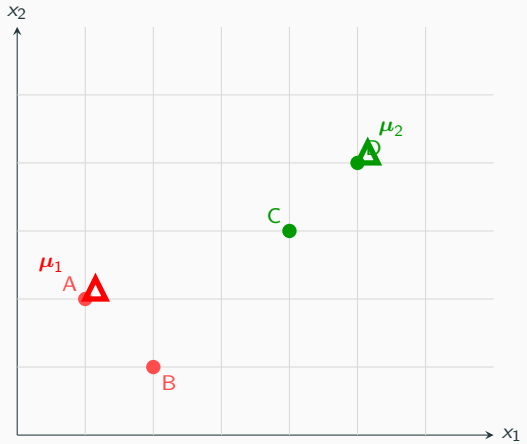
- To μ_1: $\sqrt{(2-1)^2 + (1-2)^2} = \mathbf{1.41}$
- To μ_2: $\sqrt{(2-5)^2 + (1-4)^2} = 4.24$
- $\Rightarrow$ Assign to Cluster 1

Point C (4,3):

- To μ_1: $\sqrt{(4-1)^2 + (3-2)^2} = 3.16$
- To μ_2: $\sqrt{(4-5)^2 + (3-4)^2} = \mathbf{1.41}$
- $\Rightarrow$ Assign to Cluster 2

Point D (5,4):

- To μ_1: $\sqrt{(5-1)^2 + (4-2)^2} = 4.47$
- To μ_2: $\sqrt{(5-5)^2 + (4-4)^2} = \mathbf{0}$



Clusters

C1: {A, B}
C2: {C, D}

K-Means Example: Step 3 - Update Centroids

Step 3: Recompute centroids as mean of assigned points

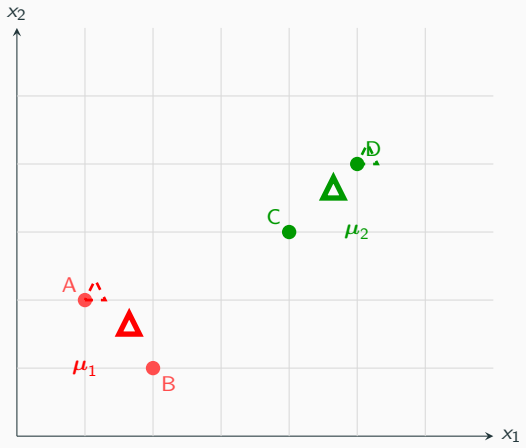
New Centroids

Cluster 1 contains: A(1,2), B(2,1)

$$\mu_1 = \frac{1}{2}[(1,2)+(2,1)]$$
$$= \left(\frac{1+2}{2}, \frac{2+1}{2}\right) = (\mathbf{1.5}, \mathbf{1.5})$$

Cluster 2 contains: C(4,3), D(5,4)

$$\mu_2 = \frac{1}{2}[(4,3)+(5,4)]$$
$$= \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (\mathbf{4.5}, \mathbf{3.5})$$



Centroids moved!

Old (dashed) → New (solid)

K-Means Example: Iteration 2 - Assignment

Iteration 2: Re-assign points to NEW centroids

Distance Calculations

Using new centroids $\boldsymbol{\mu}_1 = (1.5, 1.5)$, $\boldsymbol{\mu}_2 = (4.5, 3.5)$:

Point A (1,2):

- To $\boldsymbol{\mu}_1$: $\sqrt{(1-1.5)^2 + (2-1.5)^2} = \mathbf{0.71}$
- To $\boldsymbol{\mu}_2$: $\sqrt{(1-4.5)^2 + (2-3.5)^2} = 3.81$
- $\Rightarrow$ Cluster 1 (no change)

Point B (2,1):

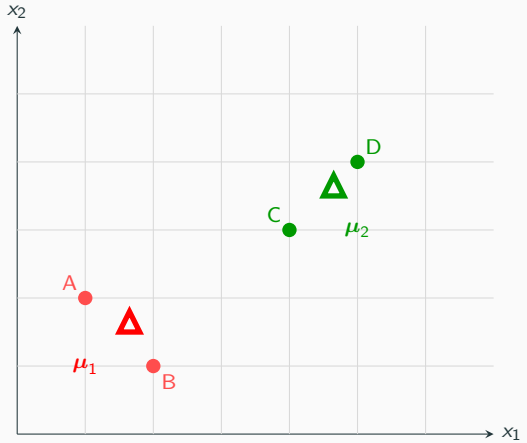
- To $\boldsymbol{\mu}_1$: $\sqrt{(2-1.5)^2 + (1-1.5)^2} = \mathbf{0.71}$
- To $\boldsymbol{\mu}_2$: $\sqrt{(2-4.5)^2 + (1-3.5)^2} = 3.54$
- $\Rightarrow$ Cluster 1 (no change)

Point C (4,3):

- To $\boldsymbol{\mu}_1$: $\sqrt{(4-1.5)^2 + (3-1.5)^2} = 2.92$
- To $\boldsymbol{\mu}_2$: $\sqrt{(4-4.5)^2 + (3-3.5)^2} = \mathbf{0.71}$
- $\Rightarrow$ Cluster 2 (no change)

Point D (5,4):

- To $\boldsymbol{\mu}_1$: $\sqrt{(5-1.5)^2 + (4-1.5)^2} = 4.30$
- To $\boldsymbol{\mu}_2$: $\sqrt{(5-4.5)^2 + (4-3.5)^2} = \mathbf{0.71}$



Result

No changes!

Assignments: Same as before

K-Means Example: Convergence

Step 4: Check convergence

Convergence Achieved!

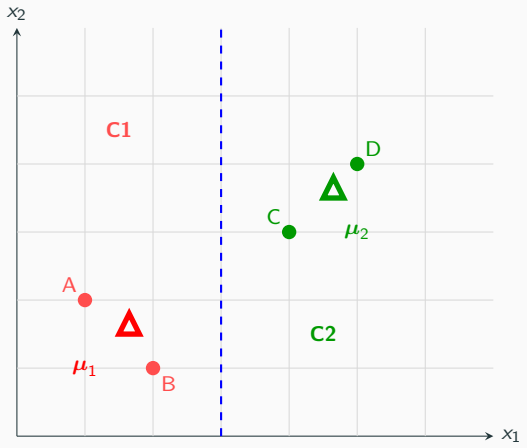
Since no points changed clusters, the algorithm has converged.

Final Clusters:

- **Cluster 1:** {A(1,2), B(2,1)}
- **Cluster 2:** {C(4,3), D(5,4)}

Final Centroids:

- $\mu_1 = (1.5, 1.5)$
- $\mu_2 = (4.5, 3.5)$



Key Insight

K-Means partitions space with linear decision boundaries (Voronoi cells)

K-Means: Voronoi Tesselation

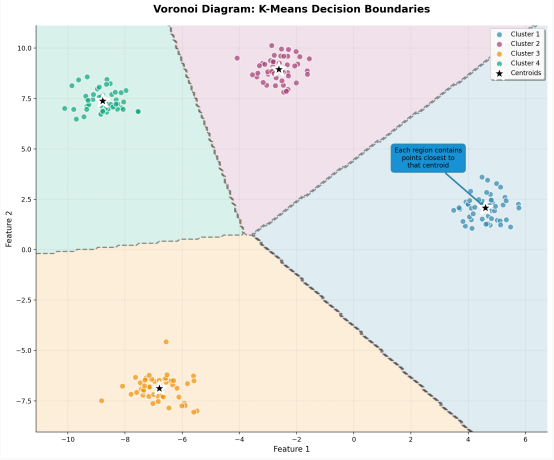
Geometric Interpretation

K-Means creates a **Voronoi diagram**:

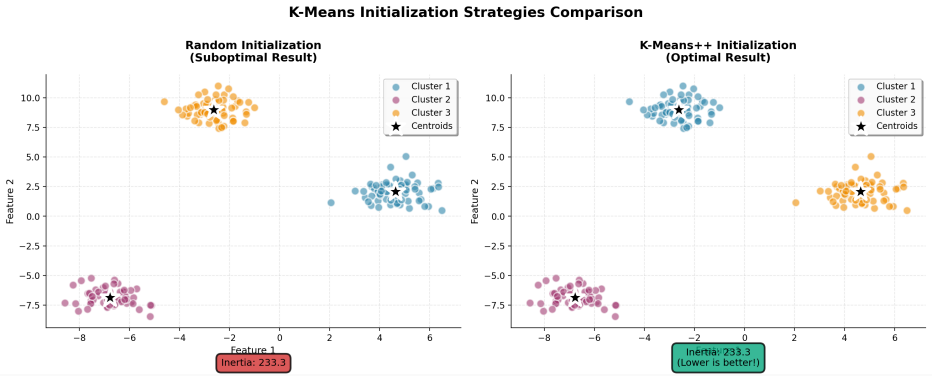
- Space partitioned into regions
- Points closest to one centroid
- Decision boundaries are **linear**
- Forms convex, polygonal cells

Implications

- Works well for spherical clusters
- Struggles with elongated shapes
- Assumes equal variance
- Sensitive to outliers



K-Means Initialization: The Challenge



Problem

Sensitive to initial centroids

Random Init Issues

- Poor local minima
- High variance
- Multiple runs needed

Practice

Run 10-100 times, keep best WCSS

K-Means++ Initialization

Smarter initialization strategy (Arthur & Vassilvitskii, 2007)

K-Means++ Algorithm

1. Choose first centroid $\boldsymbol{\mu}_1$ uniformly at random from data points
2. For $k = 2, \ldots, K$:
   - For each point $\mathbf{x}_i$, compute $D(\mathbf{x}_i)$ = distance to nearest centroid
   - Choose next centroid $\boldsymbol{\mu}_k$ with probability $\propto D(\mathbf{x}_i)^2$
3. Run standard K-Means with these initial centroids

Advantages

- Spreads out initial centroids
- Provably better: $O(\log K)$-competitive with optimal
- Lower variance, more consistent results
- Standard in scikit-learn and most libraries

Recommendation

Always use K-Means++ unless you have domain knowledge for better initialization.

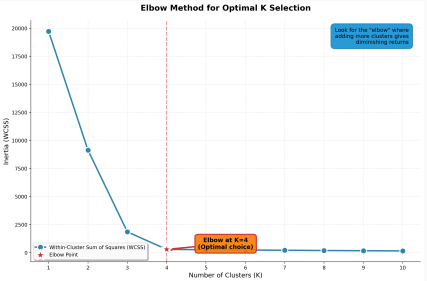
Choosing K: The Elbow Method

Elbow Method

1. Run K-Means for $K = 1, 2, \ldots, K_{\max}$
2. Plot WCSS vs K
3. Look for the "**elbow**" point
4. Choose K with diminishing returns

Interpretation

- WCSS decreases as K increases
- Elbow = fit vs complexity trade-off
- Not always clear/unique



Choosing K: Silhouette Analysis

Silhouette Coefficient

For each point $\mathbf{x}_i$:

1. a_i = avg distance to same cluster
2. b_i = avg distance to nearest other
3. Silhouette:

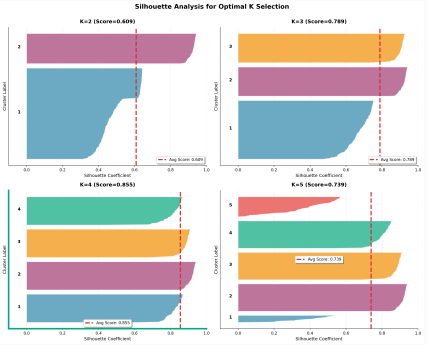
$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

Range: $s_i \in [-1, 1]$

- $s_i \approx 1$: Well clustered
- $s_i \approx 0$: On border
- $s_i < 0$: Wrong cluster

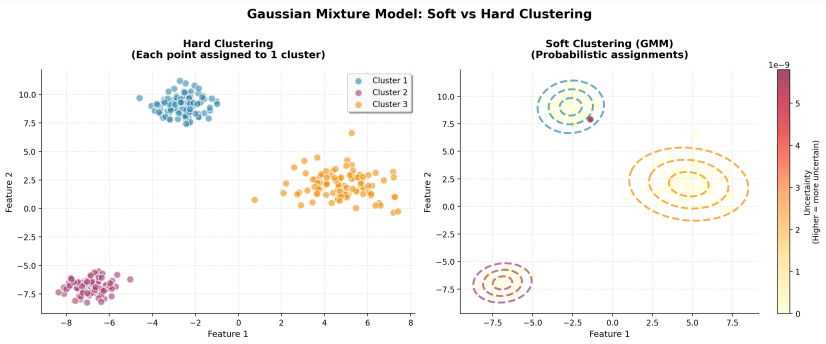
Usage

Choose K maximizing avg score



Soft Clustering: Gaussian Mixture Models

Limitations of K-Means



Key Issues

- **Hard assignments**: Binary membership
- **Spherical clusters**: Equal variance assumed
- **No uncertainty**: Can't express doubt
- **Outliers**: Forced into clusters

When K-Means Struggles

- Elongated/elliptical clusters
- Different sizes/densities
- Overlapping clusters
- Need probability of membership

Solution

Gaussian Mixture Models (GMM) provide soft, probabilistic clustering

Gaussian Mixture Models: Formulation

Model data as generated from mixture of K Gaussian distributions

Generative Model

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where:

- π_k = mixing coefficient (prior probability of cluster k), $\sum_k \pi_k = 1$
- $\boldsymbol{\mu}_k$ = mean of Gaussian k
- $\boldsymbol{\Sigma}_k$ = covariance matrix of Gaussian k
- $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ = multivariate Gaussian

Soft Assignment

Probability that point $\mathbf{x}_i$ belongs to cluster k:

$$\gamma_{ik} = p(z_i = k|\mathbf{x}_i) = \frac{\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

GMM: Expectation-Maximization Algorithm

Learn parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ using EM

EM Algorithm

Initialize: Random $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k = I$, $\pi_k = 1/K$

Repeat until convergence:

E-step: Compute responsibilities (soft assignments)

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

M-step: Update parameters

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ik}, \quad \boldsymbol{\mu}_k = \frac{\sum_i \gamma_{ik} \mathbf{x}_i}{\sum_i \gamma_{ik}}, \quad \boldsymbol{\Sigma}_k = \frac{\sum_i \gamma_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i \gamma_{ik}}$$

Properties

Monotonically increases likelihood. Guaranteed to converge to local maximum.

GMM vs K-Means Comparison

K-Means

Pros:

- Simple, fast, scalable
- Easy to implement
- Works well for spherical clusters
- Less parameters to tune

Cons:

- Hard assignments only
- Assumes spherical clusters
- Sensitive to initialization
- No measure of uncertainty

GMM

Pros:

- Soft probabilistic assignments
- Flexible cluster shapes (elliptical)
- Measures uncertainty
- Principled statistical model

Cons:

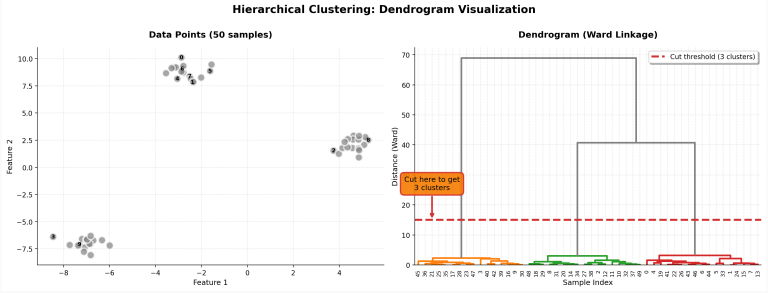
- Slower than K-Means
- More parameters ($\mathbf{\Sigma}_k$)
- Can overfit with full covariance
- Also sensitive to initialization

Note

K-Means is special case of GMM with $\mathbf{\Sigma}_k = \sigma^2 I$ and hard assignments!

Hierarchical Clustering

Hierarchical Clustering: Overview



Agglomerative (Bottom-up)

- Start: Each point = cluster
- Merge closest clusters
- End: One cluster

Divisive (Top-down)

- Start: All in one cluster
- Split clusters iteratively
- End: Each point separate

Key Advantage

No need to specify K upfront! Cut dendrogram at any height to get desired clusters.

Agglomerative Clustering Algorithm

Most common hierarchical method

Algorithm

1. **Initialize**: Each of n points is own cluster
2. **Compute**: Distance matrix between all clusters
3. **Repeat** until one cluster remains:
   - Find pair of closest clusters
   - Merge them into single cluster
   - Update distance matrix
4. **Output**: Dendrogram showing merge history

Complexity

Time: $O(n^2 \log n)$ with efficient data structures

Space: $O(n^2)$ for distance matrix

Challenge

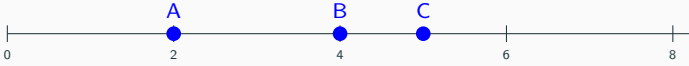
How do we measure distance between **clusters** (not just points)?

Hierarchical Clustering Example: Dataset

Let's apply Agglomerative Clustering to 5 points (Single Linkage)

Dataset (5 points, 1D)

Point	Position
A	2
B	4
C	5
D	10
E	12



Initial State

Each point = own cluster

5 clusters: {A}, {B}, {C}, {D}, {E}

Goal

Build dendrogram using **Single Linkage**

Hierarchical Example: Step 1 - Distance Matrix

Step 1: Compute pairwise distance matrix

Distance Matrix

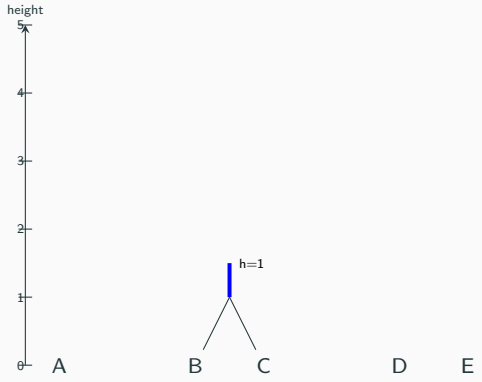
	A	B	C	D	E
A	0	**2**	3	8	10
B	**2**	0	**1**	6	8
C	3	**1**	0	5	7
D	8	6	5	0	**2**
E	10	8	7	**2**	0

Find Minimum

Smallest distance = **1** between B and C

$\Rightarrow$ Merge {B} and {C}

Dendrogram (Step 1)



Current Clusters

{A}, {B, C}, {D}, {E}

4 clusters remain

Hierarchical Example: Step 2 - Update Matrix

Step 2: Update distance matrix using Single Linkage

Single Linkage Rule

$$d(\{B, C\}, X) = \min(d(B, X), d(C, X))$$

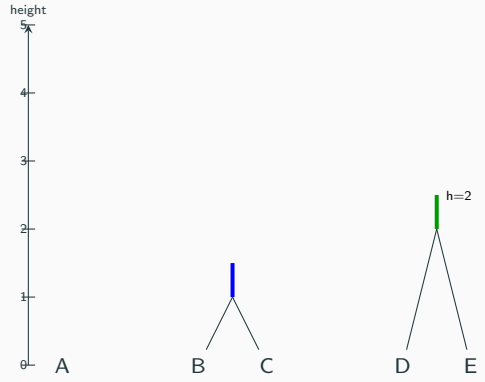
New distances to cluster {B,C}:

- $d(\{B, C\}, A) = \min(2, 3) = \mathbf{2}$
- $d(\{B, C\}, D) = \min(6, 5) = \mathbf{5}$
- $d(\{B, C\}, E) = \min(8, 7) = \mathbf{7}$

Updated Matrix

	A	{B,C}	D	E
A	0	2	8	10
{B,C}	2	0	5	7
D	8	5	0	2
E	10	7	2	0

Dendrogram (Step 2)



Next Merge

Min distance = **2**

Merge {D} and {E} at height 2

Hierarchical Example: Steps 3-4

Continue merging until one cluster remains

Step 3

Clusters: {A}, {B,C}, {D,E}

Updated distances:

- $d(A, \{B, C\}) = 2$
- $d(A, \{D, E\}) = 8$
- $d(\{B, C\}, \{D, E\}) = 5$

Min = 2: Merge A with {B,C}
New cluster: {A, B, C} at height 2

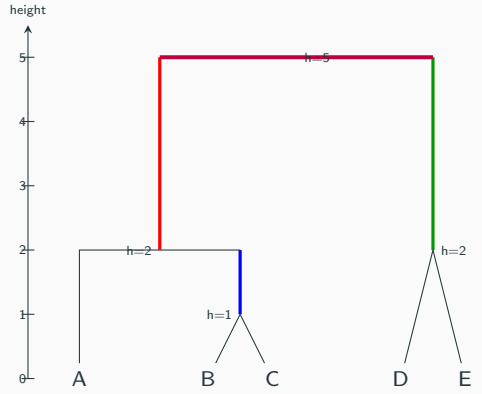
Step 4 (Final)

Clusters: {A,B,C}, {D,E}

Distance: $d(\{A, B, C\}, \{D, E\}) = 5$

Final merge at height 5
One cluster: {A, B, C, D, E}

Complete Dendrogram

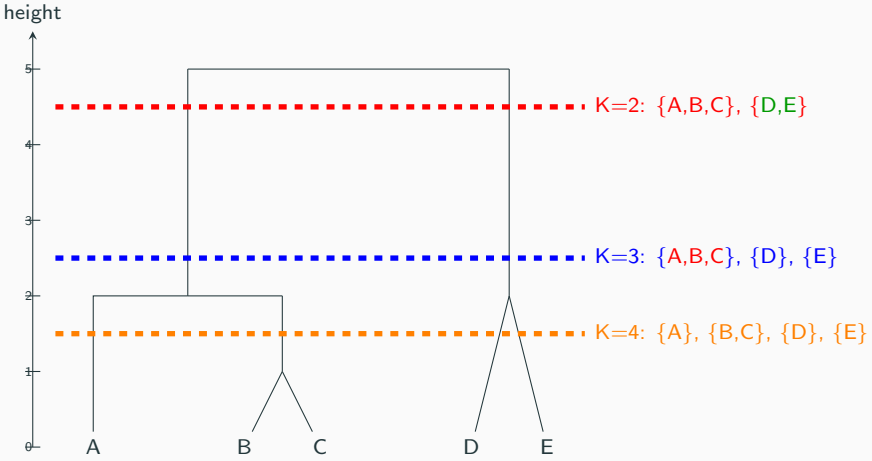


Interpretation

Cut at different heights to get different K clusters

Hierarchical Example: Cutting the Dendrogram

Extract different numbers of clusters by cutting at different heights



Key Advantage of Hierarchical Clustering

No need to pre-specify K! The dendrogram shows the full hierarchy.

Linkage Criteria

Different ways to measure inter-cluster distance

Common Linkage Methods

For clusters C_i and C_j:

1. **Single Linkage (MIN):**

$$d(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

2. **Complete Linkage (MAX):**

$$d(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

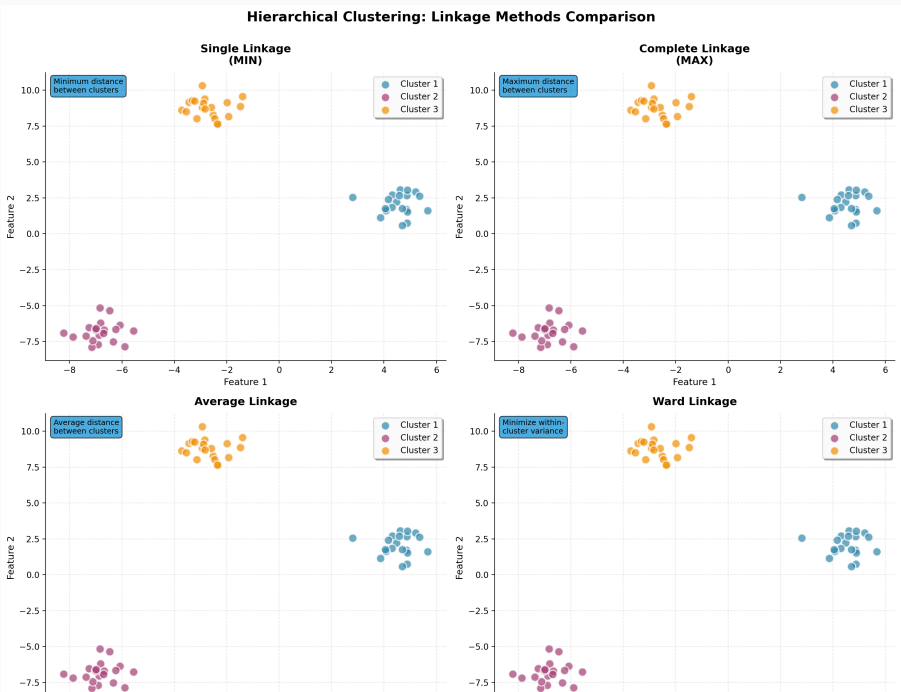
3. **Average Linkage:**

$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

4. **Ward's Linkage:**

$$d(C_i, C_j) = \text{increase in WCSS when merging } C_i \text{ and } C_j$$

Linkage Methods: Comparison



Dendrograms: Interpretation

Reading Dendrograms

- **Leaves**: Individual points
- **Height**: Merge distance
- **Branches**: Relationships
- **Cut**: Extract K clusters

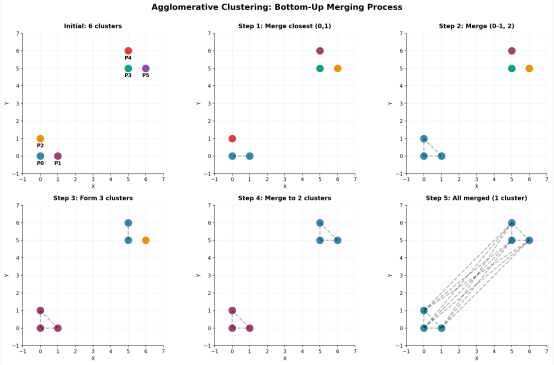
Extracting Clusters

Cut at height h:

- Higher $\rightarrow$ fewer clusters
- Lower $\rightarrow$ more clusters
- Choose via validation

Advantage

Explore different K without re-running!



Cluster Validation & Evaluation

Types of Cluster Validation

How do we assess clustering quality?

Internal Validation

Use only the data itself

Measures:

- Silhouette coefficient
- Davies-Bouldin index
- Calinski-Harabasz score
- Dunn index

Idea: Good clusters are compact and well-separated

External Validation

Compare to ground truth labels

Measures:

- Adjusted Rand Index (ARI)
- Normalized Mutual Information (NMI)
- V-measure
- Purity

Idea: Good clustering agrees with true labels

When to Use Each

Internal: Unsupervised setting (no labels) **External:** When ground truth available (benchmarking)

Internal Metrics: Formulas

1. Silhouette Coefficient

$$s = \frac{1}{n} \sum_{i=1}^{n} \frac{b_i - a_i}{\max(a_i, b_i)}$$

Range: $[-1, 1]$, higher is better

2. Davies-Bouldin Index

$$DB = \frac{1}{K} \sum_{k=1}^{K} \max_{k' \neq k} \frac{\sigma_k + \sigma_{k'}}{d(\boldsymbol{\mu}_k, \boldsymbol{\mu}_{k'})}$$

Range: $[0, \infty)$, **lower** is better

3. Calinski-Harabasz Score (Variance Ratio)

$$CH = \frac{\text{Between-cluster variance}}{\text{Within-cluster variance}} \times \frac{n - K}{K - 1}$$

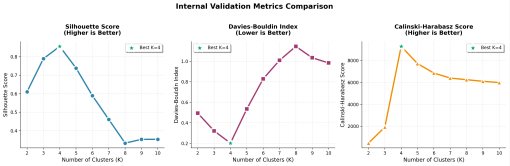
Range: $[0, \infty)$, higher is better

Usage

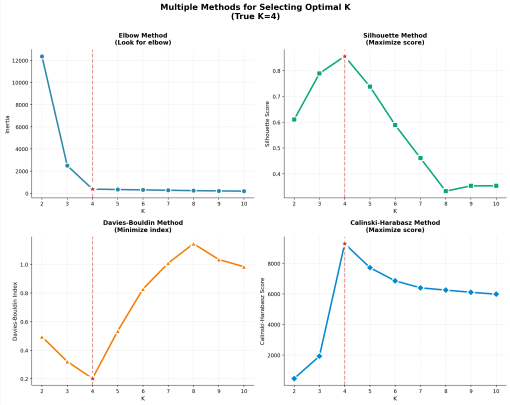
Use multiple metrics! Different metrics may favor different clusterings.

Internal Validation: Visualization

Metric Values vs K



Optimal K Comparison



Observation

Different metrics may suggest different optimal K. Use domain knowledge!

External Metrics: Formulas

Given true labels Y and predicted labels C:

1. Adjusted Rand Index (ARI)

$$ARI = \frac{\text{RI} - E[\text{RI}]}{\max(\text{RI}) - E[\text{RI}]}$$

- Range: $[-1, 1]$, higher is better
- $ARI = 1$: Perfect agreement
- $ARI \approx 0$: Random labeling
- Adjusted for chance

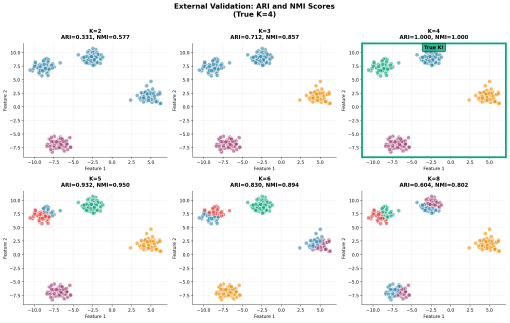
2. Normalized Mutual Information (NMI)

$$NMI(Y, C) = \frac{2 \cdot I(Y; C)}{H(Y) + H(C)}$$

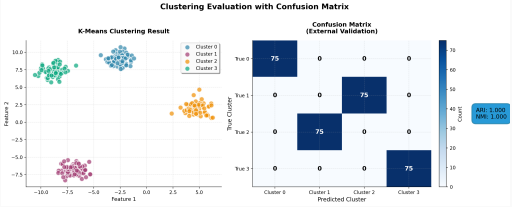
- Range: $[0, 1]$, higher is better
- $NMI = 1$: Perfect agreement
- Based on information theory
- Normalized for different K

External Validation: Example

Validation Metrics



Confusion Matrix

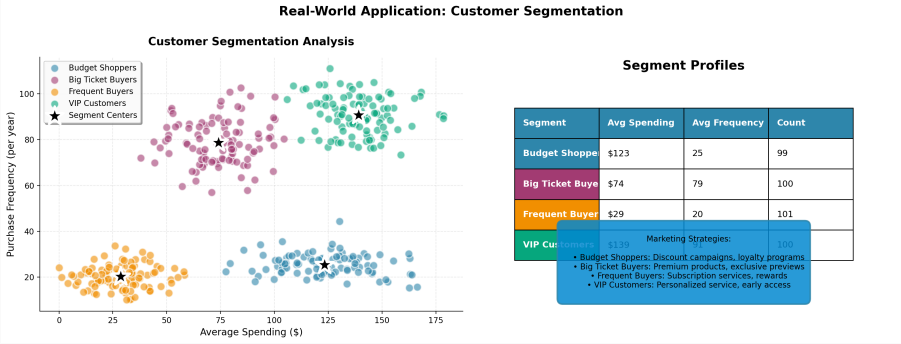


Note

External validation only for benchmarking. In real unsupervised tasks, no ground truth!

Real-World Applications

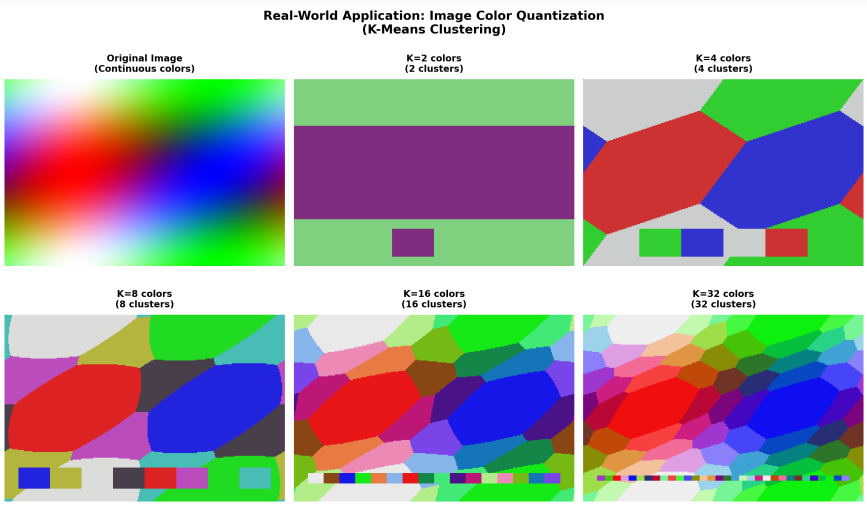
Application: Customer Segmentation



Business Impact

- **Targeted marketing**: Strategies per segment
- **Product development**: Tailor to groups
- **Resource allocation**: Focus on high-value
- **Customer retention**: Identify at-risk

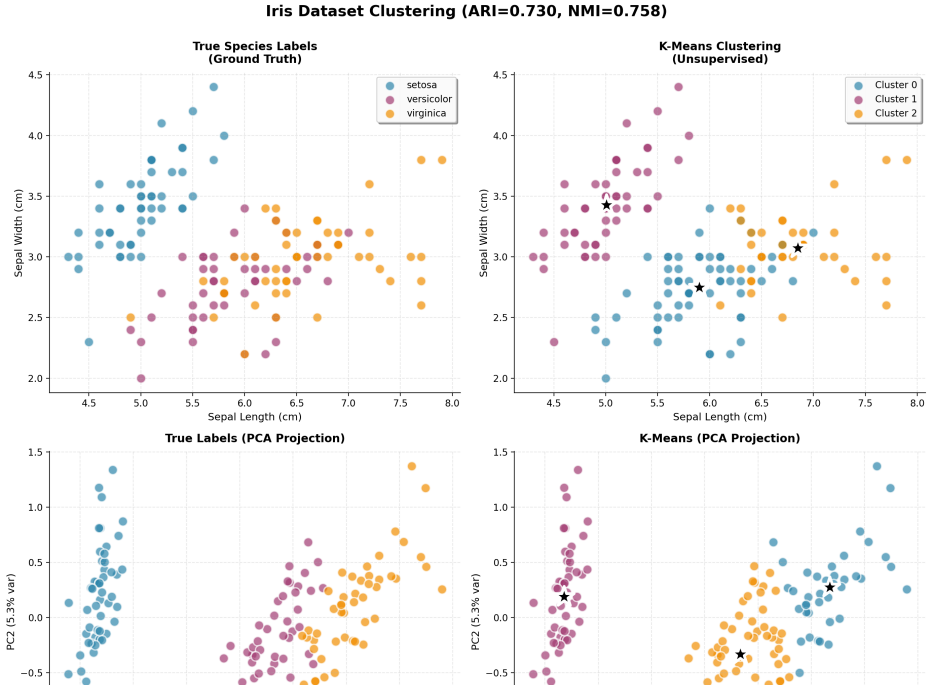
Application: Image Color Quantization



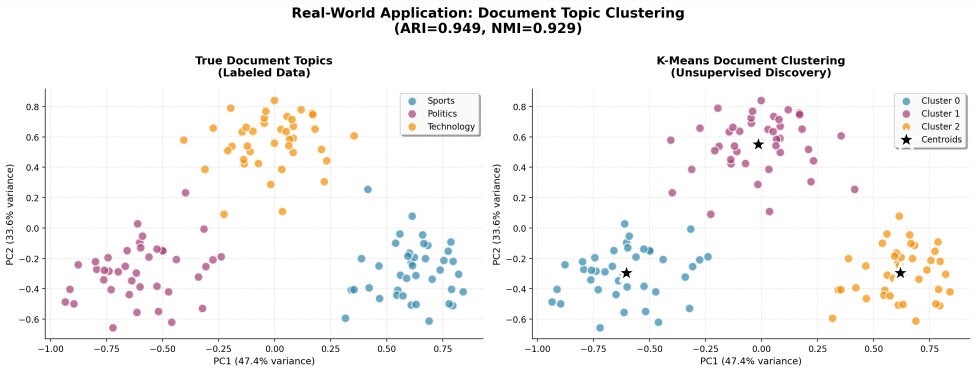
Use Cases

- **Image compression**: Reduce file size
- **Color palette**: Identify dominant colors
- **Segmentation**: Group similar pixels
- **Artistic effects**: Posterization

Application: Biological Data (Iris Dataset)



Application: Document Clustering



Text Mining Applications

- **Topic discovery**: Find themes in document collections
- **News organization**: Group similar articles
- **Search results**: Organize by topic clusters
- **Recommendation**: Find similar documents

Best Practices & Guidelines

Choosing a Clustering Algorithm

Decision Guide

Use K-Means when:

- You know K (or can estimate it)
- Data has roughly spherical clusters
- Large dataset (scalability important)
- Want fast, simple method

Use GMM when:

- Need probabilistic assignments
- Clusters have different shapes/variances
- Want to measure uncertainty
- Have computational resources

Use Hierarchical when:

- Don't know K in advance
- Want to explore multiple granularities
- Need interpretable hierarchy
- Small to medium dataset ($n < 10,000$)

Common Pitfalls & Solutions

Pitfall 1: Not Scaling Features

Problem: Features with large ranges dominate distance

Solution: Standardize features: $z = \frac{x-\mu}{\sigma}$

Pitfall 2: Using Wrong Distance Metric

Problem: Euclidean not always appropriate

Solution: Match metric to data type (cosine for text, custom for categorical)

Pitfall 3: Ignoring Outliers

Problem: Outliers can distort clusters (especially K-Means)

Solution: Detect and remove outliers, or use robust methods (DBSCAN, K-Medoids)

Pitfall 4: Blindly Trusting One Metric

Problem: Single validation metric may be misleading

Solution: Use multiple metrics + visual inspection + domain knowledge

Best Practices: Practical Tips

Data Preprocessing

- **Scale features**: Use StandardScaler or MinMaxScaler
- **Handle missing values**: Impute or remove
- **Remove duplicates**: Can bias cluster centers
- **Consider dimensionality reduction**: PCA for high-dimensional data

Model Selection

- **Try multiple K values**: Use elbow + silhouette + domain knowledge
- **Run multiple times**: Different initializations for K-Means
- **Validate results**: Use both internal and visual validation
- **Compare algorithms**: K-Means, GMM, Hierarchical

Interpretation

- **Visualize clusters**: 2D projections (PCA, t-SNE)
- **Inspect cluster centers**: What characterizes each cluster?
- **Verify with domain experts**: Do clusters make sense?
- **Iterate**: Clustering is exploratory - refine based on insights

Summary & Conclusion

Key Takeaways

Clustering Fundamentals

- **Unsupervised learning**: Discover structure without labels
- **Distance metrics**: Foundation of clustering (Euclidean, cosine, etc.)
- **Two main types**: Partitional vs Hierarchical

Key Algorithms

- **K-Means**: Fast, simple, hard assignments, spherical clusters
- **GMM**: Soft assignments, flexible shapes, probabilistic
- **Hierarchical**: No need for K, produces dendrogram, $O(n^2)$

Validation

- **Internal**: Silhouette, Davies-Bouldin, Calinski-Harabasz
- **External**: ARI, NMI (when ground truth available)
- **Selection**: Elbow method, silhouette analysis, domain knowledge

What We Covered

1. **Introduction**: Motivation, applications, clustering types
2. **Distance Metrics**: Euclidean, Manhattan, cosine similarity
3. **K-Means**: Algorithm, initialization (K-Means++), choosing K
4. **GMM**: Soft clustering, EM algorithm, comparison with K-Means
5. **Hierarchical**: Agglomerative, linkage methods, dendrograms
6. **Validation**: Internal and external metrics
7. **Applications**: Customer segmentation, image processing, bioinformatics
8. **Best Practices**: Algorithm selection, preprocessing, pitfalls

Next Steps

- **Practice**: Try clustering on real datasets
- **Experiment**: Compare different algorithms and parameters
- **Read**: Advanced topics - DBSCAN, spectral clustering, etc.

Further Reading

Textbooks

- **Bishop**: Pattern Recognition & Machine Learning (Ch. 9)
- **Murphy**: Probabilistic ML (Ch. 21)
- **Hastie et al.**: Elements of Statistical Learning (Ch. 14)

Key Papers

- Arthur & Vassilvitskii (2007): K-Means++
- Dempster et al. (1977): EM Algorithm
- Rousseeuw (1987): Silhouette Coefficient

Implementations

- **scikit-learn**: KMeans, GaussianMixture, AgglomerativeClustering
- **scipy**: Hierarchical clustering (linkage, dendrogram)
- **R**: kmeans, hclust, cluster package

Questions?

Thank you for your attention!

Noel Jeffrey Pinton
Department of Computer Science
University of the Philippines - Cebu