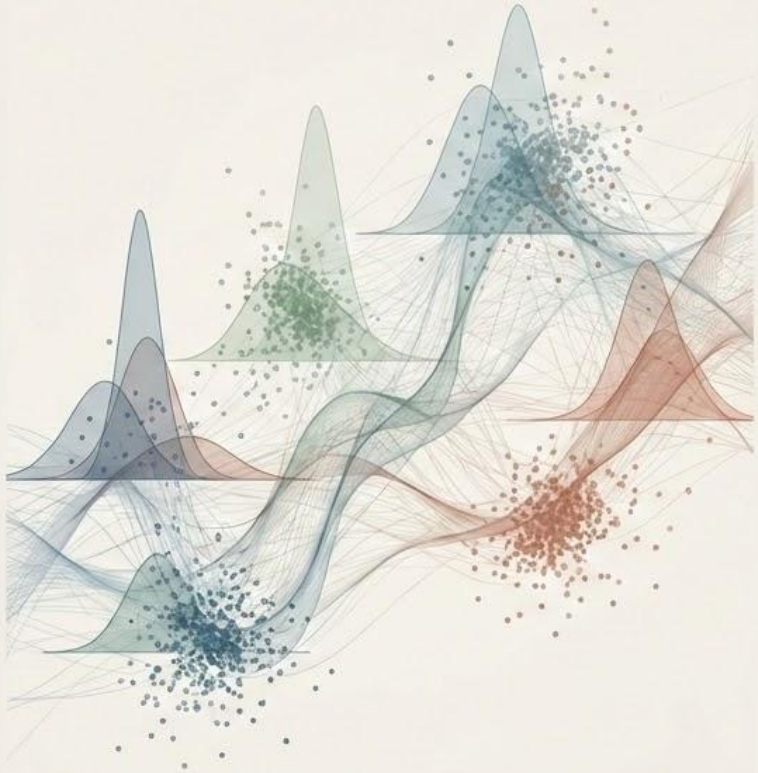
Lecture Title: Probabilistic Clustering, High-Dimensional Data, and Biclustering

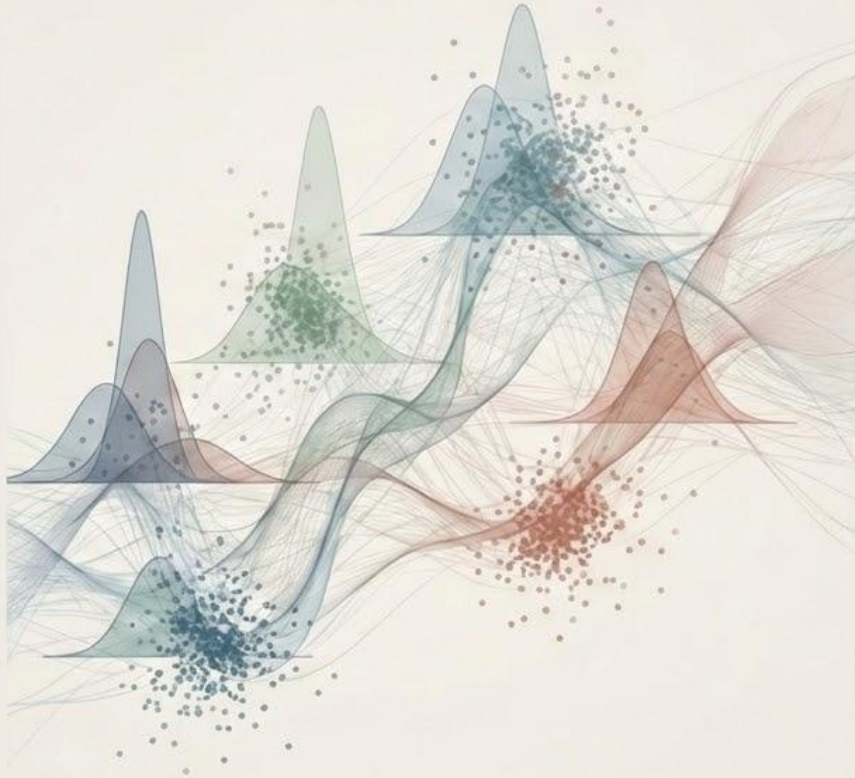
- **Duration**: 1 Hour
- **Target Audience**: Data Science/Computer Science students
- **Prerequisites**: Understanding of basic clustering, probability distributions, EM algorithm, dimensionality reduction concepts



The Complexity of Real-World Clustering

Moving Beyond ‘One Point, One Cluster’

- **The Traditional Limit:** Standard algorithms assign every single data point to exactly one, mutually exclusive cluster.
- **The Reality of Overlap:** What if a customer naturally belongs to both the "budget-conscious" segment and the "quality-seeker" segment?
- **The High-Dimensional Trap:** What happens when our data has thousands of dimensions and standard distance measures completely break down?
- **The Subspace Problem:** How do we find meaningful clusters that only exist across a small, hidden subset of those dimensions?

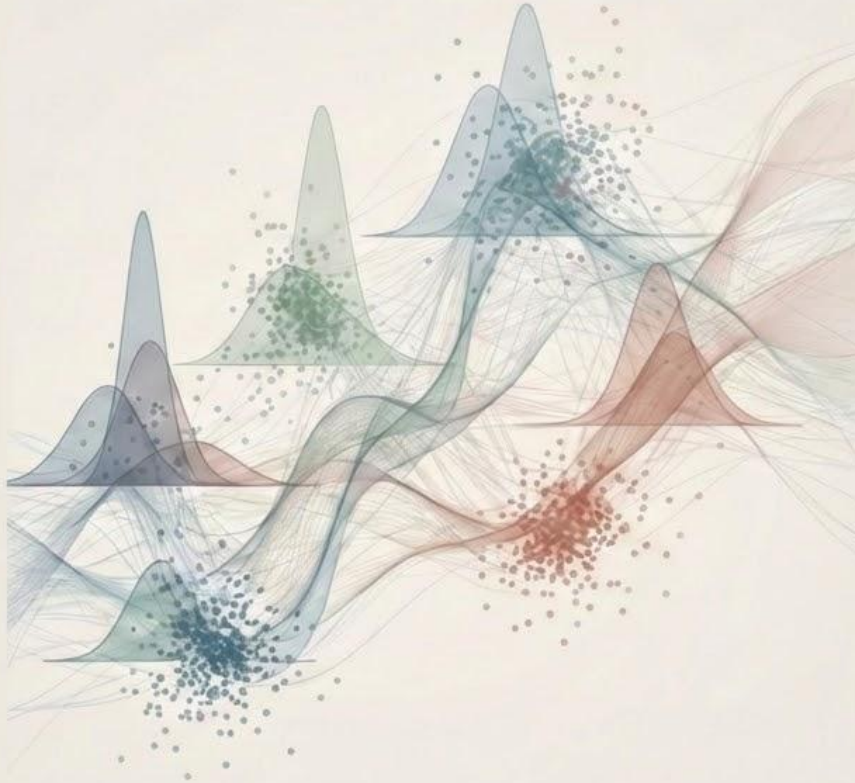


Learning Objectives

Mastering Advanced Clustering

By the end of this session, you will be able to:

- Understand probabilistic and fuzzy clustering approaches to handle overlapping data.
- Master the mechanics of the Expectation-Maximization (EM) algorithm.
- Address the mathematical and practical challenges of clustering high-dimensional data.
- Learn how to execute biclustering for simultaneous row and column grouping.
- Explore dimensionality reduction methods specifically tailored for clustering tasks.



Example 9.1. Clustering product reviews. Imagine that an e-commerce company has an online store, where customers not only purchase online, but also create reviews of products. Not every product receives reviews; instead, some products may have many reviews, whereas many others have none or only a few. Moreover, a review may involve multiple products. Thus as the review editor of the company, your task is to cluster the reviews.

Ideally, a cluster is about a *topic*, for example, a group of products, services, or issues that are highly related. Assigning a review to one cluster exclusively would not work well for your task. Suppose there is a cluster for "cameras and camcorders" and another for "computers." What if a review talks about the compatibility between a camcorder and a computer? The review is related to both clusters; however, it does not exclusively belong to either cluster.

You would like to use a clustering method that allows a review to belong to more than one cluster if the review indeed involves more than one topic. To reflect the strength that a review belongs to a cluster, you want the assignment of a review to a cluster to carry a weight representing the partial membership. □

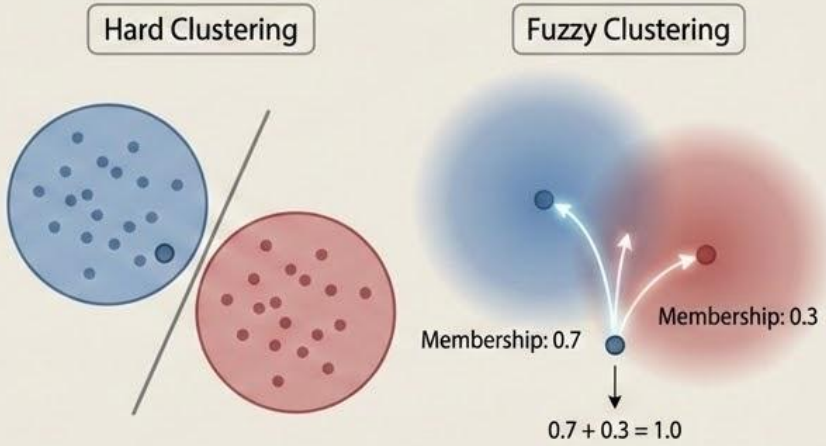
Example 9.2. Clustering to study user search intent. The online store of the e-commerce company discussed in Example 9.1 records all customer browsing and purchasing behavior in a log. An important data mining task is to use the log data to categorize and understand *user search intent*. For example, consider a user *session* (a short period in which a user interacts with the online store). Is the user searching for a product, making comparisons among different products, or looking for customer support information? Cluster analysis helps here because it is difficult to predefine user behavior patterns thoroughly. A cluster that contains similar user browsing trajectories may represent similar user behavior.

However, not every session belongs to only one cluster. For example, suppose the user sessions involving the purchase of digital cameras form one cluster, and the user sessions that compare laptop computers form another cluster. What if a user in one session makes an order for a digital camera and at the same time compares several laptop computers? Such a session should belong to both clusters to some extent. □

Introduction to Fuzzy Clustering

Moving Beyond ‘Hard’ Assignments

- The Limitation of Hard Clustering: Algorithms like k-means enforce strict boundaries. Every data point is assigned to exactly one cluster, leaving absolutely no room for uncertainty, overlap, or partial membership.
- The Fuzzy Alternative: In fuzzy clustering, a point doesn’t belong to just one group. It belongs to to all clusters simultaneously, with a specific degree of membership ranging between 0 and 1.
- The Rule: For any single data point, its membership values across all available clusters must mathematically sum to exactly 1.



Example 9.3. Fuzzy set. The more digital camera units that are sold, the more popular the camera is. In an e-commerce company, we can use the following formula to compute the degree of popularity of a digital camera, o, given the sales of o:

$$pop(o) = \begin{cases} 1 & \text{if 1000 or more units of } o \text{ are sold} \\ \frac{i}{1000} & \text{if } i \ (i < 1000) \text{ units of } o \text{ are sold.} \end{cases} \quad (9.1)$$

Function $pop()$ defines a fuzzy set of popular digital cameras. For example, suppose the sales of digital cameras at the e-commerce company are as shown in Table 9.1. The fuzzy set of popular digital cameras is $\{A(0.05), B(1), C(0.86), D(0.27)\}$, where the degrees of membership are written in parentheses. □

Table 9.1 A set of digital cameras and their sales at an e-commerce company.

Camera	Sales (units)
A	50
B	1320
C	860
D	270

Example 9.4. Fuzzy clusters. Suppose the online store of the e-commerce company has six review
The keywords contained in these reviews are listed in Table 9.2.

Table 9.2 Set of reviews and the keywords used.

Review_ID	Keywords
R_1	digital camera, lens
R_2	digital camera
R_3	lens
R_4	digital camera, lens, computer
R_5	computer, CPU
R_6	computer, computer game

We can group the reviews into two fuzzy clusters, C_1 and C_2. C_1 is for "digital camera" and "lens," and C_2 is for "computer." The partition matrix is

$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Here, we use the keywords "digital camera" and "lens" as the features of cluster C_1 and "computer" as the feature of cluster C_2. For review R_i and cluster C_j ($1 \leq i \leq 6, 1 \leq j \leq 2$), w_{ij} is defined as

$$w_{ij} = \frac{|R_i \cap C_j|}{|R_i \cap (C_1 \cup C_2)|} = \frac{|R_i \cap C_j|}{|R_i \cap \{\text{digital camera, lens, computer}\}|}.$$

In this fuzzy clustering, review R_4 belongs to clusters C_1 and C_2 with membership degrees $\frac{2}{3}$ and $\frac{1}{3}$, respectively. □

Fuzzy c-Means (FCM)

The Mathematical Foundation

The Goal:

Similar to k-means, the objective is to minimize the within-cluster sum of squares, but now we heavily weight it by the membership degree.

The Objective Function:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{m} \|\mathbf{x}_i - \mathbf{c}_j\|^2$$

The Variables:

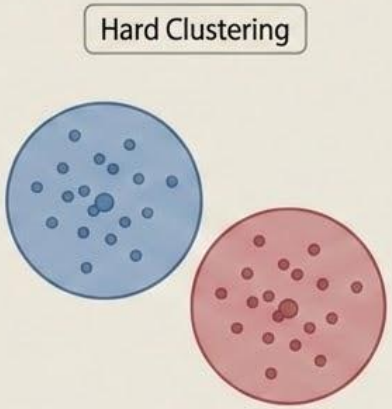
- m: The fuzziness parameter (where $m > 1$).
- u_{ij}: The specific membership degree of point *i* in cluster j.
- c_j: The centroid of cluster j.

Properties of the FCM Algorithm

The Impact of the Fuzziness Parameter (m)

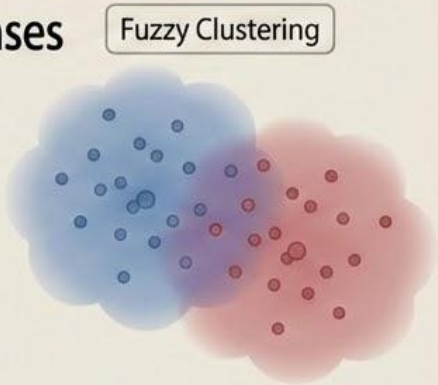
Condition: As $m \to 1$

Algorithmic Behavior: The algorithm effectively becomes rigid. It mathematically approaches standard **k-means** (hard clustering).



Condition: As m increases

Algorithmic Behavior: The cluster memberships become increasingly "fuzzy" and uniform across all points.



The Core Benefit

It mathematically and naturally captures ambiguous edge cases that sit right on the border between two distinct groups.

Applications of Fuzzy Clustering

Where Ambiguity is the Reality

Fuzzy clustering excels in domains where distinct boundaries do not naturally exist:

- **Image Segmentation**: Pixels at the borders of objects often contain a blend of colors and can partially belong to multiple distinct regions.
- **Customer Segmentation**: Human behavior is complex. Customers frequently exhibit mixed preferences and purchasing habits that span multiple standard marketing personas.
- **Gene Expression Analysis**: In bioinformatics, a single gene often plays multiple roles and is involved in several different biological functions simultaneously.

Probabilistic Model-Based Clustering

A Generative Approach

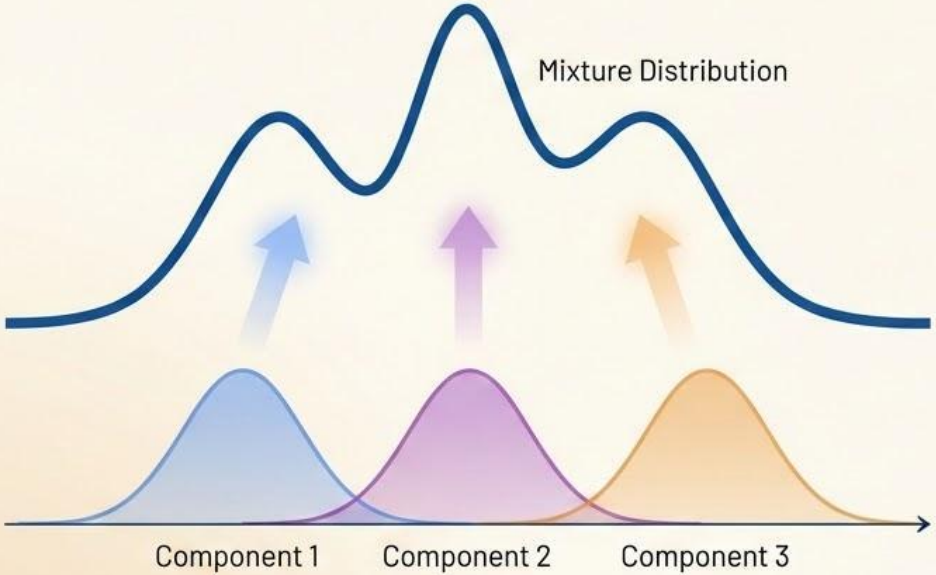


Figure: Complex data distribution (top) modeled as a mixture of simple Gaussian components (bottom), where each component is a distinct cluster.

- Assume the data is generated from a **mixture of probability distributions**.
- Crucially: Each distribution represents a distinct cluster.
- We model the overall complexity by identifying these **hidden simple components**.

Example 9.5. Probabilistic clusters. Suppose the digital cameras sold by an e-commerce company can be divided into two categories: C_1, a consumer line (e.g., point-and-shoot cameras), and C_2, a professional line (e.g., single-lens reflex cameras). Their respective probability density functions, f_{consumer} and $f_{\text{professional}}$, are shown in Fig. 9.1 with respect to the attribute *price*.

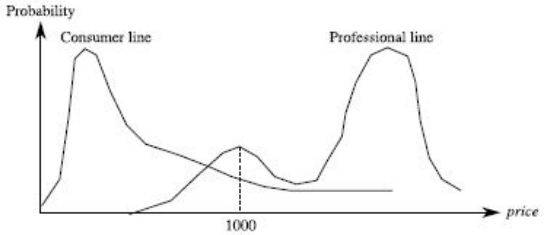


FIGURE 9.1

The probability density functions of two probabilistic clusters.

For a price value of, say, \$1000, $f_{\text{consumer}}(1000)$ is the relative likelihood that the price of a consumer-line camera is \$1000. Similarly, $f_{\text{professional}}(1000)$ is the relative likelihood that the price of a professional-line camera is \$1000.

The probability density functions, f_{consumer} and $f_{\text{professional}}$, cannot be observed directly. Instead, the company can only infer these distributions by analyzing the prices of the digital cameras it sells. Moreover, a camera often does not come with a well-determined category (e.g., "consumer line" or "professional line"). Instead, such categories are typically based on user background knowledge and can vary. For example, a camera in the *prosumer* segment may be regarded at the high end of the consumer line by some customers and the low end of the professional line by others.

As an analyst, you can consider each category as a probabilistic cluster and conduct cluster analysis on the price of cameras to approach these categories. □

Formally, let $o_1, \ldots, o_n$ be the n observed objects, and $\Theta_1, \ldots, \Theta_k$ be the parameters of the k distributions, denoted by $\mathbf{O} = \{o_1, \ldots, o_n\}$ and $\mathbf{\Theta} = \{\Theta_1, \ldots, \Theta_k\}$, respectively. Then, for any object $o_i \in \mathbf{O}$ $(1 \leq i \leq n)$, Eq. (9.5) can be rewritten as

$$P(o_i|\mathbf{\Theta}) = \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j), \tag{9.7}$$

where $P_j(o_i|\Theta_j)$ is the probability that o_i is generated from the jth distribution using parameter Θ_j. Consequently, Eq. (9.6) can be rewritten as

$$P(\mathbf{O}|\mathbf{\Theta}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j). \tag{9.8}$$

Using the parameterized probability distribution models, the task of probabilistic model-based cluster analysis is to infer a set of parameters, $\mathbf{\Theta}$, that maximizes Eq. (9.8).

Example 9.6. Univariate Gaussian mixture model. Let us use univariate Gaussian distributions as an example to illustrate probabilistic model-based clustering. That is, we assume that the probability density function of each cluster follows a 1-D Gaussian distribution. Suppose there are k clusters. The two parameters for the probability density function of each cluster are center, μ_j, and standard deviation, σ_j $(1 \leq j \leq k)$. We denote the parameters as $\Theta_j = (\mu_j, \sigma_j)$ and $\mathbf{\Theta} = \{\Theta_1, \ldots, \Theta_k\}$. Let the data set be $\mathbf{O} = \{o_1, \ldots, o_n\}$, where o_i $(1 \leq i \leq n)$ is a real number. For any point, $o_i \in \mathbf{O}$, we have

$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma^2}}. \tag{9.9}$$

Assuming that each cluster has the same probability, that is $\omega_1 = \omega_2 = \cdots = \omega_k = \frac{1}{k}$, and plugging Eq. (9.9) into Eq. (9.7), we have

$$P(o_i|\mathbf{\Theta}) = \frac{1}{k} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma^2}}. \tag{9.10}$$

Applying Eq. (9.8), we have

$$P(\mathbf{O}|\mathbf{\Theta}) = \frac{1}{k} \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma^2}}. \tag{9.11}$$

The task of probabilistic model-based cluster analysis using a univariate Gaussian mixture model is to infer $\mathbf{\Theta}$ such that Eq. (9.11) is maximized. □

The Mixture Model and Gaussian Variant (GMM)

The Mathematical Model: Defining the Components and GMM

The General Mixture Model

In probabilistic clustering, the entire data space is described as a combined model. We define the overall probability of a point **x** existing as a mathematical sum of these distinct distributions, each acting as a cluster.

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \cdot p_j(\mathbf{x}|\theta_j)$$

- ⚖ π_j **(Mixing Weight):** The probability that a data point naturally belongs to cluster *j*.
- ⛰ $p_j(\mathbf{x}|\theta_j)$ **(Probability Density):** The fundamental probability density function describing cluster *j* itself (e.g., a simple bell curve).

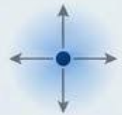
The Constraints: All mixing weights π_j across the entire system must mathematically sum to exactly 1.

Gaussian Mixture Model (GMM)

This is the most popular variant of the model, where we assume every cluster follows a multivariate Gaussian (normal) distribution.

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \cdot \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)$$

Each Gaussian component in the model has its own distinct **mean** (μ_j) (its center) and **covariance matrix** (Σ_j) (which defines its specific spread, orientation, and elliptical shape).



Mean (μ_j)

Covariance (Σ_j)

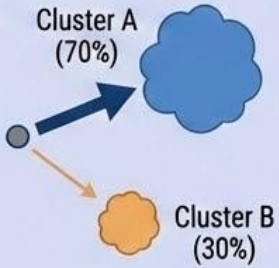
Advantages of Probabilistic Models

Capturing Nuance, Shape, and Complexity

Probabilistic models offer significant advantages over traditional “hard” clustering methods like k-means:

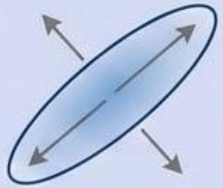
Nuanced Assignments

They do not force points into “in or out” categories. They provide **natural, continuous probabilistic cluster assignments** (e.g., a point has a 70% chance of being in Cluster A and a 30% chance of being in Cluster B). Ambiguity is embraced.



Capturing Shape

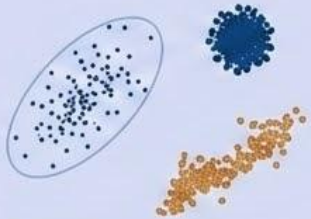
We explicitly capture the **actual geometric shape, spread, and elliptical orientation** of each cluster through its covariance matrix.



Covariance Matrix defines shape/orientation.

Handling Complexity

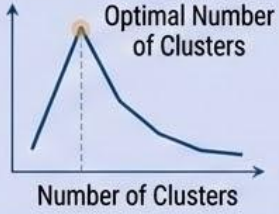
This approach naturally handles clusters with vastly different sizes, densities, and orientations—a scenario where k-means frequently fails.



(K-means often fails here)

Automatic Complexity Selection

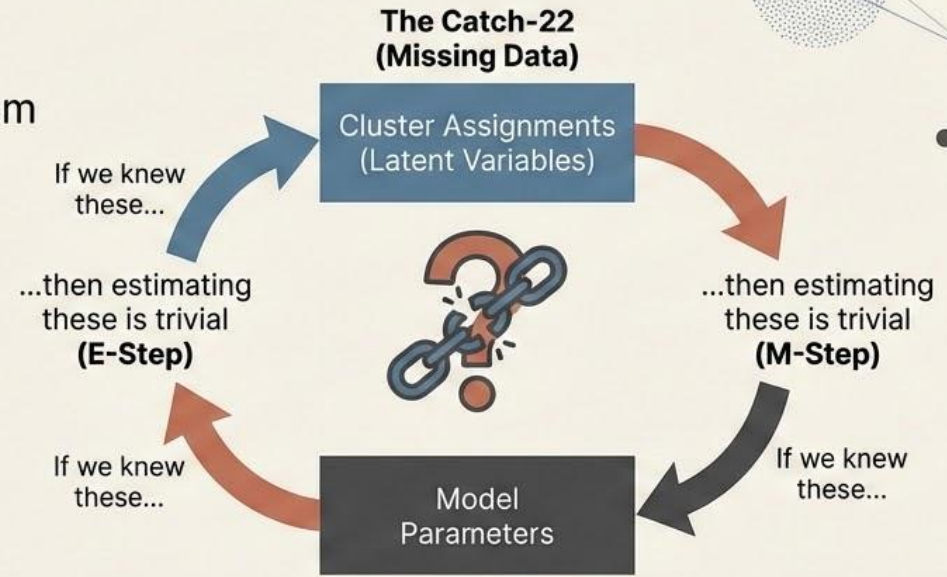
Through model selection techniques, the system can automatically and rigorously determine the mathematically optimal number of clusters for the dataset.



Expectation-Maximization (EM) Algorithm

Solving the 'Chicken-and-Egg' Problem

The Problem:
Probabilistic model-based clustering assumes data is generated from hidden components (latent variables).



Reality: We know neither! The data is incomplete because we are missing the labels indicating which cluster generated which point.

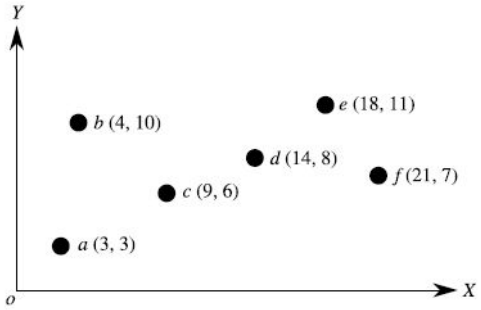


FIGURE 9.2

Data set for fuzzy clustering.

Example 9.7. Fuzzy clustering using the EM algorithm. Consider the six points in Fig. 9.2, where the coordinates of the points are also shown. Let us compute two fuzzy clusters using the EM algorithm.

We randomly select two points, say $c_1 = a$ and $c_2 = b$, as the initial centers of the two clusters. The first iteration conducts the expectation step and the maximization step as follows.

In the **E-step**, for each point we calculate its membership degree in each cluster. For any point, o, we assign o to c_1 with membership weight

$$\frac{\frac{1}{dist(o,c_1)^2}}{\frac{1}{dist(o,c_1)^2}+\frac{1}{dist(o,c_2)^2}} = \frac{dist(o,c_2)^2}{dist(o,c_1)^2+dist(o,c_2)^2}$$

and to c_2 with membership weight $\frac{dist(o,c_1)^2}{dist(o,c_1)^2+dist(o,c_2)^2}$, where $dist(,)$ is the Euclidean distance. The rationale is that, if o is close to c_1 and $dist(o, c_1)$ is small, the membership degree of o with respect to c_1 should be high. We also normalize the membership degrees so that the sum of degrees for an object is equal to 1.

For point a, we have $w_{a,c_1} = 1$ and $w_{a,c_2} = 0$. That is, a exclusively belongs to c_1. For point b, we have $w_{b,c_1} = 0$ and $w_{b,c_2} = 1$. For point c, we have $w_{c,c_1} = \frac{41}{45+41} = 0.48$ and $w_{c,c_2} = \frac{45}{45+41} = 0.52$. The degrees of membership of the other points are shown in the partition matrix in Table 9.3. □

In the **M-step**, we recalculate the centroids according to the partition matrix, minimizing the SSE given in Eq. (9.4) where we assume the parameter $p = 2$ in this example. The new centroid should be adjusted to

$$c_j = \frac{\sum_{\text{each point } o} w_{o,c_j}^2 o}{\sum_{\text{each point } o} w_{o,c_j}^2}, \tag{9.12}$$

where $j = 1, 2$.

Table 9.3 Intermediate results from the first three iterations of the EM algorithm in Example 9.7.

Iteration	E-Step	M-Step
1	$\mathbf{M}^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$	$c_1 = (8.47, 5.12)$ $c_2 = (10.42, 8.99)$
2	$\mathbf{M}^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$	$c_1 = (8.51, 6.11)$ $c_2 = (14.42, 8.69)$
3	$\mathbf{M}^T = \begin{bmatrix} 0.80 & 0.76 & 0.99 & 0.02 & 0.14 & 0.23 \\ 0.20 & 0.24 & 0.01 & 0.98 & 0.86 & 0.77 \end{bmatrix}$	$c_1 = (6.40, 6.24)$ $c_2 = (16.55, 8.64)$

In this example,

$$c_1 = \left(\frac{1^2 \times 3 + 0^2 \times 4 + 0.48^2 \times 9 + 0.42^2 \times 14 + 0.41^2 \times 18 + 0.47^2 \times 21}{1^2 + 0^2 + 0.48^2 + 0.42^2 + 0.41^2 + 0.47^2}, \right.$$
$$\left. \frac{1^2 \times 3 + 0^2 \times 10 + 0.48^2 \times 6 + 0.42^2 \times 8 + 0.41^2 \times 11 + 0.47^2 \times 7}{1^2 + 0^2 + 0.48^2 + 0.42^2 + 0.41^2 + 0.47^2} \right)$$
$$= (8.47, 5.12)$$

and

$$c_2 = \left(\frac{0^2 \times 3 + 1^2 \times 4 + 0.52^2 \times 9 + 0.58^2 \times 14 + 0.59^2 \times 18 + 0.53^2 \times 21}{0^2 + 1^2 + 0.52^2 + 0.58^2 + 0.59^2 + 0.53^2}, \right.$$
$$\left. \frac{0^2 \times 3 + 1^2 \times 10 + 0.52^2 \times 6 + 0.58^2 \times 8 + 0.59^2 \times 11 + 0.53^2 \times 7}{0^2 + 1^2 + 0.52^2 + 0.58^2 + 0.59^2 + 0.53^2} \right)$$
$$= (10.42, 8.99).$$

We repeat the iterations, where each iteration contains an E-step and an M-step. Table 9.3 shows the results from the first three iterations. The algorithm stops when the cluster centers converge or the change is small enough.

Example 9.8. Using the EM algorithm for mixture models. Given a set of objects, $\mathbf{O} = \{o_1, \ldots, o_n\}$, we want to mine a set of parameters, $\mathbf{\Theta} = \{\Theta_1, \ldots, \Theta_k\}$, such that $P(\mathbf{O}|\mathbf{\Theta})$ in Eq. (9.11) is maximized, where $\Theta_j = (\mu_j, \sigma_j)$ are the mean and standard deviation, respectively, of the jth univariate Gaussian distribution, $(1 \leq j \leq k)$.

We can apply the EM algorithm. We assign random values to parameters $\mathbf{\Theta}$ as the initial values. We then iteratively conduct the E-step and the M-step as follows until the parameters converge or the change is sufficiently small.

In the **E-step**, for each object, $o_i \in \mathbf{O}$ $(1 \leq i \leq n)$, we calculate the probability that o_i belongs to each distribution, that is,

$$P(\Theta_j|o_i, \mathbf{\Theta}) = \frac{P(o_i|\Theta_j)}{\sum_{l=1}^{k} P(o_i|\Theta_l)}. \tag{9.13}$$

In the **M-step**, we adjust the parameters $\mathbf{\Theta}$ so that the expected likelihood $P(\mathbf{O}|\mathbf{\Theta})$ in Eq. (9.11) is maximized. This can be achieved by setting

$$\mu_j = \frac{1}{k} \sum_{i=1}^{n} o_i \frac{P(\Theta_j|o_i, \mathbf{\Theta})}{\sum_{l=1}^{n} P(\Theta_j|o_l, \mathbf{\Theta})} = \frac{1}{k} \frac{\sum_{i=1}^{n} o_i P(\Theta_j|o_i, \mathbf{\Theta})}{\sum_{i=1}^{n} P(\Theta_j|o_i, \mathbf{\Theta})} \tag{9.14}$$

and

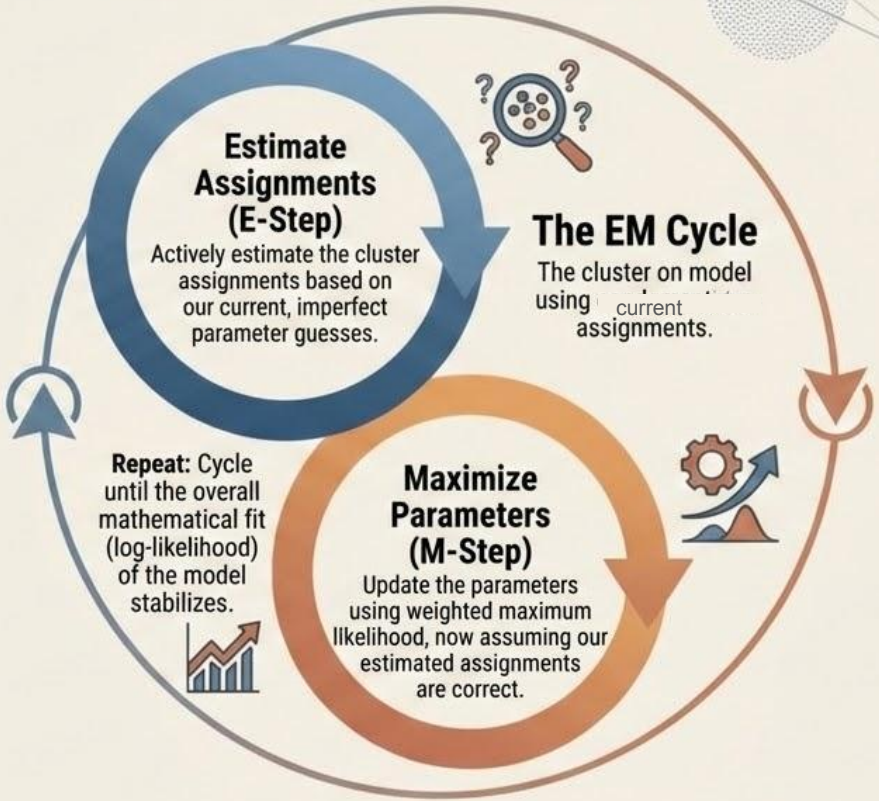
$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{n} P(\Theta_j|o_i, \mathbf{\Theta})(o_i - u_j)^2}{\sum_{i=1}^{n} P(\Theta_j|o_i, \mathbf{\Theta})}}. \tag{9.15}$$

□

The Iterative EM Solution

Alternate and Maximize

EM Core Idea
Instead of trying to solve assignments and parameters simultaneously (which is computationally difficult), we use an iterative method that alternates between two steps.

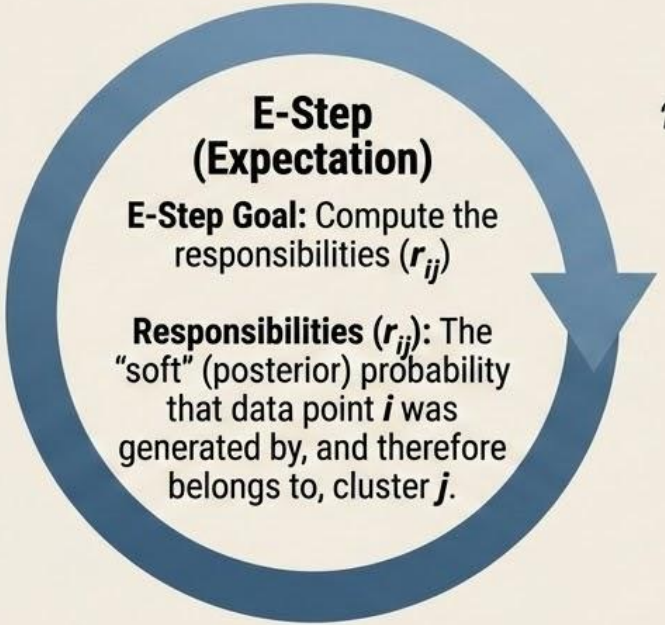


Algorithm Steps: Initialize & E-Step

Computing Responsibilities (Posterior Probabilities)

Initialize (Step 0)

- Choose initial starting parameters θ_j (including initial means, covariances, and mixing weights).



$$r_{ij} = \frac{\pi_j \cdot p(x_i|\theta_j)}{\sum_{l=1}^{k} \pi_l \cdot p(x_i|\theta_l)}$$

Numerator (Top)

How likely cluster *j* generated point *i*, weighted by cluster *j*'s overall size (π_j).

Denominator (Bottom)

The total, combined likelihood for point *i* across all clusters.

Algorithm Steps: The M-Step for GMM

Updating Parameters via Weighted Maximum Likelihood

Goal: Update the parameters $\boldsymbol{\theta_j}$ by performing weighted maximum likelihood estimation. We treat the responsibilities ($\boldsymbol{r_{ij}}$) as weights for the data points.

Updated Mean (μ_j)

$$\mu_j = \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}}$$

Interpretation: The average of the data points, but weighted by their probability of belonging to cluster ***j***.

Updated Covariance (Σ_j)

$$\Sigma_j = \frac{\sum_i r_{ij} (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_i r_{ij}}$$

Interpretation: Measuring the spread/ellipse shape based on the updated mean, again using probabilities as weights.

Updated Mixing Weight (π_j)

$$\pi_j = \frac{\sum_i r_{ij}}{n}$$

Interpretation: The average responsibility assigned to cluster ***j*** across all ***n*** points.

Convergence & Model Selection

Log-Likelihood and GMM

Repeat until Convergence

Convergence Properties

- EM guarantees non-decreasing likelihood (every E-M cycle must improve the fit or keep it the same).
- Convergence is guaranteed, but only to a local optimum.

Multiple random restarts are therefore highly recommended.

Model Selection for GMM (Determining K)

Used to balance mathematical fit (high log-likelihood, L) vs. model complexity (number of parameters, M, and clusters, K). We seek the simplest model with the best fit.

AIC (Akaike Information Criterion)

$$AIC = -2 \log L + 2M$$

BIC (Bayesian Information Criterion)

$$BIC = -2 \log L + M \log n$$

Rule: Lower values indicate better models (the criteria punish overly complex models that overfit).

Part 3: Clustering High-Dimensional Data

The Challenge of Hyper-Space

Objective

- Explore why traditional clustering algorithms, which thrive on geometric intuition, fundamentally fail when faced with hundreds or thousands of dimensions.

What is High-Dimensional?

Low-D:
2D, 3D (easy to visualize, points cluster tightly).

High-D:
(e.g., Gene expression data with 10,000 genes, or text analysis with 5,000 unique word vectors).

The Reality: High-dimensional space behaves differently from our 2D/3D intuition.

Table 9.4 Customer purchase data.

Customer	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Ada	1	0	0	0	0	0	0	0	0	0
Bob	0	0	0	0	0	0	0	0	0	1
Cathy	1	0	0	0	1	0	0	0	0	1

Example 9.9. High-dimensional data and clustering. Consider an e-commerce company that keeps track of the products purchased by every customer. As a customer-relationship manager, you want to cluster customers into groups according to what they purchased from the company.

The customer purchase data are of very high dimensionality. The company carries tens of thousands of products. Therefore a customer's purchase profile, which is a vector of the products carried by the company, has tens of thousands of dimensions.

"Are the traditional distance measures, which are frequently used in low-dimensional cluster analysis, also effective on high-dimensional data?" Consider the customers in Table 9.4, where 10 products, $P_1, \ldots, P_{10}$, are used in demonstration. If a customer purchases a product, a 1 is set at the corresponding bit; otherwise, a 0 appears. Let us calculate the Euclidean distances among Ada, Bob, and Cathy. It is easy to see that

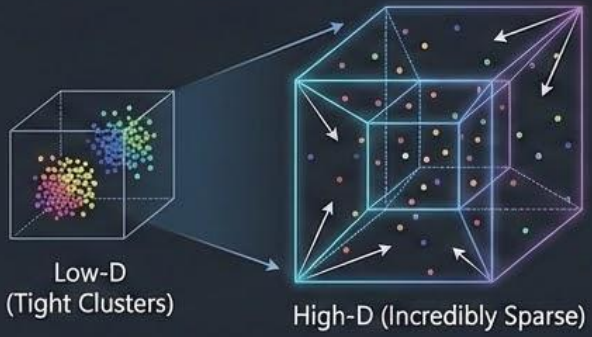
$$dist(\text{Ada}, \text{Bob}) = dist(\text{Bob}, \text{Cathy}) = dist(\text{Ada}, \text{Cathy}) = \sqrt{2}.$$

According to Euclidean distance, the three customers are equivalently similar (or dissimilar) to each other. However, a close look tells us that Ada should be more similar to Cathy than to Bob because Ada and Cathy share one common purchased item, P_1. □

The Curse of Dimensionality

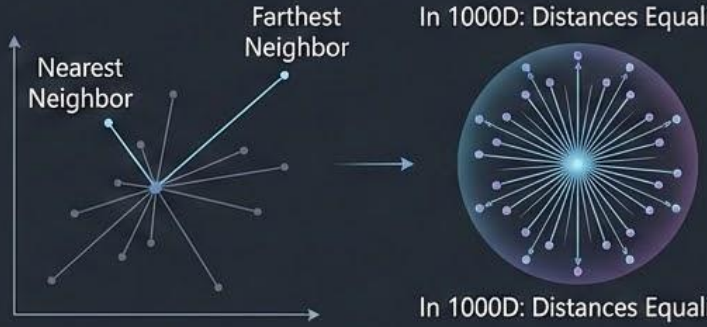
Geometry Breaks Down

Sparsity & Distance Concentration



The most damaging geometric breakdown. In low dimensions, data points can easily form tight clusters. In high dimensions, data points become incredibly sparse.

Distance Concentration Problem



In 1000D, Euclidean distance itself loses discriminative meaning. Mathematically, all pairwise distances between all points become nearly equal. The distance to the "nearest neighbor" looks almost identical to the distance to the "farthest neighbor."

Conclusion

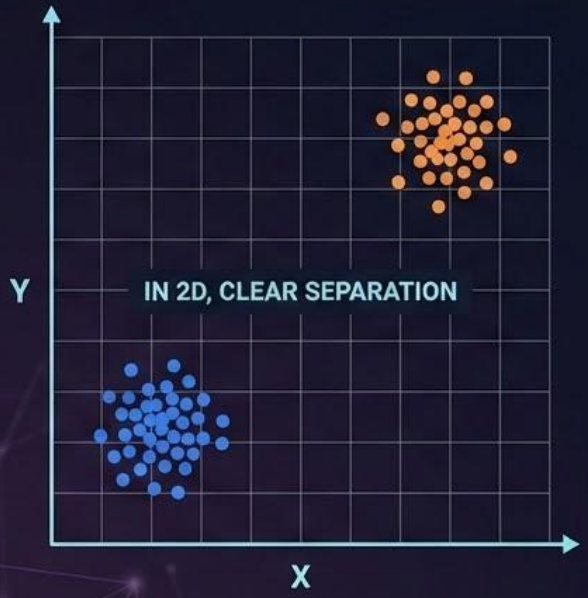
Traditional distance metrics lose all statistical meaning for segmentation. The "Curse" is that as you increase the dimensionality (add more features), the statistical power of standard algorithms is diluted to near-zero.

Visualizing the Geometry Breakdown

Sparsity and the Equidistance Trap

2D Intuition: Clear Separation

Points form dense, clear, tightly packed clusters on a simple grid. They naturally separate.



High-D Reality: Sparsity & Equidistance

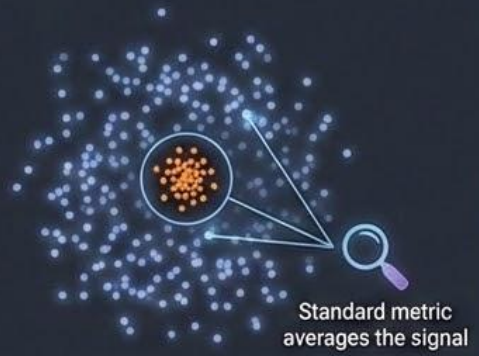
The same number of points are scattered wildly. Data points are isolated and far apart. All points are connected by glowing, almost identical distance lines, creating a uniform, confusing, equidistant web.

Key Problems in High-Dimensional Clustering

Diluting and Hiding the Signal

The geometric collapse leads to specific, fatal problems for standard algorithms:

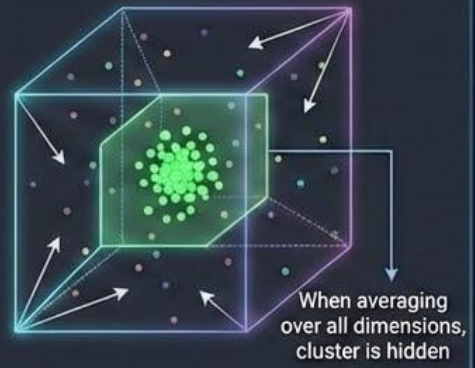
Noise Dimensions



Often, only a tiny fraction of the thousands of features actually define a true cluster (e.g., 50 genes out of 10,000 are relevant).

The standard metric averages the "signal" over the remaining 99% of noisy dimensions, obscuring the real pattern.

Subspace Clusters

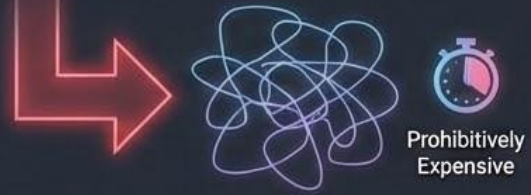


This is the most complex problem. True clusters may exist only in a specific subset of dimensions (a subspace), but they are hidden when we average the distance over all dimensions. dimensions. Traditional methods look globally, washing the signal out.

Computational Complexity

Standard k-means is $O(nkd)$

n=points k=clusters d=dimensions



When d becomes thousands, this complexity becomes prohibitively expensive for very large datasets (n).

Part 4: Clustering Methods for High-Dimensional Data

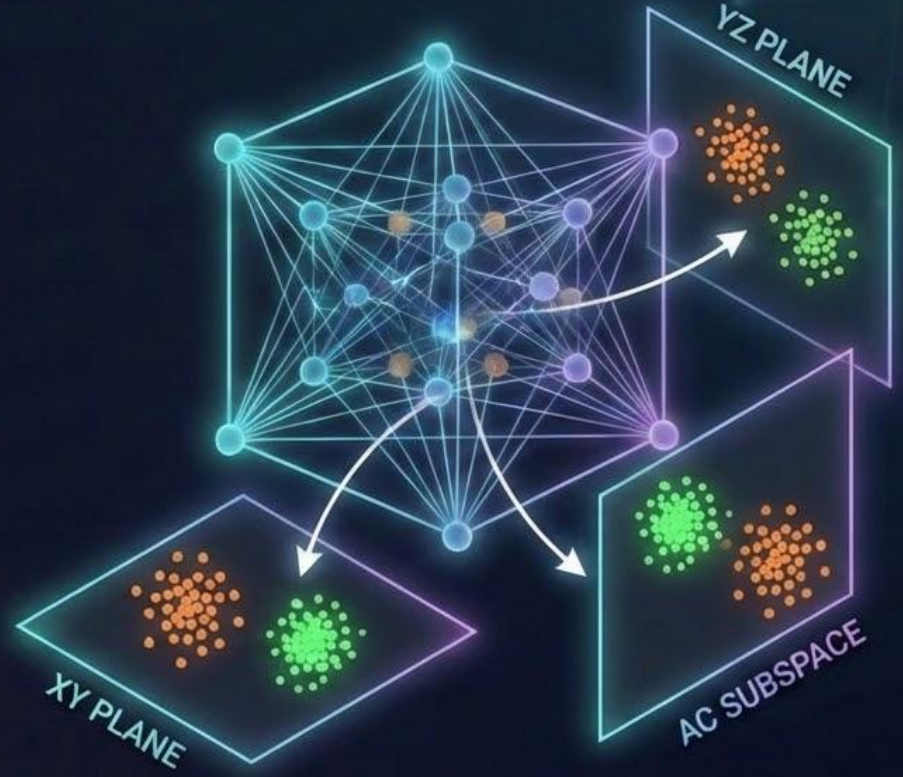
Uncovering Signal in Axis-Parallel Subspaces

We must move beyond standard distance-based clustering. We now actively look for axis-parallel subspace solutions to the Curse of Dimensionality.

The Axis-Parallel Strategy (The Solution)

Instead of calculating distances globally (diluting the signal), we project the massive, confused **high-D web down into simple, low-dimensional subspaces.**

We look only at specific, orthogonal planes (parallel to the main axes) where true clusters may be hidden.
We are mathematically "uncovering" order hidden in the noise.

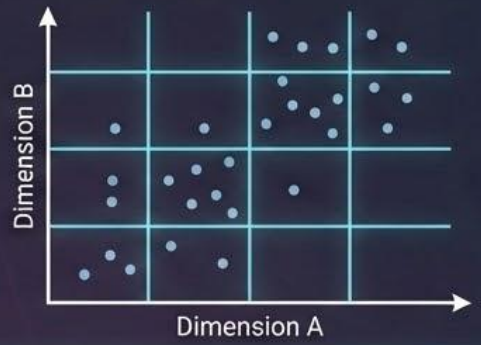


Key Algorithm 1: CLIQUE

Clustering In QUEst (A Grid-Based Approach)

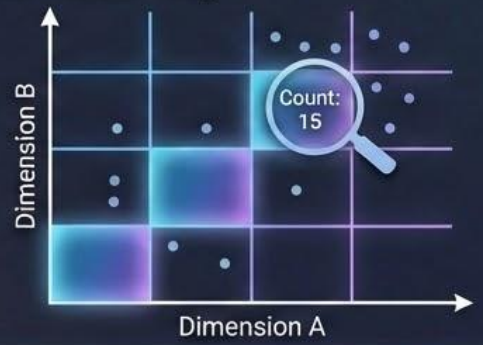
CLIQUE Strategy:
A hybrid approach that combines a grid-based and a density-based methodology for subspace discovery.

1. The Grid (Partition):



The algorithm starts by partitioning every single original dimension into simple, non-overlapping grid cells (intervals).

2. The Subspace Hunt (Density Identification):



For every potential subspace (e.g., all 2D subspaces), CLIQUE counts the data points in each grid cell. It strictly identifies dense cells.

3. The Cluster Reveal (Merge):

Once dense subspace cells are identified, it groups them, merging adjacent dense cells to form clusters. A cluster is a maximal region of dense, contiguous subspace grid units.

Key Algorithm 2: PROCLUS

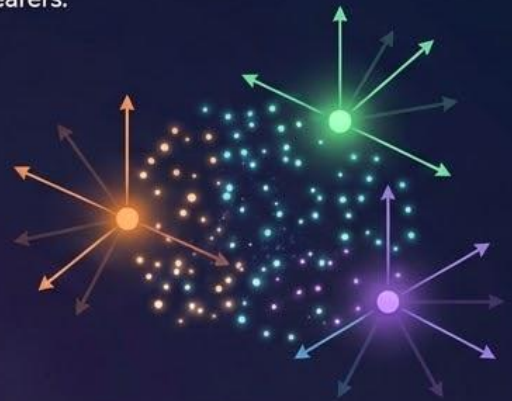
PROjected CLUStering (Representative Points)

PROCLUS Strategy

A completely different, representative point-based methodology (similar to k-medoids) designed specifically for subspace projection.

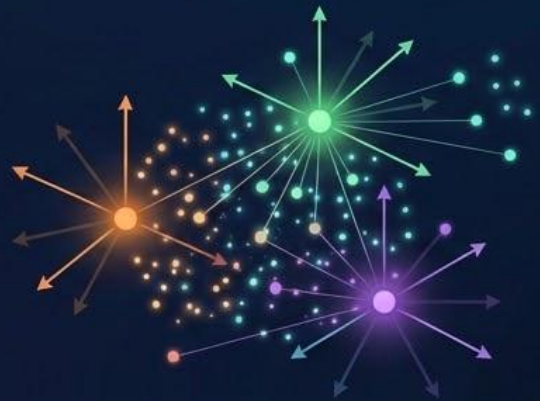
The Projective Hunt

- **Select Medoids**: It intelligently selects a set of representative points (medoids) to act as cluster prototypes (K points for K desired clusters).
- **Determine Dimensions:** For every specific cluster (medoid), it analyzes the data neighborhood to strictly determine its own set of relevant dimensions. Crucially, dimensions are not shared globally; each cluster (projection) has its own specific features that are statistical order-bearers.



The Final Assign

3. **Subspace Distances:** Points are then assigned to their nearest cluster (medoid). This assignment, however, does not use a global distance metric. It uses subspace distances (e.g., Mahalanobis or standard Euclidean, but calculated only over that specific cluster's relevant dimensions), strictly ignoring all global noise.



Characteristics & Discussion

Interpretability and Axis-Parallel Limits

Interpretable Results

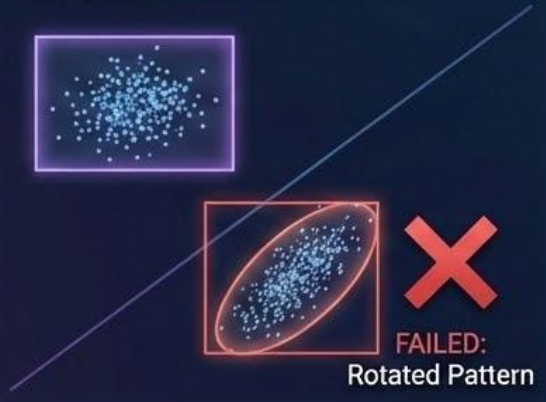
Every cluster is described by its own set of original, un-transformed features. A result might be:

"A specific consumer segment exists defined only by (Age) and (Purchase Frequency)." This is interpretable.

Axis-Parallel Boundary Trap (Constraint)

This entire family of approaches is constrained. They can only discover clusters with axis-parallel boundaries.

They completely fail if the actual pattern is rotated (skewed) or exists on non-linear manifolds that are not orthogonal to the main axes.



Scalability

Generally scalable to moderate dimensions (d in the low to mid-thousands), as long as the true subspaces remain relatively small ($d_{sub} \ll d$).

They struggle for massive global datasets (n).

Part 5: Arbitrarily Oriented Subspace Approaches

Moving Beyond the Axis-Parallel Barrier

The Limitation of Axis-Parallel (The Problem)

The algorithms we just discussed (like CLIQUE) have a strict geometric boundary trap. They completely fail when true clusters exist in rotated subspaces that are not perfectly aligned with the main axes (X, Y, Z).

The Arbitrarily Oriented Strategy (The Solution)

We move beyond standard projections and look actively for diagonal, oblique, or complex structures that can be mathematically discovered through rotation and local geometry.

Key Algorithm 1: Correlation Clustering

Finding Lower-Dimensional Affine Subspaces

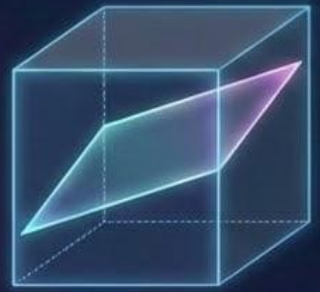
Correlation Strategy

This is a family of approaches that explicitly targets the statistical relationships (correlations) between variables, rather than just geometric proximity.

The Goal

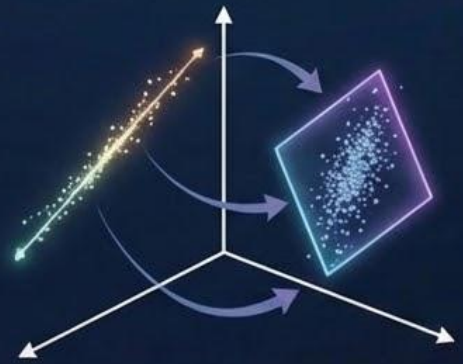
Find clusters of points where the data points lie tightly within lower-dimensional affine subspaces.

- An **Affine Subspace** is essentially a shifted linear subspace (e.g., a simple line or plane that doesn't necessarily pass through the coordinate origin).



The Defining Property

Crucially, each cluster possesses its own unique mathematical orientation in the high-dimensional space. The structure is not defined by standard subsets of axes, but by its own rotated coordinate system.

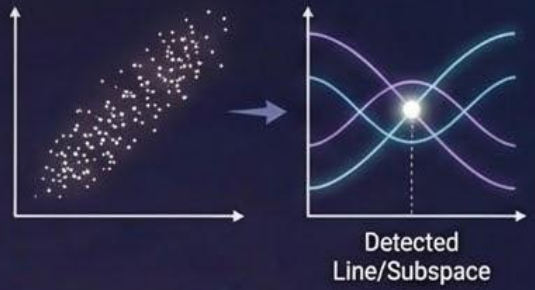


Key Algorithm 2: CASH

Hough Transform for Arbitrary Subspaces

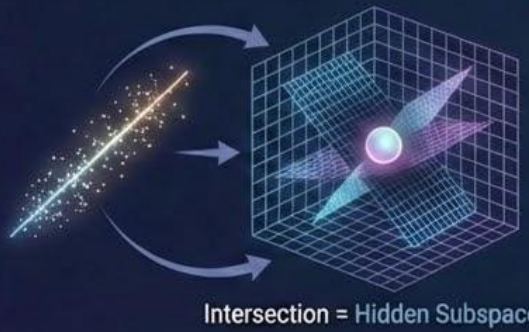
CASH & The Mechanism

- CASH: Clustering in Arbitrary SubSpaces using Hough Transform.



Uses the Hough transform, a classic mathematical technique most famous in computer vision for detecting straight lines or specific shapes in noisy pixel data.

The CASH Twist



CASH maps the high-dimensional data points into a complex, dual 'parameter space' (similar to parameterizing a line).

It then uses a robust, efficient grid-based approach within that parameter space to detect intersecting surfaces. These intersections are statistical evidence of hidden subspaces.

Characteristics

Works effectively for both standard axis-parallel clusters and highly rotated (arbitrarily oriented) patterns.

CASH is very robust against background noise, as the transformation naturally concentrates the signal.

Key Family: PCA-Based Approaches

Local Covariance Structure Analysis

PCA-Based Strategy

This isn't global dimensionality reduction. It is a local, robust methodology for analyzing geometry and orientation.

The Iterative Local Hunt

1. **Candidate Cluster Identification:**
   Start by identifying potential, robust candidate clusters (e.g., using a fast density-based pre-clustering).

2. **Perform Local PCA:**
   For every candidate cluster, the algorithm does not look at the global dataset. It performs PCA locally within that specific group.

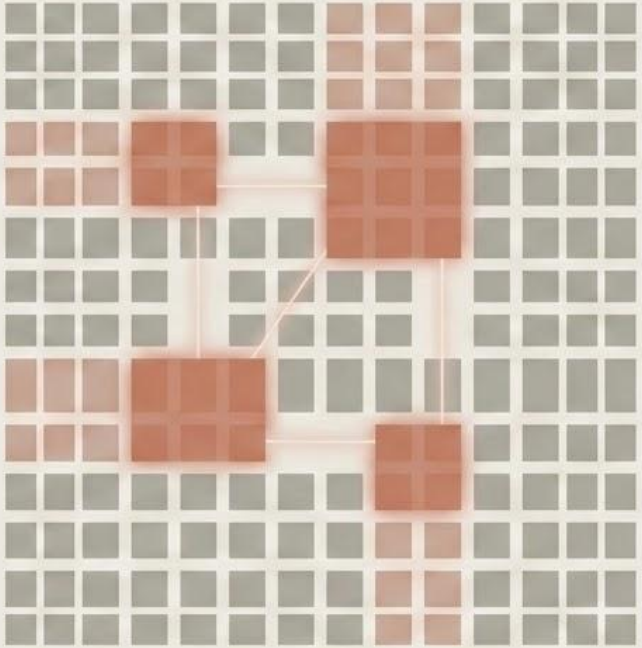
3. **Determine Intrinsic Dimensionality:**
   The local PCA reveals the group's mathematical "flattening"—its true intrinsic dimensionality (e.g., this dense "blob" is actually a perfectly flat plane, which global distance would get wrong).

4. **The Final Grouping Logic: Similar Covariance Structure**
   Finally, points are clustered together if they exhibit highly similar local covariance structure—meaning they they share the same density, mathematical shape, and unique orientation (rotation).

Introduction to Biclustering

Clustering Both Sides of the Coin

Part 4 - Moving beyond one-dimensional grouping to find complex, localized patterns in data matrices.



Example 9.10. Gene expression. Gene expression matrices are popular in bioinformatics research and development. For example, an important task is to classify a new gene using the expression data of the gene and that of other genes in known classes. Symmetrically, we may classify a new sample (e.g., a new patient) using the expression data of the sample and that of samples in known classes (e.g., tumor and nontumor). Such tasks are invaluable in understanding the mechanisms of diseases and in clinical treatment. □

Example 9.11. Using biclustering for a recommender system. Imagine that an e-commerce company collects data from customers' evaluations of products and uses the data to recommend products to customers. The data can be modeled as a customer-product matrix, where each row represents a customer, and each column represents a product. Each element in the matrix represents a customer's evaluation of a product, which may be a score (e.g., like, like somewhat, not like) or purchase behavior (e.g., buy or not). Fig. 9.6 illustrates the structure.

The customer-product matrix can be analyzed in two dimensions: the *customer* dimension and the *product* dimension. Treating each customer as an object and products as attributes, the company can

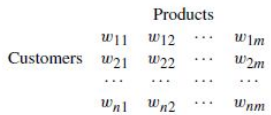


FIGURE 9.6

Customer-product matrix.

find customer groups that have similar preferences or purchase patterns. Using products as objects and customers as attributes, the company can mine product groups that are similar in customer interest.

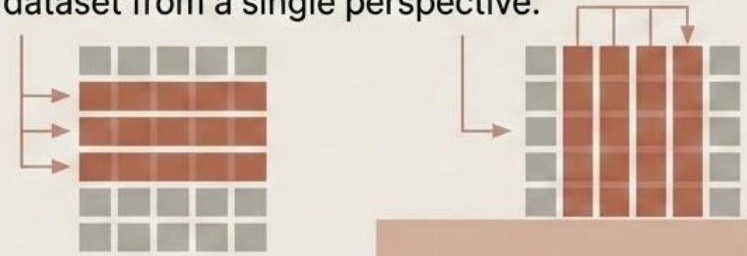
Moreover, the company can mine clusters in both customers and products simultaneously. Such a cluster contains a subset of customers and involves a subset of products. For example, the company may be highly interested in finding a group of customers who all like the same group of products. Such a cluster is a submatrix in the customer-product matrix, where all elements have a high value. Using such a cluster, the company can make recommendations in two directions. First, the company can recommend products to new customers who are similar to the customers in the cluster. Second, the company can recommend to customers new products that are similar to those involved in the cluster. □

Traditional vs. Biclustering

The Core Difference

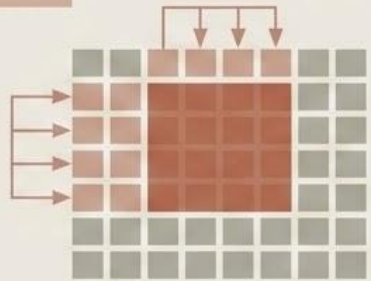
Traditional Clustering

Groups similar rows (objects) or similar columns (features) independently. It evaluates the entire dataset from a single perspective.



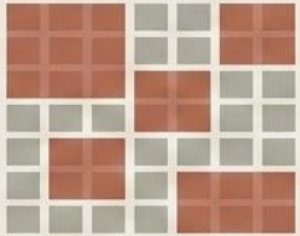
Biclustering

Simultaneously clusters rows AND columns at the exact same time.



The Goal

To find specific submatrices (smaller blocks within the whole dataset) where a subset of objects exhibits highly coherent and correlated values across a specific subset of features.

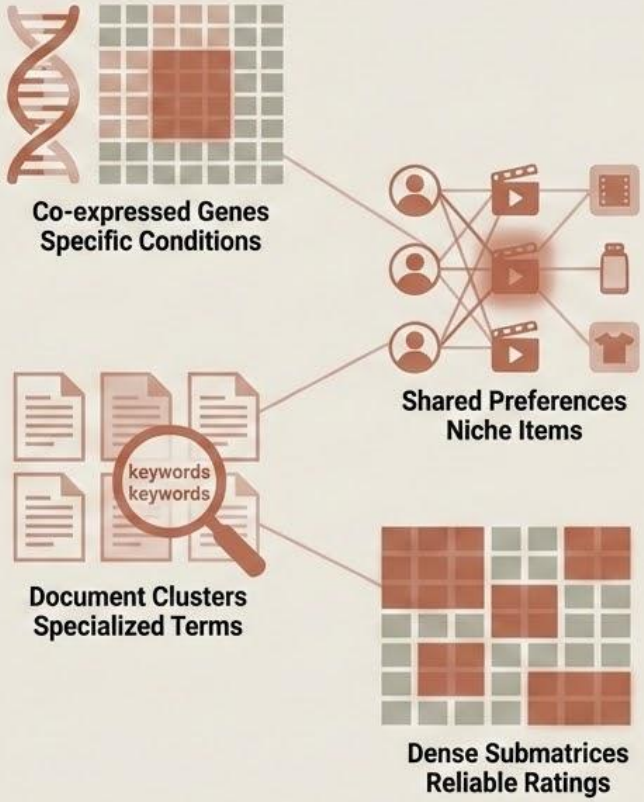


Real-World Applications

Where Simultaneous Grouping Excels

Biclustering is the go-to solution when patterns only emerge under specific conditions:

- **Gene Expression Analysis:** Identifying specific groups of genes that are perfectly co-expressed, but only under a specific subset of experimental conditions.
- **Recommender Systems:** Grouping subsets of users who share highly similar preferences for a very specific subset of items (e.g., a niche movie genre).
- **Text Mining:** Discovering distinct clusters of documents that share a highly coherent group of specific, specialized terms.
- **Collaborative Filtering:** Isolating dense submatrices filled with consistent, reliable ratings within a mostly sparse database.



The Gene Expression Example

Spotting the Hidden Submatrix

Consider this simplified matrix. A traditional algorithm might struggle to globally group these genes because their behaviors flip drastically depending on the condition.

	Condition 1	Condition 2	Condition 3	Condition 4
Gene A	10	c2:45	12	c4:48
Gene B	12	c2:47	11	c4:49
Gene C	8	c2:51	28	815
Gene D	82	10	84	13

The Hidden Bicluster

Look closely at {Gene A, Gene B}. They are highly correlated, but only when we isolate {Condition 1, Condition 3}.
A biclustering algorithm mathematically isolates this exact 2x2 submatrix block while ignoring the rest of the noise.

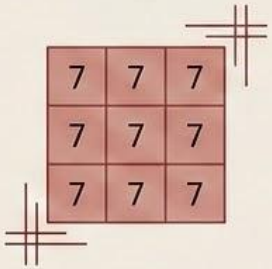
Types of Biclusters

Recognizing Patterns in the Matrix

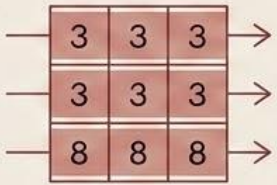
The Challenge: Not all biclusters look the same. Depending on what we are looking for (e.g., identical expression vs. similar trends), the algorithms must hunt for different mathematical structures.

The Five Primary Types:

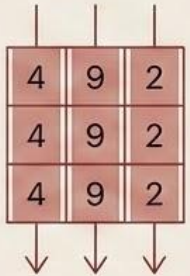
Constant Values



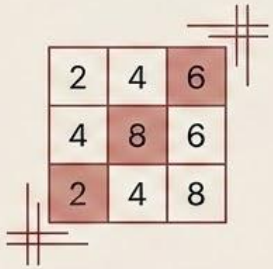
Constant Rows



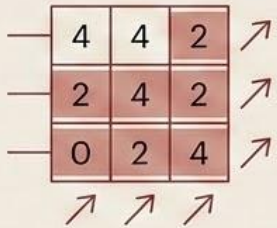
Constant Columns



Coherent Values



Coherent Evolutions



The "Constant" Biclusters: Strict Uniformity

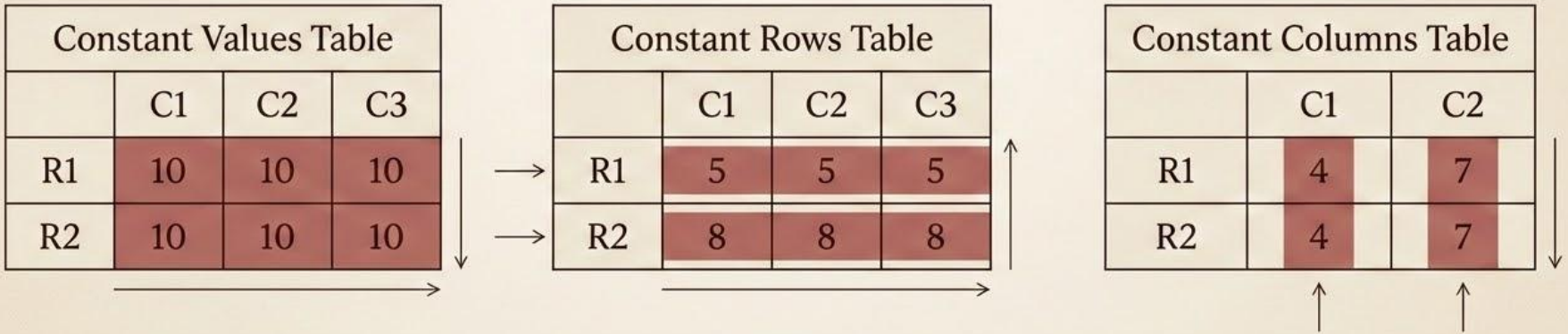
Constant Values, Rows, and Columns

These first three types look for exact, absolute uniformity across the submatrix.

1. Constant Values:
Every single cell in the submatrix is completely identical.

2. Constant Rows:
Each row has a constant value across all columns (but rows differ from each other).

3. Constant Columns:
Each column has a constant value across all rows.



Shifting Patterns and Trends

Coherent Values and Evolutions.

Coherent Biclusters

These types look for mathematical relationships and behavioral trends rather than identical numbers.

4. Coherent Values:
Values follow a strict mathematical pattern (additive or multiplicative). Notice how **R2** is exactly **R1 + 5**, and **C2** is **C1 + 10**.

	C1	C2	C3
R1	10	20	30
R2	15	25	35

5. Coherent Evolutions:
Rows or columns exhibit the exact same behavioral trend (e.g., ‘increase then decrease’), regardless of the exact numerical values.

	T1	T2	T3	Trend
R1	10	50	20	↑ then ↓
R2	2	18	5	↑ then ↓

Algorithmic Approaches to Biclustering

How We Actually Find the Patterns

The Challenge:
Finding a highly correlated submatrix hidden inside a massive global matrix is computationally heavy.

The Solutions:
We rely on four primary categories of algorithms to extract these biclusters efficiently:

Iterative Row-Column Clustering

Greedy Methods

Divide-and-Conquer

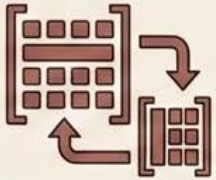
Probabilistic Methods

Iterative & Greedy Methods

Refining and Pruning the Data

Iterative Row-Column

Alternates between clustering the rows, then the columns, and continuously refines the result until it stabilizes.



Classic Example

CC (Cheng and Church) Algorithm

Key Metric: Uses a “mean squared residue score” to measure how coherent the resulting submatrix is.

$$f = \sqrt{M^2}$$

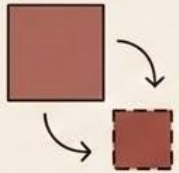
Greedy Methods

Starts with the entire, full matrix. It then aggressively and greedily removes specific rows or columns that deviate from the desired pattern.

Classic Example

δ-bicluster Approach

Key Feature: Works by pruning away the “bad” data until only a tight, highly correlated bicluster remains.



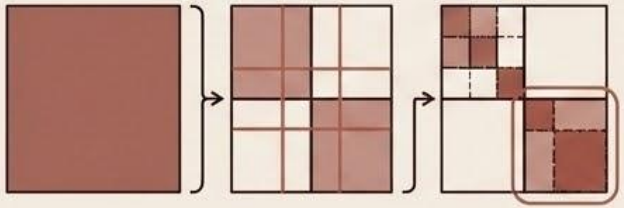
Structural & Statistical Approaches

Divide-and-Conquer & Probabilistic Methods

These methods use structural splits or underlying statistical models to find groupings.

3. Divide-and-Conquer Methods

The Approach: Recursively splits the massive global matrix into smaller and smaller sub-blocks (checkerboard style) until coherent biclusters are isolated.



Classic Example:
Spectral Biclustering.

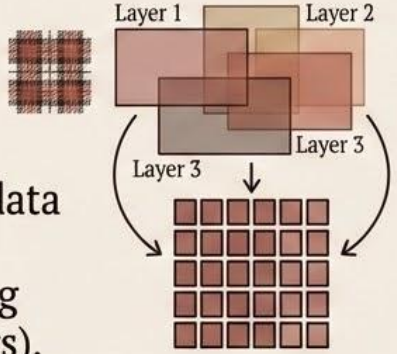
4. Probabilistic Methods

The Approach: Uses generative statistical models to represent the biclusters as hidden "latent factors" that influence the observed data.

Classic Example

The Plaid Model

(which models the data matrix as a sum of multiple overlapping "layers" or biclusters).



The MaPle Algorithm: Enumerating All Biclusters

Mining Patterns with Maximum Likelihood Estimation (MaPle)

The Shift in Strategy

Previous algorithms hunt for single optimal or use heuristics. MaPle aims for comprehensive discovery.

Core Definition

Specifically designed to discover all maximal biclusters within a dataset.

The Methodology

Abandons simple greedy searches; utilizes a sophisticated pattern growth approach to systematically build and verify clusters.

Target Bicluster Type

Highly specialized in finding biclusters with coherent values (both additive and multiplicative patterns).

Key Features of MaPle

Completeness and Coherence

Guaranteed Completeness

MaPle's defining feature is its mathematical guarantee. It will find all maximal biclusters in the data space; it does not stop at local optima.

Maximal Focus

It identifies patterns strictly as maximum possible combinations of row and column clusters (meaning the bicluster cannot cannot be expanded further without losing its coherence).

Versatile Pattern Recognition

It successfully handles and identifies both additive (e.g., Row 2 = Row 1 + 5) and multiplicative (e.g., Row 2 = Row 1 * 2) coherent patterns seamlessly.

Algorithmic Properties

How MaPle Manages Complexity

Efficient Pruning via Monotonicity

It utilizes the property of monotonicity to safely “prune” or cut off massive sections of the search space. If a small pattern fails the coherence test, MaPle knows that any larger pattern containing it will also fail, saving immense computation time.

Quality Evaluation

As its name suggests, it uses Maximum Likelihood Estimation (MLE) as its rigorous statistical metric to evaluate and score the ultimate quality of the discovered biclusters.

Scalability

Thanks to its aggressive pruning, it remains practically scalable to moderately sized matrices, though it will eventually face limits on truly massive, hyper-dimensional datasets.

Part 5 (8 Minutes) - How transforming the data space unlocks the potential of clustering algorithms in high dimensions.

Part 5: Dimensionality Reduction for Clustering

The Crucial Preprocessing Step

Why Reduce Dimensions Before Clustering?

Mitigates the Curse of Dimensionality

Restores meaningful distance metrics by reducing sparsity and distance concentration (referencing Part 3 problems).

Removes Noise and Irrelevant Features

Intelligently filters out dimensions that do not contribute to true cluster separation.

Improves Interpretability

Simpler, low-dimensional clusters are vastly easier for humans to analyze and understand.

Example 9.12. Clustering in a derived space. Consider the two clusters of points in Fig. 9.11. It is not possible to cluster these points in any subspace of the original space, $X \times Y$, because both clusters would end up being projected onto overlapping areas in the x and y axes.

What if, instead, we construct a new dimension, $-\frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y$ (shown as a dashed line in the figure)? By projecting the points onto this new dimension, the two clusters become apparent. □

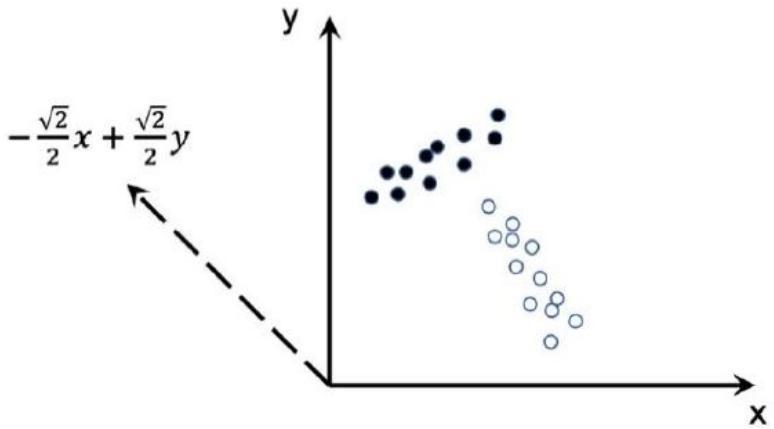


FIGURE 9.11

Clustering in a derived space may be more effective.

Example 9.13. Consider a 3-D data set that has four objects:

$$\mathbf{X} = \begin{bmatrix} 5 & 9 & 3 \\ 4 & 10 & 6 \\ 3 & 8 & 11 \\ 6 & 3 & 7 \end{bmatrix}.$$

After normalization, the matrix is

$$\mathbf{X} = \begin{bmatrix} 0.387 & 0.482 & -1.135 \\ -0.387 & 0.804 & -0.227 \\ -1.162 & 0.161 & 1.286 \\ 1.162 & -1.447 & 0.076 \end{bmatrix}.$$

The covariance matrix is

$$\mathbf{C_X} = \begin{bmatrix} 0.750 & -0.411 & -0.439 \\ -0.411 & 0.549 & -0.152 \\ -0.439 & -0.152 & 0.750 \end{bmatrix}.$$

The three eigenvectors are $[-1.339, 0.567, 1]^T$ with eigenvalue 1.252, $[0.282, -1.098, 1]^T$ with eigenvalue 0.793, and $[1.270, 1.237, 1]^T$ with eigenvalue 0.004. In other words, the transformation matrix is

$$\mathbf{P} = \begin{bmatrix} -1.339 & 0.282 & 1.270 \\ -0.567 & -1.098 & 1.237 \\ 1 & 1 & 1 \end{bmatrix}.$$

Correspondingly, $\mathbf{X}$ is transformed to

$$\mathbf{Y} = \mathbf{XP} = \begin{bmatrix} -8.798 & -5.472 & 20.483 \\ -5.026 & -3.852 & 23.45 \\ 2.447 & 3.062 & 24.706 \\ -2.735 & 5.398 & 18.331 \end{bmatrix}.$$

Since the last eigenvalue 0.004 is very small comparing to the first two eigenvalues, we can reduce the dimensionality from 3 to 2 by dropping the last dimension in $\mathbf{Y}$. That is, the data set can be represented in the 2-D space using the first two eigenvectors as the basis. The representation is

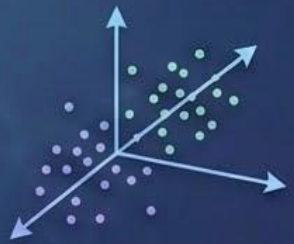
$$\begin{bmatrix} -8.798 & -5.472 \\ -5.026 & -3.852 \\ 2.447 & 3.062 \\ -2.735 & 5.398 \end{bmatrix}.$$

□

Principal Component Analysis (PCA) for Clustering

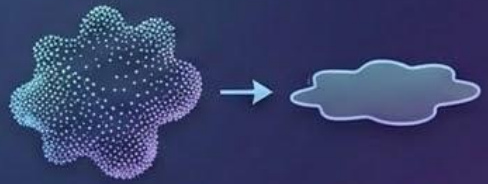
Maximizing Variance to Retain Global Structure

PCA is the most widely utilized linear dimensionality reduction method for clustering tasks.



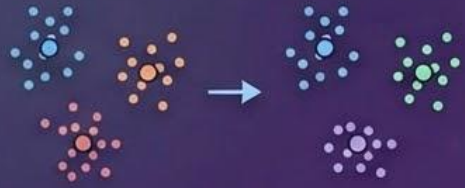
The Core Concept

PCA projects the complex, high-dimensional data onto a new set of orthogonal directions—called Principal Components (PCs)—that are ordered by the amount of total variance they explain.



Global Structure Preservation

The algorithm prioritizes retaining the overall statistical shape and global structure of the dataset.



Impact on k-means

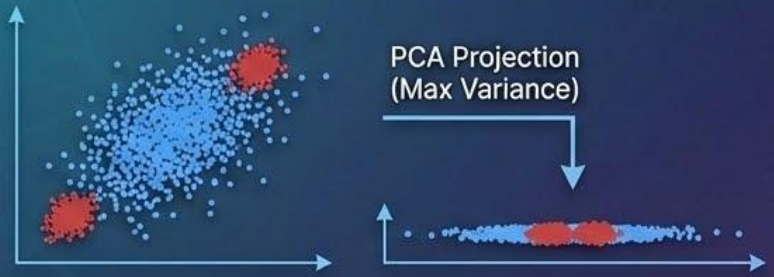
In practice, PCA frequently improves the performance and speed of k-means clustering, especially when a small, high-variance subset of PCs is retained.

Limitations of PCA & Linear Methods

The Variance Blind Spot and Geometric Rigidness

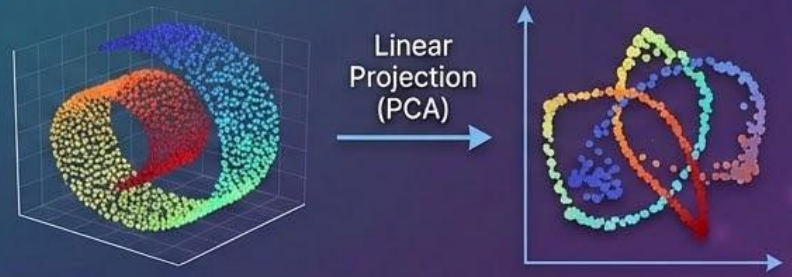
While powerful, PCA is not guaranteed to improve clustering results because its mathematical goal is not aligned with cluster discovery.

PCA May Obscure Cluster Structure



PCA prioritizes total dataset variance, not cluster separation. If true clusters are aligned with low-variance directions (e.g., small, dense groups spread over a large background variation), PCA might mathematically collapse them, hiding the signal in the process.

Non-Linear Failure



Linear dimensionality reduction methods, like PCA, apply strict, rigid geometric transformations. They completely fail when clusters possess complex, non-linear geometric structures (e.g., highly curved manifolds like the “swiss roll” or concentric circles).

Alternatives & Best Practice: A Roadmap

Choosing the Right Approach

If linear variance maximization is not optimal, consider these linear alternatives:

Alternatives for Specific Scenarios

LDA (Linear Discriminant Analysis)
For supervised or semi-supervised scenarios (maximizing class separation).

CCA (Canonical Correlation Analysis)
For multi-view data (finding common structure across different feature sets).

Factor Analysis
Provides a robust probabilistic interpretation of hidden factors.

Best Practice Checklist

Iterative Evaluation
Always evaluate and compare clustering quality strictly before and after reduction. Do not assume reduction improves the model.

Match Complexity
Consciously consider non-linear dimensionality reduction methods (like t-SNE, Isomap, or Autoencoders) If the data geometry is complex.

The Human Element
Use rigorous domain knowledge to actively guide feature selection. Often, selecting the right raw features is vastly superior to automated linear reduction.