Lecture 11 - **Dimensionality Reduction and Clustering**

BIOENG-210 Course Notes
Prof. Gioele La Manno

May 2025

Contents

1 Using PCA to Understand High-Dimensional Data

1.1 PCA as Dimensionality Reduction

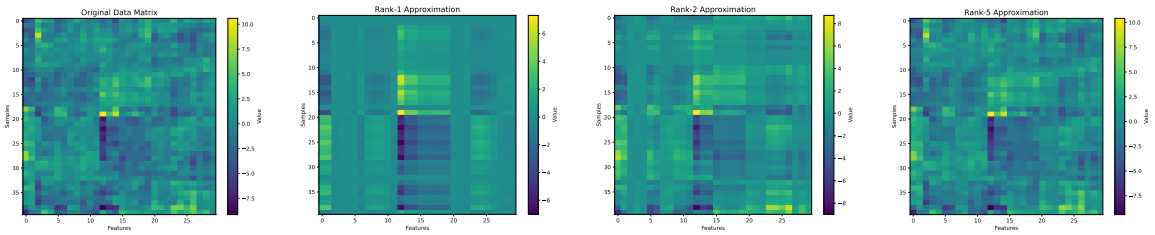
One of the primary applications of PCA is dimensionality reduction - representing high-dimensional data in a lower-dimensional space while preserving as much variance as possible.

Given our SVD decomposition $\mathbf{X}_c = \mathbf{U}\mathbf{S}\mathbf{V}^T$, we can project our data onto the first k principal components:

$$\mathbf{X}_c^{(k)} = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^T$$

where $\mathbf{U}_k$, $\mathbf{S}_k$, and $\mathbf{V}_k$ contain only the first k columns/entries of the original matrices.

This gives us a rank-k approximation of our original data. The remarkable Eckart-Young theorem guarantees that this is the best rank-k approximation in terms of minimizing the squared Frobenius norm $\|\mathbf{X}_c - \mathbf{X}_c^{(k)}\|_F^2$.



(a) Original data matrix (full rank) (b) Rank-1 approximation of the data matrix (c) Rank-2 approximation of the data matrix (d) Rank-5 approximation of the data matrix

Figure 1: **Rank-k approximations of the data matrix**

The proportion of variance explained by the first k principal components is:

$$\text{Proportion of variance explained} = \frac{\sum_{i=1}^{k}\sigma_i^2}{\sum_{i=1}^{\min(n,p)}\sigma_i^2} = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{p}\lambda_i}$$

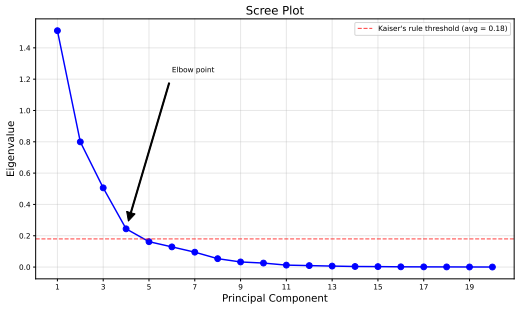
This provides a quantitative measure of how much information we retain when reducing dimensionality.

1.1.1 The Scree Plot: Deciding How Many Components to Keep

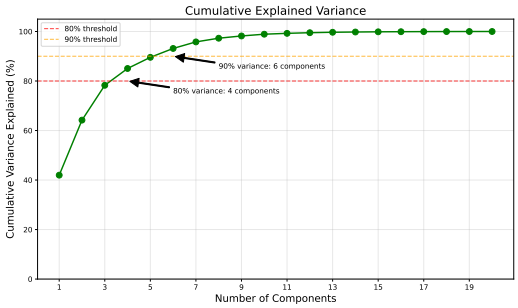
A common visualization to help decide how many principal components to retain is the scree plot, which displays the eigenvalues (or singular values squared) in descending order. Where "scree" is used as an analogy to the debris at the base of a slope.

Several heuristics can guide this decision:

- The "elbow method": Look for a point where the decrease in eigenvalues levels off
- Retain components that explain a significant proportion of variance (e.g., 80% or 90%)
- Kaiser's rule: Keep components with eigenvalues greater than the average eigenvalue
- Use a permutation test or cross-validation to determine significant components



(a) Scree plot showing eigenvalues of a covariance matrix



(b) Cumulative proportion of variance explained

Figure 2: **Using scree plots and cumulative variance to determine dimensionality**

1.2 The Shape of Your Data: Geometric Intuition

One powerful aspect of PCA is that it provides geometric intuition about our data's structure. By examining the pattern of eigenvalues, we can characterize the overall "shape" of our data cloud.

- **Spherical data**: All eigenvalues are approximately equal. The data resembles a sphere or ball in high-dimensional space, with no preferred directions of variation. Commonly seen when variables are uncorrelated and have similar variances.
- **Disk-shaped data**: Two dominant eigenvalues with similar magnitude, and the rest much smaller. The data resembles a disk, concentrated primarily in a plane. This pattern often appears in data with two underlying factors.
- **Cigarette-shaped data**: One dominant eigenvalue much larger than the others. The data is elongated along a single direction, like a cigarette. This suggests a single dominant factor driving variation.
- **Rugby ball-shaped data**: A smoothly decreasing sequence of eigenvalues with no sharp drops. The data resembles a rugby ball - elongated but with substantial width in secondary dimensions. This pattern indicates complex, multi-factorial variation.

These geometric characterizations are informal but show how PCA provide us a "vocabulary" discussing high-dimensional data, making the abstract concrete. Maybe we will not use "these data form a cigarette," in a scientific paper, we immediately convey that variation is primarily along a single dimension that can allow to reason and inform choices for its further analysis and interpretation.

1.3 Interpreting Principal Components

While PCA provides a powerful tool for visualization and dimensionality reduction, interpreting the biological meaning of principal components requires careful analysis. Principal components represent "latent factors" that explain patterns of variation in the data, essentially capturing combinations of correlated variables.

Each principal component is a linear combination of the original variables:

$$PC_i = \sum_{j=1}^{p} v_{ij} X_j$$

Where v_{ij} are the loadings (elements of eigenvector $\mathbf{v}_i$) and X_j are the original variables. The challenge in interpretation comes from these loadings having both positive and negative values, and

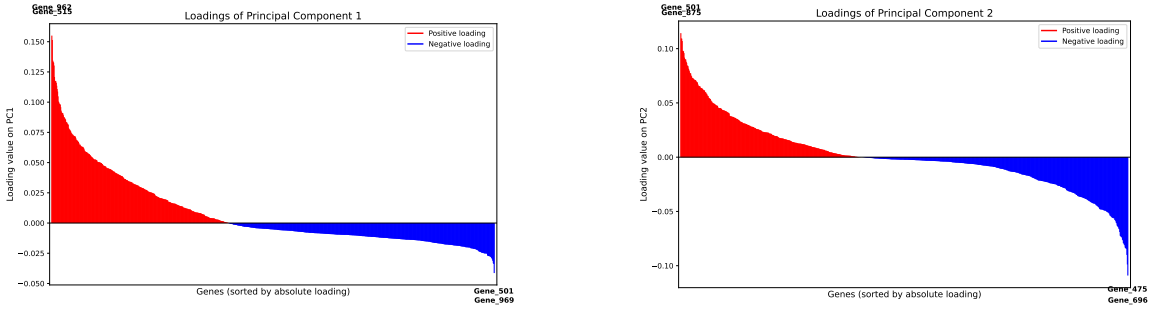
observations having both positive and negative coordinates along each PC. This means a principal component doesn't simply represent the presence or absence of a biological process, but rather contrasting patterns of gene expression.

For example, a component might separate cells with high expression of proliferation genes and low expression of differentiation genes from cells with the opposite pattern. This represents a spectrum rather than a discrete biological process, making interpretation more nuanced but also more informative when done carefully.

Several approaches can help with interpretation:

1.3.1 Loading Analysis

Examining the loadings (coefficients in the eigenvectors) can reveal which genes contribute most to each principal component. A particularly effective visualization is to sort the loadings by magnitude and display them as vertical lines.



(a) Loadings of PC1

(b) Loadings of PC2

Figure 3: **Loadings of the first two principal components**

In this visualization, each vertical line represents a gene's loading value on either PC1 or PC2. Positive loadings (shown in red) indicate genes whose increased expression moves cells in the positive direction along that principal component, while negative loadings (shown in blue) correspond to genes that push cells in the negative direction.

The sorting reveals the most influential genes at both extremes. For instance, the genes with the largest positive loadings on PC1 might be related to proliferation, while those with large negative loadings might be involved in differentiation. This pattern would suggest that PC1 captures a cell state transition from proliferation to differentiation.

The distribution shape of the sorted loadings also provides insight into the data structure. A steep curve with few genes having large loadings suggests that the component is driven by a small, specific gene module. In contrast, a gradual slope indicates broader, systemic variation across many genes.

By comparing loading patterns between PC1 and PC2, we can identify genes that contribute uniquely to each component, highlighting different biological processes captured by the principal components.

1.3.2 Correlation with Known Variables

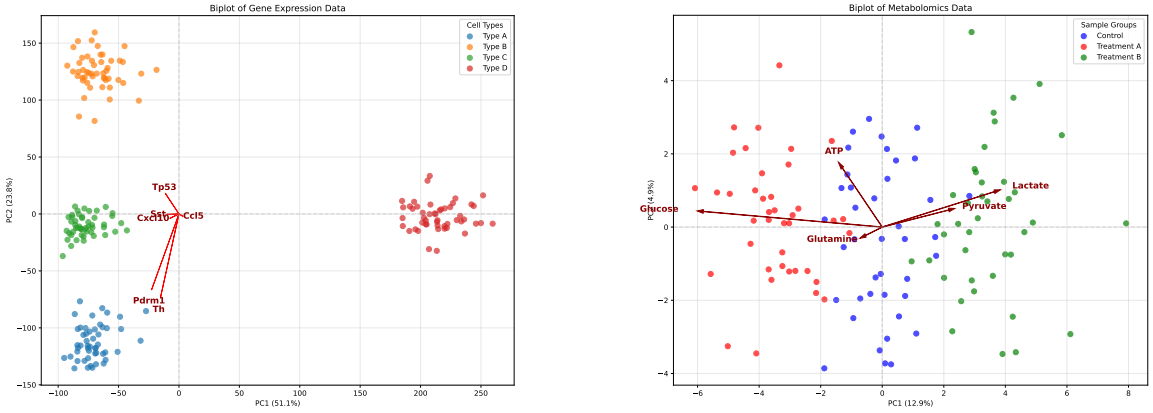
Computing correlations between principal component scores and known variables can reveal associations that aid interpretation.

$$\text{Correlation}(PC_k, \text{Variable}_j) = \text{cor}(U_{\cdot k} \cdot \sigma_k, X_{\cdot j})$$

For instance, in a patient dataset, we might find that PC2 strongly correlates with patient age, suggesting that age-related variation is a secondary factor in our data.

1.3.3 Biplots: Simultaneous Visualization of Observations and Variables

Biplots provide a powerful visualization that displays both observations and variables in the same principal component space, revealing their relationships.



(a) Biplot of gene expression data

(b) Biplot of metabolomics data

Figure 4: **Biplots of gene expression and metabolomics data**

In a biplot:

- Points represent observations (e.g., cells)
- Arrows represent variables (e.g., genes), with their direction and length indicating the loadings on the displayed principal components

To create a biplot, we project both observations and variables onto the same reduced-dimensional space defined by the principal components. Here's how a biplot is constructed:

- For observations (cells/samples):
  - We plot the principal component scores, which are the projections of observations onto the principal components
  - These are the elements of the matrix $\mathbf{U} \cdot \mathbf{S}$ (or equivalently, $\mathbf{X} \cdot \mathbf{V}$)
  - Each point represents an observation in the reduced PC space
- For variables (genes/features):
  - We represent variables as arrows (vectors) from the origin
  - The coordinates of each arrow are determined by the loadings (elements of $\mathbf{V}$)
  - The length and direction of each arrow indicate how strongly and in which direction that variable contributes to the principal components shown
- Interpretation:
  - Observations positioned in the direction of a particular variable arrow tend to have high values for that variable
  - The length of variable arrows indicates how well that variable is represented in the displayed principal components

The joint display of observations and variables in the same plot allows us to directly visualize which variables are driving the separation between observations, creating a powerful tool for exploratory data analysis.

This allows us to see which variables are responsible for the separation of observations along different axes.

1.4 Practical Considerations and Limitations

Despite its power and elegance, PCA has several limitations worth noting:

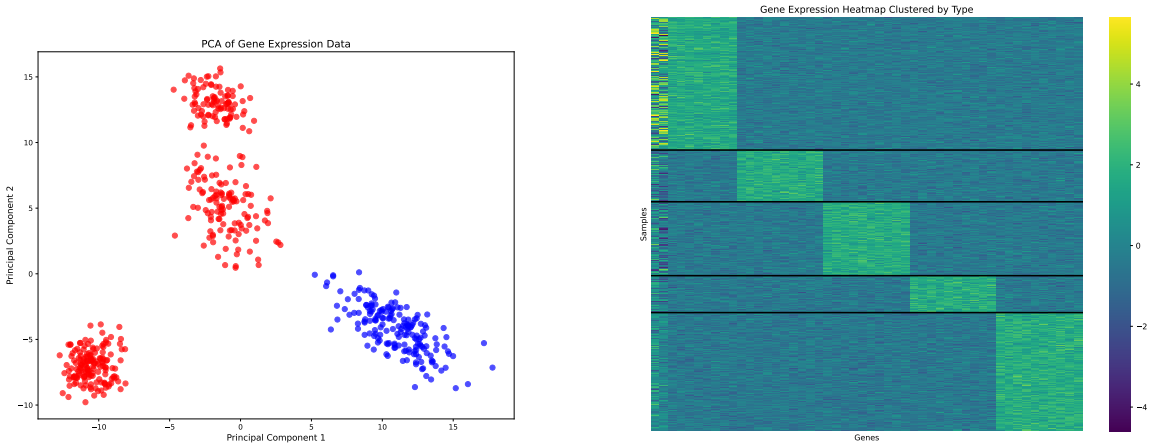
- **Sensitivity to scaling**: PCA results depend crucially on the scale of variables. Variables with larger scales will tend to dominate the first few principal components, regardless of their importance. Standardization (scaling to unit variance) is often used to address this issue.
- **Sensitivity to outliers**: As a least-squares method, PCA can be heavily influenced by outliers. Robust variants of PCA exist to address this issue.
- **Difficulty of interpretation**: Principal components are linear combinations of all original variables, which can make them difficult to interpret, especially in biological contexts.

2 Clustering

2.1 Proposing a discrete model of the data

When we apply PCA to biological data, we often observe that individual data points—whether they be cells, tissue samples, or experimental conditions—naturally organize into distinct groups in the reduced dimensional space. This clustering pattern suggests underlying biological structure that deserves closer investigation.

Consider a gene expression study where we measure thousands of genes across hundreds of patient samples. After applying PCA, we might notice that tumor samples from different patients with the same cancer subtype cluster together in the PC space. These groups were not defined a priori; they are not like the classes of a classification task, nor covariates we had access before analysis, they emerged naturally from the data's intrinsic structure.



(a) Cancer gene expression data

(b) Heatmap of the same data

Figure 5: **Clustering patterns emerging from unsupervised exploration of the data**

This observation motivates us to ask: Can we formalize the process of identifying these groups? Can we develop algorithms that objectively partition our data into meaningful clusters? This is precisely the goal of clustering analysis.

Unlike supervised learning methods where we predict known categories, clustering is an unsupervised approach that discovers structure in the data without predefined labels. In biology, this makes clustering particularly valuable for:

- Discovering novel cell types in single-cell data

- Identifying patient subtypes with different disease prognoses
- Finding functional modules in gene regulatory networks
- Classifying metabolic states from metabolomics data

Clustering describes variation in a different way than PCA, which focuses of continuous axes of variation. Clustering identifies discrete groups—from continuously distributed data to categorical partitions.

2.2 Defining the Clustering Problem

2.2.1 Formal Definition of Clustering

At its core, clustering is an optimization problem. Given a set of observations $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, clustering seeks to find a partition $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ that minimizes some objective function:

Definition 2.1 (Clustering as Optimization)**.** A clustering is a partition $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ of dataset $\mathcal{X}$ that minimizes an objective function $J(\mathcal{C})$:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} J(\mathcal{C}) \tag{1}$$

subject to:

$$C_i \cap C_j = \emptyset \quad \text{for all } i \neq j \quad \text{(clusters don't overlap)} \tag{2}$$

$$\bigcup_{i=1}^{k} C_i = \mathcal{X} \quad \text{(all points are assigned)} \tag{3}$$

This definition aligns with standard machine learning formulations where clustering is presented as finding the partition that optimizes an objective measuring cluster quality.

2.2.2 Clustering Objectives

The choice of objective function $J(\mathcal{C})$ defines the type of clustering we seek. Most objective functions balance two competing goals:

- **Intra-cluster similarity**: Observations within the same cluster should be similar
- **Inter-cluster dissimilarity**: Observations in different clusters should be dissimilar

This forces us to revise / define a bit better notion of similarity and dissimilarity (distance) between observations. In general a similarity can be defined as the inverse of a distance.

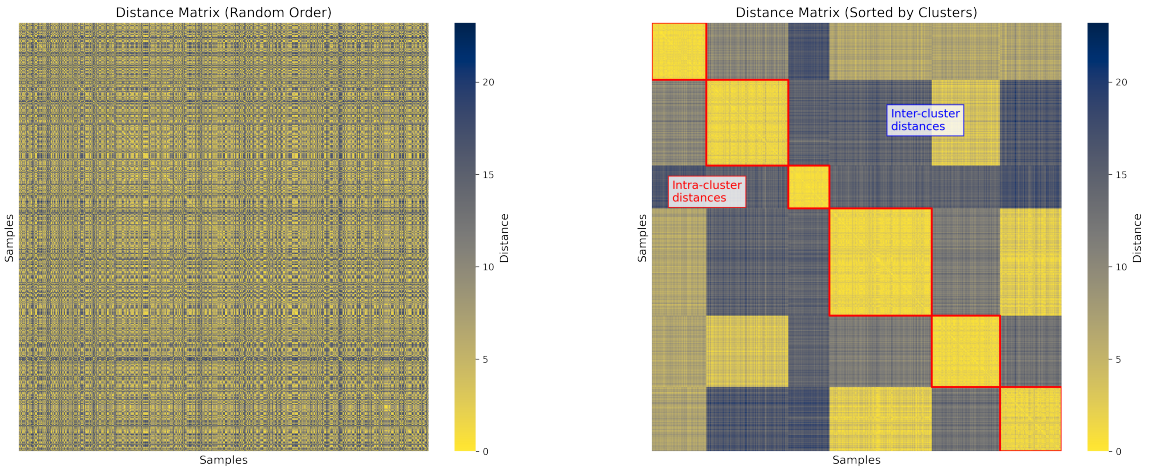
Some of the most important distance and similarity measures in statistical learning are:

2.2.3 Distance and Similarity Measures

Distance Metrics:

- **Euclidean distance**: Measures the straight-line distance between two points in Euclidean space (L2 norm)

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$



(a) Distance matrix of the data randomly ordered

(b) Distance matrix of the data sorted by clusters

Figure 6: **Distance matrices showing intra- and inter-cluster distances**

- **Manhattan distance**: Measures the distance along axes at right angles (L1 norm)

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{5}$$

- **Hamming distance**: Measures the number of positions at which two strings of equal length differ, often used for categorical data

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{1}_{x_i \neq y_i} \tag{6}$$

- **Mahalanobis distance**: Measures the distance between a point and a distribution, accounting for correlations between variables

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})} \tag{7}$$

Similarity Measures:

- **Pearson correlation**: Measures linear correlation between variables; similarity is high when points follow the same pattern regardless of scale

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{8}$$

- **Spearman correlation**: Rank-based correlation that captures monotonic relationships without requiring linearity

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(r(x_i) - \overline{r(x)})(r(y_i) - \overline{r(y)})}{\sqrt{\sum_{i=1}^{n}(r(x_i) - \overline{r(x)})^2 \sum_{i=1}^{n}(r(y_i) - \overline{r(y)})^2}} \tag{9}$$

- **Cosine similarity**: Measures the cosine of the angle between two vectors, often used for high-dimensional sparse data

$$s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{10}$$

These principles can be formalized in different ways:

Example 2.2 (Within-Cluster Sum of Squares)**.** The k-means algorithm minimizes:

$$J_{\text{WCSS}}(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{11}$$

where $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$ is the centroid of cluster C_i.

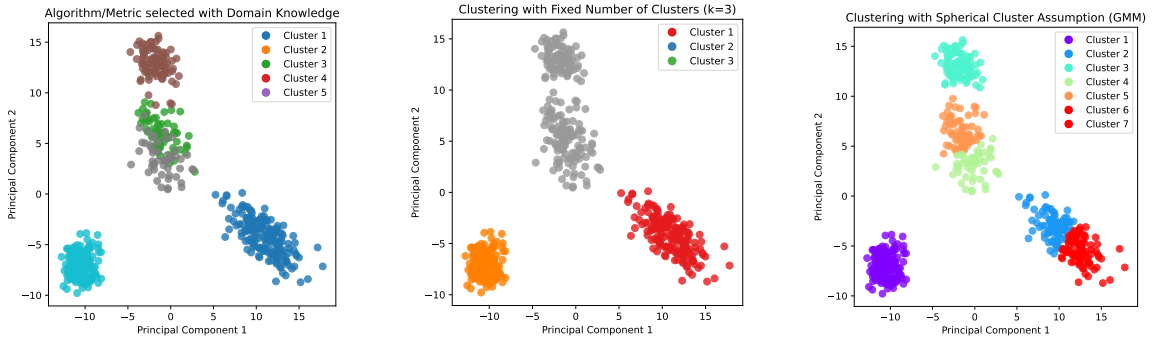
Example 2.3 (General Formulation)**.** A more general formulation optimizes:

$$J(\mathcal{C}) = \alpha \cdot \sum_{i=1}^{k} \text{within}(C_i) - \beta \cdot \sum_{i \neq j} \text{between}(C_i, C_j) \tag{12}$$

where $\text{within}(C_i)$ measures the compactness of cluster C_i, $\text{between}(C_i, C_j)$ measures the separation between clusters C_i and C_j, and α, β are weighting parameters.

In practice, different clustering algorithms correspond to different choices of objective function, distance metric, and optimization approach. The solution to the clustering problem is typically NP-hard, (meaning that it is computationally infeasible to find the exact solution in polynomial time, as it would take exponential time to check all possible partitions), leading to approximation algorithms and heuristic methods that find locally optimal solutions.

It is important to understand that there is no universally optimal clustering definition. Different applications require different similarity measures and cluster structures, leading to various algorithms that optimize different objectives.



(a) Good clustering based on domain knowledge

(b) Clustering method that sets cluster number to 3

(c) Clustering method that sets clusters to be spherical

Figure 7: **The same data can yield different clusterings depending on algorithm and parameters**

This inherent ambiguity underscores a fundamental truth about clustering: there is rarely a single "correct" answer. Different algorithms and parameter choices can yield valid but distinct partitions of the same data. In biological applications, this means that clustering results must always be interpreted with domain knowledge and validated through additional experiments.

2.3 Two Common Categories of Clustering

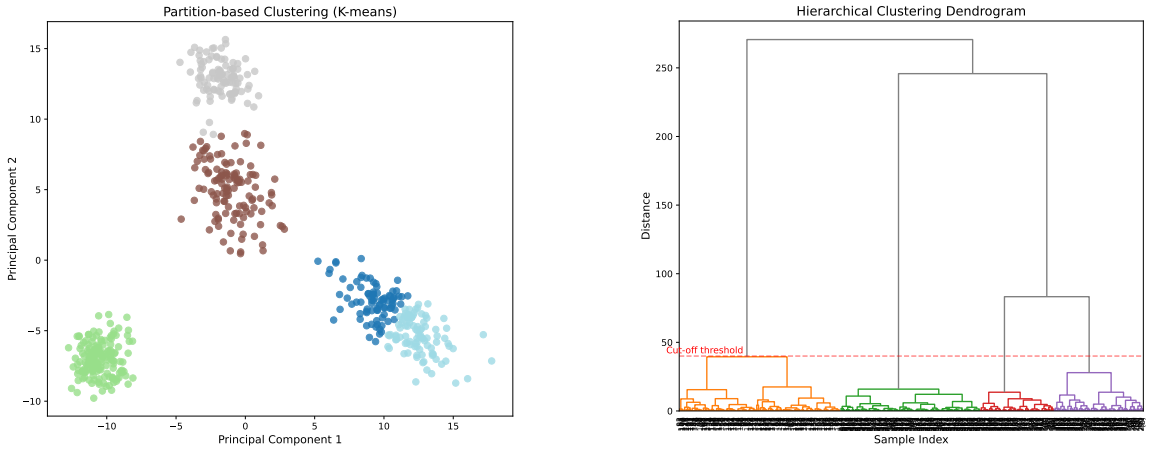
We focus on two major clustering approaches with distinct philosophies: partition-based and hierarchical clustering.

2.3.1 Partition-based Clustering

Partition-based methods divide data into a predetermined number of clusters with each observation assigned to exactly one cluster. K-means, the most popular algorithm in this category, requires specifying the number of clusters in advance, assumes roughly spherical clusters of similar size, and iteratively optimizes cluster assignments. These methods work well when the expected number of groups is known, clusters are compact and well-separated, and computational efficiency matters.

2.3.2 Hierarchical Agglomerative Clustering

Hierarchical agglomerative methods create a nested sequence of partitions, revealing relationships between clusters at different scales through a dendrogram. These methods follow a bottom-up approach, starting with individual points as singleton clusters and progressively merging the most similar pairs. This approach is valuable when the natural number of clusters is unclear, relationships between groups are important, or when data exhibits inherently hierarchical structure.



(a) Partition-based clustering

(b) Hierarchical clustering

Figure 8: **Comparison of partition-based and hierarchical clustering approaches**

The choice between these approaches depends on the biological question, data characteristics, and computational resources. Researchers often apply both as complementary analyses to gain different perspectives on their data's structure.

2.4 K-Means Clustering: A Foundational Algorithm

K-means clustering represents one of the most widely used partition-based methods in biological data analysis. Its popularity stems from its conceptual simplicity, computational efficiency, and effectiveness in many practical applications.

The algorithm aims to partition n observations into k clusters by minimizing the within-cluster sum of squares (WCSS):

$$J(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{13}$$

where $\boldsymbol{\mu}_i$ is the centroid (mean) of cluster C_i. This objective function ensures that points within each cluster are as close as possible to their cluster's center.

Consider a gene expression dataset where we seek to identify distinct cellular states. Each sample (cell or tissue) is represented as a point in a high-dimensional space, where each dimension corresponds to the expression level of a specific gene. K-means will partition these points to minimize the overall gene expression variability within each group.

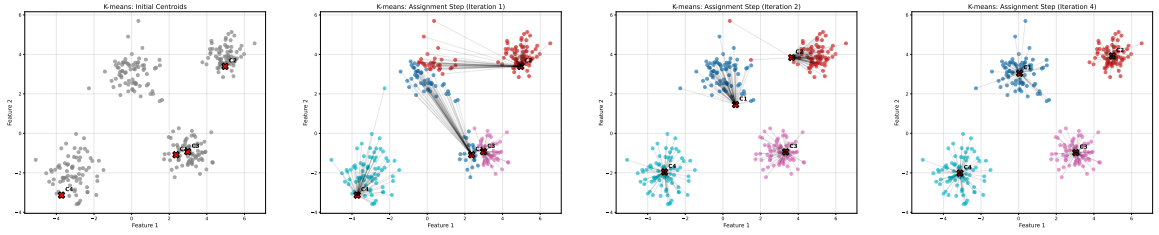
2.5 The K-Means Algorithm

The k-means algorithm follows an iterative procedure that alternates between two key steps:

1. **Assignment Step**: Assign each observation to the nearest cluster centroid
2. **Update Step**: Recalculate cluster centroids as the mean of assigned points

Algorithm 1 K-Means Clustering

Input: Dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, number of clusters k
Output: Clusters $C_1, C_2, ..., C_k$ and centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k$
Initialize centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k$ (e.g., randomly)
repeat
    // Assignment step
    for each observation $\mathbf{x}_j \in \mathcal{X}$ **do**
        Assign $\mathbf{x}_j$ to cluster C_i where $i = \arg\min_{i \in \{1,...,k\}} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$
    end for
    // Update step
    for each cluster $i \in \{1, ..., k\}$ **do**
        Update centroid $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$
    end for
until centroids no longer change significantly
return Cluster assignments $C_1, C_2, ..., C_k$ and final centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k$



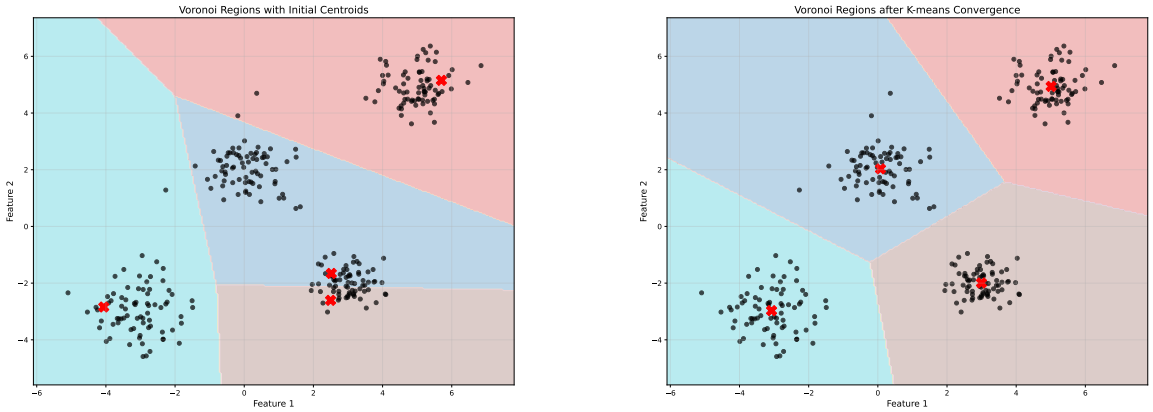
(a) Initial state - random (b) First assignment step (c) Second assignment step (d) Fourth assignment step

Figure 9: **K-means algorithm iteration process**

The algorithm creates Voronoi regions in the feature space, where each region contains all points closer to one centroid than to any other. These regions define natural cluster boundaries that evolve during iteration.

2.6 Properties and Characteristics of K-Means

K-means possesses several important properties that make it both powerful and limited:



(a) Voronoi regions with initial centroids

(b) Voronoi regions after several iterations

Figure 10: **K-means Voronoi regions**

2.6.1 Convergence Properties

Local Optima. K-means converges to a local minimum of the WCSS objective function. However, different initializations may yield different final clusterings. This is why running the algorithm multiple times with different starting points is often recommended in practice.

Computational Complexity. The algorithm has time complexity O(n·k·d) where n is the number of observations, k is the number of clusters, and d is the dimensionality. This linear scaling makes k-means suitable for large datasets, allowing it to handle the high-throughput data common in modern biological research.

2.6.2 Assumptions and Limitations

K-means makes several important assumptions that affect its performance:

- **Spherical clusters**: Works best with roughly spherical clusters of similar size
- **Euclidean distance**: May not be appropriate for all data types
- **Hard assignments**: Each point belongs to exactly one cluster
- **Sensitivity to outliers**: As centroids are means
- **Local optima**: Multiple runs with different initializations recommended

For biological data, consider:

- **Feature scaling**: Normalize expression data appropriately
- **Dimensionality**: PCA pre-processing often helps with high-dimensional data

2.7 The Challenge of Choosing K

Perhaps the most significant challenge in k-means clustering is determining the appropriate number of clusters, k. Unlike supervised learning where class labels guide model selection, clustering requires us to infer the natural grouping structure from the data itself. In biological applications, domain knowledge plays a crucial role: **Prior knowledge**: Known cell types, disease stages, or experimental conditions **Stability**: Select k values that produce stable clusters across multiple runs and subsampling

2.7.1 The Silhouette Score: Evaluating Cluster Quality

The silhouette score provides a quantitative measure of how well each observation fits within its assigned cluster. For each point, it compares the average distance to other points in the same cluster with the average distance to points in the nearest neighboring cluster.

For observation i, the silhouette coefficient is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{14}$$

where:

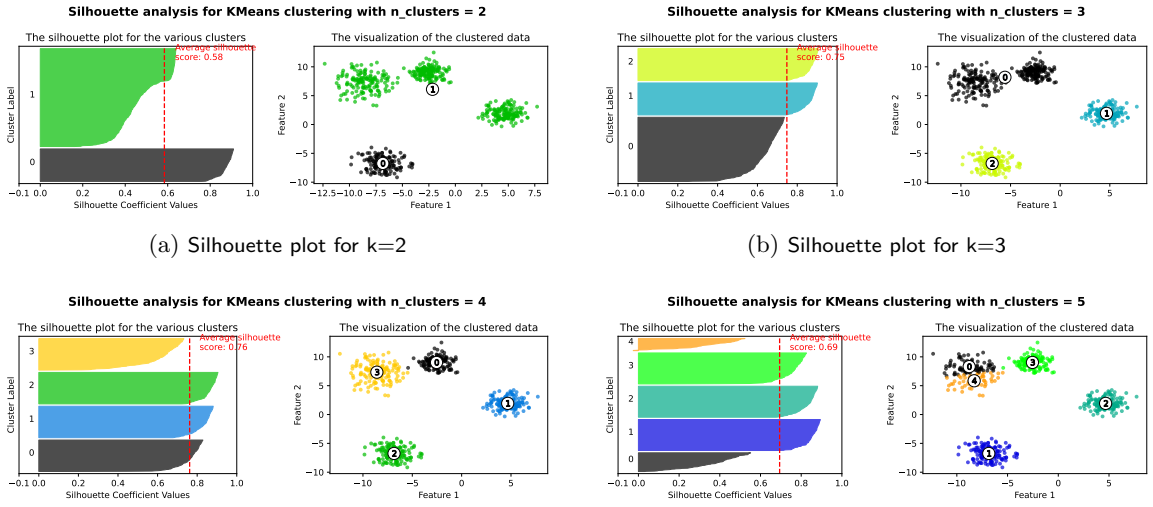
- $a(i)$ is the average distance from point i to all other points in the same cluster
- $b(i)$ is the average distance from point i to all points in the nearest neighboring cluster

The silhouette coefficient ranges from -1 to +1:

- Values near +1 indicate the point is well-matched to its cluster
- Values near 0 indicate the point is on the border between clusters
- Values near -1 indicate the point might be wrongly clustered

When plotted, these scores form bars for each point that visually resemble silhouettes, especially when sorted by cluster and score.

To interpret the silhouette plot shown below, we examine both individual silhouette coefficients and their aggregate patterns. Each vertical line represents an individual data point. Points are grouped by cluster, creating distinct regions in the plot while the red dashed line indicates the average silhouette score across all points.



(a) Silhouette plot for k=2

(b) Silhouette plot for k=3

(c) Silhouette plot for k=4

(d) Silhouette plot for k=5

Figure 11: **Silhouette analysis for cluster quality assessment**

A good clustering displays most points with high silhouette values (close to 1), consistently high values within each cluster, similar widths for different clusters indicating balanced cluster sizes, and few or no negative values which would suggest misclassifications.

When comparing multiple silhouette plots for different k values, one looks for the plot with the highest average silhouette score, consistency within clusters (few points below the average line).

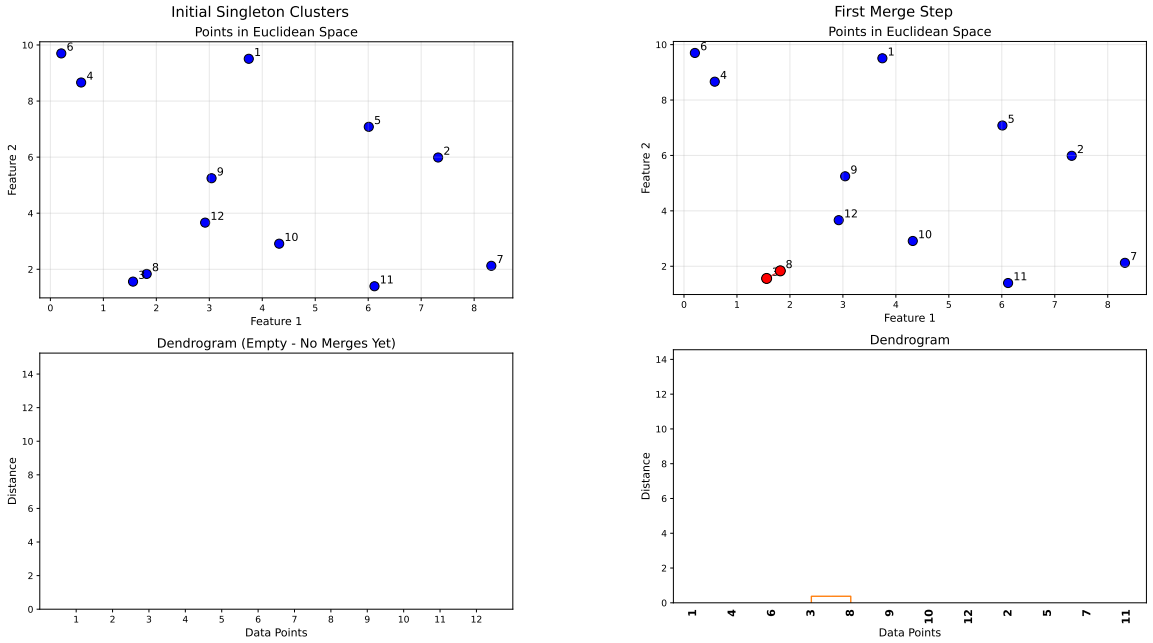
2.8 Agglomerative Hierarchical Clustering: Building Cluster Hierarchies

Agglomerative hierarchical clustering takes a bottom-up approach, starting with each observation as its own cluster and progressively merging the most similar pairs until all observations belong to a single cluster.

The basic procedure follows these steps:

Algorithm 2 Agglomerative Hierarchical Clustering

Input: Dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, distance function $d(\cdot, \cdot)$, linkage method
Output: Hierarchical cluster structure (dendrogram)
Initialize n singleton clusters $C_1 = \{\mathbf{x}_1\}, C_2 = \{\mathbf{x}_2\}, ..., C_n = \{\mathbf{x}_n\}$
Calculate distance matrix D where $D_{ij} = d(C_i, C_j)$ for all $i \neq j$
for $t = 1$ to $n - 1$ **do**
    Find clusters C_i and C_j with minimum distance: $(i, j) = \arg\min_{i,j} D_{ij}$
    Merge C_i and C_j into new cluster C_{new}
    Record merge and distance for dendrogram construction
    Remove rows/columns i and j from distance matrix D
    Calculate distances from C_{new} to all other clusters using specified linkage method
    Add C_{new} to distance matrix D
end for
return Dendrogram representing the hierarchy of merges



(a) Initial singleton clusters

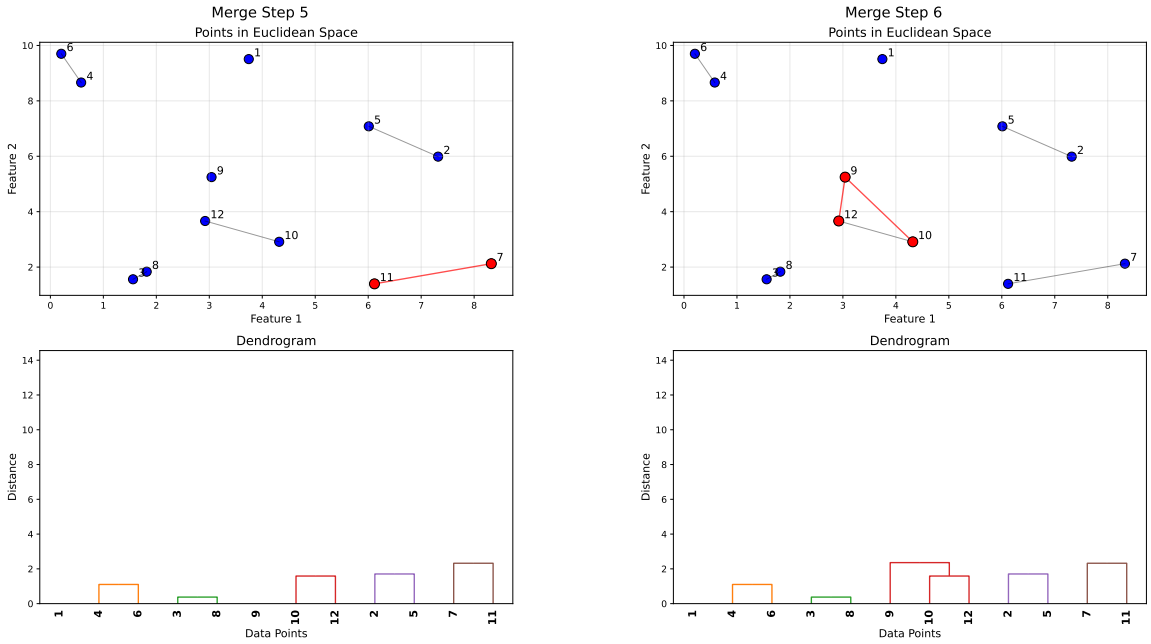
(b) First merge step

Figure 12: **Agglomerative hierarchical clustering process (Part 1)**

2.8.1 Understanding Dendrograms

The dendrogram is a tree-like diagram that records the sequence of merges performed. Each horizontal line represents a merge operation, with the height indicating the distance between the merged clusters.

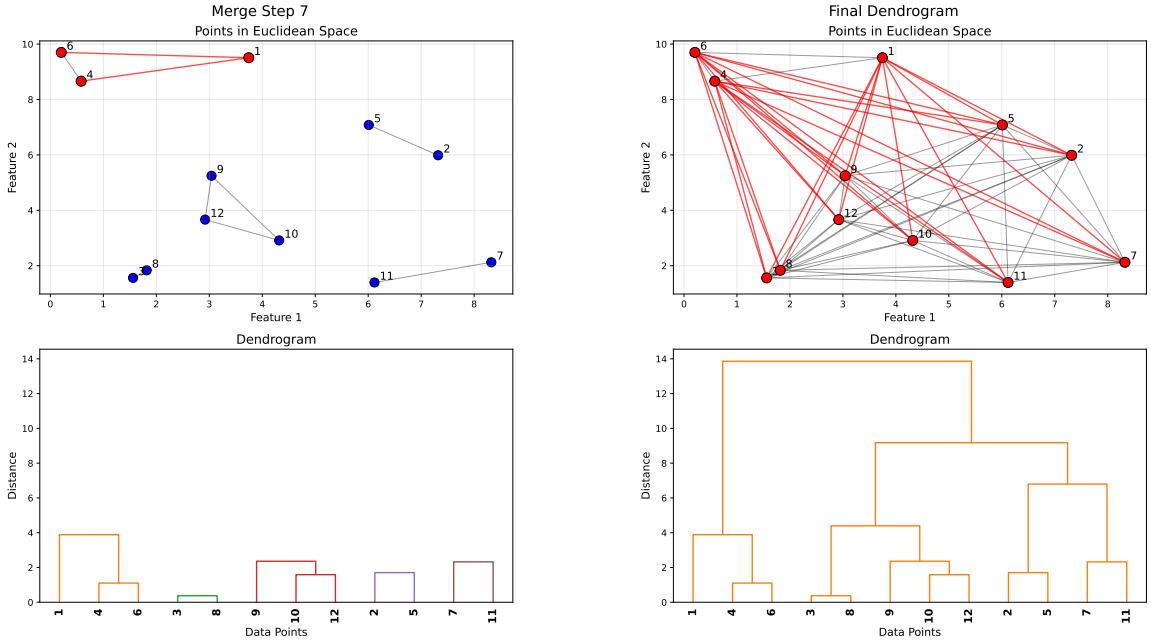
Key features of dendrograms:



(a) Subsequent merge

(b) Subsequent merge

Figure 13: **Agglomerative hierarchical clustering process (Part 2)**

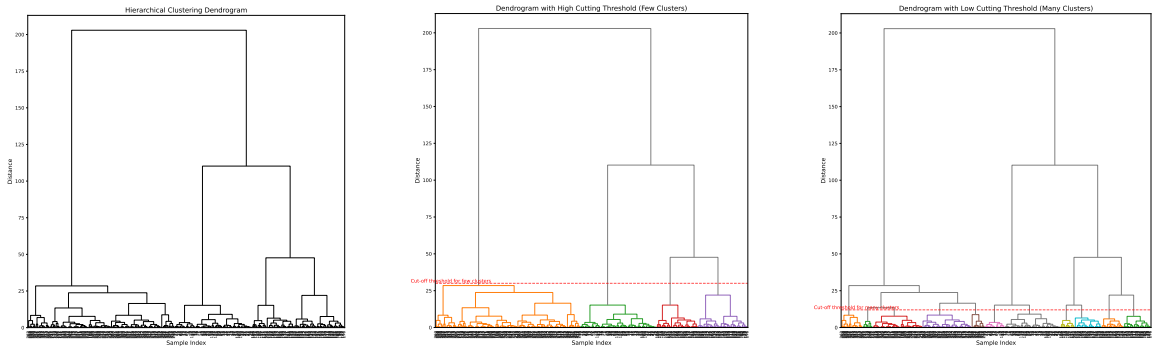


(a) Subsequent merge

(b) Final dendrogram

Figure 14: **Agglomerative hierarchical clustering process (Part 3)**

- **Leaves**: Individual observations at the bottom
- **Nodes**: Points where clusters merge
- **Height**: Distance at which merge occurs
- **Cutting**: Horizontal cuts produce different k-value partitions



(a) Dendrogram black on white (b) Dendrogram high cutoff (c) Dendrogram low cutoff

Figure 15: **Interpreting dendrograms and cluster cutting**

2.8.2 Linkage Methods: Defining Inter-Cluster Distance

A critical decision in hierarchical clustering is how to measure distance between clusters (linkage method). Different linkage methods can produce dramatically different clustering results:

- **Single Linkage**: Distance between nearest points

$$d(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \tag{15}$$

- **Complete Linkage**: Distance between farthest points

$$d(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \tag{16}$$

- **Average Linkage**: Average distance between all point pairs

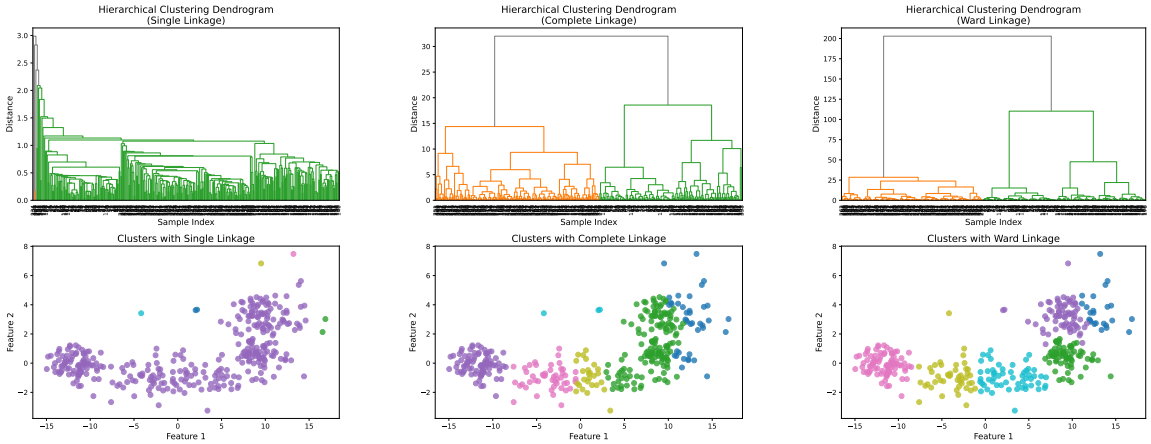
$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \tag{17}$$

- **Ward's Method**: the update formula can be written recursively as

$$d(C_i \cup C_j, C_k) = \frac{|C_i| + |C_k|}{|C_i| + |C_j| + |C_k|} d(C_i, C_k) + \frac{|C_j| + |C_k|}{|C_i| + |C_j| + |C_k|} d(C_j, C_k) - \frac{|C_k|}{|C_i| + |C_j| + |C_k|} d(C_i, C_j) \tag{18}$$

Each linkage method has distinct characteristics: Single linkage tends to create chain effects and struggles with varying density clusters Complete linkage produces more compact, similarly-sized clusters

In biological applications, Ward's method is often preferred for gene expression data as it produces tight, homogeneous clusters, while complete linkage works well for phylogenetic reconstruction where clear separation between groups is important.



(a) Single linkage clustering (b) Complete linkage clustering (c) Ward's method clustering

Figure 16: **Different linkage methods in hierarchical clustering**