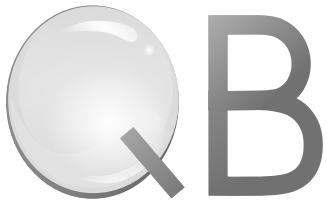


Fundamentals of Quantitative Biology

STATS 04: High Dimensional Data - Clustering

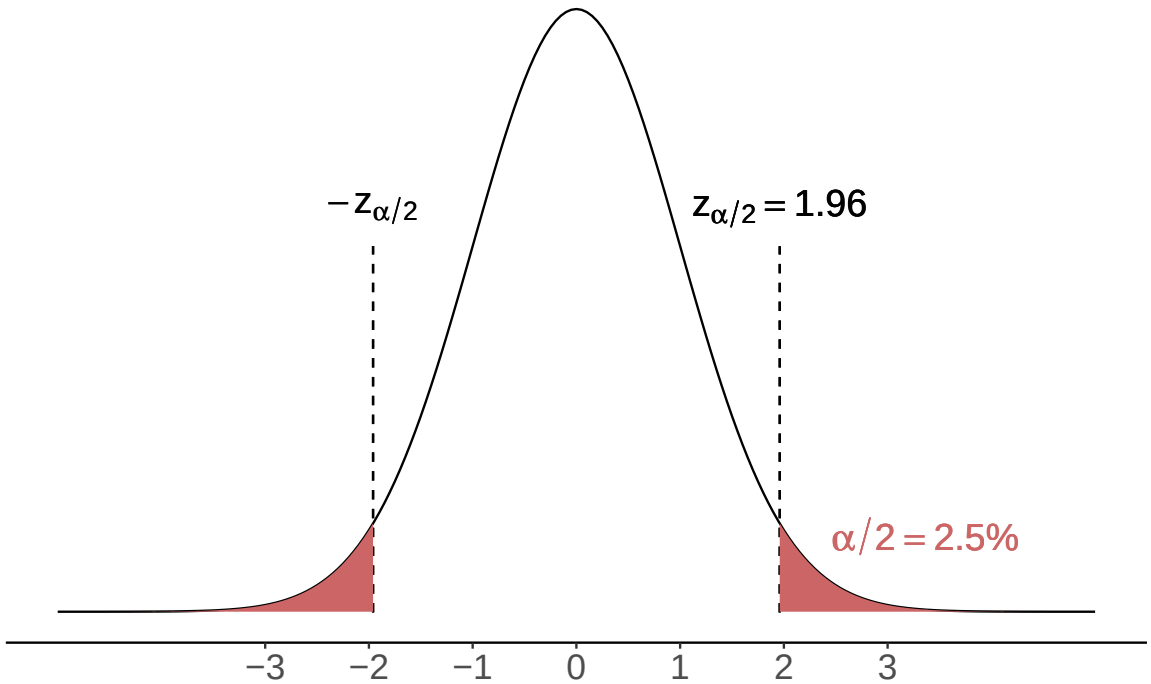
Winter semester 2024/2025

The multiple testing problem

Null Distribution for z-Tests

Model: $X_1, \ldots, X_n \sim \mathcal{N}(\mu, \sigma), \quad H_0 : \mu = \mu_0$

Test Statistic: $Z = \frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}} \sim \mathcal{N}(0,1)$, and $\alpha = 95\%$

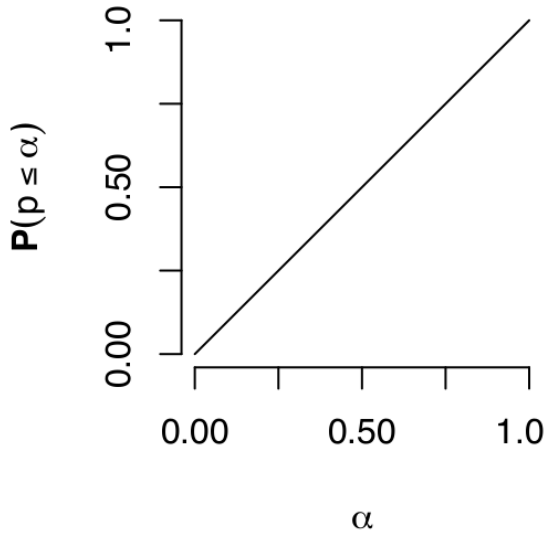


Reject H_0 if z lies within critical area (red)

Multiple Tests

p-value distribution:

If we test a *true* null hypothesis, the *p*-value distribution is uniform



This means that for $n = 1000$ and $\alpha = 0.05$, about $0.05 \times 1000 = 50$ tests will show significant results purely by chance, regardless of the shape of the null-distribution.

Running multiple tests

Example: scRNA-seq expression levels

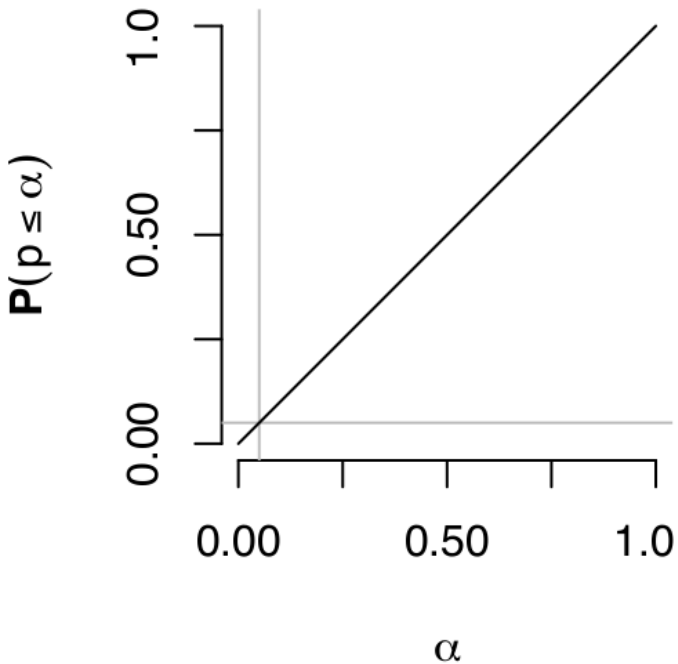
	Control				**Treatment**			
	cell 1	$\cdots$	cell n	**mean**	cell 1	$\cdots$	cell n	**mean**
gene 1	c_{11}	$\cdots$	c_{1n}	$\overline{c_1}$	t_{11}	$\cdots$	t_{1n}	$\overline{t_1}$
$\vdots$			$\ddots$				$\ddots$	
gene m	c_{m1}	$\cdots$	c_{mn}	$\overline{c_m}$	t_{m1}	$\cdots$	t_{mn}	$\overline{t_m}$

Q: Which genes show significant differences in average expression levels?

Running multiple tests

Probability that at least one p-value in m (independent) tests is $\leq \alpha$?

$$\mathbb{P}(p_1 \leq \alpha) = 1 - (1 - \alpha)^1$$

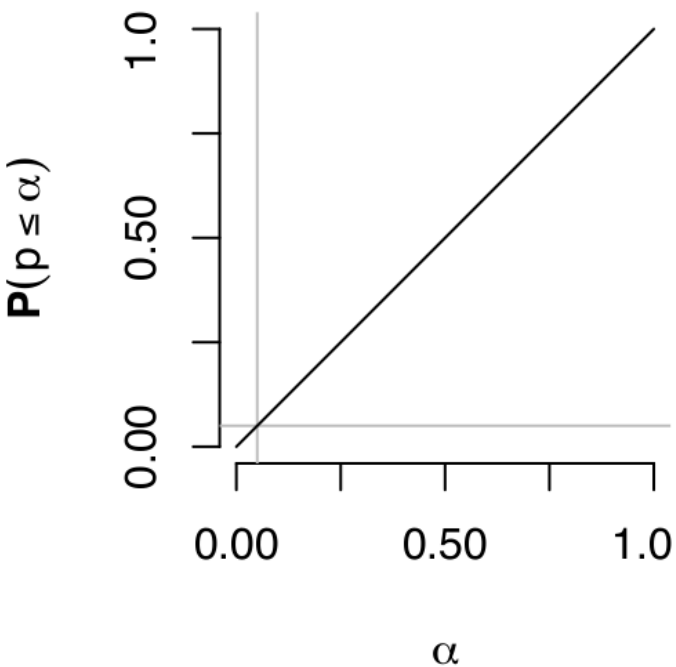


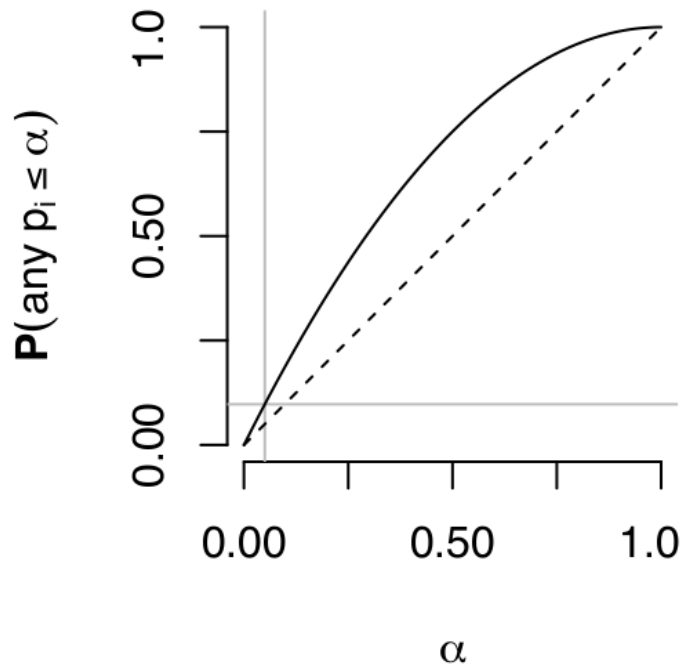
Running multiple tests

Probability that at least one p-value in m (independent) tests is $\leq \alpha$?

$$\mathbb{P}(p_1 \leq \alpha) = 1 - (1 - \alpha)^1$$

$$\mathbb{P}(p_1 \leq \alpha \text{ or } p_2 \leq \alpha) = 1 - (1 - \alpha)^2$$

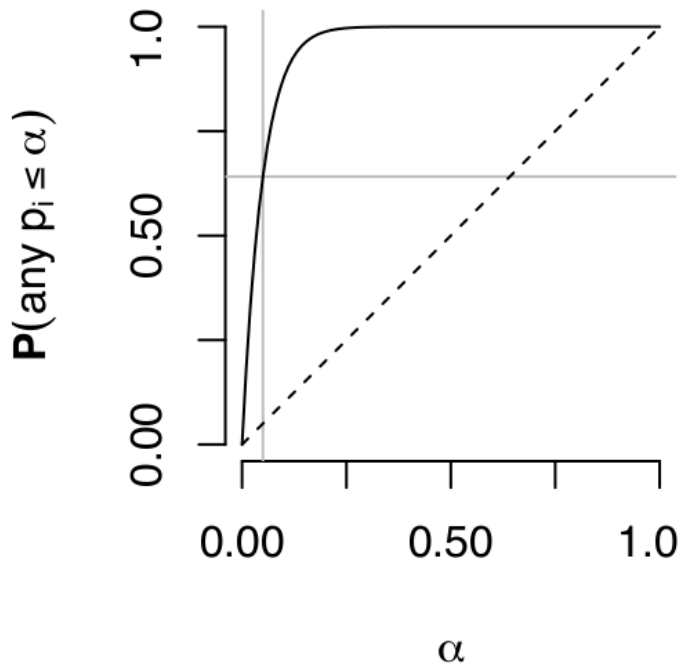




Running multiple tests

Probability that at least one p-value in m (independent) tests is $\leq \alpha$?

$$\mathbb{P}(p_1 \leq \alpha \text{ or } \ldots \text{ or } p_{20} \leq \alpha) = 1 - (1 - \alpha)^{20}$$



Running multiple tests

Probability that at least one p-value in m (independent) tests is $\leq \alpha$?

$$\mathbb{P}(p_1 \leq \alpha \text{ or } \ldots \text{ or } p_{20} \leq \alpha) = 1 - (1 - \alpha)^{20}$$

Example: testing 20 drugs with none of them having an effect. The null-hypothesis is true for all.
For each drug we have a Type I error of $\alpha = 0.05$ which is the probability of incorrectly rejecting H_0.
Considering all 20 drugs the probability of at least one false positive is

$$1 - (1 - \alpha)^{20} = 1 - (1 - 0.95)^{20} \approx 0.64$$

The multiple testing problem

The probability of making at least one Type I error increases with the number of tests!
If we test m hypotheses, we call the probability

$$\mathbb{P}(p_1 \leq \alpha \text{ or } \ldots \text{ or } p_m \leq \alpha)$$

(assuming the hypotheses are true) the **family-wise error rate**. (FWER)

= Probability of making one or more false discovery

- Without correction, FWER $> \alpha$ if $m > 1$

To ensure FWER $\leq \alpha$, we adjust (correct) the individual p-values. (or equivalently, we adjust the individual α's)

Methods to control the FWER

Assume: $p_1 \leq p_2 \leq \cdots \leq p_m$ with $H_0(1), \ldots, H_0(m)$.

Method of **Bonferroni**

- Reject $H_0(i)$ if $p_i < \frac{\alpha}{m}$ for $i = 1, \ldots, m$

Method of **Holm-Bonferroni**

- Sort p-values in ascending order $p_1, p_2, ..., p_m$ and compare them to $p_k < \frac{\alpha}{m-k+1} \Rightarrow$ Reject H_0 for the smallest p-values until one fails the condition.

Method of **Benjamini-Hochberg**

- Sort p-values in ascending order $p_1, p_2, ..., p_m$. Find the largest p-value such that $p_k \leq \frac{k}{m}\alpha \Rightarrow$ Reject all H_0 for $p_1, p_2, ..., p_k$.

High-dimensional data

From Bivariate to High-Dimensional Data

In classical bivariate statistics, each data point has two features:

$$
\begin{aligned}
\boldsymbol{D}(n, 2) &= \text{Data matrix with two features (columns)} \\
n &= \text{number of samples (rows)} \\
2 &= \text{number of features (e.g., } x \text{ (predictor) and } y \text{ (response))} \\
d(i, 1) &= \text{Value of feature } x \text{ for sample } i \\
d(i, 2) &= \text{Value of feature } y \text{ for sample } i \\
i &= 1, \ldots, n
\end{aligned}
$$

Example: Measuring lung capacity in boys or drug dosage and enzyme activity as a response etc.

High-dimensional data - format

We want to understand large and **high-dimensional** data:

$$\boldsymbol{C}(n, m) = \text{count matrix (of for example scRNA-seq experiment)}$$
$$n = \text{number of rows (cells)}$$
$$m = \text{number of columns (genes)}$$
$$c(i, j) = \text{(read) count for row (cell) } i \text{ and column (gene) } j$$

$$i = 1, \ldots, n \quad j = 1, \ldots, m$$

In the scRNA-seq example $n \gg 1000$ and $m \gg 10000$

High-dimensional data - example

Example: scRNA-seq count matrix

C	gene 1	gene 2	$\cdots$	gene m
cell 1				
cell 2		$\vdots$		
$\vdots$				
cell i	$c(i,1)$	$c(i,2)$	$\cdots$	$c(i,m)$
$\vdots$				
cell n		$\vdots$		

High-dimensional data - notation

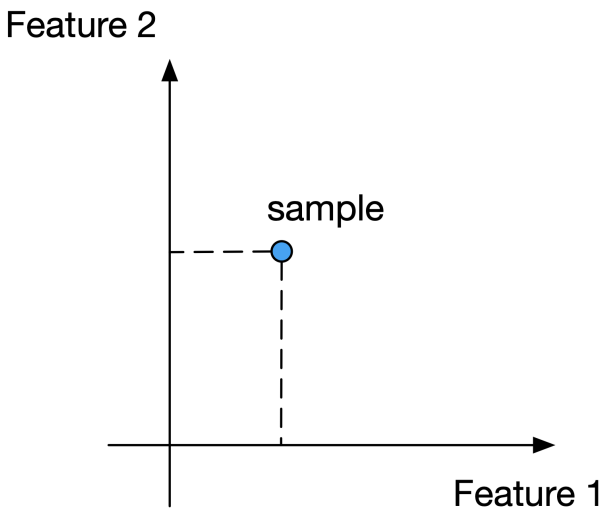
C	feature 1	feature 2	$\cdots$	feature m
sample 1				
sample 2		$\vdots$		
$\vdots$				
sample i	$c(i,1)$	$c(i,2)$	$\cdots$	$c(i,m)$
$\vdots$				
sample n		$\vdots$		

Each row is a vector in m-dimensional space
$\Rightarrow$ m-dimensional space is difficult to visualize and hard to imagine.

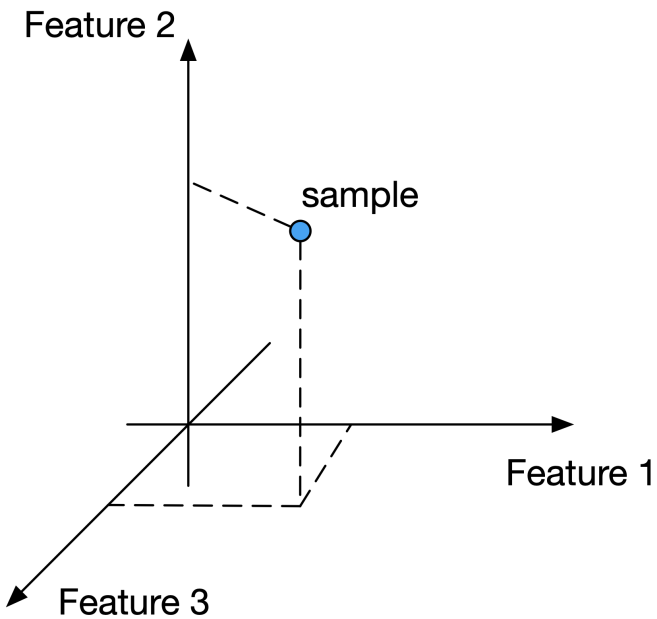
The data can be organized in a (n x m) matrix such that the m-dim space is populated (or sampled) by n points.

Low-dimensional data

a sample point in low-dimensional space ($m = 2, 3$)



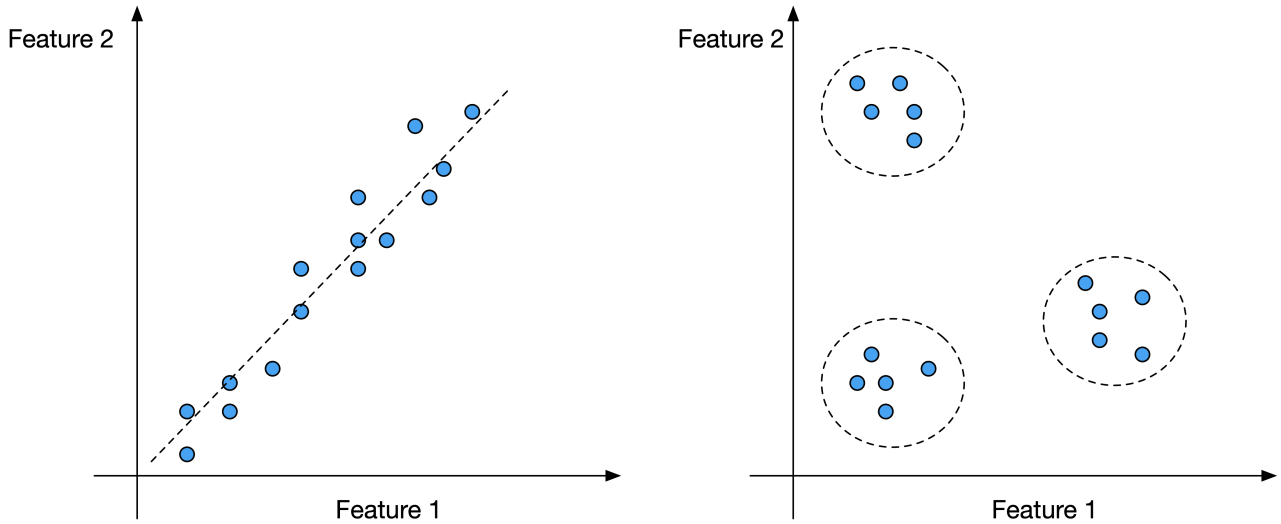
sample: (x = feature 1, y = feature 2)



sample: (x = feat. 1, y = feat. 2, z = feat. 3)

Spatial organization

Visual representations of lower-dimensional data can reveal structural insights.



Roughly, the structure falls into two distinct categories.

- Functional relation $y = f(x)$
- Clusters

Clustering

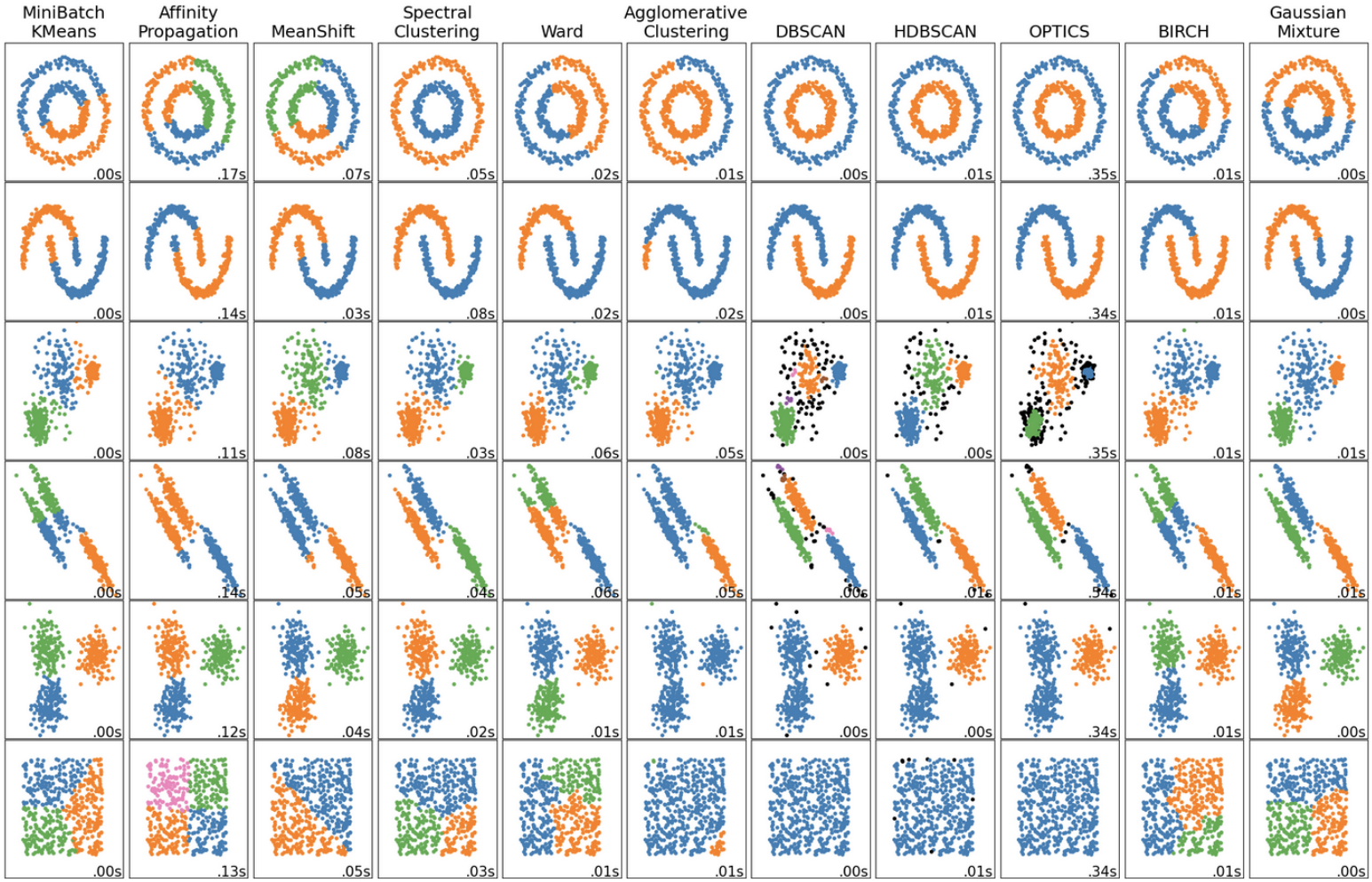
Clustering

Cluster analysis or **clustering** is the task of grouping a set of datapoints in such a way that datapoints in the same group (cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Cluster analysis is a main task of exploratory data analysis, and a common technique for statistical data analysis.

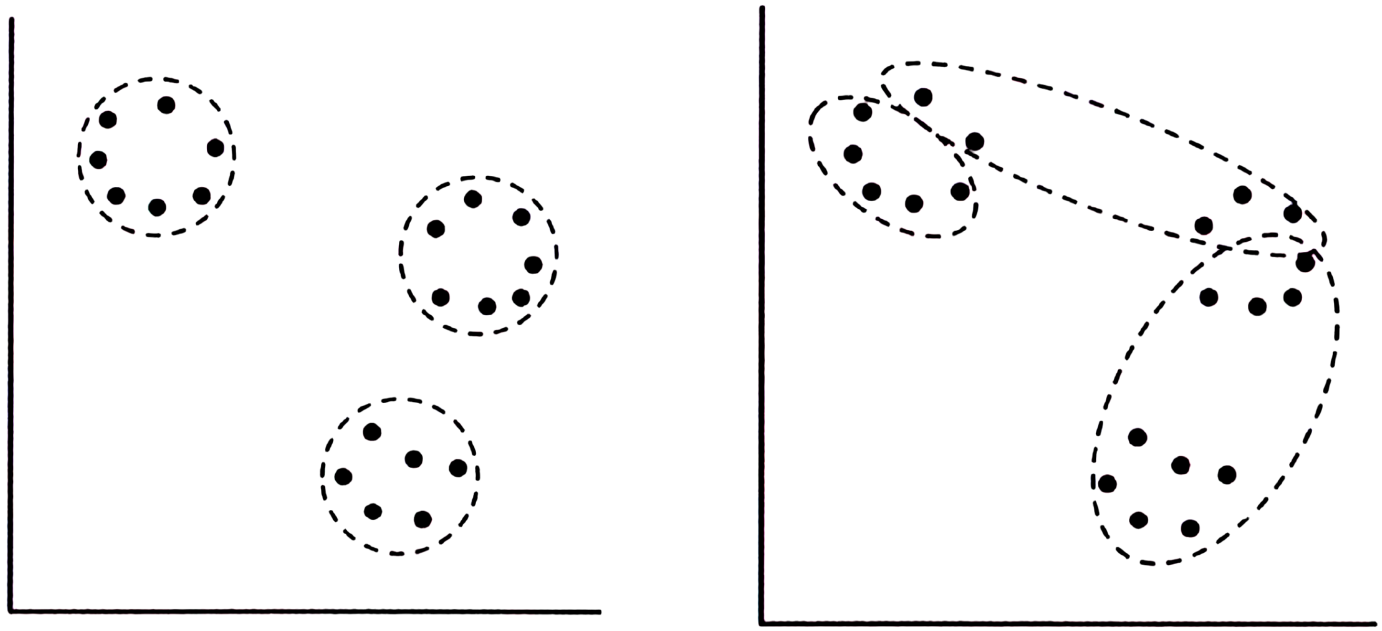
Clustering

See [scikit-learn]():



Good Clustering Principle

Good Clustering Principle: Every pair of points within the same cluster should be closer to each other than any pair of points from different clusters.



Clustering

What does **closer** mean? We need a notion of distance!

Euclidean distance between points (vectors) $v = (v_x, v_y)$ and $w = (w_x, w_y)$ in two-dimensional space, denoted by $d(v, w)$

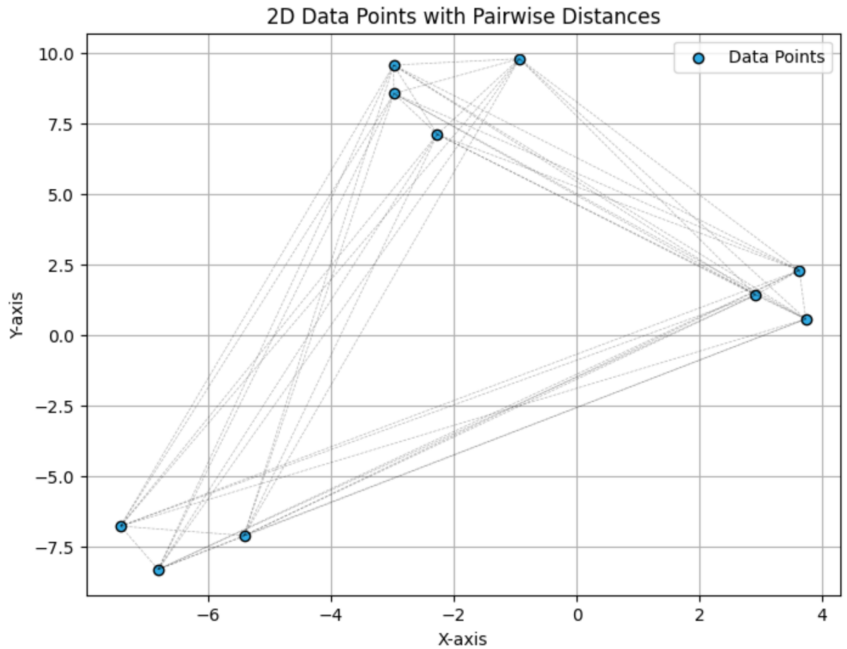
$$d(v, w) = \sqrt{(v_x - w_x)^2 + (v_y - w_y)^2},$$

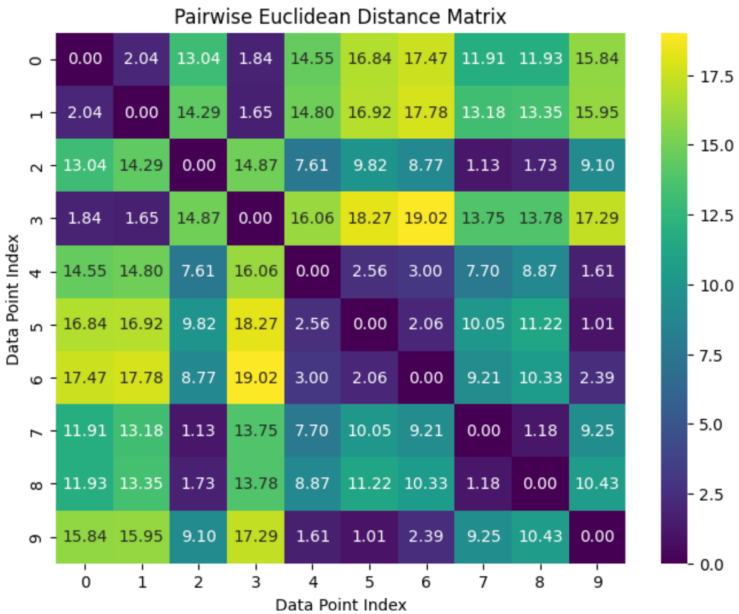
and in m dimensions:

$$d(v, w) = \sqrt{\sum_{i=1}^{m} (v_i - w_i)^2},$$

Clustering - pairwise distance

For n points we have $\binom{n}{2}$ distances $\Rightarrow$ pairwise distance matrix.

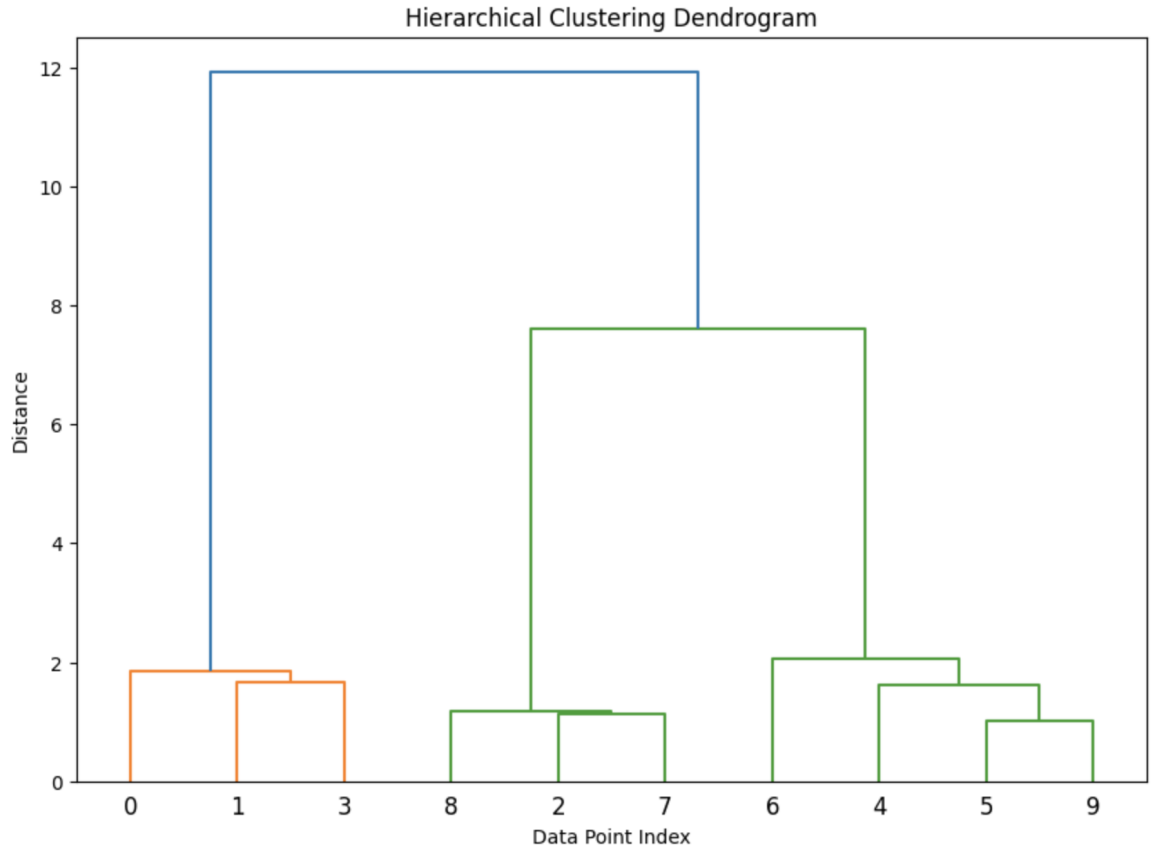




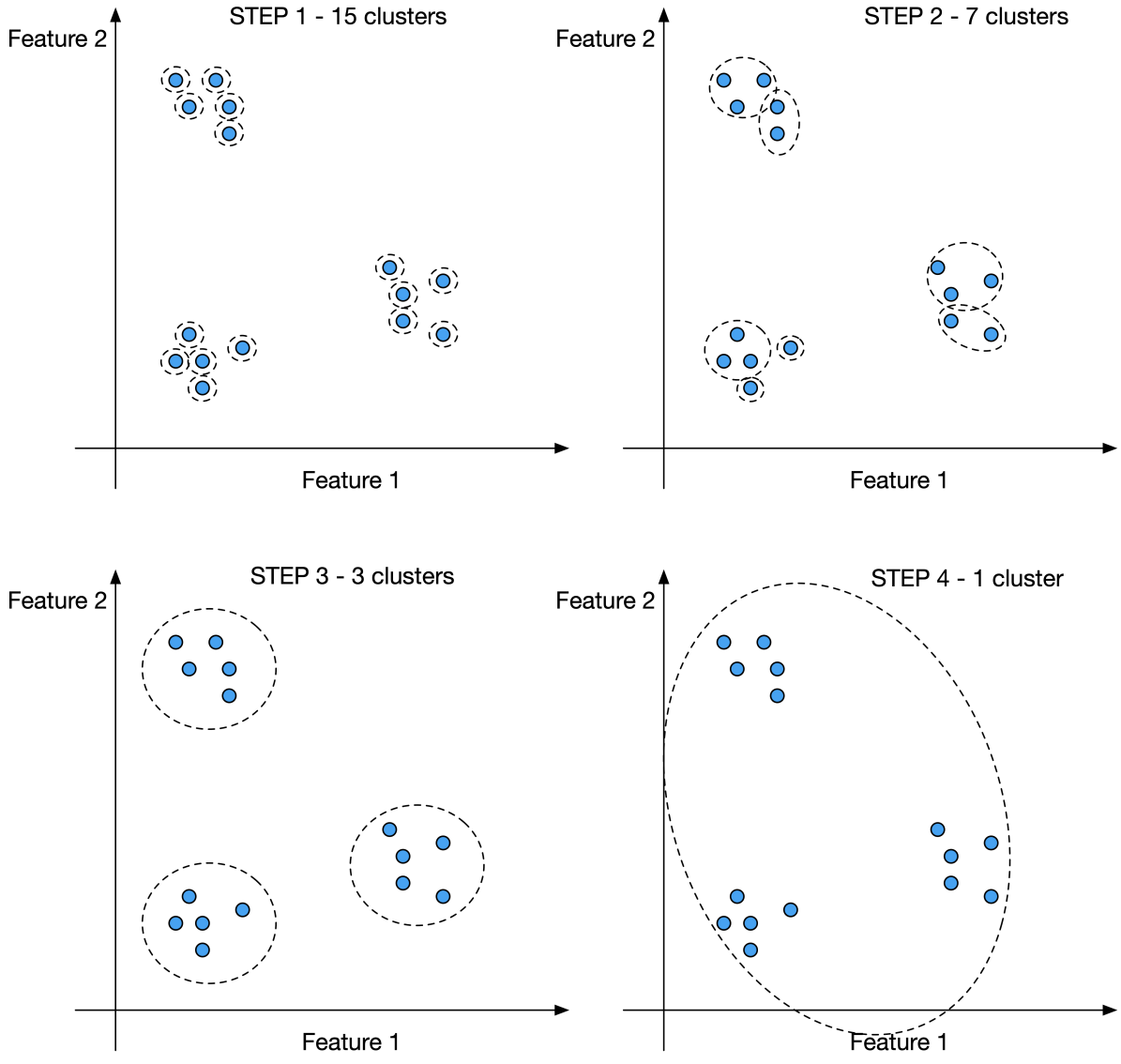
There are various methods to form clusters from here.

Hierarchical clustering

Starting with each point as its own individual cluster, we iteratively merge the closest clusters as the distance threshold increases.



Hierarchical clustering



K-means

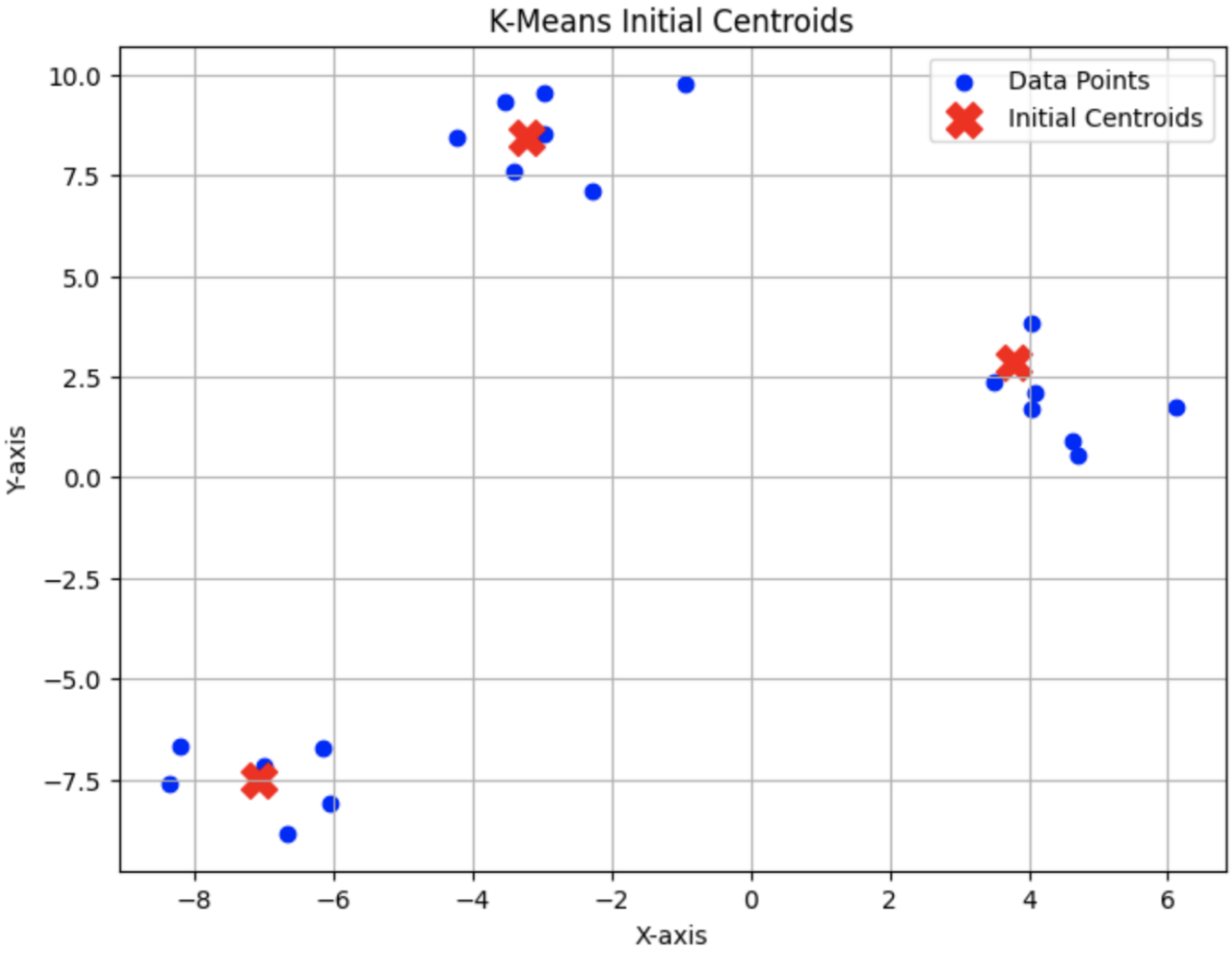
Separate samples in k groups of equal variance starting with random placement of centroids. The idea is to update the centroids position (computing the mean of all points assigned to that cluster) and minimize the **within-cluster sum-of-squares criterion**:

$$\text{WCSS} = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{c}_j\|^2$$

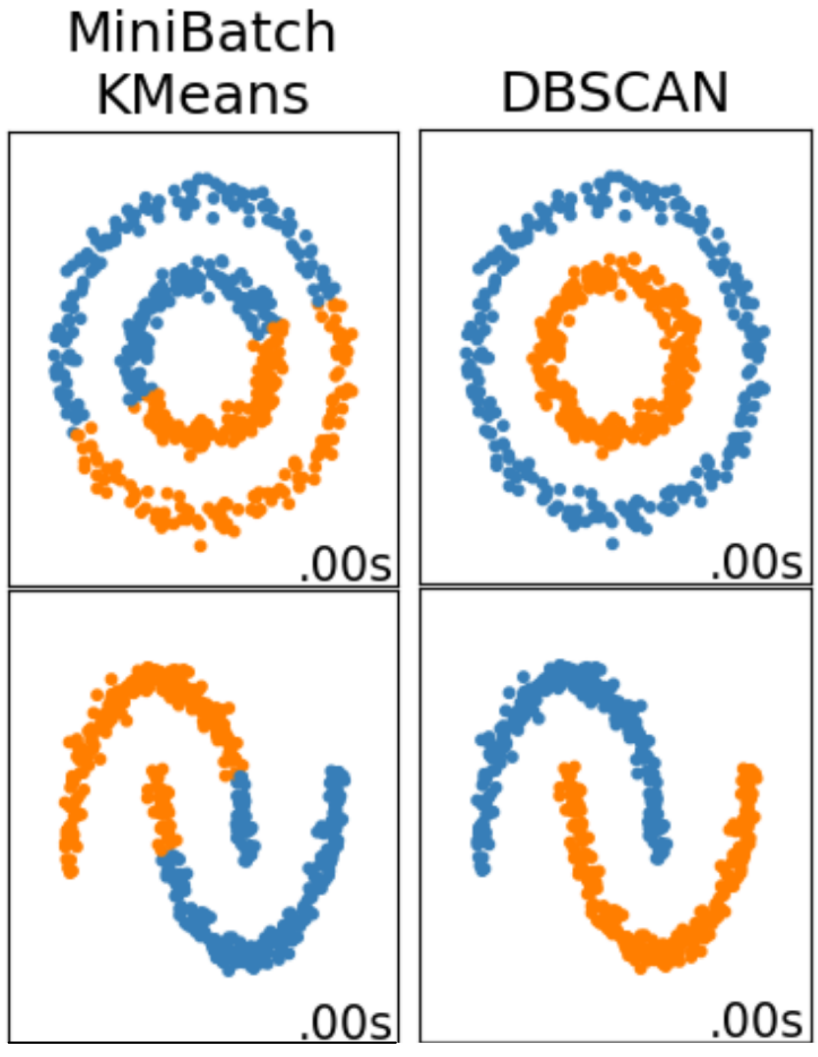
where:

- k is the number of clusters,
- C_j is the set of points in cluster j,
- $\mathbf{x}_i$ is a data point in cluster C_j,
- $\mathbf{c}_j$ is the centroid of cluster C_j.

K-means


K-Means Initial Centroids
Data Points
Initial Centroids
Y-axis
X-axis
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−8
−6
−4
−2
0
2
4
6

Limitations



Density-Based Clustering

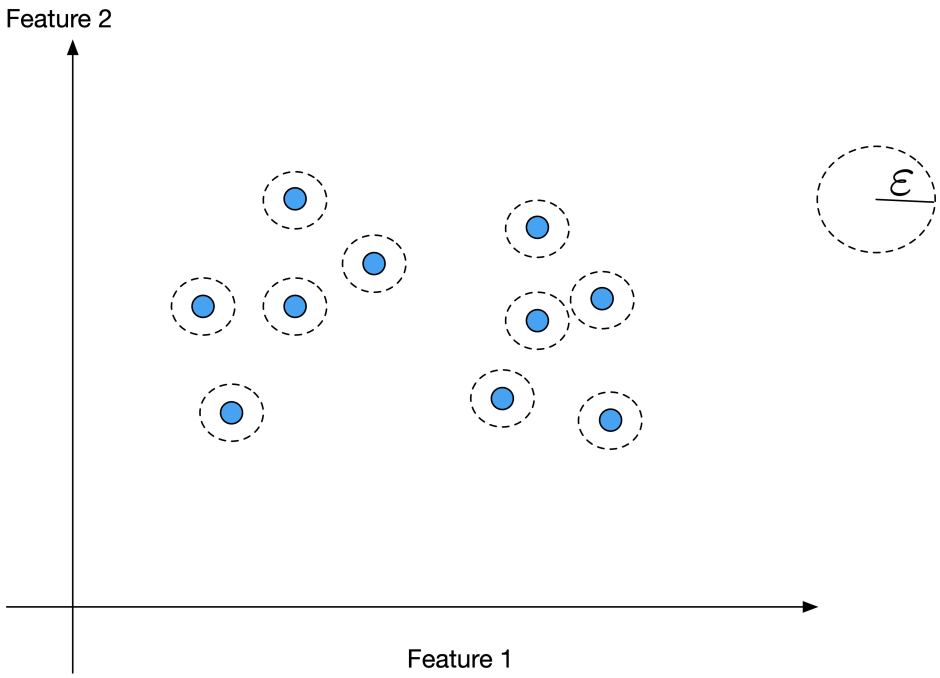
DBSCAN (**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise)

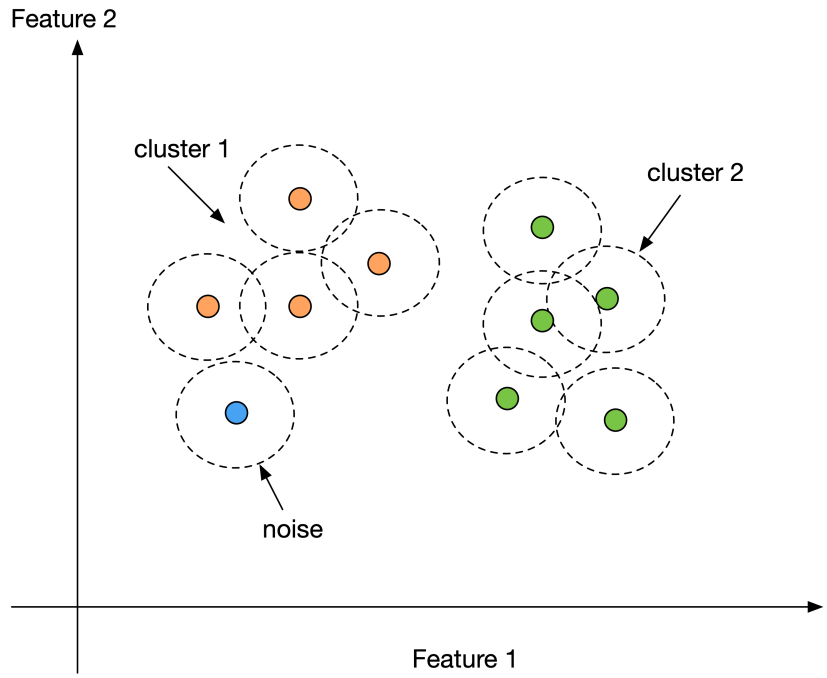
- **Epsilon** ε-environment for each data point detects neighbors.
- **MinPts:** Density given by number of points within a ε - proximity.

Advantages:

- Can detect clusters of arbitrary shape.
- Automatically identifies noise and outliers.
- No need to predefine the number of clusters.

Density-Based Clustering





Disadvantages:

- Sparse regions often labeled as noise
- Tend to fail is higher dimensions
- pairwise distance computation is costly

Quantitative Comparison Using Feature Vectors

- Feature vectors represent each sample in a multi-dimensional space.
- These vectors enable quantitative comparison between samples.
- Similarity and distance measures quantify how alike or different two samples are.

Similarity (Distance) Measures:

- **Euclidean Distance:** Measures the straight-line distance between two points.
- **Cosine Similarity:** Measures the cosine of the angle between two vectors.
- **Jaccard Similarity:** Ratio of intersection over union (genes expressed in different tissues).
- **Pearson Correlation:** Measures the linear relationship between two vectors.

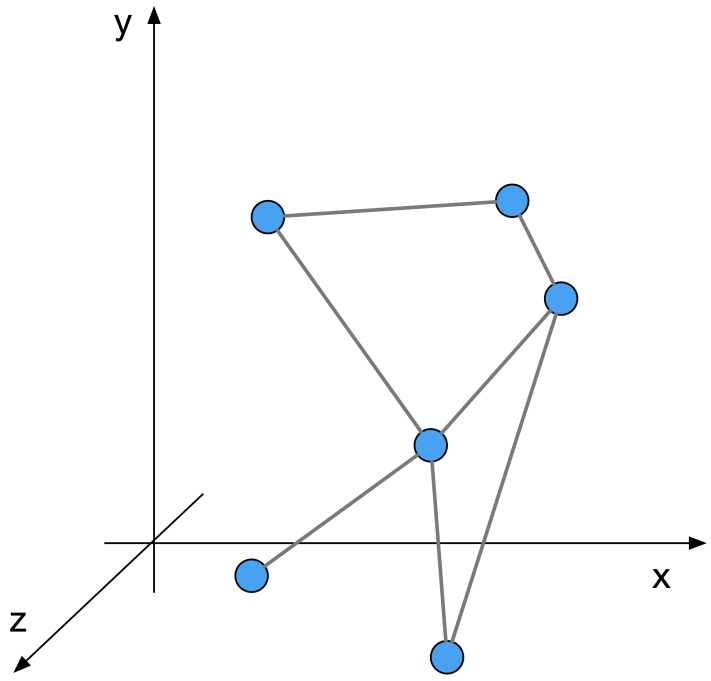
Quantitative Comparison Using Feature Vectors

Interpretation:

- Low distance or high similarity values indicate more similar samples.
- The choice of the metric depends on the data type and analysis goal.
- Conversion of distance to similarity: sim $\propto \exp(-\beta d)$
- Pairwise similarities serve as weights or can be filtered (sparsification)

Network Clustering

Remaining pairwise connections represent edges to create a network



Modularity in Network Clustering

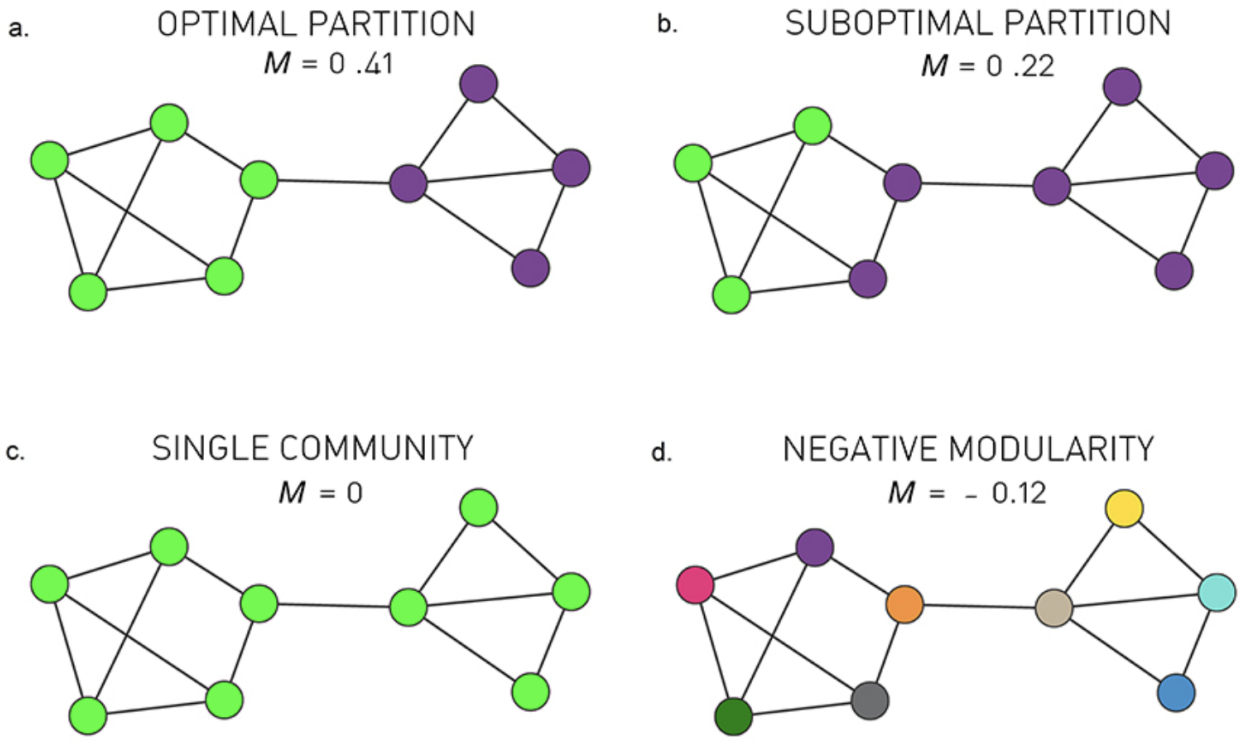
Networks can be compartmentalized considering the wiring structure. Modularization divides a network into tightly connected communities with sparse interconnections.

Modularity Formula:

$$M = \sum_c \left(\frac{l_c}{L} - \frac{k_c^2}{4L^2} \right)$$

- l_c: Number of edges within community c.
- L: Total number of edges in the entire network.
- k_c: Sum of degrees of all nodes in community c.

Modularity in Network Clustering



Adapted from Albert-László Barabási, Network Science, Cambridge University Press, 2016.

Dimensionality Reduction

Challenges of High-Dimensional Space

Working in high-dimensional spaces can be undesirable for many reasons:

- Raw data is often sparse or fat ($m \gg n$) $\Rightarrow$ curse of dimensionality.
- Analyzing the data is usually computationally intractable (hard to control or deal with).
- High-dimensional objects can have very bizarre properties.
- Visualization beyond three-dimensions is unintuitive.

The Curse of Dimensionality

- **Data Sparsity:** In high dimensions, data points become sparse, making it difficult to find meaningful patterns.
- **Distance Measures Lose Meaning:** Differences in distances between points diminish, as most points become equidistant.
- **Exponential Growth in Complexity:** The number of samples required to represent the data grows exponentially with dimensionality.
- **Model Overfitting:** High-dimensional spaces increase the risk of overfitting due to excessive flexibility in models.

Implications:

- Reducing dimensionality (e.g., using PCA, t-SNE, or UMAP) is often essential.
- Focus on selecting relevant features to mitigate noise and improve interpretability.

Latent space

Dimension(ality) reduction is the transformation of data from a high-dimensional (ambient-, data-) space into a low-dimensional (latent-) space.

- Low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

An **embedding** is a relatively low-dimensional space into which you can translate high-dimensional vectors.

The m-dimensional space

Remember the sample vectors (rows)

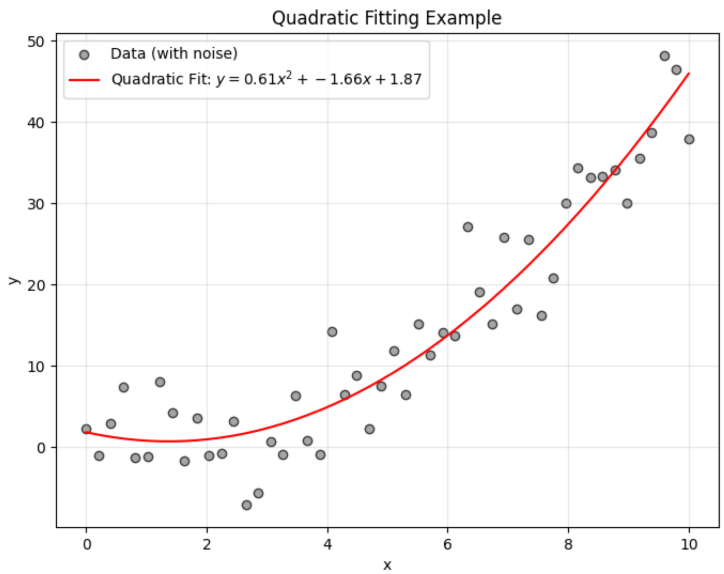
$$\boldsymbol{C}(1,\cdot),\ldots,\boldsymbol{C}(n,\cdot)$$

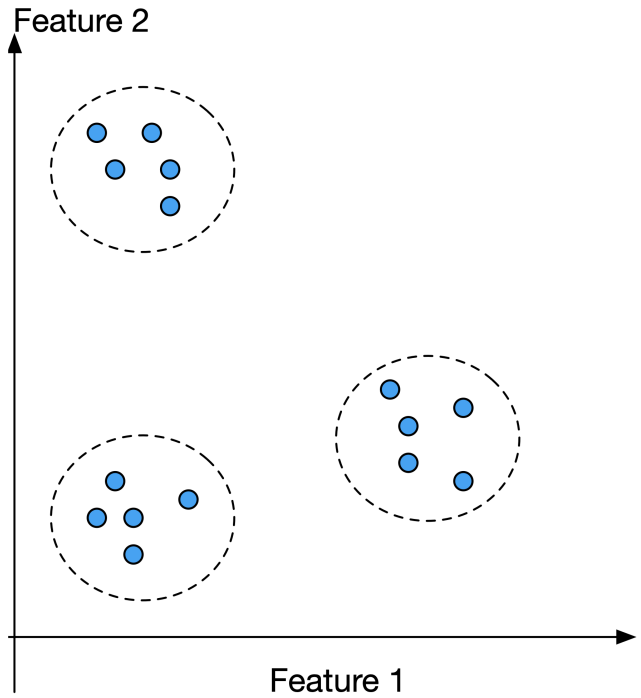
are m-dimensional (m features).
The geometric shape that the point cloud takes on in m-dimensional space can be complex, hard to interpret and impossible to visualize.

Thus we need **dimensionality reduction techniques** to better understand the data.

Embedding

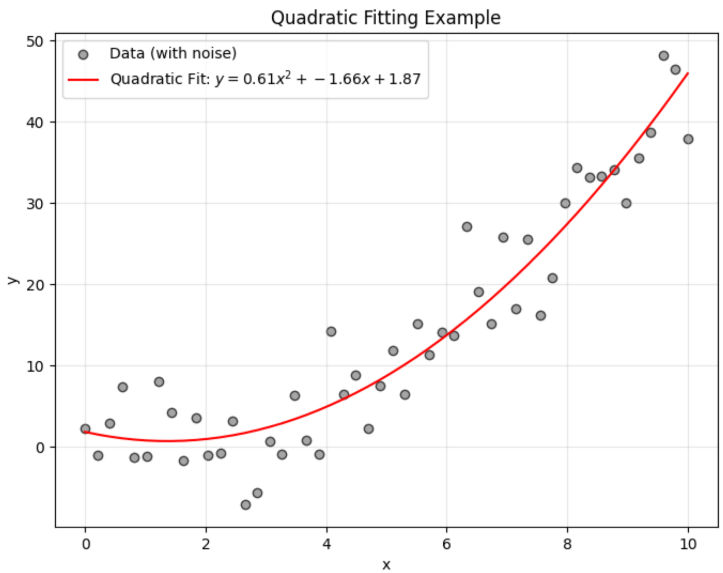
Dimensionality reduction techniques work with the local structure of data. They try to identify patterns that can reduce the details of single point locations into clusters (dense point clouds), functional relations (regression), or manifolds.





Embedding

Dimensionality reduction techniques work with the local structure of data. They try to identify patterns that can reduce the details of single point locations into clusters (dense point clouds), functional relations (regression), or manifolds.



$$(x_1|y_1), (x_1|y_1), ..., (x_n|y_n)$$
$$\Rightarrow f(x) = ax^2 + bx + c$$

Feature Selection

Some features may be more relevant than others.

- **Explanatory Features:** Capture the essential structure or patterns in the data.
- **Noisy/Redundant Features:** Irrelevant or redundant information that obscures meaningful patterns.

Extraction Techniques:

- **Variance Thresholding:** Remove features with low variance.
- **Correlation analysis:** Remove highly correlated feature to reduce redundancy .
- **Random forest:** Identifies and ranks highly explanatory features through ensemble learning.
- **Autoencoder:** Learn compressed feature representations.
- **Entropy-Based Methods:** Maximize information gain.

Linear methods

Classical techniques: Principal Component Analysis (PCA) or Multidimensional Scaling (MDS) aim to reveal underlying structures in data that may lie on or near a linear subspace.

PCA finds a low-dimensional embedding that best preserves the variance measured in the m-dimensional space

MDS computes an embedding that preserves pairwise Euclidean distances.

\+ Computationally efficient, easy to implement

− Restricted to linear relations

Principal Component Analysis (PCA)

- We have m features (genes) and assume that the data points (cells) are scattered around and primarily concentrated near a linear subspace of dimension $p \ll m$ (a p-dimensional plane).
- The new variables that span the subspace are called **principal components**, each principal component is a m-dimensional vector.
- For m-dimensional measurements we get m principal components, but we use only p of them. The choice of p depends on the percentage of variance/variation we would like to maintain in the smaller subspace.

Principal Component Analysis (PCA)

Goal: Reduce the dimensionality of data while preserving as much variance as possible.

Concept:

- PCA transforms the original data into a new coordinate system.
- The first principal component captures the direction of the highest variance.
- Each subsequent component captures the next highest variance, orthogonal to the previous ones.

Data Representation:

- Data matrix: $\mathbf{X} \in \mathbb{R}^{n \times m}$
  - n: Number of samples (rows)
  - m: Number of features (columns)
- Center the data by subtracting the mean.

Principal Components in PCA

Finding Principal Components:

- Compute the **covariance matrix**:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$$

- Perform **eigen decomposition** of $\mathbf{C}$:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

- $\mathbf{v}_i$: Eigenvectors (principal components)
  λ_i: Eigenvalues (variance explained)

Interpretation:

- Eigenvectors define the new axes (principal components).
- Eigenvalues indicate the importance (variance captured) of each component.

PCA Transformation

Projecting Data onto Principal Components:

- Select the top k (=2,3) eigenvectors to form a projection matrix $\mathbf{W}$:

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

- Transform the original data:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}$$

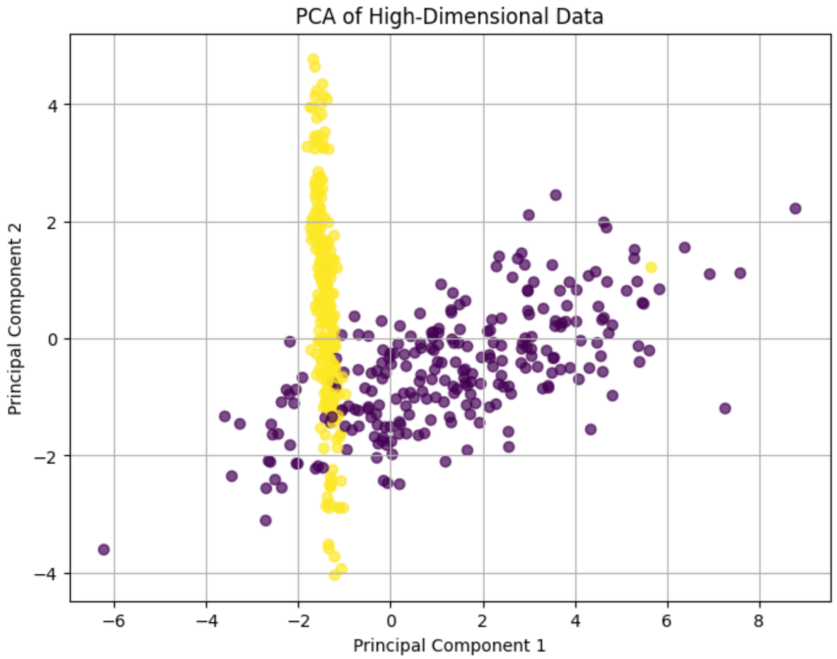
- $\mathbf{Z}$ is the lower-dimensional representation of $\mathbf{X}$.

Outcome:

- Data is now expressed in terms of principal components.
- Dimensionality is reduced with minimal loss of variance.

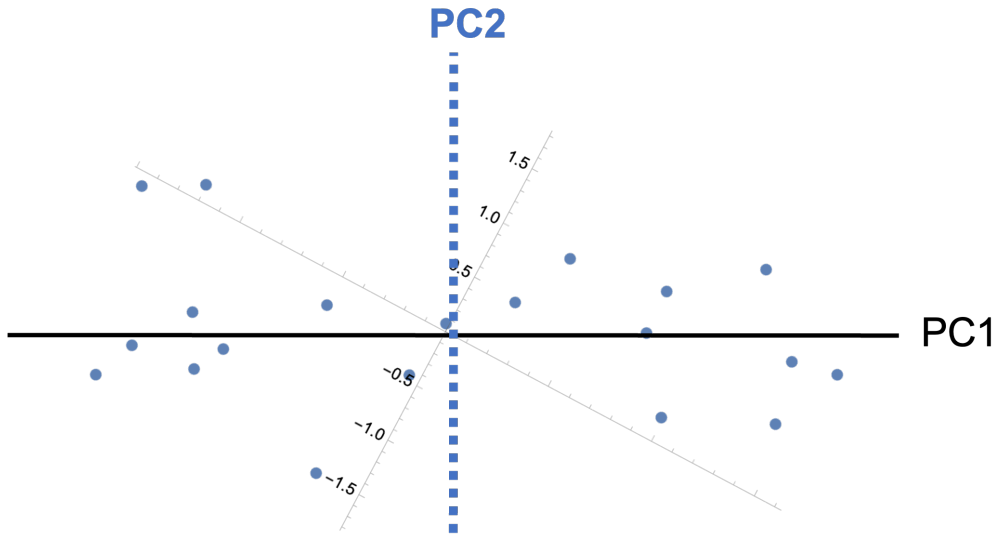
PCA Transformation

$$\mathbb{R}^{500\times 10} \rightarrow \mathbb{R}^{500\times 2}$$



Principal Component Analysis (PCA)

The principal components describe a new coordinate system:

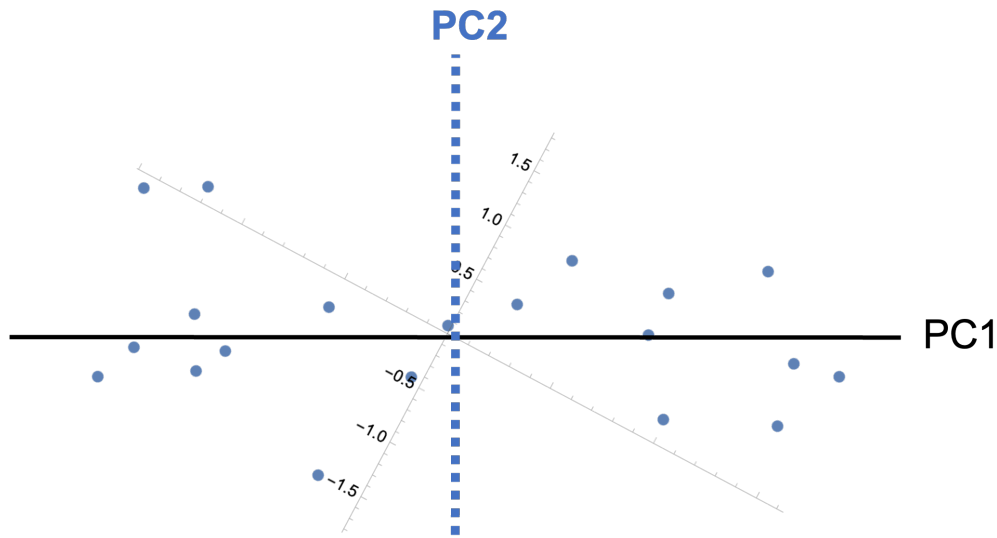


We observe: $\lambda_1 > \lambda_2 \Rightarrow$ More variation in the first principal component.

Total variation: $\lambda_1 + \lambda_2$. The ratio $\frac{\lambda_1}{\lambda_1+\lambda_2}$ describes the percentage of total variation explained by PC1.

Principal Component Analysis (PCA)

Total variation: $\lambda_1 + \lambda_2$. The ratio $\frac{\lambda_1}{\lambda_1+\lambda_2}$ describes the percentage of total variation explained by PC1.

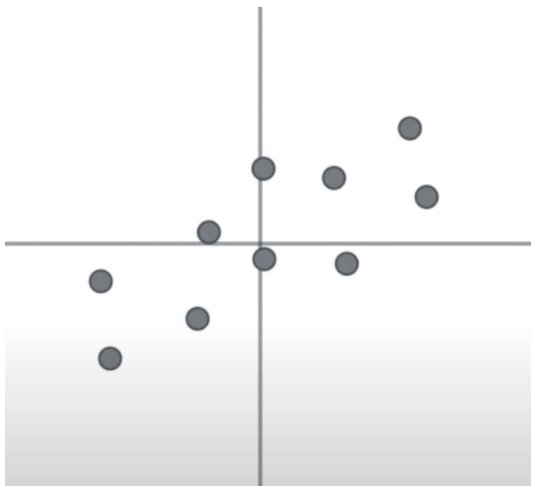


If $\frac{\lambda_1}{\lambda_1+\lambda_2}$ is large, then simply drawing PC1 and the projections of the data on PC1 is a good approximation of the 2-dim graph.

Principal Component Analysis (PCA)

An algorithm to compute the eigenvalues and eigenvectors

Center the data.



Compute the covariance between gene 1 and gene 2

$$\text{cov}(x_1, x_2) = \sum_{i=1}^{n} x_i(i)x_2(i)$$

and obtain the symmetric covariance matrix, where the variances are on the main-diagonal and the covariances on the off-diagonal elements

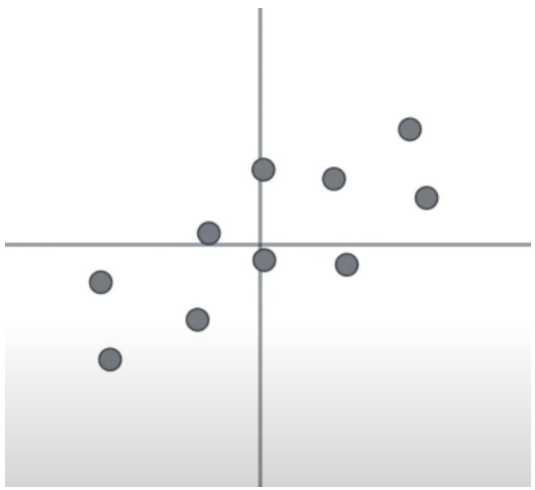
$$Cov = \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) \\ \text{cov}(x_1, x_2) & \text{var}(x_2) \end{bmatrix}$$

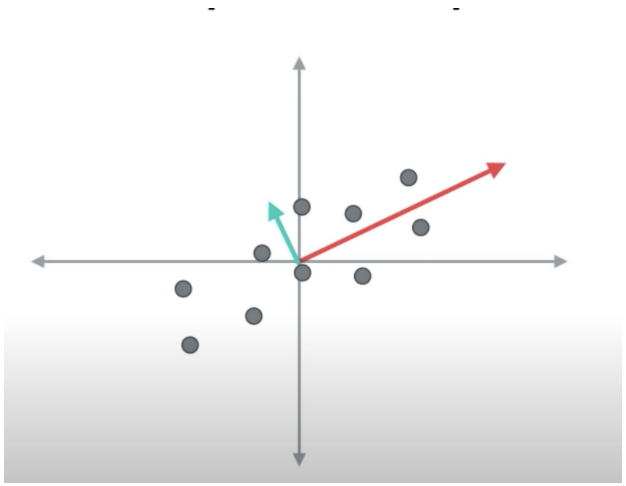
Principal Component Analysis (PCA)

An algorithm to compute the eigenvalues and eigenvectors

Compute the eigenvalues and eigenvectors of

$$Cov = \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) \\ \text{cov}(x_1, x_2) & \text{var}(x_2) \end{bmatrix}$$





Principal Component Analysis (PCA)

Summary:

- **Input:** Large table, i.e, $\boldsymbol{C}$
- Column-wise normalize $\boldsymbol{C}$ by subtracting the mean for each $i = 1, \ldots, m$
- Use the normalized data matrix to compute the $m \times m$ covariance matrix
- Compute the m real eigenvalues and eigenvectors.
- Take the biggest eigenvalue(s) and project the high m-dimensional data on the smaller space.

PCA with Python

Don't worry, the computer is your friend.

```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# 1. Standardize the data
scaler = StandardScaler()
C_scaled = scaler.fit_transform(C)

# 2. Apply PCA to get the first two components
pca = PCA(n_components=2)
C_pca = pca.fit_transform(C_scaled)

# 3. Plot the PCA results
plt.figure(figsize=(8, 6))
plt.scatter(C_pca[:, 0], C_pca[:, 1], alpha=0.7)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

Make sure that C is a proper feature matrix with samples as rows and features as columns.