Linear Algebra:

The determinant of a square matrix A is a scalar value that encodes properties about the linear mapping described by A.

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

det(A) = ad – bc

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad det(A) = a \cdot det\begin{pmatrix} e & f \\ h & i \end{pmatrix} - b \cdot det\begin{pmatrix} d & f \\ g & i \end{pmatrix} + c \cdot det\begin{pmatrix} d & e \\ g & h \end{pmatrix}$$

Property:

**n** vectors $\{\vec{v}_1, ..., \vec{v}_n\}$ in an n-dimensional space are **linearly independent** iff the matrix **A**:

$$\mathbf{A} = [\vec{v}_1, ..., \vec{v}_n] \text{ (n x n)}$$

has non-zero determinant.

**Definition**:

A **basis B** of a vector space (over a field **F**) is a **linearly independent** subset of **V** that **spans V**. **B spans V** if for every vector **v** in **V** it is possible to choose $\mathbf{v}_1$, ..., $\mathbf{v}_n$ in **F** and $\vec{b}_1$, ..., $\vec{b}_n$ in **B** such that:

$$\mathbf{v} = \mathbf{v}_1\vec{b}_1 + ... + \mathbf{v}_n\vec{b}_n$$

Rank:

The rank of a matrix A is the dimension of the vector space spanned by its column space. This is equivalent to the maximal number of linearly independent columns / rows of A.

Definition: A matrix A is full-rank iff rank(A) = min(m, n)

To store an n x m matrix A requires storing m · n values.
However, if the rank of the matrix of A is k, A can be factored as
A = UV
Where U is n x k and V is k x m which requires storing k(m + n) values.

Frobenius Distance:

$$d_F(A, B) = \|A - B\|_F = \sqrt{\sum_{i,j}(a_{ij} - b_{ij})^2}$$

Approximation:
When k < rank(A), the rank-k approximation of A (in the least squares sense) is

$$A^{(k)} = \underset{\{B|rank(B)=k\}}{\arg\min} \; d_F(A, B)$$

The Singular Value Decomposition of a rank-r matrix A has the form: $A = U\mathbf{\Sigma}V^T$
U is n x r The columns of U are orthogonal & unit length ($U^TU = I$)
V is m x r The columns of V are orthogonal & unit length ($V^2V = I$)

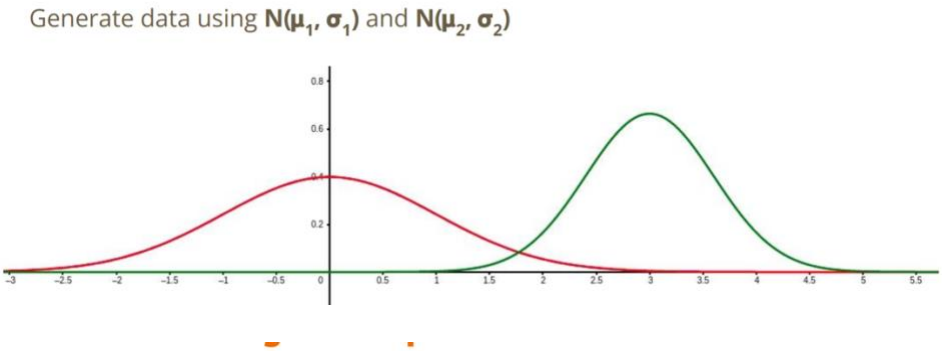
Property:

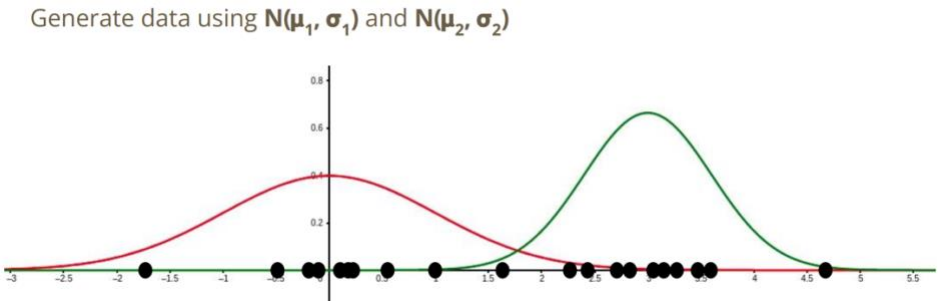
$$d_F(A, A^{(k)})^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Latent Semantic Analysis:
Inputs are documents. Each word is a feature.
We can represent each document by:
-The presence of the word (0 / 1)
- Count of the word (0, 1, … )
- Frequency of the word (ni / **Σ**ni )
- TfiDf

Anomaly Detection:
Define $O = A - A^{(k)}$, The largest rows of O could be considered anomalies

Soft Clustering:




Mixture Model:
X comes from a mixture model with k mixture components if the probability distribution of X is

$$P(X = x) = \sum_{j=1}^{k} P(C_j)P(X = x|C_j)$$

Mixture proportion
Represents the probability
of belonging to $C_j$

Probability of seeing x
when sampling from $C_j$

Gaussian Mixture Model:
A Gaussian Mixture Model (GMM) is a mixture model where

$$P(X = x|C_i) \sim N(\mu, \sigma)$$

GMM Clustering:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{n} \sum_{j=1}^{k} P(C_j)P(X_i \mid C_j)$$

It is a joint probability distribution of our data and we assume our data are independent

We can define the following:

$$l(\theta) = \log(L(\theta))$$
$$= \sum_{i=1}^{n} \log(\sum_{j=1}^{k} P(C_j)P(X_i \mid C_j))$$

For $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k]^T$ and $\boldsymbol{\Sigma} = [\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k]^T$

We can solve

$$\frac{d}{d\Sigma} l(\theta) = 0 \qquad \frac{d}{d\mu} l(\theta) = 0$$

And then we can get:

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} P(C_j|X_i)X_i}{\sum_{i=1}^{n} P(C_j|X_i)}$$

$$\hat{\Sigma}_j = \frac{\sum_{i=1}^{n} P(C_j|X_i)(X_i - \hat{\mu}_j)^T(X_i - \hat{\mu}_j)}{\sum_{i=1}^{n} P(C_j|X_i)}$$

$$\hat{P}(C_j) = \frac{1}{n}\sum_{i=1}^{n} P(C_j|X_i)$$

$$P(C_j|X_i) = \frac{P(X_i|C_j)}{P(X_i)}P(C_j)$$
$$= \frac{P(X_i|C_j)P(C_j)}{\sum_{j=1}^{k} P(C_j)P(X_i|C_j)}$$

Expectation Maximization Algorithm:
1. Start with random θ
2. Compute P(Cj | XI ) for all Xi by using θ
3. Compute / Update θ from P(Cj | XI )
4. Repeat 2 & 3 until convergence

Hierarchical Clustering:

Two main types of hierarchical clustering:
Agglomerative:
1. Start with every point in its own cluster
2. At each step, merge the two closest clusters
3. Stop when every point is in the same cluster
Divisive:
1. Start with every point in the same cluster
2. At each step, split until every point is in its own cluster

Agglomerative Clustering Algorithm
1. Let each point in the dataset be in its own cluster
2. Compute the distance between all pairs of clusters
3. Merge the two closest clusters
4. Repeat 3 & 4 until all points are in the same cluster

Questions to be considered:
Can we implement this?
Are we missing anything?
How do we compute the distance between clusters?
Distance between clusters can be thought of as distance between two sets of points. What ideas come to mind?

Single-Link Distance is the minimum of all pairwise distances between a point from one cluster and a point from the other cluster.

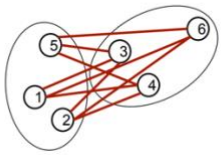
$$D_{SL}(C_1, C_2) = \min \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$

Complete-Link Distance is the maximum of all pairwise distances between a point from one cluster and a point from the other cluster.

$$D_{CL}(C_1, C_2) = \max \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$

Average-Link Distance is the average of all pairwise distances between a point from one cluster and a point from the other cluster.

$$D_{AL}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$$

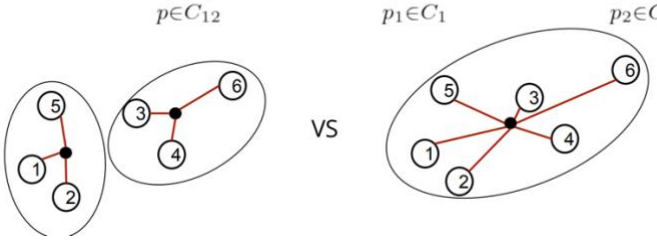


Centroid Distance is the distance between the centroids of clusters.

$$D_C(C_1, C_2) = d(\mu_1, \mu_2)$$

Ward's Distance is the difference between the spread / variance of points in the merged cluster and the unmerged clusters.

$$D_{WD}(C_1, C_2) = \sum_{p \in C_{12}} d(p, \mu_{12}) - \sum_{p_1 \in C_1} d(p_1, \mu_1) - \sum_{p_2 \in C_2} d(p_2, \mu_2)$$




Hierarchical Clustering: Finding the threshold with which to cut the dendrogram requires exploration and tuning. But in general hierarchical clustering is used to expose a hierarchy in the data (ex: finding/defining species via DNA similarity). To capture the difference between clusterings you can use a cost function, or methods that we will discuss later when we look at clustering aggregation.

Density-Based Clustering

DB Scan Algorithm
**ε** and min_pts given:
1. Find the **ε**-neighborhood of each point
2. Label the point as core if it contains at least min_pts
3. Label points in its neighborhood that are not core as border
4. Label points as noise if they are neither core nor border
5. For each core point, assign to the same cluster all core points in its neighborhood
6. Assign border points to nearby clusters

DBScan - Benefits
1. Can identify clusters of different shapes and sizes
2. Resistant to noise

DBScan - Limitations
1. Can fail to identify clusters of varying densities.
2. Tends to create clusters of the same density.
3. Notion of density is problematic in high-dimensional spaces