Unsupervised Learning Techniques

What is Unsupervised Learning?

- **No labels** – only features
- Goal: find **hidden patterns** or **structures** in data
- Common tasks:
  - Clustering
  - Anomaly detection
  - Density estimation
  - Dimensionality reduction

Clustering

- Group similar instances into **clusters**
- Applications:
    - Customer segmentation
    - Data analysis
    - Dimensionality reduction
    - Anomaly detection (outlier detection)
    - Semi-supervised learning
    - Search engines
    - Image segmentation

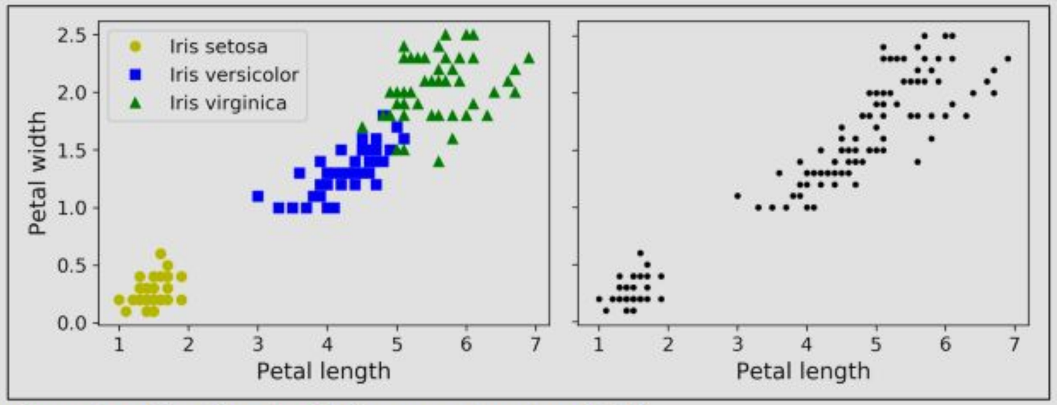


Figure 9-1. Classification (left) versus clustering (right)

K-Means Clustering

- **Algorithm**:
  - Initialize centroids randomly
  - Assign each instance to nearest centroid
  - Update centroids as mean of assigned instances
  - Repeat until convergence

Equation:

$$\mu_k = \frac{1}{|C_k|} \sum_{x \in |C_k|} x$$

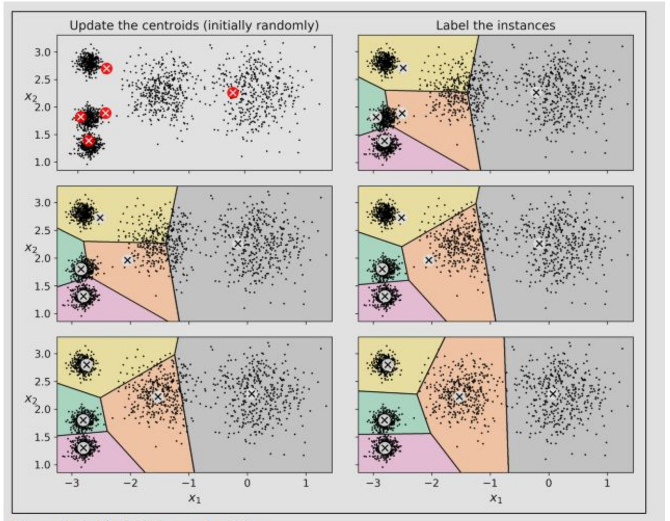


Figure 9-4. The K-Means algorithm

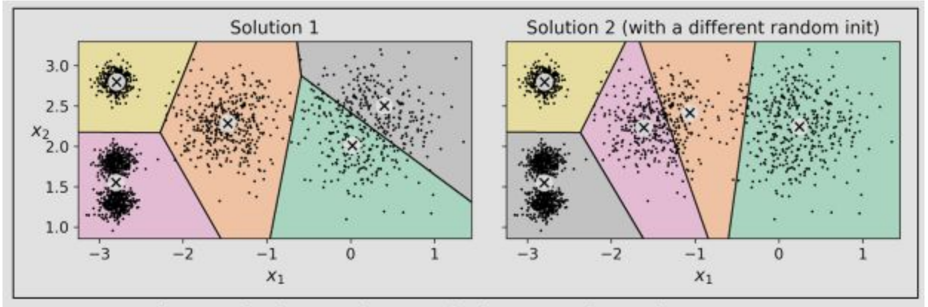


Figure 9-5. Suboptimal solutions due to unlucky centroid initializations

Voronoi tessellation

Centroid initialization methods

- Run the algorithm multiple times with different random initializations and keep the best solution
- Use a performance metric **“the model’s inertia”** to select the best model.
- **Inertia**: Sum of squared distances to nearest centroid
- Used as a cost function to minimize:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} r_{ik} \left|\left| x_i - \mu_k \right|\right|^2$$

- where $r_{ik} = 1$ if x_i is in cluster k, else 0
- Select the model with the lowest inertia

Finding the Optimal Number of Clusters

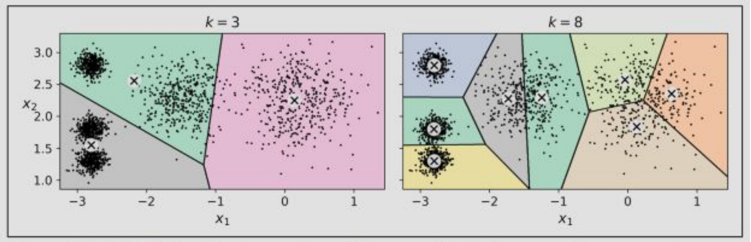


Figure 9-7. Bad choices for the number of clusters: when k is too small, separate clusters get merged (left), and when k is too large, some clusters get chopped into multiple pieces (right)

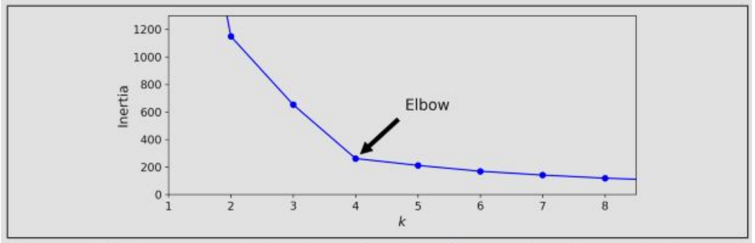


Figure 9-8. When plotting the inertia as a function of the number of clusters k, the curve often contains an inflexion point called the "elbow"

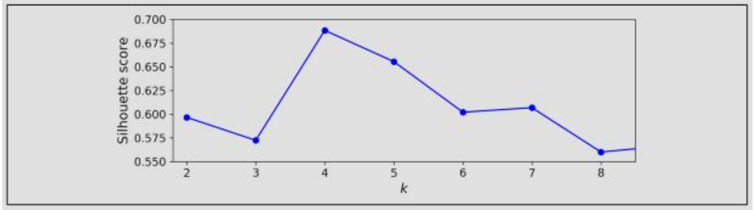


Figure 9-9. Selecting the number of clusters k using the silhouette score

- Silhouette coefficient:

$$\frac{b - a}{\max(a, b)}$$

- Where, a is the mean intra-cluster distance and b is the mean nearest-cluster distance.
- Coefficient = +1 → The instance is well inside its own cluster and far from other clusters
- Coefficient = 0 → Instance is close to a cluster
- Coefficient = –1 → Instance assigned to the wrong cluster

K-Means Limitations

- Requires number of clusters k to be chosen in advance
- Sensitive to centroid initialization
- Poor performance with non-spherical clusters
- Struggles with clusters of different sizes/densities

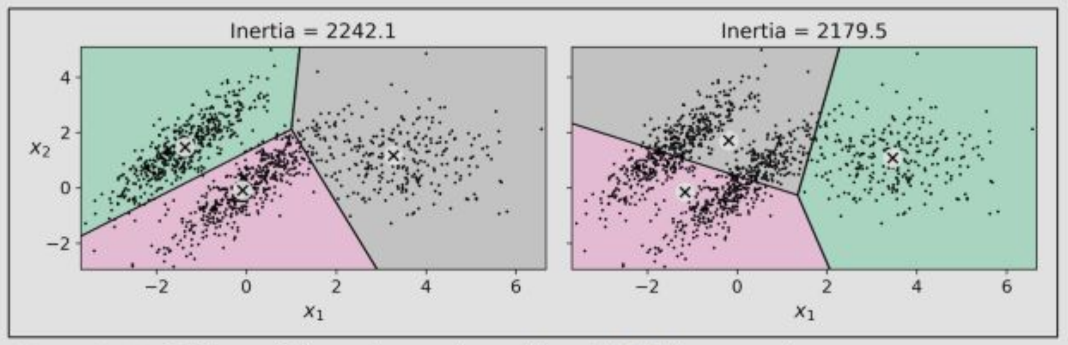


Figure 9-11. K-Means fails to cluster these ellipsoidal blobs properly

Problem Setup

- Cluster the following eight points (with (x, y)) into three clusters
    - **A1(2, 10)**
    - **A2(2, 5)**
    - **A3(8, 4)**
    - **A4(5, 8)**
    - **A5(7, 5)**
    - **A6(6, 4)**
    - **A7(1, 2)**
    - **A8(4, 9)**

Solution

- Here $k = 3$
- Initial cluster centers are:

  $\boldsymbol{m1} = (\mathbf{2}, \mathbf{10}), \boldsymbol{m2} = (\mathbf{5}, \mathbf{8})$ and $\boldsymbol{m3} = (\mathbf{1}, \mathbf{2})$
- The distance function between two points $a = (x1, y1)$ and $b = (x2, y2)$ is defined as:

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

Iteration 1 – Cluster Assignment

- Clusters formed:
  - Cluster 1 = {A1}
  - Cluster 2 = {A3, A4, A5, A6, A8}
  - Cluster 3 = {A2, A7}

Point		(2, 10)	(5, 8)	(1, 2)	Cluster
		Dist Mean 1	Dist Mean 2	Dist Mean 3	
A1	(2, 10)	0	5	9	1
A2	(2, 5)	5	6	4	3
A3	(8, 4)	12	7	9	2
A4	(5, 8)	5	0	10	2
A5	(7, 5)	10	5	9	2
A6	(6, 4)	10	5	7	2
A7	(1, 2)	9	10	0	3
A8	(4, 9)	3	2	10	2

Iteration 1 – Update Centroids

- Re-compute the new cluster centers (means) by taking the mean of all points in each cluster.
- For Cluster 1, we only have one point A1 (2, 10), which was the old mean, so the cluster center remains the same.
- For Cluster 2,

$$\left(\frac{8+5+7+6+4}{5}, \frac{4+8+5+4+9}{5}\right) = (6,6)$$

- For Cluster 3,

$$\left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5, 3.5)$$

- Now cluster centers are:
  - $\boldsymbol{m1} = (\mathbf{2}, \mathbf{10})$
  - $\boldsymbol{m2} = (\mathbf{6}, \mathbf{6})$
  - $\boldsymbol{m3} = (\mathbf{1.5}, \mathbf{3.5})$

Iteration 2 – Cluster Assignment

- Clusters formed:
  - Cluster 1 = {A1, A8}
  - Cluster 2 = {A3, A4, A5, A6}
  - Cluster 3 = {A2, A7}

Point		(2,10) Dist Mean 1	(6,6) Dist Mean 2	(1.5,3.5) Dist Mean 3	Cluster
A1	(2, 10)	0	8	7	1
A2	(2, 5)	5	5	2	3
A3	(8, 4)	12	4	7	2
A4	(5, 8)	5	3	8	2
A5	(7, 5)	10	2	7	2
A6	(6, 4)	10	2	5	2
A7	(1, 2)	9	9	2	3
A8	(4, 9)	3	5	8	1

Iteration 2 – Update Centroids

- For Cluster 1,

$$\left(\frac{2+4}{2}, \frac{10+9}{2}\right) = (3,9.5)$$

- For Cluster 2,

$$\left(\frac{8+5+7+6}{4}, \frac{4+8+5+4}{4}\right) = (6.5, 5.25)$$

- For Cluster 3,

$$\left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5, 3.5)$$

- Now cluster centers are:
  - $\boldsymbol{m1} = (\mathbf{3}, \mathbf{9.5})$
  - $\boldsymbol{m2} = (\mathbf{6.5}, \mathbf{5.25})$
  - $\boldsymbol{m3} = (\mathbf{1.5}, \mathbf{3.5})$

Iteration 3 – Cluster Assignment

- Clusters formed:
  - Cluster 1 = {A1, A4, A8}
  - Cluster 2 = {A3, A5, A6}
  - Cluster 3 = {A2, A7}

Point		(3,9.5)	(6.5,5.25)	(1.5,3.5)	Cluster
		Dist Mean 1	Dist Mean 2	Dist Mean 3	
A1	(2, 10)	1.5	9.25	7	1
A2	(2, 5)	5.5	4.75	2	3
A3	(8, 4)	10.5	2.75	7	2
A4	(5, 8)	3.5	4.25	8	1
A5	(7, 5)	8.5	0.75	7	2
A6	(6, 4)	8.5	1.75	5	2
A7	(1, 2)	9.5	8.75	2	3
A8	(4, 9)	1.5	6.25	8	1

Iteration 3 – Update Centroids

- For Cluster 1,

$$\left(\frac{2+5+4}{3}, \frac{10+8+9}{3}\right) = (3.6,9)$$

- For Cluster 2,

$$\left(\frac{8+7+6}{3}, \frac{4+5+4}{3}\right) = (7,4.3)$$

- For Cluster 3,

$$\left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5,3.5)$$

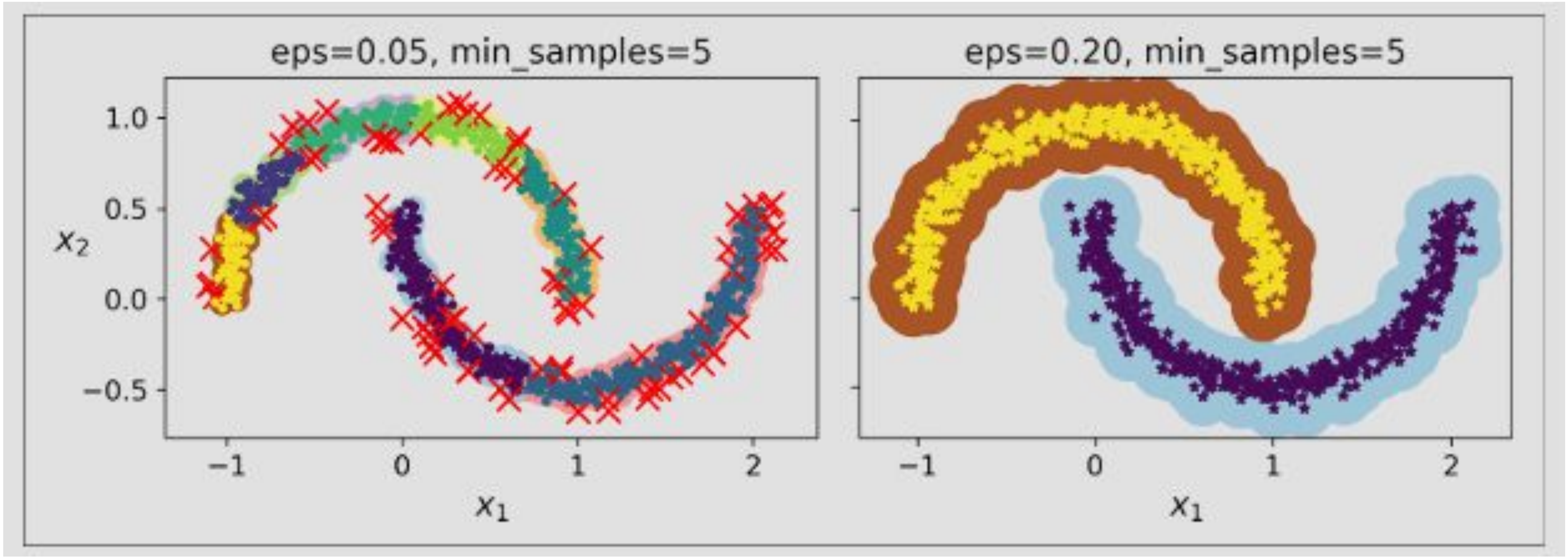
- Now cluster centers are:
    - $\boldsymbol{m1 = (3.6, 9)}$
    - $\boldsymbol{m2 = (7, 4.3)}$
    - $\boldsymbol{m3 = (1.5, 3.5)}$
- Repeat the process until the cluster assignments no longer change.

DBSCAN

- **Density-Based Spatial Clustering of Applications with Noise**
- Does not require pre-specifying number of clusters
- Can find arbitrarily shaped clusters
- Identifies outliers as noise

DBSCAN Algorithm

- For each instance:
  - The algorithm counts how many instances are located within a small distance $\boldsymbol{epsilon}$ $(\boldsymbol{\varepsilon})$ from it. This region is called the instance's $\boldsymbol{\varepsilon - neighborhood.}$
  - If an instance has at least $\boldsymbol{min\_samples}$ instances in its $\varepsilon - neighborhood$ (including itself), then it is considered a $\boldsymbol{core\ instance}$ (dense region).
  - All instances in the neighborhood of a $core\ instance$ belong to the same cluster.
  - Any instance that is not a $core\ instance$ and does not have one in its neighborhood is considered an $\boldsymbol{anomaly}$.



DBSCAN Parameters

- *epsilon* (ε): Maximum distance between two points to be considered neighbors
- *min_samples*: Minimum number of points to form a dense region

Advantages and Disadvantages of DBSCAN

- DBSCAN is robust to outliers
- It has just two hyperparameters (ϵ and $\min\_samples$)
- Computational complexity is roughly $O(mlogm)$, linear with regard to the number of instances.
- If the density varies significantly across the clusters, it can be impossible for it to capture all the clusters properly
- Require up to $O(m^2)$ memory if ϵ is large

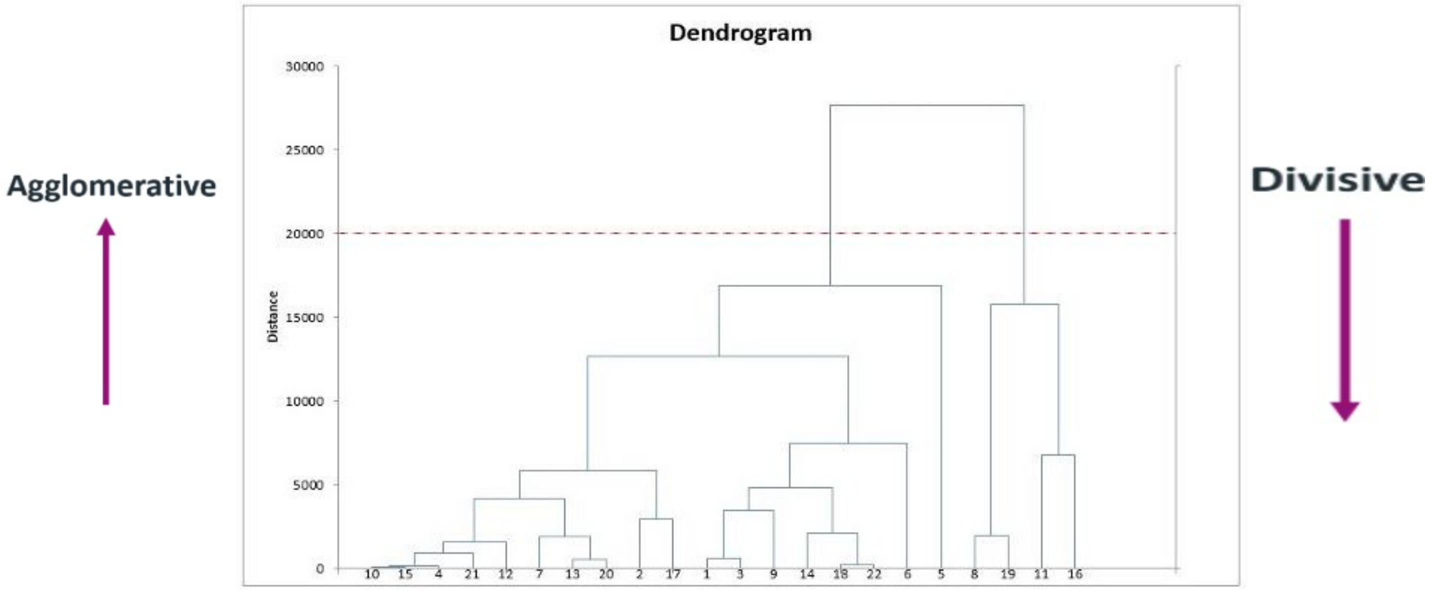
Agglomerative/Hierarchical Clustering

- Hierarchical clustering is similar to regular clustering, except that you’re aiming to build a hierarchy of clusters.
- This can be useful when you want flexibility in how many clusters you ultimately want.
- For example,
    - Imagine grouping items on an online marketplace like Amazon. On the homepage you’d want a few broad categories of items for simple navigation, but as you go into more specific shopping categories, you’d want increasing levels of granularity, i.e. more distinct clusters of items.

Steps for Hierarchical Clustering

1. Start with N clusters, one for each data point
2. Merge the two clusters that are closest to each other
   - Now you have $N - 1$ clusters
3. Recompute the distances between the clusters
   - Average-linkage clustering → compute the distance between two clusters → compute the average distance between all respective members.
4. Repeat steps 2 and 3 until you get one cluster of N data points. You get a tree (also known as a dendrogram).

If you want $k = 2$ clusters, you should draw a horizontal line around ‘distance=20000.’ You’ll get one cluster with data points 8, 9, 11, 16 and one cluster with the rest of the data points. In general, the number of clusters you get is the number of intersection points of your horizontal line with the vertical lines in the dendrogram.



Other Clustering Algorithms

- **BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)**
  - BIRCH was designed specifically for very large datasets
  - During training, it builds a tree structure containing just enough information to quickly assign each new instance to a cluster, without having to store all the instances in the tree.
- **Mean-Shift**
  - This algorithm starts by placing a circle centered on each instance; then for each circle it computes the mean of all the instances located within it, and it shifts the circle so that it is centered on the mean.
  - Its computational complexity is $O(m^2)$, not suited for large datasets.

Other Clustering Algorithms

- **Affinity propagation**
  - This algorithm uses a voting system, where instances vote for similar instances to be their representatives, and once the algorithm converges, each representative and its voters form a cluster.
  - Its computational complexity is $O(m^2)$, not suited for large datasets.
- **Spectral clustering**
  - This algorithm takes a similarity matrix between the instances and creates a low dimensional embedding from it (i.e., it reduces its dimensionality), then it uses another clustering algorithm in this low-dimensional space (K-Means.)
  - It does not scale well to large numbers of instances, and it does not behave well when the clusters have very different sizes.

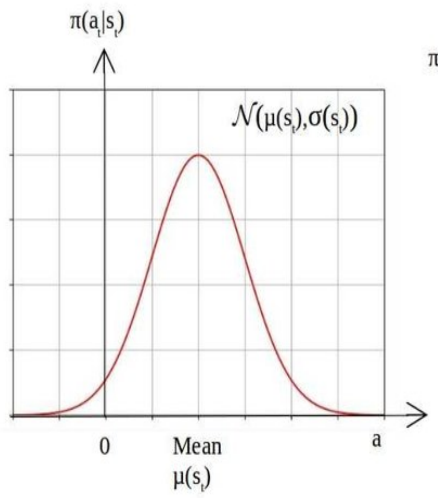
Gaussian Mixtures Models (GMM)

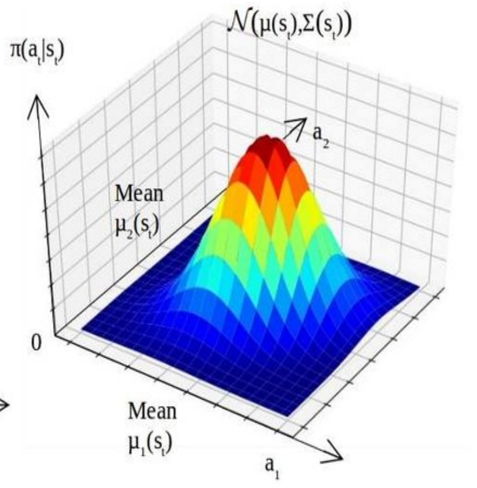
- Assumes data generated from a mixture of several Gaussian distributions
- **Soft clustering:**
  - Each instance belongs to all clusters with certain probabilities
- Flexible compared to K-means as it captures elliptical clusters

Motivation

- Real-world data is often heterogeneous.
- Single Gaussian may not capture complex distributions.
- GMM provides a framework to represent data as a combination of multiple Gaussians.

Gaussian Distribution Recap





•

- Formula:

$$N(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

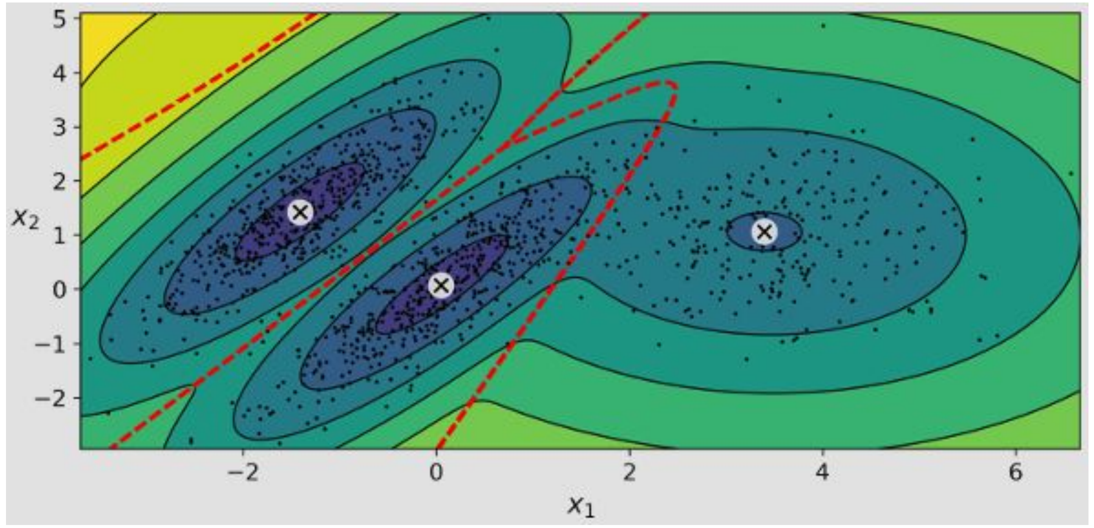
- Parameters:
  - Mean vector μ → center of distribution
  - Covariance matrix Σ → shape & orientation

What is a Mixture Model?

- Mixture model = weighted sum of component distributions.
- GMM formula:

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$$

- K = number of Gaussian components (clusters)
- π_k = mixing weight of component k ($\Sigma \pi k = 1$)
- μ_k = mean vector of component k
- Σ_k = covariance matrix of component k
- $N(x|\mu_k, \Sigma_k)$ = Gaussian distribution



Expectation-Maximization (EM) Algorithm

-

- GMM is trained using **EM Algorithm**:
  - Initialize parameters (μ, Σ, π).
  - **E-step:** Compute responsibilities (posterior probabilities).
  - **M-step:** Update parameters using responsibilities.
  - Repeat until convergence.

Choosing Number of Components (K)

- Too small → underfitting.
- Too large → overfitting.
- Use model selection criteria:
    - Akaike Information Criterion (AIC)
    - Bayesian Information Criterion (BIC)
- Elbow method or visualization can also help.

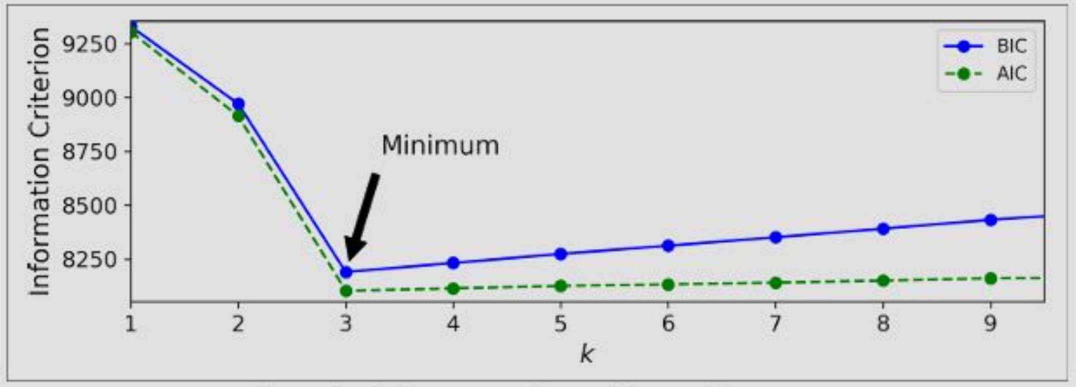


Figure 9-21. AIC and BIC for different numbers of clusters k

Advantages & Limitations

- Handles elliptical clusters.
- Provides probabilities (soft clustering).
- Flexible covariance structures.
- Computationally expensive.
- Sensitive to initialization.
- Struggles with high-dimensional data.

Association Analysis – Association Rule Mining

-
- Finding the relationships between variables (data items) in the large database.
- It determines the set of items that occurs together in the dataset.
- Association rule makes marketing strategy more effective.
    - Market Basket Analysis: People who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item.

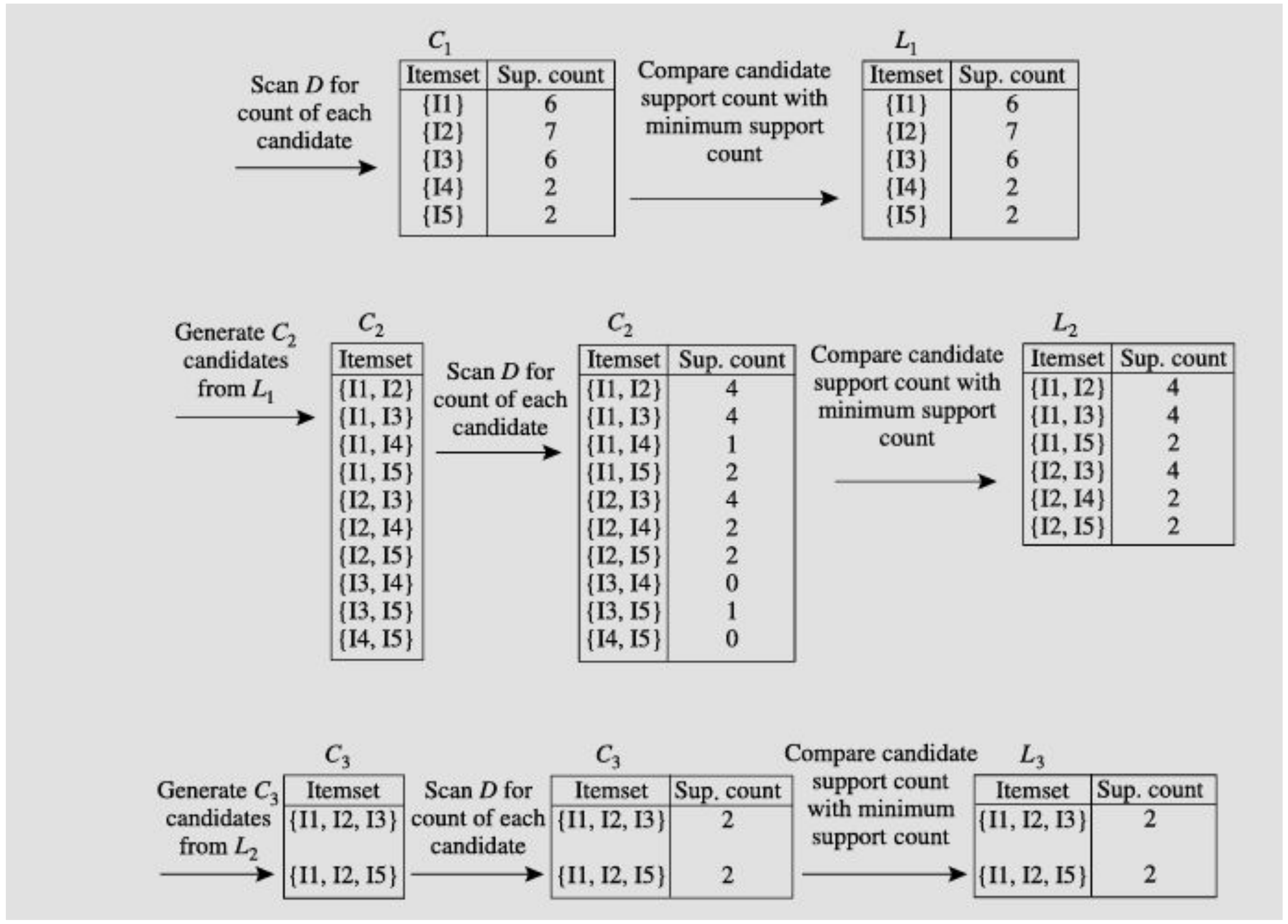
Apriori Algorithm

- The Apriori Algorithm is a breadth-first search-based algorithm which calculates the support between items.
- This support basically maps the dependency of one data item with another which can help us understand what data item influences the possibility of something happening to the other data item.
- Ex. Transaction data for an *AllElectronics*

TID	Items
T100	{I1, I2, I5}
T200	{I2, I4}
T300	{I2, I3}
T400	{I1, I2, I4}
T500	{I1, I3}
T600	{I2, I3}
T700	{I1, I3}
T800	{I1, I2, I3, I5}
T900	{I1, I2, I3}

Support Count = 2

The corresponding relative support is $2/9 = 22\%$


Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C_2 candidates from L_1
Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C_3 candidates from L_2
Scan D for count of each candidate
Compare candidate support count with minimum support count

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Generating Association Rules from Frequent Itemsets

-

- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule $s \Longrightarrow (l - s)$ if $\frac{supportcount(l)}{supportcount(s)} \geq \min conf$, where $\min conf$ is the minimum confidence threshold.

$$confidence(A \Longrightarrow B) = P\left(\frac{B}{A}\right) = \frac{supportcount(A \cup B)}{supportcount(A)}$$

- The data contain frequent itemset $X = \{I1, I2, I5\}$
- What are the association rules that can be generated from X?

- The nonempty subsets of X are: $\{I1, I2\}, \{I1, I5\}, \{I2, I5\}, \{I1\}, \{I2\}$, and $\{I5\}$
- The resulting association rules are as shown below, each listed with its confidence:
  - $\{I1, I2\} \Longrightarrow I5, confidence = \frac{2}{4} = 50\%$
  - $\{I1, I5\} \Longrightarrow I2, confidence = \frac{2}{2} = 100\%$
  - $\{I2, I5\} \Longrightarrow I1, confidence = \frac{2}{2} = 100\%$
  - $I1 \Longrightarrow \{I2, I5\}, confidence = \frac{2}{6} = 33\%$
  - $I2 \Longrightarrow \{I1, I5\}, confidence = \frac{2}{7} = 29\%$
  - $I5 \Longrightarrow \{I1, I2\}, confidence = \frac{2}{2} = 100\%$
- If the minimum confidence threshold is, 70%, then only the second, third, and last rules are output, because these are the only ones generated that are strong.

FP-Growth Algorithm

- The Frequency Pattern (FP) algorithm finds the count of the pattern that has been repeated, adds that to a table and then finds the most plausible item and sets that as the root of the tree.
- Other data items are then added into the tree and the support is calculated. If that particular branch fails to meet the threshold of the support, it is pruned.
- Once all the iterations are completed, a tree with the root to the item will be created which are then used to make the rules of the association.

Problem Setup

Transaction id	Items
T1	{a, b, c}
T2	{b, c, d, e}
T3	{a, b, d}
T4	{a, c, e}
T5	{a, b, c, e}
T6	{b, e}
T7	{a, b, c, d}
T8	{a, b, e}
T9	{b, c}

- Minimum support (absolute) = **3**

Count single-item supports

- All items meet min support (≥3). Order frequent items in **descending support** (tie-break lexicographic where needed):
- **Order:** {b:8, a:6, c:6, e:5, d:3}

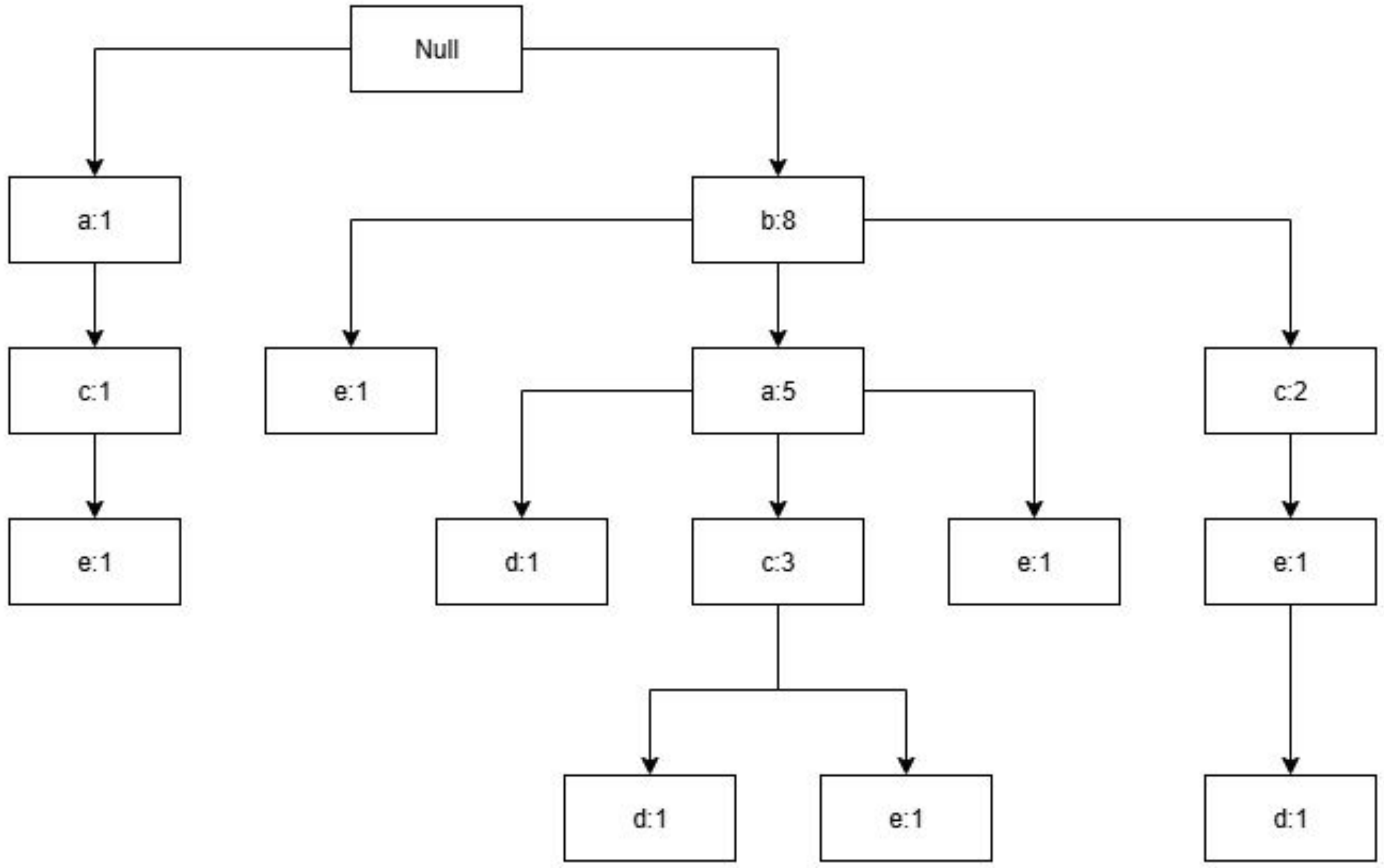
Item	Support
a	6
b	8
c	6
d	3
e	5

Sort Each Transaction

- **Frequent Pattern Set:** {b:8, a:6, c:6, e:5, d:3}

Transaction id	Items	Ordered-Items
T1	{a, b, c}	{b, a, c}
T2	{b, c, d, e}	{b, c, e, d}
T3	{a, b, d}	{b, a, d}
T4	{a, c, e}	{a, c, e}
T5	{a, b, c, e}	{b, a, c, e}
T6	{b, e}	{b, e}
T7	{a, b, c, d}	{b, a, c, d}
T8	{a, b, e}	{b, a, e}
T9	{b, c}	{b, c}

Construct FP-Tree



Mine FP-Tree

Items	Conditional Pattern Base	Conditional FP-Tree	Frequent Patterns Generated
d	{{b,a:1}, {b,a,c:1}, {b,c,e:1}}	<b:3>	{b,d:3}
e	{{a,c:1}, {b:1}, {b,a,c:1}, {b,a:1}, {b,c:1}}	<b:4,a:3,c:3>	{b,e:4},{a,e:3},{c,e:3}
c	{{a:1}, {b:2}, {b,a:3}}	<b:5,a:4>, <b,a:3>	{b,c:5},{a,c:4},{b,a,c:3}
a	{b:5}	<b:5>	{b,a:5}
b	-	-	-

Generating Association Rules from Frequent Itemsets

•

$$confidence(A \Longrightarrow B) = P\left(\frac{B}{A}\right) = \frac{supportcount(A \cup B)}{supportcount(A)}$$

- The data contain frequent itemset $X = \{b, a, c\}$
- What are the association rules that can be generated from X?
- The nonempty subsets of X are: $\{b, a\}, \{b, c\}, \{a, c\}, \{b\}, \{a\}$, and $\{c\}$

•

- The resulting association rules are as shown below, each listed with its confidence:
  - $\{b, a\} \Longrightarrow c, confidence = \frac{3}{5} = 60\%$
  - $\{b, c\} \Longrightarrow a, confidence = \frac{3}{5} = 60\%$
  - $\{a, c\} \Longrightarrow b, confidence = \frac{3}{4} = 75\%$
  - $b \Longrightarrow \{a, c\}, confidence = \frac{3}{8} = 37\%$
  - $a \Longrightarrow \{b, c\}, confidence = \frac{3}{6} = 50\%$
  - $c \Longrightarrow \{b, a\}, confidence = \frac{3}{6} = 50\%$
- If the minimum confidence threshold is, 70%, then only the third rules is output.