Unsupervised Learning

Author: Wenxiao Yang

Institute: Department of Mathematics, University of Illinois at Urbana-Champaign

All models are wrong, but some are useful.

Contents

Chapter 1 Clustering

General Goal of **Clustering Algorithm**:

- the "similarity" of the objects in the same cluster is maximized while
- the "similarity" of objects in different clusters is minimized.

Definition 1.1

*For a given set of objects $V = \{x_1, x_2, ..., x_m\}$, we call a **cluster** $\mathbf{S_k}$ a subset of these objects, and we call a **clustering** the set of all K clusters $\{\mathbf{S_1}, \mathbf{S_2}, ..., \mathbf{S_k}\}$.* ♣

Example 1.1 Clustering of $\{x_1, x_2, x_3, x_4\}$: (1). $\{\{x_1, x_3\}, \{x_2, x_4\}\}$; (2). $\{\{x_1, x_3\}, \{x_1, x_2, x_4\}\}$; (3). $\{\{x_3\}, \{x_2, x_4\}\}$.

1.1 K-Means

1.1.1 K-Means Clustering Optimization Problem

1. **Input:**

   Desired number of clusters (ex: $K = 3$)

   Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...\vec{x}_{in})$ has n numerical attributes. (We can also think of X as being an $m \times n$ matrix $X_{m \times n}$.)

2. **Goal of K-Means:**

   Out of all possible clusterings of $\{S_1, S_2, ..., S_K\}$ with K clusters that can be made from the m objects in X, find the optimal clustering $\{S_1^*, S_2^*, ..., S_K^*\}$ that minimizes the sum of the "distance" of each object and the centroid (the mean of the cluster that object is assigned to).

   Technically, we can write this as an optimization problem

$$\{S_1^*, S_2^*, ..., S_K^*\} = \underset{S_1, S_2, ..., S_K}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{x \in S_k} \|x - \mu_k\|^2$$

$$\text{Optimal Inertia} = \min_{S_1, S_2, ..., S_K} \sum_{k=1}^{K} \sum_{x \in S_k} \|x - \mu_k\|^2$$

   Inertia measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster. A good model is one with low inertia and a low number of clusters (K).

Find the clustering $\{S_1^*, S_2^*, ..., S_K^*\}$ that provides a global minimum is **NP-hard**.

We use a heuristic algorithm to find a local minimum is good enough.

1.1.2 Lloyd's Algorithm

1. **Input:**

   Desired number of clusters (ex: $K = 3$)

   Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{in})$ has n numerical attributes. (We can also think of X as being an $m \times n$ matrix $X_{m \times n}$.)

2. **Algorithm:**

   - **Step 1: Centroid Initialization Step**

     Randomly select K centroids $\{\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_K\}$, where $\vec{\mu}_k = (\mu_{k1}, \mu_{k2}, ..., \mu_{kn})$

   - **Step 2: Cluster Assignment Step**

     Assign each object x_i in the dataset to it's closest centroid (specifically the *smallest squared euclidean distance*)

   - **Step 3: Centroid Update Step**

     Find the mean of each cluster created in step 2. These means are now the new centroids.

   - **Step 4: Stopping Criterion**

     If the old centroids and the new centroids are the same, stop the algorithm. Otherwise, go back to step 2.

3. **Output:** Clustering with K clusters $\{V_1, V_2, ..., V_K\}$.

Lloyd's algorithm is known as a **non-deterministic** algorithm because, even with the same input, it can exhibit different behaviors on different runs.

1.1.3 Benefits and Drawbacks

Benefits

- Fast algorithm.
- Computationally efficient.
- It scales well as the number of objects or attributes grows really large. (However, k-means is not great for "big data".)
- One of the easiest to understand.

Drawbacks

- Only works well with some types of data.

  The K-means algorithm works best for data when "the underlying clustering" of the data has the following properties:

  (1). Each cluster has roughly the same number of objects;

  (2). The clusters are spherical;

  (3). The clusters have the same sparsity;

  (4). There is good separation between the clusters;

  (5). You know the right number of clusters to ask for;

  (6). Attributes are numerical (non-categorical);

  (7). Data does not have a lot of noise or outliers.

  (**Caveat:** Just because some of these assumptions are not met does not mean necessarily the algorithm will perform worse.)

- Need to know the "right" number of clusters to ask for in advance. (We use k-means elbow plot method)
- It is a non-deterministic algorithm.

1.1.4 Elbow Method

Elbow Plot

1. For $k = 1$ to K:

   [a] Cluster the data several times into k clusters.

   [b] Calculate the average inertia of these resulting clusterings.

2. Plot "k vs. average inertia".

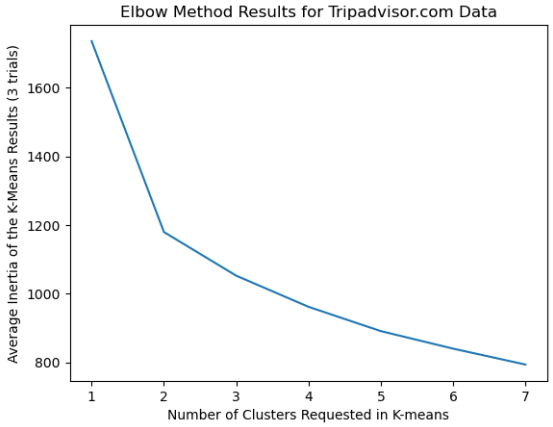


Figure 1.1: Elbow Plot Example

Interpretation of Elbow Plot

1. If there is not a dramatic elbow, then this suggests that either:
    1. The dataset is not clusterable or
    2. K-means is not a suitable algorithm for detecting the underlying clusters.
2. If there is a dramatic elbow, then this suggests that:
    1. There is a clustering structure and
    2. The k-means clustering algorithm is suggesting that there are about K clusters where the plot levels off.

In the example of the figure

1. We see a somewhat dramatic elbow in the plot. This suggests that there is some clustering structure in the dataset and that k-means is capable of identifying some clustering structure.

2. We see that that plot levels off dramatically at k=2 clusters. So this suggests that asking the k-means algorithm to return k=2 clusters will be the most insightful.

1.2 Types of Clusters Definitions

As we know the K-means algorithm can only work well with data that fulfills specific properties, we define some common **types of clusters** that could be considered in a numerical dataset to help introduce our new algorithms.

Definition 1.2 (Well-Separated Cluster)

*A **well-separated cluster** defines a cluster only when the data contains natural clusters that are far apart from each other. (This definition is vague in how far apart do clusters have to be.)*

♣

Why K-means may not work well?: Well-Separated Cluster can be non-spherical.

Definition 1.3 (Density-Based Cluster)

*A **density-based cluster** defines a cluster as a dense region of objects that is surrounded by a region of lower density. (This definition is vague in how dense it needs to be considered a cluster.)*

♣

Why K-means may not work well?: Density-Based Cluster can have noise.

Definition 1.4 (Graph-Based Cluster)

***Graph-based cluster** is a group of objects that are connected to one another, but have no connection to objects outside the group. (This definition is vague in how do we decide objects are connected.)*

♣

Why K-means may not work well?: Graph-based cluster can be non-spherical and not well separated.

Definition 1.5 (Contiguity-Based Cluster (a type of graph-based cluster definition))

*In **contiguity-based cluster** (a type of graph-based cluster definition), two objects are connected only if they are within a specified distance of one another.*

♣

Types of contiguity-based clustering algorithms: spectral clustering.

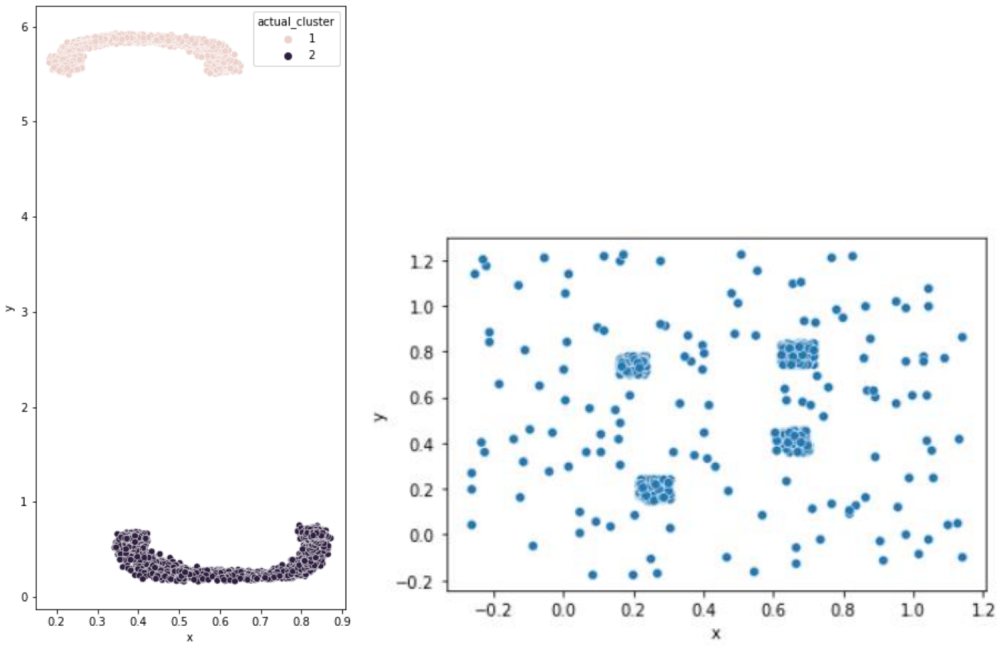


Figure 1.2: (1). Well-Separated Cluster; (2). Density-Based Cluster

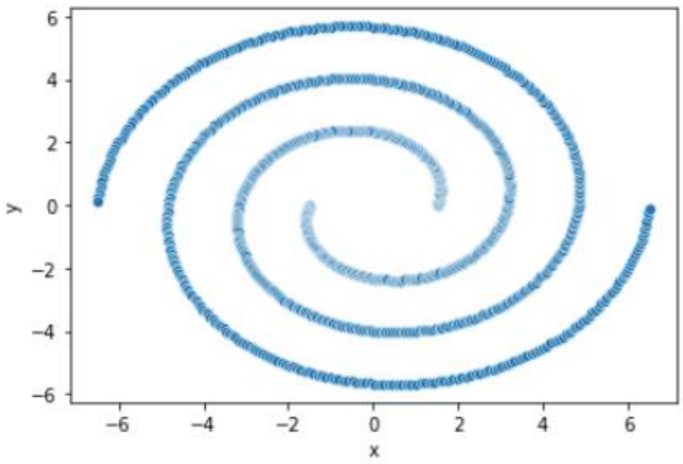


Figure 1.3: Contiguity-Based Cluster

Definition 1.6 (Prototype-Based Cluster)

*A **prototype-based cluster** defines a cluster as a set of objects in which each object is closer (or more similar) to the prototype (e.g. mean, median) that defines the cluster than to the prototype of any other cluster.*

♣

Why K-means may not work well?: Prototype-Based Cluster may be not well-separated and have outliers.

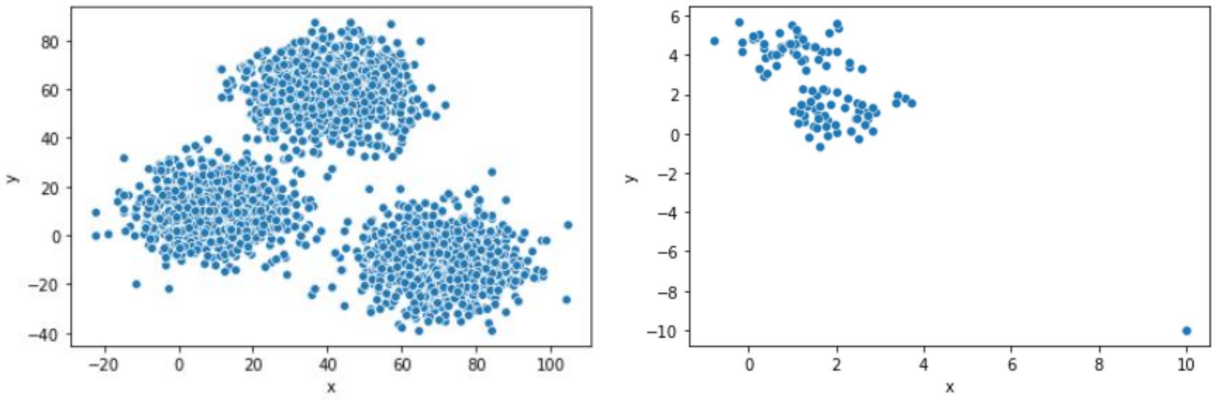


Figure 1.4: Prototype-Based Cluster

Types of prototype-based clustering algorithms:

- **K-means**: Prototype is the mean of the cluster.
- **K-median**: Prototype is the median of the cluster.

1.3 K-Medians

1.3.1 K-Medians Clustering Optimization Problem

Goal of K-Means:

Out of all possible clustering of $\{S_1, S_2, ..., S_K\}$ with K clusters that can be made from the m objects in X, find the optimal clustering $\{S_1^*, S_2^*, ..., S_K^*\}$ that minimizes the sum of the Manhattan distances (i.e., L_1 distances) of each object and the centroid (the median of the cluster that object is assigned to).

Technically, we can write this as an optimization problem

$$\{S_1^*, S_2^*, ..., S_K^*\} = \underset{S_1, S_2, ..., S_K}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{x \in S_k} \|x - c_k\|_1$$

$$\text{Optimal Inertia} = \min_{S_1, S_2, ..., S_K} \sum_{k=1}^{K} \sum_{x \in S_k} \|x - c_k\|_1$$

1.3.2 K-Medians Heuristic Algorithm

1. **Input:**

   Desired number of clusters (ex: $K = 3$)

   Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{in})$ has n numerical attributes. (We can also think of X as being an $m \times n$ matrix $X_{m \times n}$.)

2. **Algorithm:**

   - **Step 1: Centroid Initialization Step**

     Randomly select K centroids $\{\vec{c}_1, \vec{c}_2, ..., \vec{c}_K\}$, where $\vec{c}_k = (c_{k1}, c_{k2}, ..., c_{kn})$

   - **Step 2: Cluster Assignment Step**

     Assign each object x_i in the dataset to its closest centroid (specifically the *smallest Manhattan distance*)

   - **Step 3: Centroid Update Step**

     Find the median of each cluster created in step 2. These medians are now the new centroids.

   - **Step 4: Stopping Criterion**

     If the old centroids and the new centroids are the same, stop the algorithm. Otherwise, go back to step 2.

3. **Output:** Clustering with K clusters $\{V_1, V_2, ..., V_K\}$.

1.4 K-Medoids

Definition 1.7 (Medoid)

*In the context of clustering, we define a **medoid** as an actual object in a cluster whose sum of distance to all the objects in the cluster is minimal.*

♣

1.4.1 K-Medoids Clustering Optimization Problem

Goal of K-Medoids:

Out of all possible clustering of $\{S_1, S_2, ..., S_K\}$ with K clusters that can be made from the m objects in X, find the optimal clustering $\{S_1^*, S_2^*, ..., S_K^*\}$ that minimizes the sum of the distances (any distance metric) of each object and the centroid (the medoid of the cluster that object is assigned to).

Technically, we can write this as an optimization problem

$$\{S_1^*, S_2^*, ..., S_K^*\} = \underset{S_1,S_2,...,S_K}{\text{argmin}} \sum_{k=1}^{K} \sum_{x \in S_k} \text{dist}(x, c_k)$$

$$\text{Optimal Inertia} = \min_{S_1,S_2,...,S_K} \sum_{k=1}^{K} \sum_{x \in S_k} \text{dist}(x, c_k)$$

1.4.2 K-Medoids Clustering Algorithm

1. **Input:**

   Desired number of clusters (ex: $K = 3$)

   Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{in})$ has n numerical attributes. (We can also think of X as being an $m \times n$ matrix $X_{m \times n}$.)

2. **Algorithm:**

   - **Step 1: Centroid Initialization Step**

     Randomly select K centroids $\{\vec{c}_1, \vec{c}_2, ..., \vec{c}_K\}$, where $\vec{c}_k = (c_{k1}, c_{k2}, ..., c_{kn})$

   - **Step 2: Cluster Assignment Step**

     Assign each object x_i in the dataset to its closest centroid (specifically the *using distance metric you've chosen*)

   - **Step 3: Centroid Update Step**

     Find the medoid of each cluster created in step 2. These medians are now the new centroids.

   - **Step 4: Stopping Criterion**

     If the old centroids and the new centroids are the same, stop the algorithm. Otherwise, go back to step 2.

3. **Output:** Clustering with K clusters $\{V_1, V_2, ..., V_K\}$.

1.4.3 K-Medoids vs. K-Means

Benefit of K-Medoids over K-Means:

1. The medoid is more robust to outliers.
2. Guaranteed to converge using any distance metric we want (K-means has to use squared euclidean distance).

Benefit of K-Means over K-Medoids:

K-Medoids is more computationally complex than k-means:

1. K-means: O (number of objects $\times$ number of attributes $\times$ number of clusters $\times$ number of iterations)
2. K-medoids: O $\left((\text{number of objects})^2 \times \text{number of attributes} \times \text{number of clusters} \times \text{number of iterations}\right)$

1.5 Types of Clustering Algorithms Results

1.5.1 Partitional vs. Hierarchical Clustering Results

Definition 1.8 (Partitional Clustering)

*We call a **partitional clustering** a division of the set of data objects into k subsets (clusters) such that each object is in exactly one subset.*

♣

Example 1.2 $\{1, 2, 8\}, \{3, 7\}, \{4, 5, 6\}$

Definition 1.9 (Hierarchical Clustering)

*In a **hierarchical clustering** we allow for clusters to have nested subclusters.*

*A hierarchical clustering is displayed as a set of nested clusters displayed as a **dendrogram** tree. The dendrogram can reflect which objects and clusters are closer to each other than others.*

♣

Example 1.3 $\{\{1, 3\}, \{\{\{6, 9\}, 10\}, \{11, 15\}\}\}, \{4, \{12, 19\}\}, \{\{\{2, 14\}, \{\{17, 20\}, 18\}\}, \{5, 8\}\}, \{7, \{13, 16\}\}$.

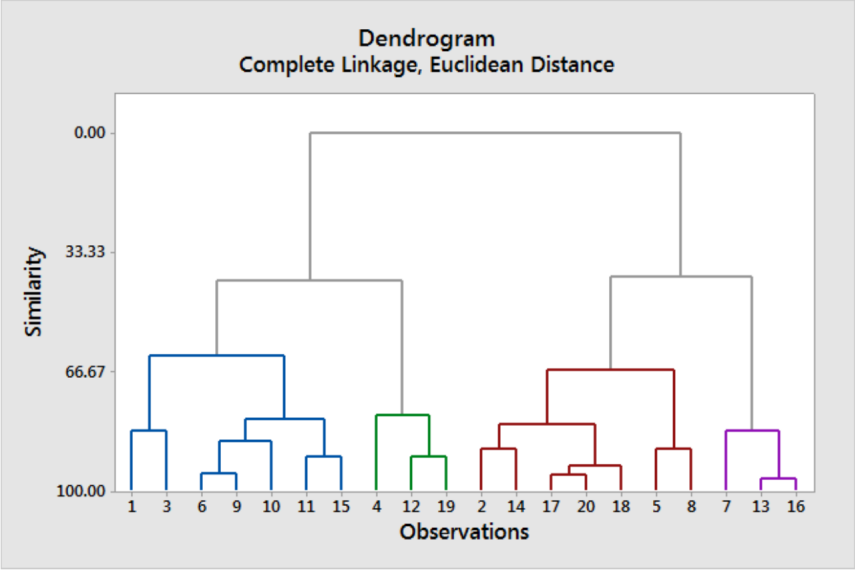


Figure 1.5: Hierarchical Clustering Example

1.5.2 Exclusive vs. Overlapping vs. Fuzzy Clustering Results

Definition 1.10

Exclusive Clustering *will assign an object to an exactly one cluster.*

♣

Example 1.4 $\{1, 3, 5\}, \{2, 4\}$.

Definition 1.11

Overlapping Clustering *can allow for an object to be assigned to more than one cluster.*

♣

Example 1.5 $\{1, 3, 5\}, \{2, 4, 5\}$

Definition 1.12

In a ***Fuzzy Clustering*** *every object belongs to every cluster with a membership weight that is between* 0 *(absolutely doesn't belong to the cluster) to* 1 *(absolutely belongs).*

1. *Usually the sum of each object's weights must sum to* 1.
2. w_{ij} = *the probability that object* i *belongs to cluster* j.

♣

Chapter 2 Clustering Evaluation Metrics

2.1 Clusterability Evaluation Metric: Is the dataset clusterable?

Definition 2.1

*A dataset is **clusterable** if there exist some distinct groupings of observations in a dataset.* ♣

Then, how distinct do the observation need to be is a question.

2.1.1 Hopkin's Statistics

- **Input:** Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{in})$ has n numerical attributes.
- **How to calculate:**

  (1) Create a set of random artificial data point closest distances $\{u_1, u_2, ..., u_p\}$ as follows.

  a) Generate p random artificial data points $\{\vec{y}_1, \vec{y}_2, ..., \vec{y}_p\}$ distributed across the range of the dataset.

  b) For each random artificial data points $i = 1, ..., p$ calculate

  $$u_i = \min_{\vec{x} \in X} \text{dist}(\vec{y}_i, \vec{x})$$

  (2) Create a set of random actual data point closest distances $\{w_1, w_2, ..., w_p\}$ as follows.

  a) Random select p actual points $\{\vec{z}_1, \vec{z}_2, ..., \vec{z}_p\}$ from the dataset.

  b) For each randomly selected actual points $i = 1, ..., p$ calculate

  $$w_i = \min_{\vec{x} \in X} \text{dist}(\vec{z}_i, \vec{x})$$

  (3)

  $$\textbf{Hopkins Statistic} = \frac{\sum_{i=1}^{p} w_i}{\sum_{i=1}^{p} w_i + \sum_{i=1}^{p} u_i}$$

- **How to interpret:** The dataset is clusterable if the Hopkins Statistic close to 0 and is not clusterable if the Hopkins Statistic close to 0.5.
- **Intuition:**

- **Additional Tips and Information:** $p = 10\% \times$ (the number of observations in the dataset); Hopkins Statistic is non-deterministic evaluation metric.

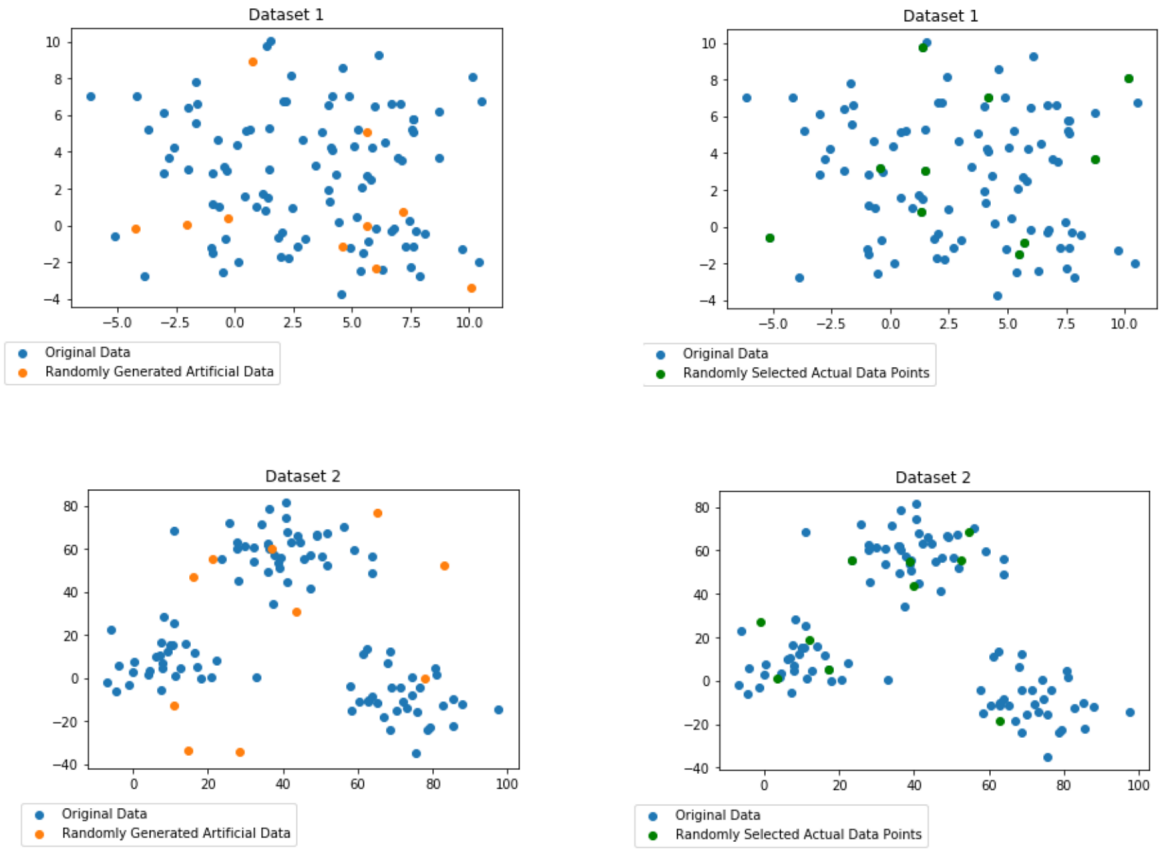


Figure 2.1: When Hopkins Statistic works well and not well

2.2 Unsupervised Clustering Evaluation Metrics: How cohesive and well separated are the clusters in the clustering?

2.2.1 Definition

Definition 2.2

Unsupervised clustering evaluation metrics *evaluate the goodness of the clustering without using pre-assigned class labels.*

♣

Types of Unsupervised Clustering Evaluation Metrics:

1. **Cohesion** measures how closely related the objects in a cluster are.
2. **Separation** measures how distinct or well-separated a cluster is from other clusters.
3. **Validity of a clustering** can be expressed as some function of **cohesion** and **separation** of all the clusters in a clustering.

2.2.2 Graph-based view of cohesion and separation for a clustering

A graph-based view of calculating cohesion and separation for a clustering involves first creating a **proximity matrix** (graph) of the objects in the dataset, that measures the "proximity" of each pair of objects in the dataset.

		Cluster 1			Cluster 2	
		1	2	3	4	5
Cluster 1	1	1	0.9	0.95	0.2	0.1
	2	0.9	1	0.8	0.15	0.05
	3	0.95	0.8	1	0.02	0.03
Cluster 2	4	0.2	0.15	0.02	1	0.8
	5	0.1	0.05	0.03	0.8	1

Figure 2.2: Proximity Matrix

Different ways to measure proximity:

1. **Similarity metric measure of proximity:** The more similar two objects are the lower the proximity measure is. e.g. Euclidean distance.
2. **Dissimilarity metric measure of proximity:** The more similar two objects are the higher this proximity measure is. e.g. The number of attribute agreements between two categorical objects.

Cohesion of a graph-based cluster is the sum of proximities between all pairs of points within the same cluster.

$$\text{cohesion}(C_i) = \sum_{\vec{x} \in C_i, \vec{y} \in C_i} \text{proximity}(\vec{x}, \vec{y})$$

Separation of a graph-based cluster is the sum of proximities between all pairs of points in the two different clusters.

$$\text{separation}(C_i, C_j) = \sum_{\vec{x} \in C_i, \vec{y} \in C_j} \text{proximity}(\vec{x}, \vec{y})$$

2.2.3 Silhouette Coefficients (Scores)

1. **Cohesion Metric:** Measure of how "well assigned" object x_i is to cluster C_k:

$$a_i = \frac{1}{|C_k| - 1} \sum_{\vec{x}_j \in C_k, \vec{x}_i \neq \vec{x}_j} \text{dist}(\vec{x}_i, \vec{x}_j)$$

2. **Separation Metric:** Find the Average Distance of object x_i to it's "neighboring cluster."

$$b_i = \min_{k' \neq k} \frac{1}{|C_{k'}|} \sum_{\vec{x}_j \in C_{k'}} \text{dist}(\vec{x}_i, \vec{x}_j)$$

3. **Silhouette Coefficient (Score)** of x_i

$$s_i = \begin{cases} \frac{b_i - a_i}{\max\{a_i, b_i\}}, & \text{if } |C_i| > 1 \\ 0, & \text{if } |C_i| = 1 \end{cases}$$

The silhouette of a cluster visualizes the silhouette values s_i of all the points in it in the decreasing order. A silhouette plot shows the silhouettes of all the clusters in random order. Additionally, it inserts blank spaces between consecutive clusters and can color them differently.

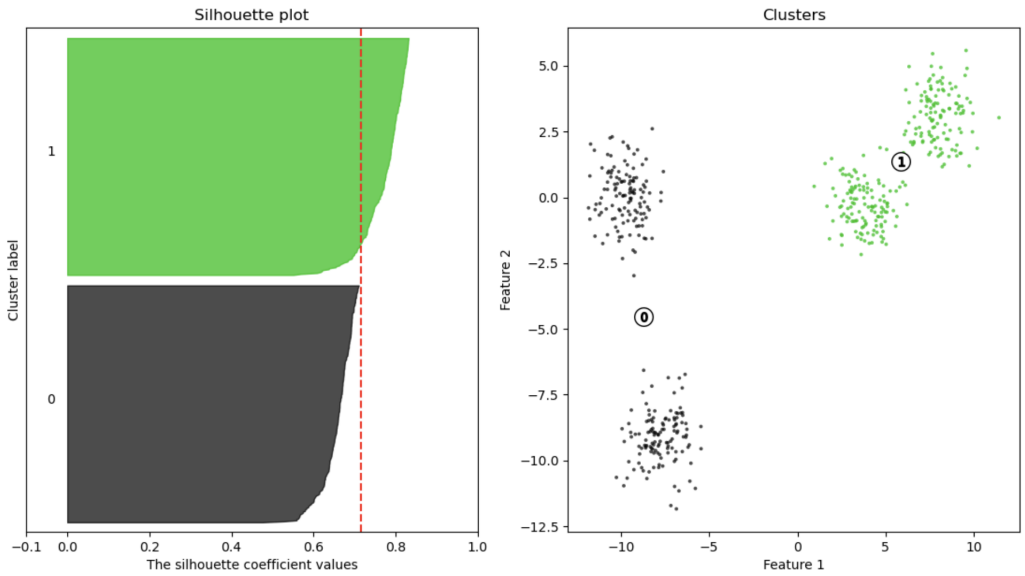


Figure 2.3: Silhouette Plots

1. **Silhouette Coefficient** s_i of object $\vec{x}_i$ is large and positive: the object is closer to objects in the cluster that it is assigned to than objects in other clusters.
2. **Silhouette Coefficient** s_i of object $\vec{x}_i$ is close to 0: the object is equally close to objects in the cluster that it is assigned to than objects in other clusters.
3. **Silhouette Coefficient** s_i of object $\vec{x}_i$ is large and negative: the object is further away from to objects in the cluster that it is assigned to than objects in other clusters.

Warning: Silhouette coefficients and plots (based off of Euclidean distances) will not be effective at assessing clustering separation and cohesion for all types of datasets. (e.g. the contiguity-based cluster.) We need to revise the distance metric.

2.2.4 Prototype-Based View of Cohesion and Separation for a Clustering

1. **Cohesion of a prototype-based cluster:** the sum of proximities between all points in a given cluster and the prototype of that cluster.

2. **Separation of a prototype-based cluster:** the proximity of the two cluster prototypes.

2.3 Cluster Number Evaluation Metrics: What is the 'correct' number of clusters?

2.3.1 General Elbow Plot Method

Definition 2.3 (General Elbow Plot Method)

When determining whether a clustering structure can be detected by a particular clustering algorithm (K-means, K-medians, K-medoids), we can use an elbow plot that plots the value of the objective function that we are trying to minimize of the clusterings found by that particular clustering algorithm with $k = 1, k = 2, ...$ *clusters respectively.*

♣

The **drawback** of this method is each of these methods are dependent on the ability of a clustering algorithm to detect the clustering structure.

2.3.2 Average Silhouette Score Plot Method

(1). **Creating an Average Silhouette Score Plot**

For $k = 1$ to K:

- Cluster the data several times into k clusters.
- Calculate the average of the average silhouette scores of these resulting clusterings.
- Then plot the average of these average silhouette scores for each k.

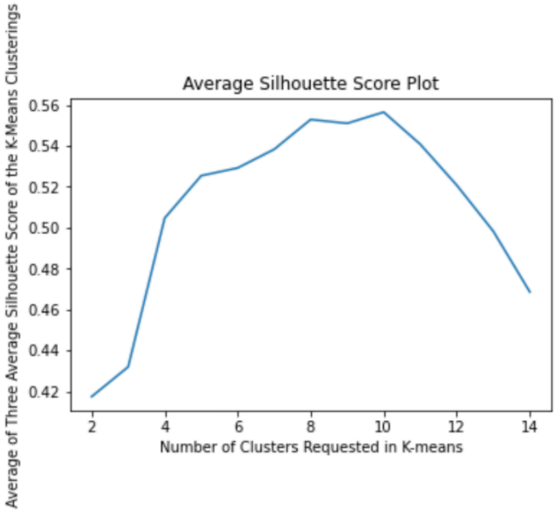


Figure 2.4: Average Silhouette Score Plot

(2). **How to interpret:** Choose the number of clusters with the highest average silhouette score.

(3). **Warning:** We need to make sure the distance metric we are using to measure the silhouette score with is a useful metric in measuring the cohesion and the separation of the clusters in the dataset.

For instance, when using the Euclidean distance to measure distance in the silhouette score, the average silhouette score is not as effective in measuring clustering cohesion and separation of **non-convex shapes**.

(4). **Benefit:** Using an average silhouette score plot do not assume using one particular algorithm/clustering problem to measure cluster performance.

2.4 Supervised Clustering Evaluation Metrics: How similar is the clustering to a set of (external) pre-assigned class labels?

Definition 2.4 (Supervised Clustering Evaluation Metrics)

Supervised clustering evaluation metrics *evaluate the clustering by using a set of pre-assigned class labels. This can be useful for examining the association between our pre-assigned cluster labels and the clustering structure identified by our clustering algorithms.*

♣

2.4.1 Rand Index and Jaccard Coefficient of Two Partitions

A partition of a set $\{x_1, x_2, ..., x_m\}$ is a collection of K non-empty subsets (i.e. clusters/classes) of the set such that every element of the set is in exactly one of the subsets (i.e. clusters/classes).

Definition 2.5 (Rand Index and Jaccard Coefficient)

$$\text{Rand Index (Statistic)} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

$$\text{Jaccard Coefficient} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

- f_{00} = *number of pairs of objects in the dataset that are* ***apart*** *in partition 1 and partition 2*
- f_{11} = *number of pairs of objects in the dataset that are* ***together*** *in partition 1 and partition 2*
- f_{01} = *number of pairs of objects in the dataset that are* ***apart*** *in partition 1 and* ***together*** *in partition 2*
- f_{10} = *number of pairs of objects in the dataset that are* ***together*** *in partition 1 and* ***apart*** *in partition 2*

♣

Interpretation and Ranges:

1. Rand Index (Statistic)$\in [0,1]$: 0 means all possible object pairs disagree between the two partitions; 1 means the partitions are exactly the same.
2. Jaccard Coefficient$\in [0,1]$: 0 means none of the possible object pairs that have at least one partition that has put them together are together in both partitions; 1 means all possible object pairs that have at least one partition that has put them together are together in both partitions.

2.4.2 Adjusted Rand Index

In Rand Index (Statistic), 0 means all possible object pairs disagree between the two partitions and 1 means clusterings are identical. However, we want a desired evaluation metric such that, 0 means random labeling independently of the number of clusters and samples and 1 means clusterings are identical

Definition 2.6 (Adjusted Rand Index)

$$\text{Adjusted Rand Index} = \frac{\text{rand index} - \text{expected rand index}}{\text{maximum rand index} - \text{expected rand index}}$$

♣

For two partitions $\{U_1, U_2, ..., U_K\}$ and $\{V_1, V_2, ..., V_{K'}\}$ of the same set of m objects, we can define:

1. $m_{k,k'}$ =number of objects in common in subset U_k and $V_{k'}$.
2. a_k =the total number objects in subset U_k.
3. $b_{k'}$ =the total number objects in subset $V_{k'}$.

$$\text{Adjusted Rand Index} = \frac{\sum_{k,k'} \binom{m_{k,k'}}{2} - \left[\sum_k \binom{a_k}{2} \sum_{k'} \binom{b_{k'}}{2}\right] / \binom{m}{2}}{\frac{1}{2}\left[\sum_k \binom{a_k}{2} + \sum_{k'} \binom{b_{k'}}{2}\right] - \left[\sum_k \binom{a_k}{2} \sum_{k'} \binom{b_{k'}}{2}\right] / \binom{m}{2}}$$

Adjusted rand index can actually be negative for two clusterings that have very low similarity.

2.5 Clustering Comparison Metrics: Which clustering is better for a given dataset?

1. **Inertias** of the two clusterings: Inertia should only be used to compare clustering generated from the **same dataset**.
2. **Average silhouette scores** of the two clusterings: We can use this metric to compare the cohesion and separation of clusterings (and they **could** be two clusterings of **different dataset**),

Chapter 3 Checking Clustering Conditions with t-SNE Plots

3.1 t-SNE: t-Distributed Stochastic Neighbor Embedding

We want to use t-SNE plots help us visualize some aspects of the underlying clustering structure of the data.

3.1.1 Goal

Goal: project multidimensional data onto a 2 or 3-dimensional plane while preserving **clustering structure** of the dataset.

We want to use t-SNE plots help us visualize some aspects of the underlying clustering structure of the data:

1. Whether there exists a **clustering structure** in the data.
2. Approximation of the **number of clusters**.
3. Approximation of the **cluster shapes**.
4. Approximate **number of objects** in each cluster.
5. Approximation of whether clusters are **separated** or not.
6. Approximation how any **pre-assigned class labels** associate with the underlying **clustering structure of the data**.
7. Approximation of how any **cluster labels (from a clustering algorithm)** associate with the **underlying clustering structure of the data suggested by the t-SNE algorithm**.

3.1.2 Input/Output for the Algorithm

Input:

- Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{in})$ has n numerical attributes.
- Number of Dimensions: Dimension of the data you want to project the data onto (usually 2).
- Number of Iterations: maximum number of iterations for the algorithm.

  How to select: (1). At least 200; (2). Automatically set to 1000 in Python (which tends to work for many, but not all datasets); (3). Keep iterating until you see a stable configuration of the shapes.
- Perplexity, which says (loosely) how to balance attention between local and global aspects of your data. The parameter is, in a sense, a guess about the number of close neighbors each point has.

  How to select: (1). Works best when $5 \leq$ perplexity ≤ 50; (2). Perplexity $<$ number of objects.

x	y
-37.556343	32.625168
-27.310349	44.866043
-39.212635	25.225029
-37.236549	50.355652
-54.262161	3.018528

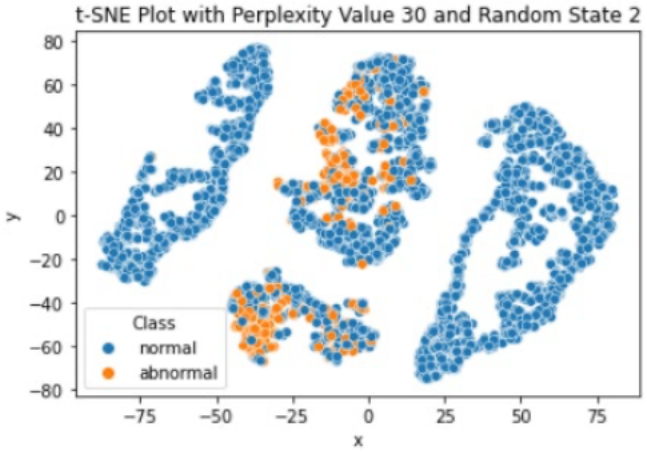


Figure 3.1: Example of Output

3.1.3 Main Idea of The Algorithm

1. **Optimization Problem:** Given an original high dimensional dataset, we need to solve an optimization for the optimal value of the decision variables that represent the low-dimensional projected coordinates for each observation in the original dataset.

   $\vec{x}_i = [x_{1,1}, ..., x_{1,n}] \Rightarrow \vec{y}_i = [y_{i,1}, y_{i,2}],\ i = 1, 2..., m.$

2. **Crates a Similarity matrix P for the Points in the Original Dataset:** Similarity between a given point $\vec{x}_i$ and another point $\vec{x}_j$ is a function of a normal distribution centered at $\vec{x}_i$ with a standard deviation σ_i that changes based on the point and how many "neighbors" you think each point in the dataset has. Each entry of the matrix is the **similarity** between i and j.

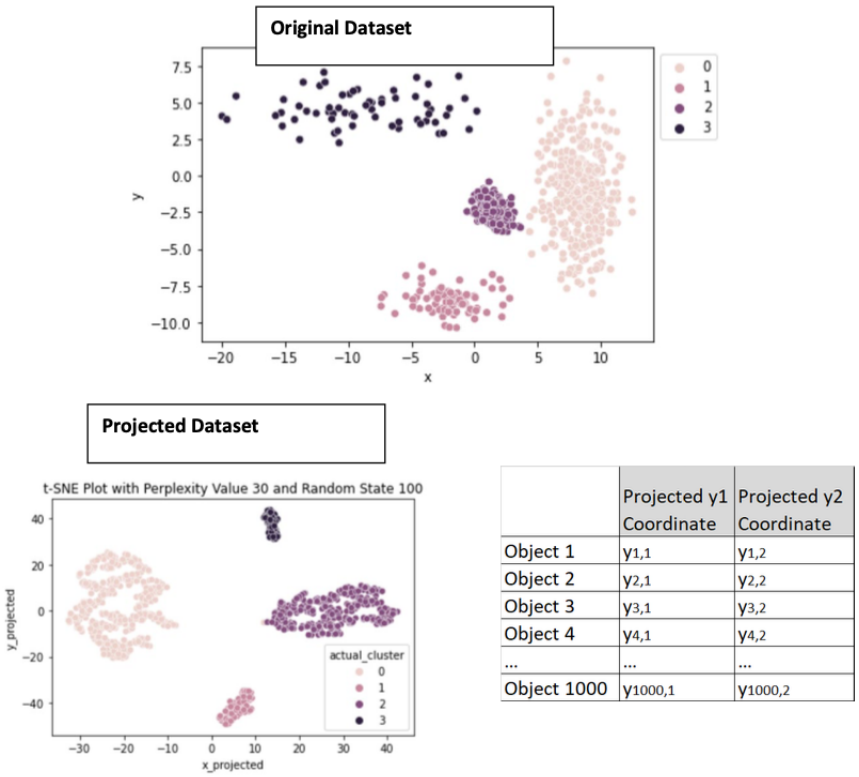
$$p_{ij} = \frac{1}{2m}\left(p_{j|i} + p_{i|j}\right)$$

$$\bullet\ p_{j|i} = \frac{\exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\vec{x}_i - \vec{x}_k\|^2}{2\sigma_i^2}\right)}$$

$$\bullet\ p_{i|j} = \frac{\exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma_j^2}\right)}{\sum_{k \neq j} \exp\left(-\frac{\|\vec{x}_j - \vec{x}_k\|^2}{2\sigma_j^2}\right)}$$

3. **Creates a Similarity Matrix Q for the Points (i.e., Decision Variables we are Trying to Solve for) in the Projected Dataset:** Similarity between a given point $\vec{y}_i$ and another point $\vec{y}_j$ is a function about t-distribution centered at $\vec{y}_i$.

$$q_{ij} = \frac{(1 + \|\vec{y}_i - \vec{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\vec{y}_k - \vec{y}_j\|^2)^{-1}}$$



	Projected y1 Coordinate	Projected y2 Coordinate
Object 1	$y_{1,1}$	$y_{1,2}$
Object 2	$y_{2,1}$	$y_{2,2}$
Object 3	$y_{3,1}$	$y_{3,2}$
Object 4	$y_{4,1}$	$y_{4,2}$
...	...	...
Object 1000	$y_{1000,1}$	$y_{1000,2}$

Figure 3.2: Optimization Problem

4. Solve the optimization problem:

$$\min_{\{\vec{y}_i\}_{i=1}^m} \text{dist}(P,Q) = D_{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

A heuristic solution to this optimization problem can be found via a Gradient Descent algorithm. Algorithm is heavy on both time and space resources $O(n^2)$. Computationally ineffective for datasets with more than 10000 observations.

Chapter 4 Hierarchical Clustering

4.1 Agglomerative and Divisive Hierarchical Clustering Algorithms

Hierarchical clustering algorithms allow us to display a series of nested clusterings, graphically displayed in a dendrogram.

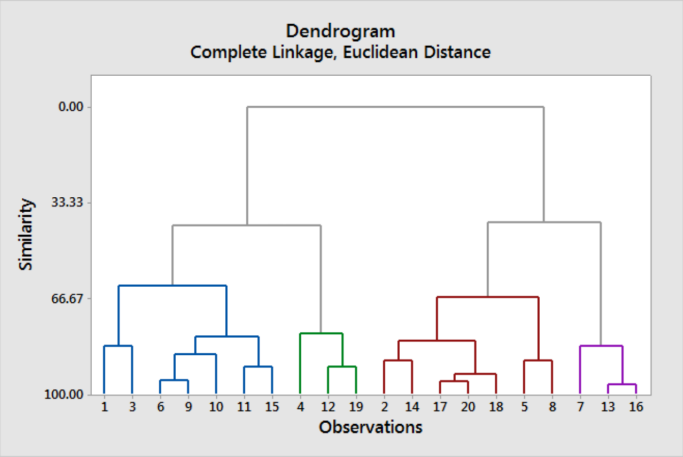


Figure 4.1: Hierarchical Clustering Example

1. An **agglomerative hierarchical clustering algorithm** starts with all objects **apart in singleton clusters** and then iteratively joins clusters together until all objects are in the same cluster.
2. A **divisive hierarchical clustering algorithm** starts with all objects **together** and then iteratively divides clusters together until all objects are apart in singleton clusters.

4.2 Agglomerative Hierarchical Clustering

4.2.1 General Algorithm for Agglomerative Hierarchical Clustering

1. Create an **initial proximity matrix of clusters** by calculating the proximity of all objects to each other.

2. **Repeat:**

   Merge the clusters in the current proximity matrix of the clusters that have the smallest proximity.

   Update the proximity matrix to reflect the proximity between the new merged cluster and the remaining clusters.

Proximity Matrix (of the Objects)					
	Object 0	Object 1	Object 2	Object 3	Object 4
Object 0	0.00	5.10	4.27	4.03	4.12
Object 1	5.10	0.00	1.12	3.91	5.00
Object 2	4.27	1.12	0.00	2.83	3.91
Object 3	4.03	3.91	2.83	0.00	1.12
Object 4	4.12	5.00	3.91	1.12	0.00

Figure 4.2: Proximity Matrix

3. **Until** only one cluster remains.

4.2.2 with Single Linkage

In **single linkage algorithms**, the **proximity between two clusters** is defined as the proximity between the two closest points in the two clusters.

Downsides of using hierarchical agglomerative clustering with single linkage proximity measure

1. Not as effective in identifying the ”main clusters” in datasets where the clusters are not well separated.
2. Sensitive to outliers and noise. It will in these cases sometimes detect the presence of outliers and noise at the expense of detecting the actual clustering structure.
3. Algorithm is more computationally complex than k-means.

Benefits of using hierarchical agglomerative clustering with single linkage proximity measure

1. You can use it as another means to detect outliers and noise in your dataset.
2. It can detect:

   non-convex clusters

   Clusters of different sparsities

   Clusters with different number of objects in them

4.2.3 with Complete Linkage

In **complete linkage algorithms**, the **proximity between two clusters** is defined as the proximity between the two furthest away points in the two clusters.

Downsides of using hierarchical agglomerative clustering with complete linkage proximity measure

1. Tends to split larger clusters, never enabling the algorithm to completely separate the main clusters that we're looking for.
2. Tends to favor spherical clusters
3. It tends not to detect noise or outliers.
4. Has a higher computational complexity than k-means.

Benefits of using hierarchical agglomerative clustering with complete linkage proximity measure

1. It is more robust to noise and outliers.
2. It does work better for clusters that are not well separated.

4.2.4 with Average Linkage

In **average linkage algorithms**, the **proximity between two clusters** is defined as the average distance between all pairs of points in the two clusters.

Why use average linkage agglomerative hierarchical clustering?

1. Average linkage strikes a "middle ground" solution in between the results of complete linkage and single linkage.
2. Less sensitive to noise and outliers than single linkage (but more so than complete linkage).
3. Less effective at identifying non-convex clusters (or those that are of different sparsities and sizes) than single linkage, but more effective than complete linkage.
4. Less effective at identifying non well separated clusters than complete linkage (but more effective than single linkage).
5. Less prone to split large clusters than complete linkage, but more prone than single linkage.

4.2.5 with Ward's Linkage

In **Ward's linkage algorithms**, the **proximity between two clusters** is defined as the change in objective function we get when merging the two clusters.
(Most common objective function value to use for Ward's linkage is **inertia**. The initial inertia value of the clustering with all singleton clusters is 0)

Benefits of using Ward's linkage in agglomerative hierarchical clustering.

1. Tends to identify spherical clusters really well

Drawbacks of using Ward's linkage in agglomerative hierarchical clustering.

1. Tends to favor the same types of clusters that k-means tends to favor (because they both use inertia to define what a cluster is).

   Spherical clusters.

   Well separated clusters.

   Clusters with equal sparsities and the same size.

   Know the number of clusters that you want

   Numerical attributes

2. Also sensitive to noise and outliers

4.3 Divisive Hierarchical Clustering

4.3.1 General Algorithm for Divisive Hierarchical Clustering

1. Split the dataset into 2 or more clusters.
2. **Repeat:** Select two or more clusters in the current clustering and split each of them into two or more clusters.
3. **Until** all objects are in singleton clusters (i.e. clusters of size one).

4.3.2 with the Bisecting k-Means Algorithm

Input:

- Dataset of m objects $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_m\}$, where each object $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{in})$ has n numerical attributes.
- Final Number of Desired Clusters in Final Clustering: K [if you don't want the algorithm to end with all objects in singleton clusters]

Algorithm:

1. **Initialize** the list of clusters to contain **the cluster consisting of all points**.
2. **Repeat:**

   [a.] **Remove a cluster** from the list of clusters

   (Algorithm specification: how do we select the cluster?)

[b.] for $i = 1$ to *number of trials* do: Bisect the selected cluster using k-means algorithm (asking for k=2 clusters).

[c.] Select the clustering from 2b with **the lowest inertia**.

[d.] Add the two clusters from the clustering selected in 2c to the list of clusters.

3. Until the list of clusters contains the K that we want.

Common ways to choose which cluster to split.

1. Choose the cluster with the most amount of points in it. (Benefit: you'll end up with more balanced clusters)
2. Choose the clusters with the highest inertia. (Benefit: you'll end up with a clustering with lower inertia)

Why use Bisecting k-means over k-means?

- Bisecting k-means FORCES there to be a nested clustering structure of your returned clusterings. Just using k-means multiple times (asking for $k = 2, 3, 4, \ldots$ clusters) does not force a nested cluster relationship.
- If k is large, then bisecting k-means can find the clustering with k clusters faster than k-means by itself.

Cons of Bisecting k-means

- Bisecting k-means tends to be more computationally complex (than k-means) when the desired cluster number k is small.