An Overview of Clustering: Combinatorial Algorithms, Hierarchical Clustering and Self-Organizing Maps

Beatrice Musizza

March 2024

Contents

1 Introduction

From enabling customer segmentation to unraveling complex biological relationships, clustering is increasingly emerging as a powerful tool.
This thesis will comprise two distinct parts. This initial part will be dedicated to theoretical foundation of clustering, presenting the related elements of Statistical Learning. The dissertation is based on *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, by Hastie, T., Tibshirani, R., & Friedman, J. [1].
Subsequently, a second part will be devoted to two practical applications of clustering method to real-world case studies.

In this first segment, starting with a brief description of Unsupervised Learning in Section 2, we will then discuss dissimilarity and present clustering algorithms (Section 3 and 4), with a particular focus on K-means (Section 5) and K-medoids (Section 6). We will then present Hierarchical Clustering (Section 7) and self-organizing maps (Section 8).

2 Unsupervised Learning

Consider a given set of predictor variables $X^T = (X_1, ..., X_m)$ and one or more response variables $Y = (Y_1, ..., Y_m)$. In this scenario, we refer to the process of predicting the response values as *supervised learning*. During supervised learning, the "student" is trained on a training sample, providing an answer $\hat{y}_i$ based on x_i in this training sample. The "teacher" corrects this answer by considering the error made, typically characterized by a loss function. In general, the objective is to minimize this loss. Numerous techniques address supervised learning, offering the capability to predict response values.

On the other hand, we have *unsupervised learning*. Unsupervised learning aims to identify properties solely related to X, without the presence of one or more response variables and without a "teacher" providing correct answers. In this domain we find the clustering techniques.
The aim of clustering is to find regions in the X-space that contain groups exhibiting similarities. The identification of these regions can determine if the probability density of X, $Pr(X)$, can be described by simpler densities distinguished for classes.

A notable characteristic of unsupervised learning is the heuristic judgment of the result's quality, due to the absence of a direct measure of success, such as the loss function used in supervised learning.

3 Clustering: Object Grouping, Dissimilarity Metrics and Attribute Considerations

3.1 Goals

The main goal of Cluster Algorithms is to group a collection of objects into 'clusters'. This has to be done in a way that allows objects in each cluster to be more related to one another than to objects belonging to different clusters. By doing this we can also aim to arrange clusters into a natural hierarchy using methods that involve successive grouping. Cluster Analysis can also be used to assess if the data can be considered as composed of different groups, with objects in each group presenting different properties.

3.2 The notion of dissimilarity

To pursue these goals, the notion of dissimilarity between individual objects is necessary.

Dissimilarity quantifies how different two objects are from each other. A high dissimilarity indicates that the objects are less alike.
The properties of a dissimilarity are:

1. Non-negativity: The dissimilarity between two objects is always non-negative:

$$d(x, y) \geq 0$$

2. Reflexivity: The dissimilarity between an object and itself is zero:

$$d(x, x) = 0$$

3. Symmetry: The dissimilarity between object x and object y is the same as the dissimilarity between y and x:

$$d(x, y) = d(y, x)$$

Moreover, a distance is a specific type of dissimilarity that presents also the following property:

4. Triangle Inequality: The dissimilarity between two objects and the sum of dissimilarities through an intermediate object is greater than or equal to the direct dissimilarity between the two objects:

$$d(x, z) \leq d(x, y) + d(y, z)$$

For dissimilarity measures, the triangle inequality may not necessarily hold. In other words, dissimilarity measures are more general and do not need to satisfy the triangle inequality.

Furthermore, an ultrametric distance is a special case of a distance. In this case also have:

5. Ultrametric Inequality: The the triangle inequality is strengthened to the following form, a stronger version of the triangle inequality:

$$d(x, z) \leq \max\{d(x, y), d(y, z)\}$$

In the context of clustering and dissimilarity measures, ultrametric distances have specific applications, especially in hierarchical clustering. In Section 7.1.1. we will present the cophonetic dissimilarity, which obey to the ultrametric inequality.

The choice of a dissimilarity metric is inherently subjective. This decision should be based considering the criteria by which two objects can be considered distant. We begin by describing how this difference is typically articulated.

3.2.1 Proximity Matrices

Sometimes, data can be already presented as differences. This type of data are represented by proximity matrices. In this case, the data are initially recorded as differences. For example, we can have researchers conducting surveys in which participants are asked to assess the dissimilarity between different objects.
In this scenario, we will have a matrix D of size $N \times N$, where N is the total number of objects. An entry of the matrix, $d_{ii'}$, will contain the measure of dissimilarity between object i and object i'. Most algorithms presume that the matrix has non-negative entries and zero diagonal elements. Considering that the difference between an element and itself should be zero, this is reasonable. Moreover, most algorithms require a symmetric dissimilarity matrice. The absence of this characteristic is linked to some form of inconsistency. If the original matrix D is not symmetric, it is possible to apply the transformation $(D + D^T)/2$, which will produce a symmetric matrix (with the same value on $d_{ii'}$ and $d_{i'i}$) with a value that is the average of the values that we had on $d_{ii'}$ and $d_{i'i}$.
This type of data are usually based on subjective judgments, and as such, they rarely represent distances in the strict sense, meaning that they usually do not satisfy the triangle inequality $d_{ii'} \leq d_{ik} + d_{i'k}$. For this reason, some care has to be taken: algorithms that assume distances , and so this property, can not be used with this type of data.

3.2.2 Dissimilarity Based on Attributes

Most of the time, we have objects x_{ij} for $i = (1, 2, ..., N)$ measured as a set of variables $j = (1, 2, ..., p)$. These j-variables are also called attributes and represent different features of the objects.
To measure the difference between objects when we have this representation, and not the proximity matrix, we have different options for the measure of difference. The choice of the measure for the degree-of-difference is based on the type of attribute and the context.

We start by describing alternatives based on the attribute type.

Quantitative variables: In this case, the most intuitive way to define dissimilarity is simply to measure the distance through a monotone-increasing function of their absolute differences:

$$d_j(x_{ij}, x_{i'j}) = l(|x_{ij} - x_{i'j}|)$$

Another common choice is to use the squared difference:

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

which places more emphasis on big differences.

Ordinal variables: Given this variables, we usually standardize them by replacing their M original values with:

$$\frac{i - \frac{1}{2}}{M}, i = 1, ..., M. \tag{3.1}$$

This transformation allows maintaining the reciprocal distance, but in a [0,1] range, and so makes ordinal values more comparable across different ordinal values.

Nominal variables: In this instance, we have to explicitly delineate all the degrees-of-difference between pairs. This is usually done by a symmetric $M \times M$ matrix, where M is the number of distinct values assumed by the nominal variable.

3.2.3 Weights of Attributes

When we have defined the p-individual dissimilarities $d_j(x_{ij}, x_{i'j})$, $j = (1, 2, ..., p)$, we have to combine them into an overall measure of dissimilarity between objects $D(x_i, x_{i'})$. The most common and intuitive choice is the weighted average:

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot d_j(x_{ij}, x_{i'j}); \quad \sum_{j=1}^{p} w_j = 1. \tag{3.2}$$

If we want all the attributes to have the same influence on the overall measure of dissimilarity $D(x_i, x_{i'})$, we will need to consider that the influence of the j-th attribute X_j depends on its average dissimilarity $\bar{d}_j$ across all pairs of objects. Indeed, if the j-th attribute has a high average dissimilarity, it means that, on average, the values of dissimilarities for that attribute are higher across pairs of observations. In this case, the j-th attribute will generally contribute more to the overall dissimilarity measure.
For this reason, to give all attributes equal influence, a suitable choice is setting $w_j \sim \frac{1}{\bar{d}_j}$.

Even if this approach may appear reasonable, it is important to be careful with it. If the value of the average dissimilarity of the j-th attribute ($\bar{d}_j$) is high, it could mean that, in the context of the problem domain, that attribute is significant in determining if two objects are different.
Flattening all the contributions of the attributes in order to make their influence the same, risks hiding some natural importance.

In conclusion, it is important to put the right care in considering the dissimilarities $d_j(x_{ij}, x_{i'j})$ and their weights w_j when working with clustering. This is because there is no simple generic prescription for accomplishing this task, but specifying an appropriate dissimilarity measure is more important in obtaining success, than the choice of the clustering algorithm.

3.2.4 Missing Values

If some observations have missing values for certain attributes, a common approach is to omit each attribute-observation pair $x_{ij}, x_{i'j}$, that has at least one missing value. If two observations have no measured attribute values in common, we can proceed by removing those observations from the analysis.

To avoid losing data, another approach is to replace the missing attribute entries by using the mean or median of each attribute over the nonmissing data.

Another method, applicable to certain categorical data, involves considering 'missing' as another possible categorical value. This is reasonable only if, in the domain, it makes sense to consider two objects that both have missing values as similar.

4 Combinatorial Algorithms

4.1 Categories of Clustering Algorithms

Clustering algorithms can be divided into three categories:

- Combinatorial algorithms: They assign each observation to a group or cluster without regard to a probability model describing the data.
- Mixture modeling: This technique assumes that the analyzed data distribution is a mixture of multiple probability distributions. These approaches are commonly referred to as 'model-based clustering'.
- Mode seekers: They do not assume a specific functional form for the underlying distribution, attempting to estimate distinct modes of the probability density function. This might involve identifying peaks or high-density regions in the data.

We will discuss by far the most popular type, which are combinatorial algorithms.

4.2 Main Characteristics

In combinatorial algorithms, each observation is uniquely labeled by an integer $i \in \{1, \ldots, N\}$.
It is required to pre-specify the number of clusters $K < N$. Each cluster is labeled by an integer $k \in \{1, \ldots, K\}$.
Each observation is assigned to one and only one cluster. These assignments can be considered as an encoder $k = C(i)$, a many-to-one mapping, that assigns the i-th observation to the k-th cluster.
The goal is to find the optimal encoder $C^*(i)$ that minimizes the loss function describing the degree to which the clustering goal is not met. The approach that we will follow is to directly specify the loss function and try to minimize it through some optimization algorithm.
In general, the encoder $C(i)$ is explicitly defined by providing a direct mapping, which assigns a cluster to each observation i.

4.3 Mathematical formulation of the problem

A natural loss function would simply consider the sum of all the dissimilarities of the objects assigned to the same cluster, and the sum this value for all the clusters.

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'}). \tag{4.1}$$

This function describes the similarity of objects in the same cluster. It is referred to as the "within-cluster" point scatter.
Considering T the total point scatter, which is a constant given the data, this

would be the sum of the dissimilarities between all objects:

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d_{ii'}. \tag{4.2}$$

This total distance can be seen as the sum of the dissimilarities between objects belonging to the same cluster and dissimilarities between objects not belonging to the same cluster :

$$T = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\Big(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'}\Big) \tag{4.3}$$

Using the same notation as before and denoting the between-cluster point scatter as $B(C)$, we can write:

$$T = W(C) + B(C). \tag{4.4}$$

$B(C)$ will tend to be large when observations assigned to different clusters are far apart.

From (4.4), we have:

$$W(C) = T - B(C)$$

and we can see that minimizing $W(C)$ is equivalent to maximizing $B(C)$.

4.4 Greedy descent methods

The process for finding the solution, in principle, is clear: one simply minimizes $W(C)$ or equivalently maximizes $B(C)$ over all possible assignments of the N data points to K clusters. Unfortunately, complete enumeration is feasible only for very small datasets.

Feasible strategies are based on greedy descent methods. In general, an initial assignment is specified. Then, at each step of the algorithm, a local optimum choice is made by updating the cluster assignments. Different cluster algorithms differ in the procedure by which they update these cluster assignments at each iteration.

The algorithms terminate when the procedure is no longer able to provide improvement.

Greedy algorithms are not exact and may converge to local optima. By making locally optimal choices at each step, they examine only a fraction of all possible assignments. We will now present some specific procedures that involve this approach, starting with K-means.

5 K-means

This algorithm requires all variables to be of the quantitative type, since the squared Euclidean distance is chosen as the dissimilarity measure.

5.1 The Within-Cluster Scatter Optimization

When the squared Euclidean distance is chosen as dissimilarity, the within-cluster scatter $W(C)$ can be computed as the sum of the squared distances between each object in a cluster and the mean of that cluster's objects.
By defining $\bar{x}_k = (\bar{x}_{1k}, \ldots, \bar{x}_{pk})$ as the mean vector associated with the k-th cluster and $N_k = \sum_{i=1}^{N} I(C(i) = k)$:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2 \tag{5.1}$$

$$= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2. \tag{5.2}$$

Thus, we want to solve:

$$C^* = \min_{C} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2. \tag{5.3}$$

An iterative descent algorithm can consider minimizing clustering dissimilarity as the task of finding the optimal C and optimal means. Indeed, considering the natural definition of the mean vector for any set of observations S:

$$\bar{x}_S = \arg\min_{m} \sum_{i \in S} ||x_i - m||^2 \tag{5.4}$$

for finding C^*, we can solve the enlarged optimization problem:

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - m_k||^2 \tag{5.5}$$

5.2 The K-means Algorithm

1. Randomly initialize the centers of the clusters.
2. Assign each data point to the nearest center.
   Based on Euclidean distance, this is:

$$C(i) = \underset{1 \leq k \leq K}{\operatorname{argmin}} ||x_i - m_k||^2 \tag{5.6}$$

3. Recalculate the centers of the clusters based on the newly assigned data points.
4. Repeat Steps 2 and 3 until convergence. Convergence occurs when the centers no longer change significantly (or a specified number of iterations is reached).

A visual representation of the algorithm is presented in Figure 1.
Given the inherent non-exact nature of the K-means clustering algorithm, the initial selection of means significantly impacts the final result. It is advisable to initiate the algorithm with many different random values and then opt for the solution that presents the smallest value of the objective function.

5.3 An Application in Image and Signal Compression

K-means finds an intriguing application in the domain of image and signal compression presented by Gersho and Gray [3]. Consider an image composed of $N \times N$ pixels, each representing a grayscale value ranging from 0 to 255, and so requiring 8 bits for storage. The objective is to compress the image using the K-means algorithm.

The initial step involves dividing the image into 2×2 blocks of pixels. Each block is then treated as a vector in R^4. This vectors are the input data for the algorithm. The subsequent task perform the K-means algorithm and cluster these blocks into K clusters. All the blocks are then approximated with the center values of their respective clusters.
This clustering process is termed the *encoding step*, and the collection of centers is called the *codebook*.
In terms of storage savings, for each block, we store the label of the center with which it is approximated. With K clusters, we need $\log_2(K)$ bits for storing the cluster labels.
Thus, for the entire image, we require $\log_2(K) \cdot \frac{N \cdot N}{4}$ bits.
Typically, the storage required for the codebook itself, which comprises $K \times 4$ numbers, is considered negligible.
Comparing this to the initial storage required of $N \times N \times 8$ bits, the compressed image storage amounts to $\frac{\log_2(K)}{4 \cdot 8}$ of the original. This is commonly expressed as a rate in bits per pixel: $\frac{\log_2(K)}{4}$.

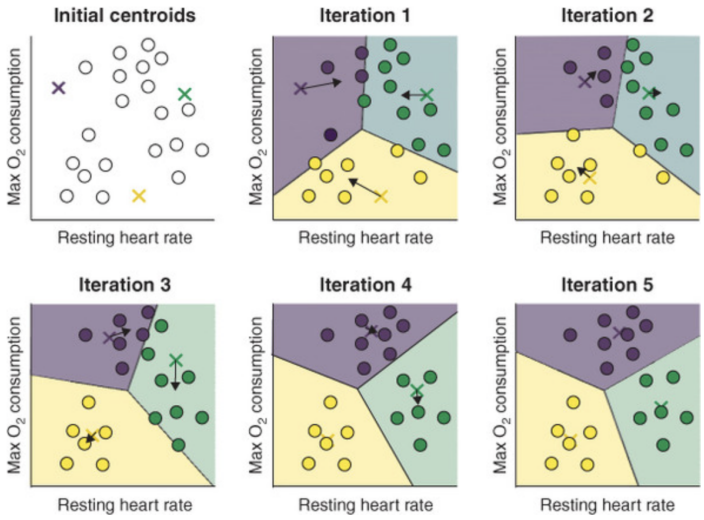

Figure 1: *Five iterations of k-means clustering, from Rhys [2]. In the top-left plot, three initial centers are randomly generated (Step 1).Objects are then assigned to the cluster of their nearest center (Step 2). At each iteration, each center moves to the mean of the cases in its cluster (indicated by arrows) and observations are reassigned.*
The straight lines show the partitioning of points, each sector being the set of points closest to each center. This partitioning is called the Voronoi tessellation.

To determine the most suitable number of clusters, a trade-off is considered between the rate of space gained and the incurred distortion. A rate/distortion curve is commonly employed for this purpose.
Moreover, we note that we employed a fixed-length code, requiring $\log_2(K)$ bits to identify each of the K codewords in the codebook. As not all codewords have the same probability of occurrence, a variable-length code can be used, allowing for even greater space savings.

6 K-medoids

6.1 Restrictions of K-means

K-means presents several constraints:

- The data must be of quantitative type since the dissimilarity measure has to be a Euclidean distance.
- The data has to be raw observations, the algorithm is not directly applicable to distances since it requires the calculation of the means.
- It lacks robustness against outliers, as using squared Euclidean distance gives more influence to larger distances.

These limitations can be overcome by the use of K-medoids. The trade-off is in computation.

6.2 The K-medoids Algorithm

The only part of K-means that requires Euclidean distance is step 3 of the algorithm (Section 5.2): in this step, the cluster representatives $\{m_1, \ldots, m_K\}$ are computed as the means of each cluster.

The main difference with K-means lies in the fact that in K-medoids, the clusters are restricted to be one of the observations assigned to each cluster. By doing that, an explicit optimization is required to compute $\{m_1, \ldots, m_K\}$. This is presented in Step 2 of the following procedure.

1. Randomly initialize the medoids of the clusters.
2. Assign each data point to the nearest medoid. This is the resolution of:
$$C(i) = \underset{1 \leq k \leq K}{\operatorname{argmin}} D(x_i, m_k), \tag{6.1}$$
where $D(\cdot)$ represents the dissimilarity measure.
3. Recalculate the medoids of the clusters based on the newly assigned data points. Find the observation in the cluster that minimizes the total distance to other points in the cluster:
$$i_k^* = \underset{i:C(i)=k}{\operatorname{argmin}} \sum_{C(i')=k} D(x_i, x_{i'}). \tag{6.2}$$
4. Repeat Steps 2 and 3 until convergence. Convergence occurs when the medoids no longer change significantly (or a specified number of iterations is reached).

This algorithm overcomes the constraints of K-means: quantitative data is not required, and distance matrices can be used as input (there is no need to explicitly compute cluster centers; rather, we just keep track of the indices i_k^*). Additionally, the dissimilarity measure $D(\cdot)$ can be defined in a way that makes it robust against outliers.

6.3 Computation Required

Step 2 of K-medoids requires an amount of computation proportional to $K \cdot N$, like Step 2 of K-means. However, Step 3 of K-medoids requires an amount of computation which increases to $O(N_k^2)$, while Step 3 of K-means requires an amount of computation which increases to $O(N_k)$. Thus, we have removed the restrictions at the expense of computation.

6.4 Practical Issues

For applying K-means or K-medoids it is necessary to select the number of clusters K^* and an initialization. We start by providing the general strategies for the initialization.

6.4.1 Initialization

For this process, suggestions are simple random selection and forward stepwise assignment. The latter is adapted for K-medoids and involves an initial set of centers, typically chosen randomly, and a greedy addition of new centers. New centers are added one at a time to minimize the criterion:

$$\min_{C,\{i_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{C(i)=k} d_{ii_k} \tag{6.3}$$

which is the complete optimization problem tried to be solved by the K-medoids algorithm, (6.1) and (6.2).

The forward stepwise assignment, therefore, can be described as:

1. Start with an initial set of K centers, typically chosen randomly.
2. For each iteration, identify the data point that minimizes (6.1) when added as an additional center. Add the selected data point to the set of cluster centers.
3. Repeat steps 2 and 3 until K cluster centers have been found.

6.4.2 The Number of Clusters K^*

There are typically two main scenarios:

- K is defined as part of the problem. In this case, we want to segment the data. An example is a company that employs K people to organize different activities for children. The goal is to partition the children into K segments, one for each animator, such that the children assigned to each one are as similar as possible, and the activities could be more personalized and adapted for them.

- We want to understand the extent to which the observations fall into natural distinct groupings. In this case, the natural number of groups is unknown, and we need to estimate it from the data.

In this context, cross-validation is not suitable. A higher number of groups might improve results simply because a greater number of centers can more effectively cover the data space. More centers will be closer to data points, resulting in a reduction of within-cluster dissimilarity as K increases.

Thus, we implement and heuristic approach starting from the assumption that there is a true underlying number of groups. Considering this, for $K < K^*$, the clusters returned by the algorithm will each contain a subset of the true underlying clusters. On the contrary, for $K > K^*$, the algorithm must divide at least one of the natural groups. So we can say that for $K > K^*$, we will have a smaller decrease in the within-cluster dissimilarity compared to the ones obtained for $K < K^*$.
This means: $\{W_K - W_{K+1}|K < K^*\} \gg \{W_K - W_{K+1}|K \geq K^*\}$.
An strategy is to estimate K^* as the "kink" in the plot of W_K as a function of K.

6.4.3 Gap Statistic

The gap statistic, introduced by Tibshirani et al.[4] , provides an automatic way of locating the "kink" in the data.

The procedure can be described as follows:

- Generate N simulated uniformly distributed datasets: Create datasets uniformly distributed within a rectangle containing the original data. A suggested number of simulated dataset is 20.
- Apply K-means clustering to each simulated dataset with different values for k.
- For each value of k and each simulated dataset, calculate the within-cluster sum of squares.
- Calculate the average expected value of within-cluster sum of squares ($logW_{k,n}$) for $n = 1, .., N$.
- Calculate the gap statistic for each value of k: the gap is the difference between the average expected value of $logW_k^{unif}$ for the N simulations and the observed $logW_k^{data}$ from the actual data.

$$G(K) = logW_k^{unif} - logW_k^{data} \tag{6.4}$$

- Calculate the half-width of the error bars as $s' = s_K\sqrt{1 + \frac{1}{N}}$, where s_K is the standard deviation of $logW_k^{unif}$ over the N simulations.

- Select the minimum K that presents a gap greater than that of $K+1$, considering the standard error s'_{K+1}. This means:

$$K^* = \arg\min_K \left\{ K \,\middle|\, G(K) \geq G(K+1) - s'_{K+1} \right\} \tag{6.5}$$

This formula selects the minimum value of k where the gap is statistically significant, so where it is unlikely to have this gap caused by noise.
The intuition between this procedure is that the gap statistic can be seen as a measure of how much the observed $logW_k^{data}$ is larger than the expected $logW_k^{unif}$, for a given number of clusters. A large gap suggests that the data are well-clustered, while a small gap suggests that the data are not well-clustered.

6.4.4 Shilouette Method

Another very commonly used method to assess the optimal number of clusters K^* is the shilouette method. This method evaluates the quality of clustering by measuring how well each data point fits into its assigned cluster compared to other clusters. The silhouette score, ranging from -1 to 1, provides a higher score for better clustering results.

Typically, the silhouette score is computed for various values of k (number of clusters). The clustering solution with the highest silhouette score is considered the most suitable, and the related k as K^*.

For calculating the shilouette score for a specific clustering solution, the following steps are required:

- Calculate the cluster cohesion: For each data point within a cluster, compute the average distance from the data point to the other points within the same cluster (a_i).
- Calculate the cluster separation: For the same data point, calculate the average distance from the data point to all points in the nearest neighboring cluster (b_i). To choose the nearest neighboring cluster for a given data point, it is necessary to calculate the average distance from that data point to all points in each cluster, and select the cluster with the smallest average distance.
- Compute the silhouette score: The silhouette score for each data point is given by the formula:

$$\text{Silhouette score}_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

.

The numerator, $b - a$, quantifies the difference between separation and cohesion. The denominator, $\max(a, b)$, serves as a normalization factor, ensuring that the resulting score is within the range of [-1, 1].

- Compute the overall silhouette score for the clustering solution. This overall silhouette score is calculated as the average of the silhouette scores for all data points.

7 Hierarchical Clustering

Hierarchical clustering methods present different levels of hierarchy. Clusters at each level of the hierarchy are created by merging clusters at the next lower level.
These methods require the specification of a measure of dissimilarity between groups, which would be based on the dissimilarity chosen for individual observations. With these approaches, it is possible to obtain ordering information about dissimilarities among clusters.
There are two main strategies for hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). As the name suggests, agglomerative methods recursively merge a selected pair of clusters into a single cluster. At each level, only two clusters are merged, resulting in a grouping at the next higher level with one less cluster.
On the other hand, divisive methods recursively split one cluster into two new clusters. The chosen cluster is the one whose division allows to create two new clusters with the largest between-group dissimilarity.
Talking about the number of clusters K^*, the user can then select the level of the hierarchy, and consequently, the number of clusters, that better reproduces the natural structure of the data. To pursue this objective, the gap statistic (Section 7.3) can be used.

7.1 Dendograms

The recursive binary splitting (for divisive) and agglomeration (for agglomerative) can be represented by a rooted binary tree. In these representations, the root node represents the entire dataset, and each of the N terminal nodes represents one observation.
A dendrogram, a rooted binary tree where the height of each node is proportional to the value of the intergroup dissimilarity between its two daughters, can be created. This is possible because these methods exhibit a monotonicity property: the dissimilarity between merged clusters monotonically increases with the level of the merger. This monotonicity is maintained thanks to the logical processes underlying hierarchical clustering methods (see later).

Dendrograms are widely utilized as they offer a comprehensive and interpretable description of clustering, outlining the rationale that led to the formation of specific clusters. The clarity and explainability of dendrograms are among the main reasons for the popularity of hierarchical clustering methods.

Another advantage of dendrograms is that they provide a graphical display

of the selection of the number of clusters, K^*. Indeed, it is possible to cut the dendrogram horizontally at a particular height, and the vertical lines that intersect the horizontal line represent the disjoint clusters. Moreover, visually understanding which clusters could be candidates for natural clusters is possible. If two clusters merge at a high level compared to the merger values of the subgroups contained within them lower in the tree, this indicates that the dissimilarity between them is higher compared to the dissimilarity within the subgroups. Considering that $W(K+1) \ll W(K))$, as the natural groups are successively assigned to separate clusters, these groups that merge at high values are likely to be natural clusters.

7.1.1 Cophenetic Dissimilary

The dendrogram should be primarily viewed as a description of the clustering structure of the data imposed by the specific algorithm employed, rather than as an representation of the inherent structure of the data.
First of all, different hierarchical methods and minor changes in the data can result in significantly different dendrograms. Moreover, the data must possess the hierarchical structure produced by the algorithm.

Additionally, certain characteristics of cophenetic dissimilary underscore the need to interpret dendrograms with caution.
Cophenetic dissimilarity is a valuable concept for understanding how well the clustering process respects the initial dissimilarities between observations.
The cophenetic dissimilarity $C_{ii'}$ between two observations (i, i') is defined as the intergroup dissimilarity at which observations i and i' are *first* joined together in a cluster.

To assess how faithfully the clustering process preserves the original dissimilarities, we use the cophenetic correlation coefficient. This coefficient measures the correlation between the $N(N-1)$ pairwise observation dissimilarities $d_{ii'}$ and their corresponding cophenetic dissimilarities. This correlation provides insights into the extent to which the clustering process maintains or distorts the initial dissimilarities present in the raw data. If the cophenetic correlation coefficient is close to 1, it suggests that the clustering has preserved the original dissimilarities. In contrast, a lower correlation indicates that the clustering results may not accurately reflect the original dissimilarities among the observations in the dataset.

We can identify two main reasons why is not common to achieve a high cophenetic correlation coefficient, and so a clustering process that preserves the original dissimilarities.

First, the cophenetic dissimilarity obeys the *ultrametric inequality* which aligns with the idea that clusters are formed based on the minimum dissimilarity criterion.

$$C_{ii'} \leq \max(C_{ik}, C_{i'k}) \tag{7.1}$$

for any three observations (i, i', k).

This inequality is challenging to respect for arbitrary data sets. For example, in a Euclidean coordinate system, to respect this property, all the triangles formed by all triples of points must be isosceles triangles with unequal sides no longer than the length of the two equal sides.

Moreover, the tree structure imposed by hierarchical clustering algorithms flattens the original dissimilarities between observations. Initially, we have a total of $N(N-1)/2$ values for the dissimilarities, with the same number of observations (N), we can have only $N-1$ distinct values of cophenetic dissimilarities.

In conclusion, the difficulty to respect the ultrametric inequality for arbitrary data and the reduction in the number of distinct cophenetic values are among the reasons why it is not common to obtain a high cophenetic correlation coefficient in practice. This means that it is difficult to obtain dissimilarities between clusters (represented in the dendograms) that preserve the original dissimilarities.

7.2 Agglomerative Clustering

As previously mentioned, the agglomerative clustering strategy involves recursively merging the two closest clusters into a single cluster.

To determine which clusters to merge, it is necessary to establish a measure of dissimilarity between two clusters, each representing a group of observations. Three main methods exist for this purpose. In all cases, the intergroup dissimilarity is computed from the set of pairwise observation dissimilarities $d_{ii'}$, where i is in the first cluster and i' is in the second cluster. The methods differ in how they summarize the intergroup dissimilarity from these individual dissimilarities. We present the methods and a description of their advantages and disadvantages.

Given two clusters G and H:

- **Single linkage** takes the intergroup dissimilarity to be that of the closest pair:

$$d_{\mathrm{SL}}(G, H) = \min_{i \in G, i' \in H} d_{ii'} \tag{7.2}$$

- **Complete linkage** takes the intergroup dissimilarity to be that of the furthest pair:

$$d_{\mathrm{CL}}(G, H) = \max_{i \in G, i' \in H} d_{ii'} \tag{7.3}$$

- **Group average** takes the intergroup dissimilarity to be the average dissimilarity between the groups:

$$d_{\mathrm{GA}}(G,H) = \frac{1}{N_G \cdot N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'} \tag{7.4}$$

where N_G and N_H are the numbers of observations in G and H, respectively.

All the methods present pros and cons. If the data have a strong clustering tendency, then the choice of the method is not that important; however since, in general, the results differ based on the method chosen,it is important to be aware of these advantages and disadvantages.

With Single linkage, we risk to observe the phenomenon of chaining. Considering that a single low value $d_{ii'}$ is enough to merge two clusters, there is a tendency to combine observations linked by a series of close intermediate observations. Here, the problem is the violation of the compactness property of clusters, which refers to the degree to which a cluster occupies a small, well-defined area in the feature space. Compactness focuses on the total spatial extent of a cluster.
On the other hand, with complete linkage, there can be a tendency to have clusters with some observations closer to members of other clusters than they are to some members of their own cluster. Here, compactness is usually assured, meaning that clusters tend to have small diameters D_G:

$$D_G = \max_{i \in G, i' \in G} d_{ii'}. \tag{7.5}$$

With complete linkage, the risk is to violate the "closeness" property of clusters, which refers to the extent to which data points within the same cluster are close to each other. It focuses on pairwise distances between points within a cluster.

Group average is considered a compromise between the two preceding methods. However, a significant risk is associated with the method's sensitivity to the numerical scale of dissimilarities. This sensitivity arises because group average takes into account the specific values of dissimilarities between individual observations. In contrast, single linkage and complete linkage are based on the rank order of dissimilarities, making them invariant to monotone transformations.

7.2.1 Group Average Statistical Consistency

Group average clustering is the only method among the three that exhibits a statistical consistency property.
We expect that in the limit of a large sample, the method provides a consistent estimate of the true relationship between the underlying populations. The idea is that, as the sample size increases indefinitely ($N \to \infty$), the clustering results

should converge to the true underlying structure of the data.
In the context of attribute-value data $X_T = (X_1, \ldots, X_p)$, we assume that each cluster k is a random sample from some population joint density $p_k(x)$. Considering dissimilarities, we start with the expectation:

$$\mathrm{E}[g(G,H)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,x')p_G(x)p_H(x')\,dx\,dx'. \tag{7.6}$$

For group average, we consider $g(X,X')$ as the joint dissimilarity function $d(G,H)$. $p_G(x)$ and $p_H(x')$ are the probability density functions of the random variables X and X'.

For the group average dissimilarity $d_{GA}(G,H)$ as N approaches infinity, we obtain:

$$\mathrm{E}[d_{GA}(G,H)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d(x,x')p_G(x)p_H(x')\,dx\,dx'. \tag{7.7}$$

In contrast, for single linkage, $d_{SL}(G,H)$ approaches zero as $N \to \infty$, independent of $p_G(x)$ and $p_H(x)$. For complete linkage, $d_{CL}(G,H)$ becomes infinite as $N \to \infty$, again independent of the two densities.
Thus, only with group average do we gain a clear understanding of what aspects of the population distribution are being estimated by the intergroup dissimilarity.

7.3 Divisive Clustering

Divisive clustering involves recursively splitting one cluster into two new clusters. The main advantages of this approach are associated with situations where a small number of clusters is desired. In such cases, maintaining a lower computational complexity is possible compared to additive clustering algorithms.
To decide how to perform the split at each iteration, there are two main paradigms.

The first one employs K-means or K-medoids with K=2. However, problems arise with this approach because the results depend on the starting configuration specified at each step, and it may not necessarily produce splittings guaranteeing monotonicity, a requirement for dendrogram representation.

The other commonly used approach is the divisive algorithm proposed by Macnaughton Smith et al. [5].

1. Place all observations in a single cluster G.
2. Choose the observation whose average dissimilarity from all the other observations is the largest.
3. Make this observation the first member of a second cluster H.

4. Choose the observation in G whose average distance from those in H, minus that for the remaining observations in G, is the largest and transfer it to H.

5. Repeat the last step until the corresponding difference in averages becomes negative. Indeed, if, for a specific observation in cluster G, the average distance to the observations in cluster H minus the average distance to the remaining observations in cluster G is positive, it implies that, on average, this particular observation is closer to the existing members in cluster H than it is to the remaining members in cluster G.
   The result of this first 5 Steps is a split of the original cluster into two daughter clusters, which form the second level of the hierarchy.

6. Apply this splitting procedure to one of the clusters at the current level.

Continue until all clusters become singletons or all members of each one have zero dissimilarity from one another.

To choose the cluster to split, the procedure suggested by Kaufman and Rousseeuw [6] involves selecting, at each level, the cluster with the largest diameter (7.5).
An alternative is to choose the one with the largest average dissimilarity among its members, represented as

$$\bar{d}_G = \frac{1}{N_G} \sum_{i \in G} \sum_{i' \in G} d_{ii'}$$

where N_G is the number of observations in cluster G.

8 Self-Organizing Maps

Self-organizing maps are a tool used to organize and visualize high-dimensional data in a lower-dimensional space, capturing the inherent structure within the data.

We denote the high-dimensional space of the data as R^p. We consider the lower-dimensional space to be R^2. Common choices for the representation dimensions are R^2 or R^3.
In this case, the Self-Organizing Map (SOM) will use a two-dimensional rectangular grid. In this two-dimensional grid, we have K prototypes. Prototypes can be visualized as circles in the grid. To each prototype, we attach a vector $m_j \in R^p$, representing a vector of weights. These prototypes are also parameterized with an integer coordinate pair $l_j = (l_{j1}, l_{j2})$, where l_{j1} and l_{j2} are integers representing the position of the prototype along the first and second dimensions of the grid.

Prototypes in a SOM act as representative points or centroids within the input space. These prototypes are adjusted during the learning process to capture the underlying patterns in the data. These adjustments are made by updating their weight vectors. They serve as "archetypes" that sum up the features present in the dataset. After the learning process, prototypes are then mapped in a lower dimensional space. This mapping allows the visualisation.

The grid dimensions are defined by q_1 and q_2, representing the possible values for the coordinates, $l_{j1} \in Q_1$ with $Q_1 = \{1, 2, \ldots, q1\}$ and $l_{j2} \in Q_2$ with $Q_1 = \{1, 2, \ldots, q2\}$. Q_1 and Q_2 are sets representing the possible values for the coordinates. The total number of prototypes (K) is given by the product $q_1 \times q_2$, which corresponds to the total number of prototypes in the two-dimensional grid.

The SOM creation process proceeds as follows for each observation x_i :

1. Compute the Euclidean distance in R^p between the vector x_i and each m_n (with $n = \{1, 2, \ldots, q1 \times q2\}$, representing the prototypes.
2. Indentify the closest prototype m_j.
3. Move all neighbors m_k of m_j toward x_i via the update:

$$m_k \leftarrow m_k + \alpha(x_i - m_k).$$

   This makes x_i and m_k closer in the R^p space.

The "neighbors" of m_j are defined to be all m_k such that the distance between l_j and l_k is smaller than the threshold r, which is a hyperparameter of the model. This neighborhood always includes the closest prototype m_j itself.

Over time, the learning rate (α) and the neighborhood size typically decrease to allow the SOM to converge gradually. A fixed number of iterations (each involving processing a single observation or data point) can also be defined as a stopping criterion.

Once the process of updating weights is completed, we have weights that reflect positions of observations in the input space. The prototypes can then be mapped down onto the two-dimensional grid. The coordinates of a prototype, its position in the grid, can be calculated based on its weights. Considering that the weights of prototypes are adjusted based on the input data, prototypes that respond similarly to similar inputs end up being close to each other in the grid, preserving the topological relationships of the input data.

When the distance r is small enough so that each neighborhood contains only one point, only one prototype will be updated for each observation. In this scenario, the SOM algorithm is an online version of K-means clustering. Indeed, in this case, each observation influences only the one prototype, and we have a situation where the prototypes become analogous to the centroids in the context of K-means clustering.
The difference lies in the fact that in K-means, the assignment of points and the update of centroids are performed considering all points in each iteration. The SOM algorithm represent an online version: it considers one point at a time and then updates the interested prototypes at every point considered.

9 Conclusion

In conclusion, this thesis has delved into the theoretical foundations of clustering techniques. We began with an introduction to unsupervised learning and the goals of cluster algorithms, emphasizing the heuristic nature of evaluating results in the absence of a direct measure of success.

We then introduced the notion of dissimilarity, which plays a crucial role in understanding how clustering algorithms operate. The discussion covered proximity matrices and dissimilarity based on attributes. For the latter, various types of variables were taken into consideration. We also explored the concept of weights of attributes, concluding that flattening all the contributions of the attributes to make their influence equal risks hiding some natural importance. Finally, we addressed ways to handle missing values.

Combinatorial algorithms were introduced, with a mathematical formulation of the clustering problem, focusing on minimizing within-cluster dissimilarity using greedy descent methods.

K-means and K-medoids were presented in detail, highlighting their differences and applications. An application in image and signal compression showcased the practical relevance of these algorithms.

In conclusion, we discussed the practical issues related to these algorithms: the initialization and the choice of the number of clusters. We presented methods for these, including forward stepwise assignment for initialization and the Gap statistic for choosing the optimal number of clusters $K*$.

The hierarchical clustering section discussed linkage criteria, introducing single linkage, complete linkage, and group average, along with their pros and cons. We also emphasized the importance and advantages of dendrograms and the fact that the dendrogram should be primarily viewed as a description of the clustering structure of the data imposed by the specific algorithm. We also introduced the concept of cophenetic dissimilarity. Divisive clustering, though less common, was presented as an alternative with its own advantages.

The thesis concluded with the explanation of self-organizing maps and of their status as an online version of K-means.

In the following practical part, we will apply some of these methods

References

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics.

[2] Rhys, H. (2020). *Machine Learning with R, the Tidyverse, and mlr - Chapter 16*. Manning Publications.

[3] Gersho A., Gray R.M.(1992). *Vector Quantization and Signal Compression*. The Springer International Series in Engineering and Computer Science

[4] Tibshirani R., Walther G.,Hastie T., (2001) *Estimating the Number of Clusters in a Data Set Via the Gap Statistic*. Journal of the Royal Statistical Society Series B: Statistical Methodology, Volume 63, Issue 2, July 2001, Pages 411–423.

[5] Macnaughton Smith, P., Williams, W., Dale, M. and Mockett, L. (1965). *Dissimilarity analysis: a new technique of hierarchical subdivision*, Na- ture 202: 1034–1035.

[6] Kaufman L., Rousseeuw P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics.