CHAPTER 1

Clustering

Bart Baesens, Wouter Verbeke, Johannes De Smedt, Jochen De Weerdt, Hans Weytjens

Corresponding author: Bart.Baesens@kuleuven.be

1.1 CHAPTER OBJECTIVES

In this chapter, you learn

- to understand the basic idea of clustering and its applications;
- how hierarchical clustering methods work;
- ways to measure distances between observations and clusters;
- to determine the optimal clustering solution using dendrograms, the elbow method, the Dunn index, the Davies-Bouldin index and the silhouette coefficient;
- how non-hierarchical clustering methods work such as *k*-means, clustering with mixture models, mean shift clustering, DBSCAN and Self-Organizing Maps (SOMs);
- evaluate and interpret clustering solutions using statistical measures as well as visualisations;

1.2 INTRODUCTION

Clustering is a descriptive analytics activity whose aim is to group a data set of observations into clusters or segments so that the homogeneity within a cluster is maximized and the heterogeneity between clusters is maximized. In other words, segments should be internally cohesive and externally well separated from each other. The discovered clusters or segments can then be individually targeted using tailored strategies and policies. Since there is no continuous nor categorical target variable to steer the learning process, clustering is an example of unsupervised learning.

Clustering techniques have been applied in diverse settings. In marketing, they can be used to do market segmentation (e.g., brand differentiation) and as such improve the understanding of the customer population for targeted marketing or advertising. In text analytics, clustering can be applied to group text documents discussing similar topics. In fraud detection, it can be useful to detect clusters with only a few observations (e.g., insurance claims, credit card payments, transfers in an anti-money laundering setting), yet distant from the rest of the population as such representing anomalous and thus fraudulent behavior. An example

of this is shown in Figure 1.1 where the three red observations make up a separate, small cluster in terms of the recency and frequency variables. These could then be labeled as suspicious and be further analyzed in terms of their characteristics and potentially fraudulent behavior.

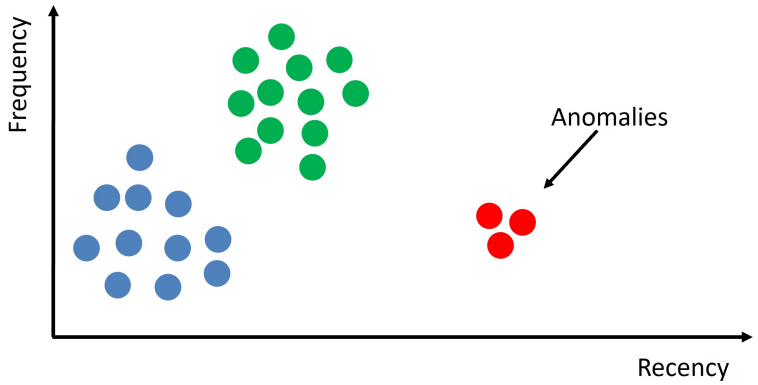

Figure 1.1: Clustering fraudulent claims.

Different types of clustering algorithms have been developed in the literature. At a high level, they can be categorized as either hierarchical or non-hierarchical. Hierarchical methods seek to build a hierarchy of clusters whereas non-hierarchical methods typically cluster observations into a pre-determined number of clusters using an iterative procedure. Popular examples of hierarchical algorithms are agglomerative and divisive algorithms. Popular examples of non-hierarchical algorithms are *k*-means clustering, DBSCAN and self-organizing maps (SOMs). We elaborate on these in what follows.

1.3 HIERARCHICAL CLUSTERING

1.3.1 Introduction

Hierarchical clustering procedures aim at building a hierarchy of clusters. In Figure 1.2 both divisive and agglomerative hierarchical clustering are visualised. The two most extreme clustering solutions are depicted to the left, where all observations are put into their individual cluster, and to the right, where all observations are put into one large cluster. Agglomerative methods start from the left and move to the right by each step merging observations and/or clusters. Divisive methods start from the right and move to the left by each time splitting clusters into smaller parts. The optimal clustering solution is then situated somewhere in between as we discuss in what follows.

1.3.2 Distance metric

As already mentioned, clustering aims at grouping observations based on similarity. In order to do so, a similarity measure needs to be adopted. Various similarity or distance measures have been introduced in the literature as follows:

$$\begin{aligned} d(\mathbf{x},\mathbf{y})_{Manhattan} &= \|\mathbf{x}-\mathbf{y}\|_1 = \textstyle\sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{y}_i| \\ d(\mathbf{x},\mathbf{y})_{Euclidean} &= \|\mathbf{x}-\mathbf{y}\|_2 = \sqrt{\textstyle\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{y}_i)^2} \\ d(\mathbf{x},\mathbf{y})_{cosine} &= \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2} \end{aligned} \tag{1.1}$$

You can see the various distance measures visualized in Figure 1.3. The red dot at location (1,1) is the reference observation for calculating the distances. Darker regions correspond to closer distances. Note how the Manhattan distance results into diamond shaped regions, the Euclidean distance into circular

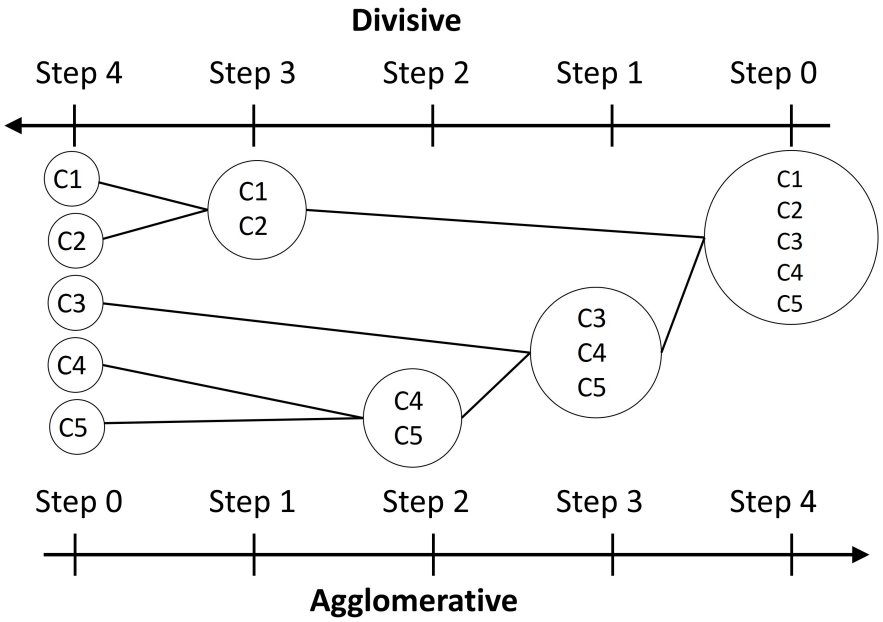

Figure 1.2: Hierarchical clustering: divisive versus agglomerative clustering.

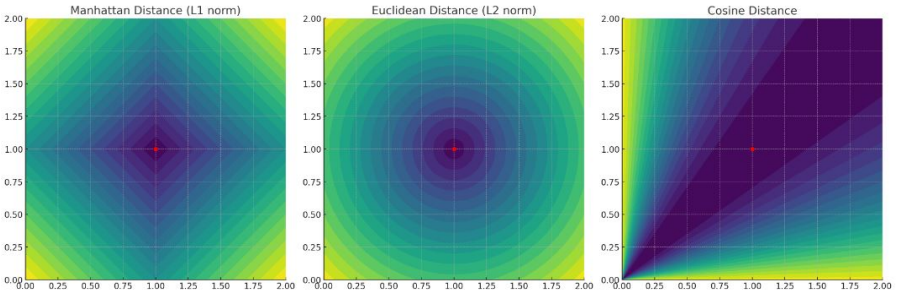

Figure 1.3: Visualisation of Manhattan, Euclidean and cosine distances.

	Accident	Injuries	Doctor Report	Previous Claims	Claim amount >1k
Transaction 1	Yes	No	Yes	Yes	Yes
Transaction 2	Yes	No	No	Yes	Yes
...					

Table 1.1: Jaccard index, Simple Matching Coefficient and Hamming Distance.

shapes and the cosine distance into an orientation based pattern. The Euclidean distance is by far the most commonly used distance metric.

Manhattan distance.
The Manhattan distance is named after the grid-like street layout of Manhattan, New York, where city blocks are arranged in a rectangular pattern. Also called L1 distance or taxicab distance, it represents the total distance a taxi would need to drive in a city laid out in a grid, having to drive along the streets rather than "as the crow flies" (Euclidean distance). The mathematical concept was first considered in the 19th century but gained its common name in the mid-20th century with the rise of computational geometry.

As a final note, before calculating distances it is highly recommended to standardize variables using e.g., the z-scores (as discussed in Chapter**??**). This is to ensure that a variable measured on a wide scale (e.g., income from 0 to 1,000,000) does not overly influence another (e.g., years customer from 1 to 10) in the distance calculations.

When categorical variables are present, various options can be pursued. The Jaccard distance measures the similarity between two sets and is calculated as the size of the intersection divided by the size of the union of the sets. For the example in Table 1.1, the Jaccard similarity would be calculated as 3/5 or 60% since both transactions have Yes values for 3 out of 5 transactions. The Jaccard distance then equals 1 minus the Jaccard similarity. Note that by definition the Jaccard index only considers the presence of values (so the Yes-Yes combinations). It can easily be generalized to categorical variables with more than two values. The Simple Matching Coefficient (SMC) also considers the absence (so the No-No combinations). In our example, the SMC would hence equal 4/5. The SMC distance metric then becomes 1/5. The Hamming distance simply calculates the number of different values. In our example of Table 1.1 it would equal 1. Note that the Hamming distance was especially developed for binary data, so in case of more than two values for a categorical variable, it would require it to be, e.g., one-hot encoded (see Chapter**??**).

When dealing with data sets that have both continuous as well as categorical data, various options can be pursued. A first one is to standardise the continuous data, one-hot encode the categorical data (as discussed in Chapter**??**) and then use the Euclidean, Manhattan or cosine distance metrics for all variables. Another option is to use the Gower distance which simply averages the Euclidean/Manhattan/Cosine distance for the continuous variables and the SMC distance for the categorical variables. Let's illustrate this with a small example. Suppose we have the data in Table 1.2. Let's now calculate the Gower distance between Bart and Sarah using the normalized Manhattan distance for continuous variables and the SMC distance for categorical variables. We normalize using the range to make sure the distances are always between 0 and 1.The ranges for the continuous variables are: Recency: 15-2=13; Frequency: 10-2=8; Monetary: 1000-200=800. We can then calculate the Manhattan distances between Bart and Sarah:

$$\begin{aligned}\text{Recency} &: |10-2|/13 = 8/13 = 0.6154\\ \text{Frequency} &: |5-10|/8 = 5/8 = 0.625\\ \text{Monetary} &: |1000-200|/800 = 800/800 = 1.0\end{aligned}$$

So the average Manhattan distance becomes: $(0.3846 + 0.375 + 0.25)/3 = 0.3365$. The SMC distance for both categorical variables between Bart and Sarah is 0.5 (both are single). Hence, the Gower distance becomes: $(0.3365 + 0.5)/2 = 0.41825$. Note that this assumes equal weight between the continuous

	Recency	Frequency	Monetary	Marital Status	Employed
Bart	10	5	1000	Single	Yes
Sarah	15	2	800	Single	No
Tom	2	10	200	Married	Yes

Table 1.2: Calculating the Gower distance.

and categorical variables. A straightforward extension is to use the number of continuous and categorical variables as weights in the calculation, or alternatively use business knowledge in determining the weights.

John C. Gower.
John C. Gower (1930-2019) was a British statistician known for pioneering work in multivariate analysis and data visualization. While working at Rothamsted Experimental Station, he developed the Gower distance metric in 1971 to compare mixed data types (numerical, categorical, binary).

Finally, remark that to facilitate the distance calculation and cluster interpretation later on, it might be recommended to do some unsupervised feature selection upfront. For the continuous variables, one could calculate, e.g., the Pearson correlation and for the categorical variables the Chi-squared or mutual information correlation. In case of highly correlated variables, one can then opt to keep only one of them.

1.3.3 Linkage methods

Besides the distances between individual observations, clustering methods also need to measure distances between clusters of observations. This is done using a linkage criterion that will help decide when two clusters should merge into a larger one (or similarly, when a large cluster should be divided into two smaller ones). In single-linkage clustering the distance between clusters is measured as the minimal distance between any two observations from the different clusters. Complete linkage clustering on the other hand, measures the distance between two clusters as the maximum distance between any two observations of those clusters. It avoids a problem occurring with single-linkage clustering, known as chaining, where clusters are sometimes merged simply because two observations happen to lie close to one another, even though most observations in the clusters lie quite far from each other. Complete linkage is reported to find compact clusters of roughly equal diameter. A compromise between these two extremes is centroid-linkage, where the distance between two clusters is simply defined as the distance between the cluster centers, similar as in k-means clustering as we discuss below. In average linkage, the distance is measured as the average distance between observations of each cluster. These distance metrics between two clusters C_1 and C_2 with cluster means μ_{C_1} and μ_{C_2} are summarized as follows

$$\begin{aligned} D_{single-linkage}(C_1, C_2) &= \min_{\mathbf{x}\in C_1, \mathbf{y}\in C_2} d(\mathbf{x}, y) \\ D_{complete-linkage}(C_1, C_2) &= \max_{\mathbf{x}\in C_1, \mathbf{y}\in C_2} d(\mathbf{x}, \mathbf{y}) \\ D_{centroid-linkage}(C_1, C_2) &= d(\mu_{C_1}, \mu_{C_2}) \\ D_{average-linkage}(C_1, C_2) &= \frac{1}{|C_1|\cdot|C_2|} \sum_{\mathbf{x}\in C_1} \sum_{\mathbf{y}\in C_2} d(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{1.2}$$

You can see these methods also illustrated in Figure 1.4.

Ward's method is another approach to do hierarchical clustering. This method starts with each observation in its individual cluster. To steer the clustering, it uses the Sum of Squares (SSE) (also called within-cluster variance) calculated as follows:

$$SSE = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_{i.})^2 \tag{1.3}$$

where k is the number of clusters at a given step, n_i is the number of observations in cluster i, x_{ij} is observation j in cluster i and $\bar{x}_{i.}$ is the cluster average of cluster i. Note that at the start of the clustering

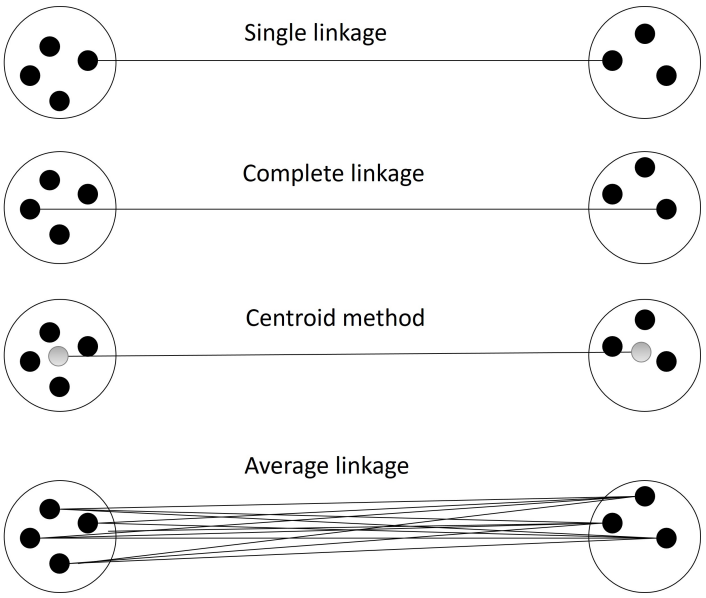

Figure 1.4: Linkage methods for hierarchical clustering.

the SSE will equal 0 since all observations are in their own cluster. Ward's method will then consider all possible pairs of clusters and step by step pick the pair of clusters that gives the smallest increase in SSE until all observations are in one cluster. The optimal number of clusters can then be determined using any of the methods we discuss below.

1.3.4 Deciding upon the number of clusters

Hierarchical clustering does not require the number of clusters to be known beforehand. However, if a final clustering solution is wanted (e.g., for market segmentation purposes), one still needs to decide on the optimal level in the hierarchy and hence on the optimal number of clusters.

A first important aid is a dendrogram. This is a tree-like diagram that records the sequences of merges. Let's illustrate this with an example. Assume that we want to cluster birds in terms of their characteristics, such as the way they look, the noise they make, what they eat, where they live, etc. You can see the clustering process illustrated in Figure 1.5. First, we group chicken and duck, then parrot and canary. Step 3 adds pigeon to the chicken and duck cluster. Step 4 clusters owl and eagle. Step 5 merges the clusters obtained in steps 2 and 3 and step 6 merges everything together. An obvious question is where we should stop clustering? In Figure 1.6 you can see the corresponding dendrogram for our bird-clustering example. The bottom of a dendrogram represents the individual data points. As you move up the dendrogram clusters are formed until all observations are in one cluster at the top or root of the dendrogram. The vertical scale gives the distance between clusters merged. These distances are typically derived from the linkage method used (e.g., single, complete, centroid, average, or Ward's method). The dendrogram can then be cut where the distance jumps become too high suggesting a natural division or clustering in the data. In our example, the red line indicates our optimal clustering. In other words, there are three clusters. We can then try and assign meaningful interpretations to our clusters. For example, Cluster 1 has chicken, duck, and pigeon and can be labeled as poultry. Cluster 2 has parrot and canary and can be labeled as exotic birds. Cluster 3 has owl and eagle and can be labeled as predators. Note that a good labelling of the clusters is very important to facilitate the interpretation thereof. It is typically done by close collaboration between the data scientist and business user.

A bit related is the elbow method. This method calculates the sum of squared errors (SSE) for various

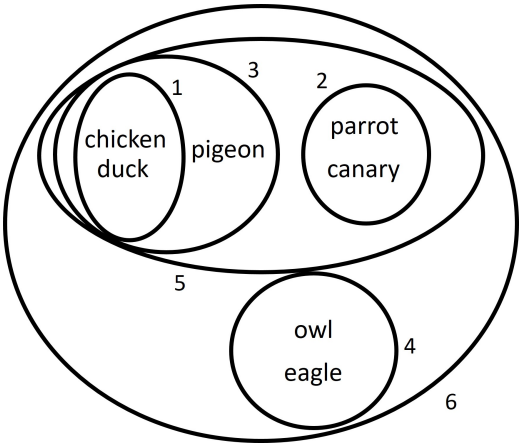

Figure 1.5: Bird clustering example.

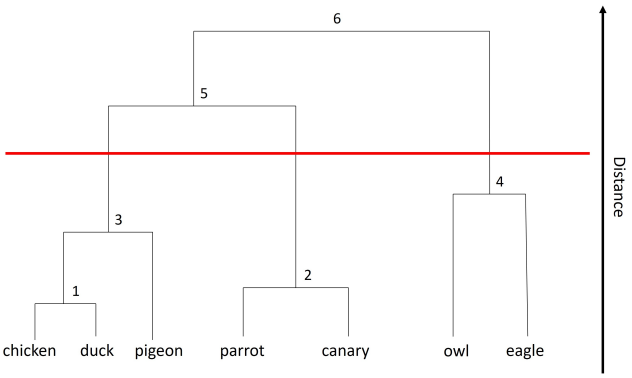

Figure 1.6: Dendrogram for bird clustering example.

number of clusters, k as follows:

$$\text{SSE} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mu_i\|^2 \tag{1.4}$$

It then plots SSE against k and looks for an elbow point as shown in Figure 1.7. Based on the plot, one could decide to put the optimal number of clusters to four as this is where the elbow point is reached.

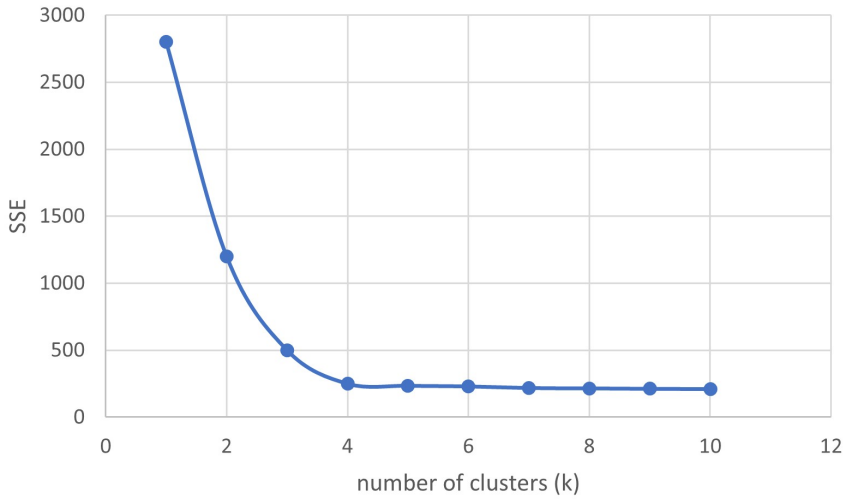

Figure 1.7: Elbow method for determining the number of clusters.

Besides graphical aid, also various metrics have been introduced to determine the optimal number of clusters. The Dunn index starts by calculating the cluster diameter or the maximum distance between any two observations within the same cluster [Dunn, 1974]. Let's call this D_{intra}. It then computes the distance between each cluster observation and the observations in all the other clusters. The minimum of all these distances is referred to as D_{inter}. The Dunn index can then be calculated as:

$$\text{Dunn index} = \frac{D_{inter}}{D_{intra}} \tag{1.5}$$

Obviously, a higher Dunn index corresponds to a better clustering solution. It is a relative index whose value depends upon the data set and clustering technique and is most useful to compare clustering solutions.

The Davies-Bouldin index can also be used to decide upon the optimal clustering [Davies and Bouldin, 1979]. It starts by calculating the within-cluster variability as follows:

$$S_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mu_i\| \tag{1.6}$$

where C_i represents cluster i, $|C_i|$ is the number of observations in cluster i and μ_i the center of cluster i. It then computes the between-cluster variability for each pair of clusters C_i and C_j as follows:

$$M_{ij} = \|\mu_i - \mu_j\| \tag{1.7}$$

Next it calculates the ratio for each cluster pair C_i and C_j:

$$R_{ij} = \frac{S_i + S_j}{M_{ij}} \tag{1.8}$$

Obviously, we want the R_{ij} to be small for good clustering solutions since this indicates good separation. Let's now consider the worst-case value D_i and calculate:

$$D_i = \max_{j \neq i} R_{ij} \tag{1.9}$$

By considering the maximum, the index focuses on the most similar (or least well-separated) pair involving C_i. If D_i turns out to be big, it indicates significant cluster overlap of cluster C_i with another cluster.

In case of a clustering solution with k clusters we can now calculate the Davies-Bouldin (DB) index as the overall average:

$$DB = \frac{1}{k} \sum_{i=1}^{k} D_i \tag{1.10}$$

Obviously, a lower value of the Davies-Bouldin index corresponds to a better clustering since it shows good separation between clusters and low variability within clusters.

The silhouette coefficient is also a popular measure for deciding upon the number of clusters [Rousseeuw, 1987]. For each observation i it first calculates the average (e.g., Euclidean) distance $a(i)$ to all other observations within its cluster. This represents the intra-cluster distance and should preferably be as small as possible. Next, for every cluster C that does not contain observation i, the average distance of every observation in C to observation i is calculated. The smallest of all these average distances is labelled $b(i)$. Obviously, a higher value for $b(i)$ indicates better separation between the clusters. We can then calculate the silhouette coefficient for each observation i as follows:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{1.11}$$

By definition, $s(i)$ ranges between -1 and +1. A value close to +1 indicates that the observation is well located within its cluster, close to 0 that it's on the boundary between clusters and -1 that it might be in the wrong cluster. The silhouette coefficient of a clustering solution can then be calculated as the average of the silhouette coefficients of all observations with higher values indicating a better clustering solution.

As a final remark, note that the business relevant interpretation of a clustering solution can also be an important factor to help decide upon the optimal number of clusters.

1.4 k-MEANS CLUSTERING

k-means clustering is a very popular non-hierarchical clustering method. It starts by selecting k observations which will serve as our initial cluster centroids or seeds. Each observation is then assigned to the cluster that has the closest centroid (for example, in Euclidean sense). When all observations are assigned, the positions of the k centroids are recalculated. This process is repeated for a fixed number of steps, or until the cluster centroids no longer change. The algorithm is described in 1.

Let's illustrate this with an example. Assume we start from a set of observations that are described by two characteristics as shown in Figure 1.8. We now perform k-means clustering and set k to 2. In other words, we try to find two clusters in the data. In Step 1, two observations are randomly chosen as our cluster centers. They are depicted by the black circle and the red triangle, respectively in Figure 1.9. The remaining observations are now assigned to both cluster centers by calculating the Euclidean distance. This results in the black and red cluster as shown in Figure 1.10. In the next step, the cluster centers are re-calculated. They are again depicted by the black circle and red triangle as shown in Figure 1.11. In Figure 1.12 you can see all observations re-assigned based on the new cluster centers. In the next step, the new cluster centers are calculated as shown in Figure 1.13. In Figure 1.14 you can see the observations re-assigned. This is where the k-means clustering ends and results in the black and red clusters.

Algorithm 1 K-Means Clustering

1: **Input:** Dataset $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, number of clusters k
2: **Output:** Cluster assignments for each observation $\mathbf{x}_i \in D$
3: Initialize k centroids $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$ randomly from the observations in D
4: **repeat**
5: Assign each observation $\mathbf{x}_i$ to the nearest centroid:

$$\text{Assign } \mathbf{x}_i \text{ to cluster } j = \arg\min_{j \in \{1,\ldots,k\}} \|\mathbf{x}_i - \mathbf{c}_j\|$$

6: Update each centroid $\mathbf{c}_j$ by computing the mean of all observations assigned to it:

$$\mathbf{c}_j \leftarrow \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$$

7: **until** centroids $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$ do not change significantly
8: **return** Cluster assignments for all observations $\mathbf{x}_i$

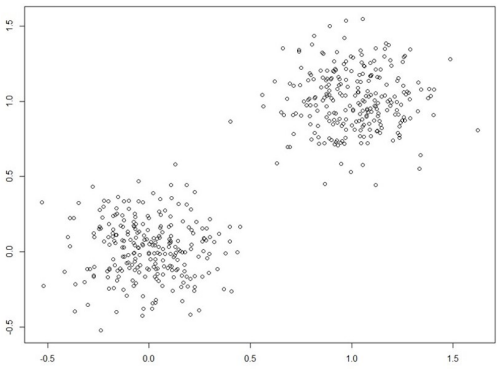

Figure 1.8: Data set for k-means clustering.

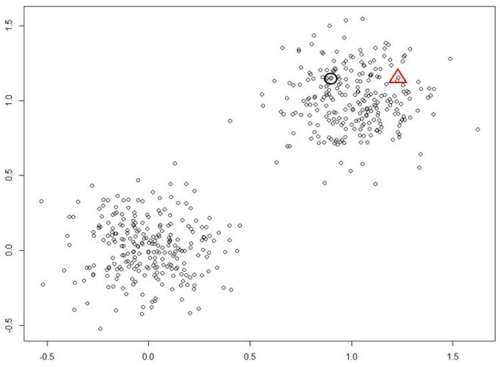

Figure 1.9: Selection of cluster centres.

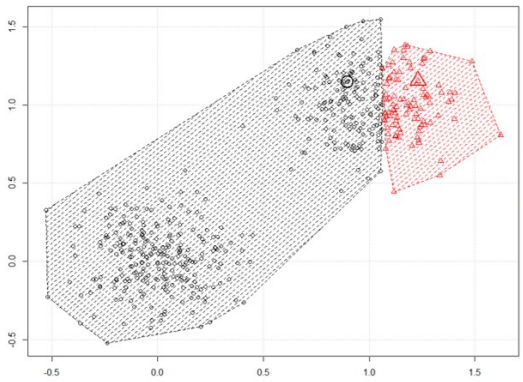

Figure 1.10: Cluster assignment.

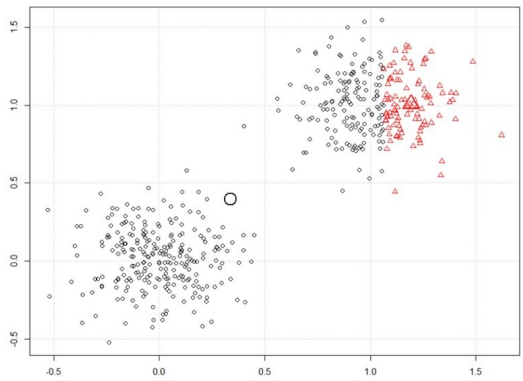

Figure 1.11: Recalculating the cluster centers.

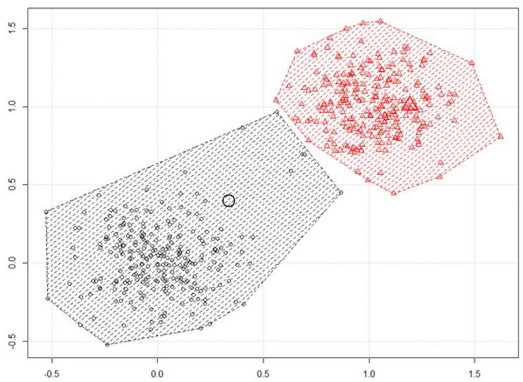

Figure 1.12: Cluster assignment.

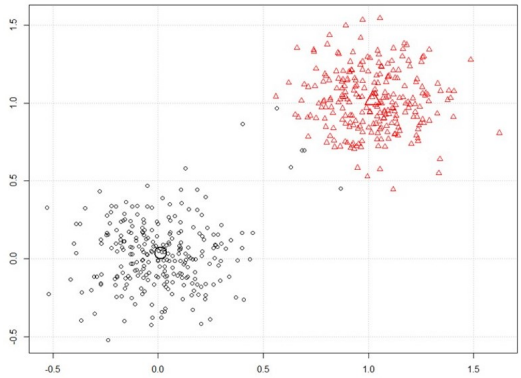

Figure 1.13: Calculating the new cluster centers.

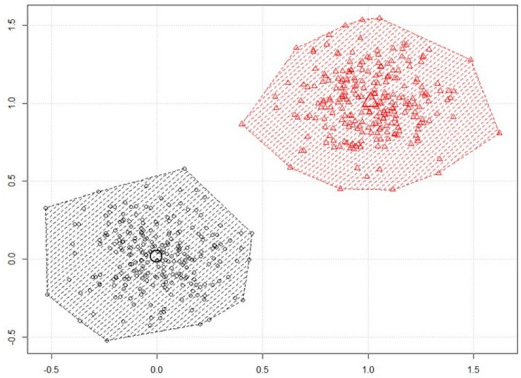

Figure 1.14: Final cluster assignment.

The iterative process of calculating the new cluster centers and assigning all data points to the closest one continues until the cluster centroids do not change significantly. Alternatively, in case of large data sets, slow convergence or observations switching back and forth between clusters, k means can also be terminated after a fixed number of steps.

This rather simple algorithm leads to the overall minimization of the distance from the data points to their cluster centers. It works especially well for clusters that are spherical and roughly the same in size but is not well suited to discover non-convex or irregularly shaped clusters. Remember that a cluster is considered convex if any line segment connecting two observations within the cluster does not leave the cluster.

To decide upon k, one could try out different values and then use any of the graphical aids or metrics we discussed in Section 1.3.4 to determine the optimal value. Furthermore, to check for the stability of the clustering solution, it is recommended to try out different initial seeds where the solution that yields the lowest overall distance measured, e.g., in terms of SSE or any of the metrics discussed in Section 1.3.4, can be chosen.

Several variants of k means clustering have been introduced in the literature. k median clustering uses the median to calculate the cluster centroids and is as such more robust to outliers. $k++$ clustering improves the choice of the initial seeds. It picks the first seed at random and then chooses each next seed with a probability proportional to the squared distance from the nearest already chosen seed. The ensures the seeds are spread out more nicely. Once all seeds are chosen, standard k means is run. In k-medoid clustering the cluster centers represent actual data points instead of fictitious ones which contributes to both cluster interpretability as well as robustness to outliers [Kaufman and Rousseeuw, 2008].

Spherical *k*-means is an extension that works especially well for high-dimensional data spaces, such as those encountered in text clustering where each document is typically represented as a vector of term frequencies or TF-IDF (term frequency-inverse document frequency) scores. In this case, the similarity between observations is better captured by angles rather than by Euclidean distances. Hence, spherical *k*-means uses the cosine metric as the distance metric. It assumes all observations are normalized to unit length and as such lie on the surface of a unit sphere. The goal of spherical *k*-means is then to maximize the overall cosine similarity between observations and their cluster centroids. Finally, fuzzy *k* means clustering allows an observation to belong to multiple clusters using a membership function (which ranges between zero and one) and as such supports overlapping clusters.

k-means clustering.
A famous statistical humor story involves the origin of k-means clustering. In the 1950s, when Bell Labs was working on classifying telephone signals, their researchers supposedly picked ”k” because it was the only letter not yet used in their equations that day. This may be apocryphal, but it amusingly suggests that one of the most widely-used machine learning algorithms got its name not through careful mathematical derivation, but because ”k” happened to be available.

1.5 CLUSTERING WITH MIXTURE MODELS

Another type of clustering assumes that observations in the same cluster are generated by the same distribution. Finding these distributions then constitutes the objective of the clustering problem. Since one can never be completely sure which clustering solution is the best, a cluster membership probability is assigned to each observation as such resulting into a soft clustering.

In its most simple form, one assumes each cluster follows a Gaussian distribution though depending upon the problem statement, other distributions can be used as well. Once the number of clusters is determined, the clustering corresponds to finding proper values for the mean and variance of each cluster distribution.

The mixture probability density modeled is as follows:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \mu_\mathbf{k}, \mathbf{\Sigma_k}) \tag{1.12}$$

K is the number of Gaussians in the mixture and needs to be specified up front, π_k is the weight for each component with $\sum_{k=1}^{K} \pi_k = 1$ and $0 \leq \pi_k \leq 1$ and $\mathcal{N}(\mathbf{x} \mid \mu_\mathbf{k}, \mathbf{\Sigma_k})$ is the Gaussian probability density function for component k. A typical method to determine the mixture parameters is Expectation Maximization (EM) Dempster et al. [1977]. The algorithm is detailed in 2.

As with *k*-means clustering several runs can take place with different initial randomization to check for robustness. The resulting solution is a set of soft clusters, where observations lie within several clusters, with a probability for each. If one wants hard clusters, one can simply assign the observation to the cluster with the highest cluster probability.

Figure 1.15 provides an example with a mixture of three Gaussian models for the Frequency and Monetary variables.

1.6 MEAN SHIFT CLUSTERING

Mean Shift clustering was originally proposed in [Comaniciu and Meer, 2002]. It works as described in Algorithm 3. It starts with each observation representing a cluster center, such that the number of observations equals the number of initial clusters. Next, the distance is measured between each cluster center $\mathbf{c}$ and the observations in its neighborhood $\mathbf{x}_i$, which is defined either as $\|\mathbf{c} - \mathbf{x}_i\| \leq h$ or using a

Algorithm 2 Expectation-Maximization (EM) for Gaussian Mixture Models

1: **Input:** Data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, number of clusters K
2: **Output:** Mixture weights π_k, means μ_k, and covariances $\mathbf{\Sigma}_k$ for each cluster
3: Initialize π_k, μ_k, and $\mathbf{\Sigma}_k$ randomly for $k = 1, \ldots, K$
4: **repeat**
5: **E-Step:** calculate the probability γ_{ik} that data point $\mathbf{x}_i$ was generated by component k

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i \mid \mu_\mathbf{k}, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i \mid \mu_j, \mathbf{\Sigma}_j)}$$

6: **M-Step:** Update the parameters
- Update mixture weights:

$$\pi_k = \frac{N_k}{N}, \quad \text{where } N_k = \sum_{i=1}^{N} \gamma_{ik}$$

- Update means:

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} \mathbf{x}_i$$

- Update covariances:

$$\mathbf{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T$$

7: Check for convergence (e.g., change in log-likelihood below a pre-specified threshold or minimal change in parameter values)
8: **until** Convergence criterion is met
9: **Assign clusters:** For each data point $\mathbf{x}_i$, assign it to the cluster with the highest responsibility γ_{ik} if hard clustering is desired.

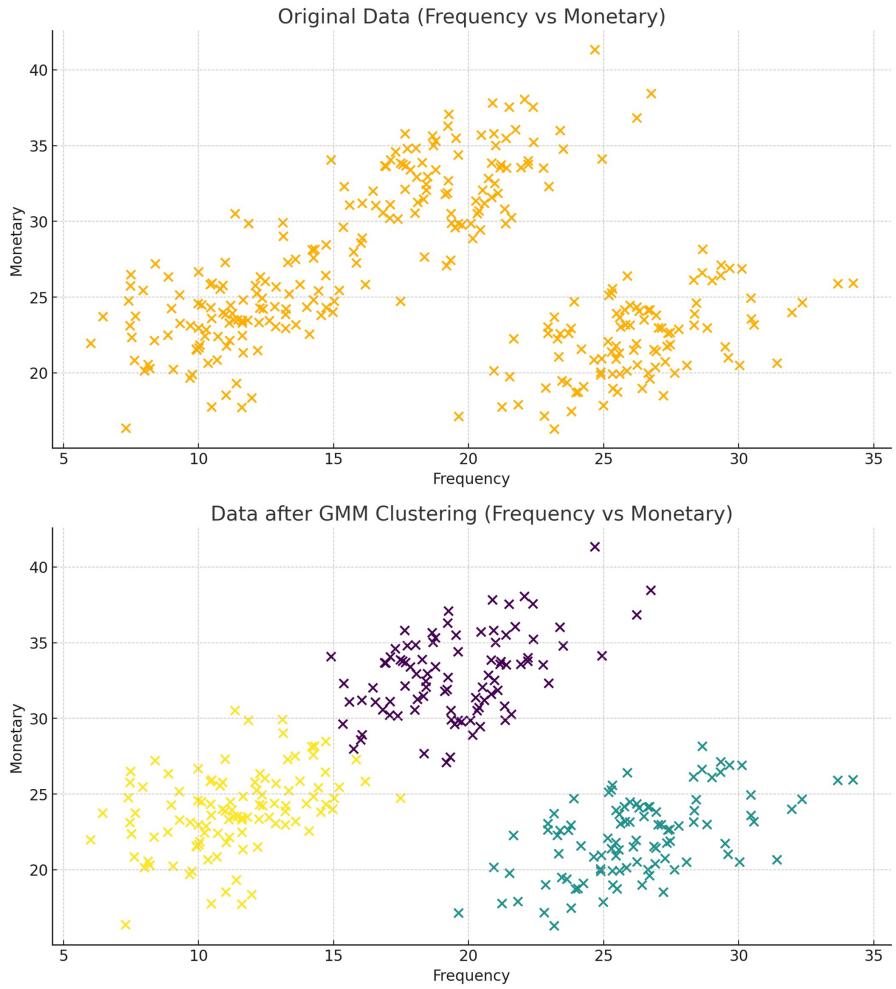

Figure 1.15: Clustering with mixture models.

Gaussian kernel as:

$$\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{c}\|^2}{2h^2}\right).$$

Mean Shift then iteratively updates the position of each cluster center by directly assigning it to the weighted mean of the positions of all observations in its neighborhood. This process is repeated until convergence, where the cluster centers no longer move significantly. The final cluster centers represent the modes of the underlying density function, and each data point is assigned to the nearest mode. This results in a partitioning of the data into clusters based on density peaks, without requiring a predefined number of clusters. Mean Shift is particularly effective for non-linearly separable data and can detect clusters of arbitrary shapes.

The bandwidth parameter h controls the size of the neighborhood. A larger (smaller) h means a broader (narrower) neighborhood, resulting in fewer (more) clusters. Two common methods for setting h are Silverman's rule:

$$h = \left(\frac{4\sigma^5}{3n}\right)^{\frac{1}{5}},$$

and Scott's rule:

$$h = n^{-\frac{1}{d+4}},$$

for Gaussian kernels where n is the number of observations, d the number of variables and σ the (pooled) standard deviation. Alternatively, one can also determine h by maximizing a clustering quality metric (e.g.,

Algorithm 3 Mean Shift Clustering

Input: Dataset $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, bandwidth h
Output: Cluster assignments for each data point in D
Initialize C to include every observation $\mathbf{x}_i$ as initial centroids
repeat
    for each centroid $\mathbf{c}$ in C **do**
        Compute weights for all observations $\mathbf{x}_i \in D$:

$$w(\mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{c}\|^2}{2h^2}\right)$$

        Update $\mathbf{c}$ using the weighted mean:

$$\mathbf{c} \leftarrow \frac{\sum_{\mathbf{x}_i \in D} w(\mathbf{x}_i) \cdot \mathbf{x}_i}{\sum_{\mathbf{x}_i \in D} w(\mathbf{x}_i)}$$

    end for
until centroids $\mathbf{c}$ in C do not change significantly
Assign each observation $\mathbf{x}_i \in D$ to the cluster with the nearest centroid in C
return Cluster assignments

Silhouette Score, Davies-Bouldin Index, elbow method, see Section 1.3.4) for varying values of h. This approach is more computationally intensive but can yield better results.

Figure 1.16 provides an example of mean shift clustering on a two dimensional data set featuring monetary and frequency variables. It can be seen that three clusters are found using a Gaussian kernel with a bandwith parameter of $h = 1$.

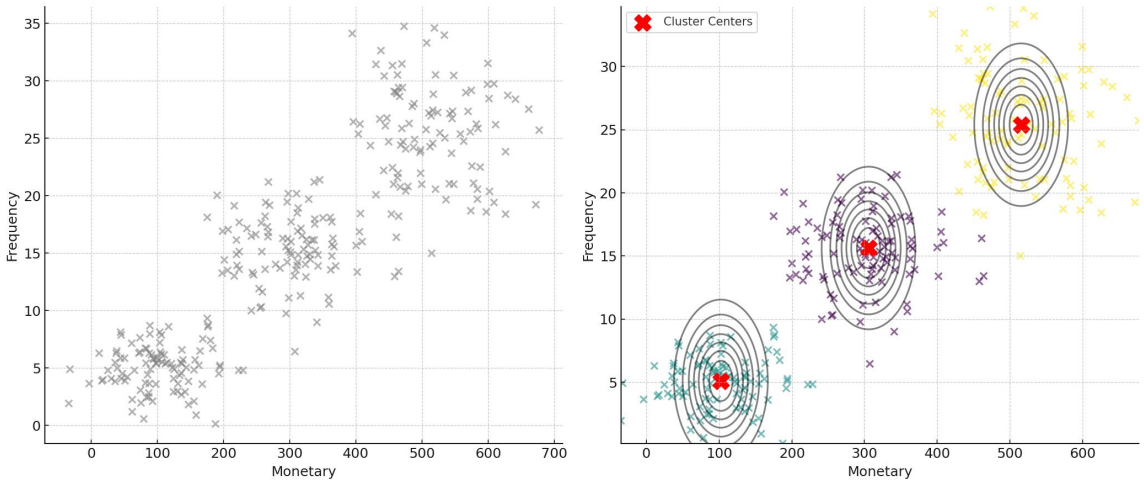

Figure 1.16: Mean Shift clustering.

1.7 DBSCAN

DBSCAN stands for density-based spatial clustering of applications with noise. One of its key strengths is that it discovers clusters of arbitrary shape even non-convex ones in contrast to k-means[Ester et al., 1996] (see Figure 1.17). Remember that for convex clusters any line connecting two observations never leaves the cluster. This is clearly not the case for both examples depicted in Figure 1.17.

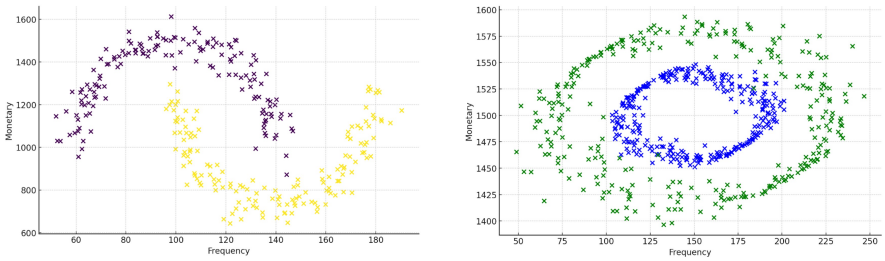

Figure 1.17: Examples of non-convex clusters.

The basic intuition of DBSCAN is to cluster observations in high density regions and mark observations that lie in low-density regions as outliers. A first key concept is the ϵ-Neighborhood which consists of all observations within a radius of ϵ from the observation p defined as

$$N_\epsilon(p) = \{q \in \text{data set } D \mid \text{dist}(p, q) \leq \epsilon\} \tag{1.13}$$

You can see this illustrated in Figure 1.18. The ϵ-neighborhood of observation p is four, whereas the ϵ-neighborhood of observation q is two. A second key parameter is *MinPts*. The density of an observation is considered high with the ϵ-neighborhood contains at least *MinPts* observations. If *MinPts* equals four, the density of p is considered high whereas the density of q is considered low.

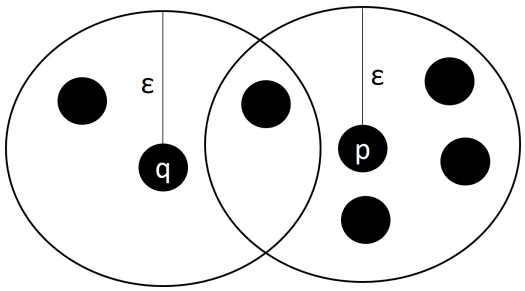

Figure 1.18: The ϵ-neighborhood.

DBSCAN differentiates between three types of observations: an observation is a core observation if the density around it is high. Hence, it has more than *MinPts* observations within its ϵ-neighborhood. These are clearly observations located at the interior of a cluster. A border observation is an observation with low density (or with fewer than *MinPts* observations within its ϵ-neighborhood) but located in the neighborhood radius of a core observation. Border points are reached by the cluster but typically cannot further expand the cluster because of low density. An outlier or noisy observation is not a core nor border observation. You can see this illustrated in Figure 1.19 in case *MinPts* equals five.

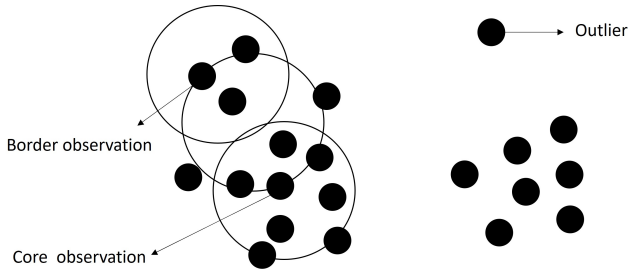

Figure 1.19: Core observation, border observation and outlier in DBSCAN.

An observation q is directly density-reachable from an observation p if p is a core observation and q is

in p's ϵ-neighborhood. In Figure 1.20 with $MinPts = 4$, it can be seen that q is directly density-reachable from p but p is not directly density-reachable from q as the measure is asymmetric and q is not a core observation.

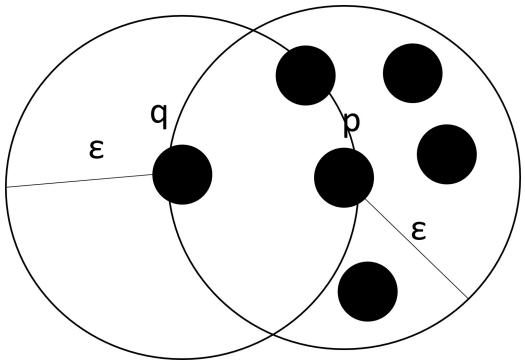

Figure 1.20: Direct density-reachability in DBSCAN.

An observation p is density-reachable from an observation q if there is a chain of points $p_1, \ldots, p_n$, with $p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i. You can see this illustrated in Figure 1.21 if $MinPts$ equals five. You can see that $p1$ is directly density-reachable from q, $p2$ is directly density-reachable from $p1$ and p is directly density-reachable from $p2$. Hence, there is a chain from q to p and p is density-reachable from q. This measure is also not symmetric and q is not density-reachable from p as p is not a core observation.

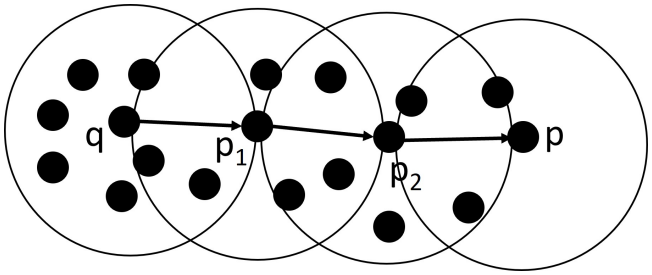

Figure 1.21: Density-reachable in DBSCAN.

Finally, two observations p and q are density-connected if they are both density-reachable from an observation o with respect to ϵ and $MinPts$. You can see this illustrated in Figure 1.22. Note that density-connectivity is a symmetric measure.

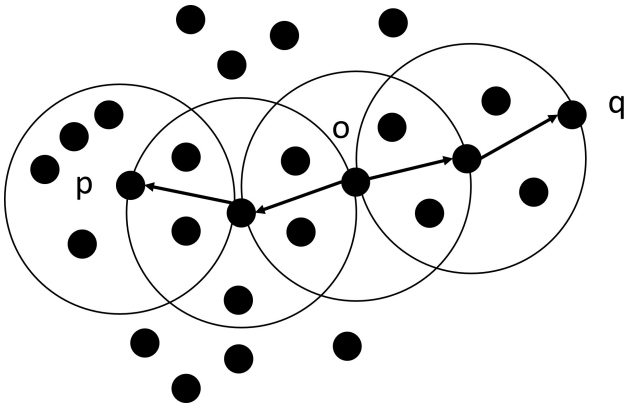

Figure 1.22: Density connectivity in DBSCAN.

A cluster can now be defined as a set of density-connected observations which is maximal with respect to density-reachability. The observations not belonging to any cluster can then be labeled as outliers or noise. We now have the necessary background to start introducing the DBSCAN clustering technique. Assume we have a data set D and two parameters: ϵ and $MinPts$. A cluster C is a subset of D satisfying two conditions:

- maximality condition: $\forall p, q$ if $p \in C$ and q is density-reachable from p, then $q \in C$.
- connectivity condition: $\forall p, q \in C$, p and q are density-connected

Note that a cluster contains both core points as well as border points.

The DBSCAN algorithm then works as follows: As shown above, at the end of the algorithm, the observations not belonging to any cluster are considered as noise.

Algorithm 4 DBSCAN

```
for each observation x_i ∈ D do
    if x_i is not yet clustered then
        if x_i is a core observation then
            find all observations density-reachable from x_i
            assign them to a new cluster
        else
            assign x_i to noise
        end if
    end if
end for
```

A common heuristic to set $MinPts$ is to set it to the dimensionality of the data set plus one. To determine ϵ a k-distance graph can be used. This graph plots the distance to the k-th nearest neighbor (which can be set to, e.g., $MinPts$). The ϵ parameter can then be determined where this plot bends sharply upwards and an elbow is obtained. Do note that often a trial-and-error approach is pursued with the number of outliers detected, or any of the metrics discussed in Section 1.3.4 (e.g., SSE, Dunn index, Davies-Bouldin index, etc) as important guiding criteria.

DBSCAN can handle clusters of different shapes and sizes. It can find non-convex clusters quite easily. When using DBSCAN, there is also no need to fix the number of clusters upfront which differentiates it from k-means clustering. However, one of the key drawbacks of DBSCAN is that it is not well suited to handle varying densities.

1.8 SELF ORGANIZING MAPS

Teuvo Kohonen.
Teuvo Kohonen (1934-2021) was a Finnish computer scientist who pioneered early artificial neural networks. He is best known for inventing the Self-Organizing Map (SOM), also called Kohonen maps, while at the Helsinki University of Technology. His 1982 paper introducing SOMs has been cited over 20,000 times. He received numerous awards including the IEEE Neural Networks Pioneer Award and was an Academician of the Academy of Finland, a rare honor. Despite his contributions to neural networks, he humorously noted that he never used a personal computer, preferring to write his code on mainframes.

Self-organizing maps (SOMs) are a special type of neural network that enables clustering. They were first introduced by Teuvo Kohonen in 1982 [Kohonen, 1982]. SOMs are unsupervised learning algorithms that enable you to visualize and cluster high-dimensional data on a low-dimensional grid of neurons. Basically, a SOM is a feed-forward neural network with two layers: an input layer and an output layer. The neurons in

the output layer are usually arranged in a two-dimensional grid that is rectangular or hexagonal as you can see illustrated in Figure 1.23. In the former case, every neuron has at most eight neighbors, whereas in the latter case, every neuron has at most six neighbors.

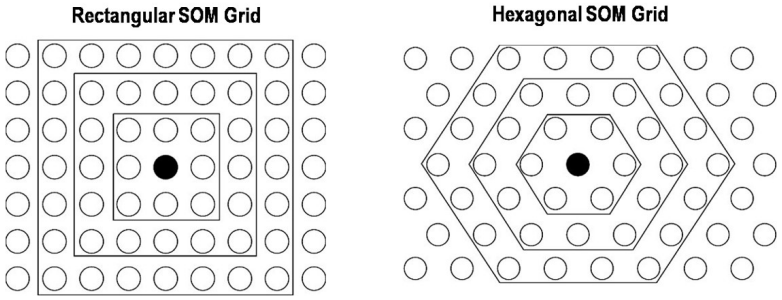

Figure 1.23: Example SOM architectures.

Each input is connected to all neurons in the output layer with corresponding weights $\mathbf{w} = [w_1, \ldots, w_n]$, with n equal to the number of inputs. In other words, every output neuron has its own combination of input weights. The weights are then randomly initialized (for example, by drawing them from a standard normal distribution). When a training observation $\mathbf{x}$ is presented to the SOM, the weight vector $\mathbf{w}_c$ of each neuron c is compared with $\mathbf{x}$ using, e.g., the Euclidean distance

$$d(\mathbf{x}, \mathbf{w}_c) = \sqrt{\sum_{i=1}^{n} (x_i - w_{ci})^2} \tag{1.14}$$

Note that, as always, it is important to standardize the data first (for example, by calculating the z-scores) as discussed in Chapter XXX. The neuron that lies closest to $\mathbf{x}$ in the Euclidean sense is the winner or best matching unit (BMU). The weight vector of the BMU and its neighbors in the SOM are then updated using the following learning rule:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{w}_i(t)] \tag{1.15}$$

where t represents the time index during training and $h_{ci}(t)$ defines the neighborhood of the BMU c, which defines the region of influence.

The neighborhood function$h_{ci}(t)$ is a non-increasing function of time and distance from the BMU. Here you can see two examples of neighborhood functions.

$$\begin{aligned} h_{ci}(t) &= \alpha(t) \exp(-\tfrac{||r_c - r_i||^2}{2\sigma^2(t)}) \\ h_{ci}(t) &= \alpha(t) \text{ if } ||r_c - r_i||^2 \leq \text{threshold; 0 otherwise} \end{aligned} \tag{1.16}$$

Note that r_c and r_i represent the location of the BMU and neuron i on the map, $\sigma^2(t)$ represents the decreasing radius, and $\alpha(t)$ is the learning rate ranging between 0 and 1. Popular examples are

$$\begin{aligned} \alpha(t) &= \tfrac{A}{t+B} \\ \alpha(t) &= \exp(-At) \end{aligned} \tag{1.17}$$

The decreasing learning rate and radius give a stable map after a certain amount of training. Training is then stopped when the BMUs remain stable, or after a fixed number of iterations, such as 500 times the number of SOM neurons. As training proceeds, the map will start to self-organize and the neurons will move toward the input observations, thereby creating clusters.

After the SOM training has been completed, we can start interpreting it. Various visualization aids can be used for this purpose. A U or unified distance-matrix essentially superimposes a height Z dimension on top of each neuron. It can be visualized using a distance map which visualizes the average distance between a

neuron and its neighbors, whereby typically dark colors indicate a large distance and can be interpreted as cluster boundaries. A second useful feature is a component plane, which visualizes the weights between each specific input variable and its output neurons. As such, it provides a visual overview of the relative contribution of each input attribute to the output neurons.

In Figure 1.24 you can see a example distance map for a 5-by-5 SOM. Remember, a distance map basically visualizes the U-matrix, as defined earlier, as a heat map. The color coding indicates the average distance between a neuron and neighboring neurons in terms of the connecting weights. In this figure, light colors indicate a large distance, and therefore, can be interpreted as cluster boundaries.

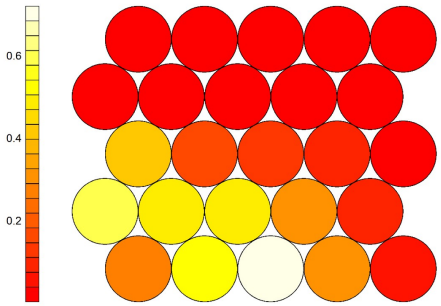

Figure 1.24: Visualising the U-matrix.

As an example, suppose we want to cluster countries based on various characteristics such as the Corruption Perception Index (CPI) ranges between 0 for highly corrupt countries and 10 for highly clean countries, political rights, literacy and other information measuring a country's demographic and macro-economic information for the years 1996, 2000, and 2004 [Huysmans et al., 2006]. Figure 1.25 shows the result of clustering this data onto a 15 by 15 hexagonal SOM map. Notice that uppercase notation refers to the country in 2004, lowercase to the country in 2000, and sentence case to the country in 1996. From the map, it can be observed that many European countries are situated in the upper right part of the SOM. The Scandinavian countries are also nicely grouped in the top half of the SOM. Other similar clusters can be found by inspecting the map.

In Figure 1.26 you can see an example of component planes for the variables: CPI, literacy and political rights. The component plane for CPI to the left is colored in black and white. Dark regions indicate regions with low values for the CPI index and thus with high corruption. Notice that the Scandinavian countries score very well with low corruption. The lower right corner of the SOM is very dark corresponding to countries with significant corruption. The component plane for literacy can then also be inspected. Darker regions correspond to lower literacy. By combining the component planes for CPI and literacy, it can easily be seen that lower literacy usually implies higher corruption. Finally, also the component plane for political rights can be inspected. Again, it shows that the lower right corner of the map faces issues in terms of political rights.

For a small number of variables (for example, four or five at most), the codebook vector graph is an excellent visualization of the contribution to the SOM of each variable [Verbeke et al., 2017]. A codebook vector graph combines basically different component planes, as introduced earlier, in a single visualization. It shows the relative importance of variables for each neuron in the map, by visualizing the relative sizes of the weights of the connections between each neuron and the input variables. It can provide insight into the composition of the data set in terms of segments as well as variations in the variables that define these segments. In Figure 1.27 you can see an example of a codebook vector graph. Note that we defined five

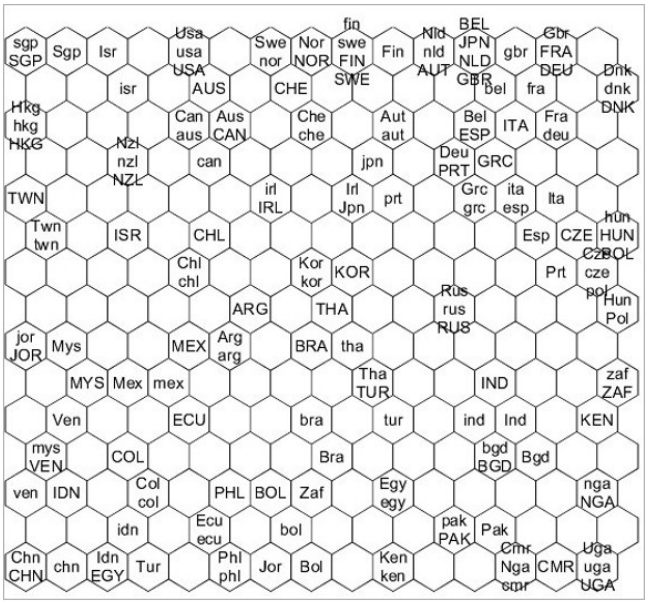

Figure 1.25: Clustering countries using a SOM.

clusters on the map. The contributions of the variables listed at the bottom can be clearly seen in each of the clusters defined.

SOMs have various advantages and disadvantages. First, they are a very useful tool for exploratory data analysis and for gaining some initial insights about the data. They can also be combined with decision trees to further characterize the discovered segments. However, they are difficult to compare because there is no objective function to be evaluated and compared. Also, extensive experimental evaluation is needed to determine the size.

1.9 INTERPRETING CLUSTERING SOLUTIONS

Evaluating a clustering solution is by no means a trivial exercise since there exists no universal criterion. From a statistical perspective, the sum of squared errors (SSE) can be computed. When comparing 2 clustering solutions, the one with the lowest SSE can then be chosen.

Besides a statistical evaluation, a clustering solution will also be evaluated in terms of its interpretation. To facilitate the interpretation of a clustering solution, various options are available. A first one is to compare cluster distributions with population distributions across all variables on a cluster by cluster basis. This is illustrated in Figure 1.28 where the distribution of a cluster $C1$ is contrasted with the overall population distribution for the Recency, Frequency and Monetary variables. It can be clearly seen that Cluster $C1$ has observations with low recency values and high monetary values, whereas the frequency is relatively similar to the original population.

Another way to explain a given clustering solution is by building a decision tree with the ClusterID as the target. Assume we have the output depicted in Table 1.3 from a k-means clustering exercise with k equal to four. Note that the last column contains the cluster identifier. We can now build a decision tree with the ClusterID as the target variable as shown in Figure 1.29. This decision tree gives us a clear insight into the distinguishing characteristics of the various clusters. For example, Cluster 2 is characterized by observations having recency $<$ 1 day and monetary $>$ 1000. Hence, using decision trees, we can easily assign new observations to the existing clusters. This is an example of how supervised or predictive techniques can be used to explain the solution from a descriptive analytics exercise.

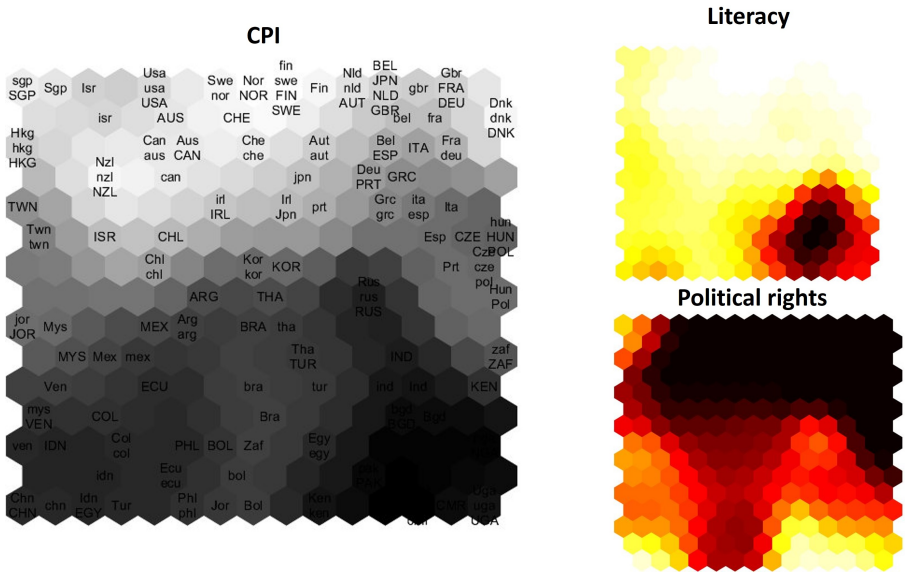

Figure 1.26: Example component planes.

Claim	Recency	Frequency	Monetary	....	ClusterID
Claim1					Cluster2
Claim2					Cluster4
Claim3					Cluster3
Claim4					Cluster2
...					

Table 1.3: Data set for cluster interpretation.

1.10 SEMI-SUPERVISED CLUSTERING

In semi-supervised clustering, known labelling information about the observations or relationships among the observations is used to obtain better clustering solutions. This can be especially handy in situations where data is scarce and these background constraints can help guide the clustering solution to desirable outcomes. Two semi-supervised clustering methods can be identified: pointwise methods where labelling information on a subset of the observations is available and pairwise methods, where must-link or cannot-link constraints exist between some pairs of observations.

A straightforward point-wise clustering based method is seeded k-means clustering, which is basically a regular k-means clustering method that initializes the cluster centroids to the centers of the observations with known class labels [Basu et al., 2002]. For a two-class (good/bad) problem for example, the average over all observations with prediction 'good' is calculated and becomes the first cluster centroid. Similarly a cluster centroid is calculated for the predicted 'bads'. Next, the traditional k-means clustering algorithm is applied. It remains possible that observations with the same initial labels do end up in different clusters. If this is not wanted, constrained k-means clustering can be applied, where observations with known labels are not allowed to change cluster.

Examples of must(cannot)-link constraints are constraints specifying that two documents or images should belong to the same cluster because they feature similar topics (e.g., textual description or images of lions). In constrained k-means, both constraints are checked when re-assigning observations to clusters. In case of violation, no re-assignment takes place. Alternatively, the constraints can be enforced by artificially modifying the distance metric (i.e., decreasing it for must-link and increasing it for cannot-link) or adding constraint violation penalties to the objective function. For more information, we refer to [Chapelle et al.,

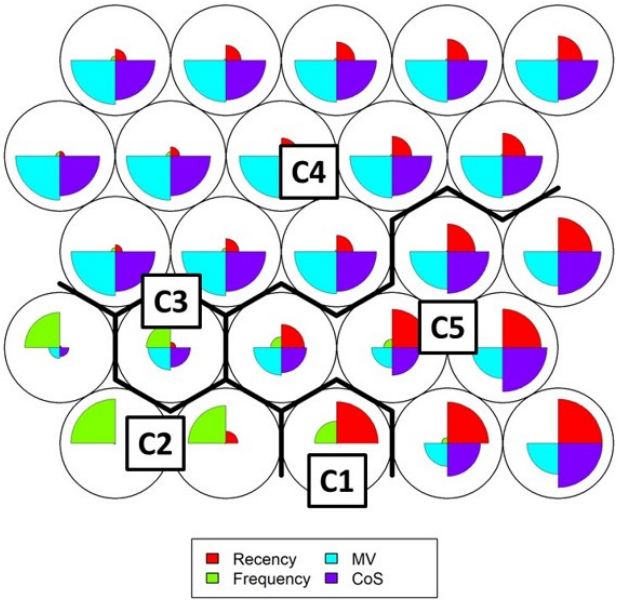

Figure 1.27: Example codebook vector graph.

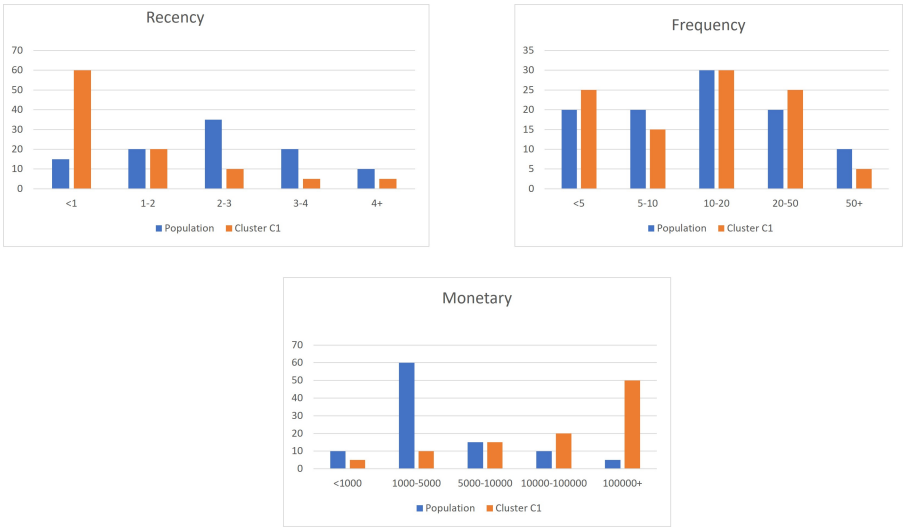

Figure 1.28: Comparing cluster distributions with population distributions.

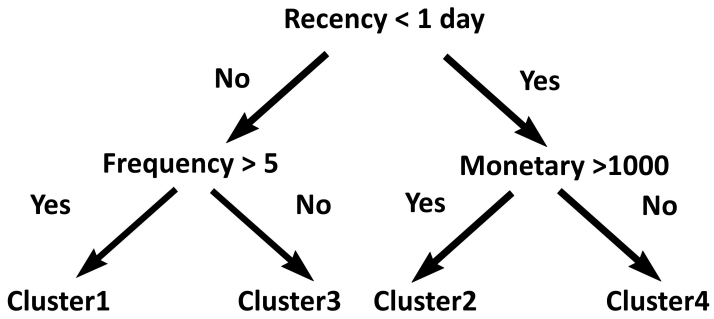

Figure 1.29: Decision tree for cluster interpretation.

2006].

1.11 EXTERNAL CLUSTER VALIDATION

The idea of external cluster validation is to compare a clustering solution to a reference clustering obtained from either another parameterization or algorithm, (a) human expert(s), or even the ground truth when available. It could be used to verify whether patterns in the data have shifted due to, e.g., population drift. Various metrics have been proposed for this purpose.

Suppose we have two clustering solutions, S_1 and S_2. Let's now calculate the following

$$\begin{aligned} a &= \text{number of observation pairs that are in the same clusters in } S_1 \text{ and } S_2 \\ b &= \text{number of observation pairs that are in different clusters in } S_1 \text{ and } S_2 \\ c &= \text{number of observation pairs that are in the same clusters in } S_1 \text{ and in different clusters in } S_2 \\ d &= \text{number of observation pairs that are in different clusters in } S_1 \text{ and in the same cluster in } S_2 \end{aligned} \tag{1.18}$$

The Rand index is then computed as follows:

$$\text{Rand} = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{1.19}$$

It represents the frequency of agreements between both clusterings across all possible observation pairs and ranges between 0 and 1.

Another way is to calculate Meila's Variation of Information (VI) [Meila, 2007]. The entropy $H(C)$ of a clustering C measures the uncertainty in the clustering and is calculated as:

$$H(C) = -\sum_{i=1}^{k} \frac{|C_i|}{N} \log_2 \frac{|C_i|}{N} \tag{1.20}$$

where C_i is the set of elements in cluster i, k is the number of clusters, $|C_i|$ is the number of observations in cluster i and N is the total number of observations. Mutual Information (MI) measures the amount of information shared between two clustering solutions, $C1$ and $C2$, and is defined as:

$$I(C1, C2) = \sum_{i=1}^{k1} \sum_{j=1}^{k2} \frac{|C1_i \cap C2_j|}{N} \log_2 \frac{N|C1_i \cap C2_j|}{|C1_i||C2_j|} \tag{1.21}$$

where $C1_i$ are clusters in $C1$, $C2_j$ clusters in $C2$ with $k1$ and $k2$ being their respective numbers of clusters. Intuitively, we can think of $I(C1, C2)$ as follows. Suppose we are given a random observation. The uncertainty about its cluster in $C2$ is measured by $H(C2)$. Suppose we are told which cluster the

observation belongs to in $C1$. $I(C1, C2)$ then measures the reduction in uncertainty about which cluster this observation belongs to in $C2$ after knowing this information, averaged over all observations. Meila's Variation of Information (VI) can then be calculated as:

$$VI(C1, C2) = H(C1) + H(C2) - 2I(C1, C2) \tag{1.22}$$

$VI(C1, C2)$ is positive number with a lower value indicating higher similarity between both clustering solutions. For identical clusterings, $VI(C1, C2)$ will equal 0.

1.12 CONCLUDING REMARKS

Clustering is one of the most popular descriptive analytics activities with excellent algorithms already developed several decades ago. In this chapter we discussed various clustering techniques and summarize their key properties in Table 1.4.

Clustering Technique	**Scalability (Observations)**	**Scalability (Variables)**	**Non-Convex Clusters**	**Parameterization**
Hierarchical Clustering	Low	Medium	Yes	Medium
K-Means	High	Low	No	Easy
Mixture Models	Medium	Medium	Yes	Difficult
Mean Shift	Medium	Medium	Yes	Medium
DBSCAN	Medium	Medium	Yes	Medium
Self-Organizing Maps	Medium	High	Yes	Medium

Table 1.4: Comparison of Clustering Techniques

Clustering has been widely used in marketing and risk analysis to profile customer segments. In fact, the goal of clustering can be two-fold. First, the clusters revealed could serve to help define marketing actions targeting each of them in different ways. Alternatively, the result of a clustering exercise can also be the input for a second stage predictive model aimed at further differentiating behavior within a cluster in a supervised way.

Bibliography

S. Basu, A. Banerjee, and R.J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, pages 19–26, 2002. URL `http://www.cs.utexas.edu/users/ai-lab?basu:ml02`.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006. ISBN 0262033585.

D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. doi: 10.1109/34.1000236.

David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979. doi: 10.1109/TPAMI.1979.4766909.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

J.C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974. doi: 10.1080/01969727408546059.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

J. Huysmans, D. Martens, B. Baesens, J. Vanthienen, and T. Van Gestel. Country corruption analysis with self organizing maps and support vector machines. In H. Chen, C. C. Wang, F. Y.and Yang, D. Zeng, M. Chau, , and K. Chang, editors, *Intelligence and Security Informatics*, pages 103–114, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33362-3.

Leonard Kaufman and Peter J. Rousseeuw. *Partitioning Around Medoids (Program PAM)*, pages 68–125. John Wiley Sons, Inc., 2008. ISBN 9780470316801. doi: 10.1002/9780470316801.ch2. URL `http://dx.doi.org/10.1002/9780470316801.ch2`.

Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43 (1):59–69, 1982.

M. Meila. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98 (5):873–895, 2007. ISSN 0047-259X. doi: https://doi.org/10.1016/j.jmva.2006.11.013. URL `https://www.sciencedirect.com/science/article/pii/S0047259X06002016`.

P.J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL `https://www.sciencedirect.com/science/article/pii/0377042787901257`.

W. Verbeke, B. Baesens, and C. Bravo. *Profit Driven Business Analytics: A Practitioner's Guide to Transforming Big Data into Added Value*. The Wiley and SAS Business Series. Wiley, 2017. ISBN 978-1-119-28655-4.