Chapter 1

State of the Art

This section first gives the definition of clustering along with its use cases and its main difficulties. Then several commonly met methods are presented along with the category of algorithms they belong to. Eventually we focus on the specific subfield of multi-view clustering, giving a definition and a state of the art of the research works which are conducted on it.

1.1 Definition of clustering

Everyday, people all around the world generate 2.5 exabytes (2.5×10^{18} bytes) of data of all kinds: videos, texts, audio, activities on social networks... Since its creation, clustering has become an active field of research because of the possibilities it offers when applied to such a variety of data. Using clustering, it becomes possible to better understand the underlying structures of a dataset. These structures can then be used for further real-world application such as market study, advertisement targeting, product improvement and recommendation.

In [22], the clustering is defined as "[...] the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters)". The term unsupervised refers to one of the two subfield of Machine Learning: supervised and unsupervised learning. A problem is said to be supervised if the learning algorithm is trained knowing both the input and the corresponding output (classification for example). On the opposite, a problem is said to be unsupervised if one has only access to its input, which is the case of the clustering for which one has to define clusters only knowing input data.

The first two clustering algorithms found in the literature are Hierarchical Clustering [44] and K-means [26] respectively defined in 1963 and 1967. Those algorithms define two of the main clustering subfields: the methods based on a hierarchical classification of the input, and the methods based on its partition. The different clustering subfields will be described later in this section.

1.2 Challenges

However, clustering in itself is a very broad field, and each method has to adress one or several challenges among these which have emerged since the creation

of the field. In this subsection is detailed the challenges faced by clustering methods: the 4 V's of Big Data, the similarity measure selection, the clustering quality assessment and the data topology.

1.2.1 The 4 Vs of Big Data

The very first challenge is related to the data to be analyzed. It is not only related to clustering, but also to any problem related to Big Data. This challenge is known as the four V of Big Data, namely the Volume, the Variety, the Veracity and the Velocity of input data. One may sometimes find some other V such as Variability, Visualization and Value, however, the first four ones are usually taken as the concensus of the field. These points each refer to an aspect of the input data that might make a Big Data analysis difficult: Volume refers to the ever growing quantity of data generated every day, Variety refers to the diversity of data available (audio, video, text...), Veracity refers to the uncertainty inherent to the data gathering process and Velocity refers to the speed at which the data may be generated and may therefor need to be analyzed. The choice of a clustering algorithm has to be made considering which of these problems one may have to adress, because to date, no known algorithm can adress the four V together.

1.2.2 Similarity measures

The second problem related to clustering lies in the similarity measures each algorithm has to use. Clustering relies on the idea that similar individuals must belong to the same cluster while dissimilar ones should not. Therefor, one of clustering challenges is to define the criterion which make it possible to compare two individuals. Problems such as point clustering in "physical" space can be adressed using distance measures among which the most well-known is the euclidean distance, a.k.a. the $l2$-norm and defined as follow:

$$d(x, y) = \sum_{k=1}^{D} (x_k - y_k)^2 \tag{1.1}$$

With D the dimension of x and y, and with x_k the k-th coordinate of x in its feature space. This measure is intuitive because it refers to the concept of distance which we use everyday. However, even if it is intuitive and can give good results in lower dimensional spaces, it becomes another problem when the number of dimensions increases. The choice of the inter-individuals distance must consider points such as the space dimensionality, the kind of features used in the dataset, and even the goal of the clustering. A guide on this problem can be found in [10].

Also, the choice of a measure defines data topology in the sense that it defines the shape of the clusters that may be found. The topology of data is also a point to consider when choosing a clustering method because some may (or not) be used for certain kind of topologies. The diagrams displayed in documentation of the Python package *scikit-learn*[1] present how some methods may or may not adapt to some topologies. For example, methods such as K-means based with

[1] http://scikit-learn.org/stable/modules/clustering.html

$l2$-norm defines ball-shaped clusters. This may not give the expected result when considering data forming concentric circles for example.

From now, every time the term "neighbor" will be used, it will implicity implies that a distance between data points has been defined. The definition of this distance can be viewed as an additional parameter for every method using it.

1.2.3 Quality measures

Eventually, another problem appears when one has applied the selected clustering method on its dataset: how can one measure the quality of a clustering? With supervised problem such as classification, it is easy to determine the quality of the result by simply considering criterion such as the percentage of object well classified. You can even use this criterion during the learning phase to deduct update rules as in [43]. However, the problem is different for clustering: most of the time you do not have access to a criterion which you can optimize during the learning process. If the problem only involves data in a visualizable space (between 1 and 3 dimensions), it is easy to visualize the data and to see if the clusters created are intuitively good. However, most of the time the problem involved have a dimensionality that is far higher than 3, making it hard to visualize graphically the content of each cluster. Some methods such as Principal Component Analysis [46] and Locally Linear Embedding [36] allow to give an estimated projection of the data (and of their corresponding clusters) in a visualizable space. Some work in the literature are solely dedicated to the definition of a clustering quality, such as presented in [1].

However, even if it is hard to give an intuitive measure of the quality of a clustering, two points are intuitevely studied to get an idea of the clustering quality: its compactness, namely how close are the individuals from a same cluster, and its separability, namely how far are the individuals from different clusters. some measures allow to compare two clustering one to each other. Among these measures, the most commonly met are the Dunn index [11], the Davies-Bouldin index [7], the Silhouette index [35] the Wemmert-Gancarski Index [45]. Some measures even allow to compare two clusterings, namely the Rand index [33] and its Adjusted version [20]. The four first measures are based on the idea that clustering should minimize the distance between members of a same cluster (compactness) while maximizing distance between members of different clusters (separability), which corresponds to the intuitive goal that one may want to achieve when performing clustering. While the four first make it possible to say that one clustering is "better" than the other, the two last only define the similarity between two clustering, without telling which one is the best. Each one is detailed in the following paragraphs.

Dunn index

The Dunn index is defined as the ratio between an inter-cluster measure D and a measure of scatter Δ_i. It is defined as follow:

$$DU = \frac{\min\limits_{i \neq j} D(c_i, c_j)}{\max\limits_{i \in [1..C]} \Delta_i} \tag{1.2}$$

With C the number of cluster. D and Δ_i can be defined in different ways, creating each time a new version of the Dunn index. Here are presented the most used versions:

$\mathbf{D(c_i, c_j)}$	$\mathbf{\Delta_i}$
$\min_{x \in c_1, y \in c_2} d(x, y)$	$\max_{x,y \in c_i} d(x, y)$
$\max_{x \in c_1, y \in c_2} d(x, y)$	$\min_{x,y \in c_i} d(x, y)$
$d(\mu_i - \mu_j)$	$\frac{1}{\lvert c_i \rvert} \sum_{x \in c_i} (x - \mu_i)$
$\frac{1}{\lvert c_1 \rvert \lvert c_2 \rvert} \sum_{x \in c_1} \sum_{x \in c_2} d(x, y)$	$\frac{1}{\lvert c_i \rvert \times (\lvert c_i \rvert - 1)} \sum_{x,y \in c_i, x \neq y} d(x, y)$

Table 1.1: Different set of distance inter-clusters and measure of scatter for the Dunn index

With μ_i the center of the i-th cluster defined as follow:

$$\mu_i = \frac{1}{|c_i|} \sum_{x \in c_i} x \tag{1.3}$$

Davies-Bouldin index

Based on the same idea than the Dunn index, the Davies-Bouldin index is defined as follows:

$$DB = \frac{1}{|C|} \sum_{i=1}^{|C|} \max_{j \neq i} \frac{\Delta_i + \Delta_j}{D(c_i, c_j)} \tag{1.4}$$

With C the clustering and $|C|$ the corresponding number of clusters, Δ_i a measure of scatter for the i-th cluster and $D(c_i, c_j)$ a distance measure as defined for the Dunn index. The most commonly met version of the Davies Bouldin index in the one based on the third line of Table 1.1.

Silhouette index

The Silhouette index (SI) can also be used to assess the compactness of the clusters as well as their separation. However, the main difference between this index and the Dunn or Dabies-Bouldin ones is that it can be computed either for a given datum x, or for a fiver cluster c_i, or for the whole clustering. For a datum x, a_x is defined as the mean distance between x and the elements

belonging to its cluster, and b_x is defined as the mean distance between x and the elements which do not belong to its cluster. Then, the Silhouette index in defined as follow:

$$SI(x) = \frac{b_x - a_x}{\max(a_x, b_x)} \tag{1.5}$$

From which it follows that

$$SI(x) = \begin{cases} \frac{b_x}{a_x} - 1, & \text{if } a_x > b_x \\ 0, & \text{if } a_x = b_x \\ 1 - \frac{a_x}{b_x}, & \text{if } a_x < b_x \end{cases} \tag{1.6}$$

This implies that $SI(x)$ takes values between -1 and 1. A value near -1 means that the distance between x and the element from its cluster is bigger than the one between x and all the other elements, meaning that x does not belong to the good cluster. On the opposite, if x is near 1, it means that it is closer to the elements from its cluster, which may indicate that x is in the right cluster. When this value is computed for every x in a cluster, the SI of the cluster can be computed as follow:

$$SI(c_i) = \frac{1}{|c_i|} \sum_{x \in c_i} SI(x) \tag{1.7}$$

And eventually, when $SI(c_i)$ is computed for all the clusters, the SI of the whole clustering can be computed:

$$SI(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} SI(c_i) \tag{1.8}$$

If one would like to use this index, it has to be noticed that it is computationaly expensive to compute it. Even if an intermediary structure is used to store the distances already computed, the index will have to compute $\frac{1}{2}n(n-1)$ distances, with n the number of elements in the dataset.

Wemmert-Gancarski index

The Wemmert-Gancarski (WG) index is another index based on the pair compactness-separability. It is defined as follow

$$WG(c_i) = \begin{cases} 0 \text{ if } \frac{1}{|c_i|} \sum_{x \in c_i} \frac{d(x,\mu_i)}{d(x,\mu_j)} > 1 \\ 1 - \frac{1}{|c_i|} \sum_{x \in c_i} \frac{d(x,\mu_i)}{d(x,\mu_j)} \text{ otherwise} \end{cases} \tag{1.9}$$

This index takes its values between 0 (bad score) to 1 (good score). To get the result on a complete clustering, the following formula is applied:

$$WG(C) = \frac{\sum_{i=1}^{|C|} |c_i| WG(c_i)}{\sum_{i=1}^{|C|} |c_i|} \tag{1.10}$$

Rand index

To compare two clustering, the Rand index considers the pairs of elements which are clustered together (or not) in each clustering. Four sets are defined (a, b, c, and d), the union of which contain all the pairs of elements of the dataset studied. The belonging of a pair to a set follows Table 1.2.

	Same cluster in C_1	**Same cluster in C_2**
a	yes	yes
b	no	no
c	yes	no
d	no	yes

Table 1.2: Sets of pairs of elements used for the definition of the Rand index

Using these sets, the Rand index can be defined as follow:

$$RI(C_1, C_2) = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} = \frac{2(a+b)}{n(n-1)} \tag{1.11}$$

The Rand index takes its values between 0 (totally different) and 1 (identical). A corrected version of the Rand index has been proposed in order to correct the possibility that two clusterings are equal because of randomness. This Adjusted Rand index is defined as follow:

$$ARI(C_1, C_2) = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}} \tag{1.12}$$

With n_{ij} the number of elements in common between the i-th cluster of C_1 and the j-th cluster of C_2, $a_i = \sum_j n_{ij}$ and $b_j = \sum_i n_{ij}$.

This multiplicity of criterion brings to light that there are several challenges to adress while performing clustering, and there is to date no method which can tackle all of them at once. Thus, the methods which have emerged have their own strengths and weaknesses. The next section presents an overview of the most well-known clustering algorithms used today with their specificities.

1.3 Most used clustering methods

In this section is presented a non-exhaustive selection of clustering algorithms which are the most commonly met in practice. Since the creation of the field, many methods have been proposed, and some of them share basic ideas which are then developed in different ways. Thus this section will be divided following the main categories of existing algorithms with at least one example for each kind. These categories are the following:

- Hierarchical methods
- Vector quantization methods
- Density based methods
- Stochastic methods
- Other methods

The categorization presented here only pretends to give to the reader an overview of the most commonly met clustering algorithms. Some other categorization may be found in the literature, as the one presented in [22], in [47] and most recently in [13].

1.3.1 Hierarchical methods

This kind of algorithm is the first to appear in 1963 with the method presented in [44]. Its core idea is to build a hierarchy of clusters by joining existing clusters (agglomerative methods, each data point starts in its own cluster) or by separating them (divise methods, all the dataset belongs to the same initial cluster). The point of the method is to iteratively build a dendrogram by looking at a specific criterion. A method is defined both by the way the dendrogram is built and by the criterion used to pair or divise the clusters. One can either apply the method until all points belong to the same cluster and graphically determining the best number of clusters, or apply the method until a specific criterion is reached (number of cluster or inter-cluster distance for example).

The first most commonly met hierarchical clustering method is the Agglomerative method presented in [44] which pairs clusters by considering a criterion to optimize. The way the dendrogram is built being specified, the variations of the algorithm depends on the criterion which is used to pair the clusters. The most used criterions are the Ward's function (defined in [44]) which aims at minimizing the intra-cluster variance at each step, and the linkage functions which consider a specific distance inter-cluster to minimize. These latter can be either the maximum or the minimum or the mean distance between points of two clusters (respectively called the complete-linkage, the single-linkage and the average linkage clusterings). There exists some other criterion which are not detailed here, however the interested reader can refer to [28] which establishes a detailed survey on the criterion used for Agglomerative clustering. A general framework for an agglomerative hierarchical clustering algorithm is presented in Algorithm 1.

Algorithm 1: General framework of a hierarchical agglomerative clustering algorithm

Initialization: Create a cluster for each element
Each cluster becomes a leaf for the dendrogram
while *there is more than one cluster left* **do**
    Compute pairwise inter-clusters similarities Merge the two most similar clusters Update the dendrogram
end
Cut the dendrogram to get the desired number of cluster

The second most commonly met Hierarchical method is the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) one [48]. This method has been defined in a context of growing interest for data mining of large dataset, and its main claim is to reduce the consumption of ressources used during the clustering, as it presents a compact representation of a dataset and only requires a single path through the dataset to create the tree.

1.3.2 Vector quantization methods

Vector quantization methods are based on the idea that a clustering can be summarized by a set of prototype vectors, thus the belonging of a datum to one cluster is determined by comparing it to the set of existing prototypes. There are two main kinds of Vector Quantization methods: these based on deterministic belongings for which a data point belong to one and only one cluster, and these based on stochastic belongings for which belongings are expressed as a set of probabilities, one per prototype.

The Deterministic Vector Quantization methods are historically the first ones that have been created. The very first method, still used today, is the K-means algorithm [26]. This method is based on a pair of steps done iteratively until convergence of the prototypes. First a set of K prototypes is randomly generated in the feature space, then each data point is associated with its closest prototype. When all the belongings have been determined, the prototypes are updated as the means of their associated data points: they become the "center" of the cluster they represent. The last two steps are repeated until the prototypes updates norms are sufficiently small to be considered as negligeable. The initial method has allowed the emergence of many variations such as k-medians clustering [21], k-medoids clustering [23] depending on the prototype update rule. An extension to this algorithm has been proposed in [25] in which the authors define the Self-Organizing Maps (SOM). This methods is based on the same idea than K-means, except that the prototypes are generally organized as a map in which each node update will also impact its neighbors, depending on a neighboring function based on the Manhattan distance between two nodes in the map. This allows to not only capture prototypes, but also to organize them following a possible underlying structure in the dataset. This methods also make it possible to get a two or three dimensions representation of a possibily high feature space. Indeed in a supervised case, by assigning to each node the class of the majority of its associated data points, it it possible to graphically see the underlying proximity structure in the dataset. The k-means algorithm is illustrated in Algorithm 2.

Deterministic Vector Quantization ideas have been adapted in a stochastic context: now, a data point does not belong to a single cluster, but rather is defined by a set of K belonging probabilities, with K the total number of clusters. A comparision can be made between the two step algorithm used by K-means and the Expectation-Maximization (EM) algorithm defined in [8] for which the update of parameters are performed using the likelihood of data being generated by the model using these parameters. In the first case, the method optimizes the variance of the distance between the prototypes and its associated points, in the second case the method optimizes the likelihood of the dataset being generated by the clustering model. The EM method thus makes it possible to define a stochastic counterpart to K-means called Gaussian Mixture Model.

Algorithm 2: K-Means algorithm

Initialization: Choose a value for K
Randomly initialize the K centroids μ_i
while *a stopping criterion is not met* **do**
  forall $x_n \in X$ **do**
    Assign x_n to the cluster c_i with the closest centroid
    $s_{n,i} = \begin{cases} 1 & \text{if} \quad i = argmin_i \|x_n - \mu_i\|^2 \\ 0 & \text{otherwise} \end{cases}$
  end
  forall μ_i **do**
    $\mu_i = \frac{\sum_n x_n \cdot s_{n,i}}{\sum_n s_{n,i}}$
  end
end

For this method, one considers that the data have been generated by a set of Gaussian functions. The EM method here make it possible to learn the the mean, the variance and, in the multidimensional case, the covariance matrix of the functions.

Considering stochastic belongings, it is also possible to draw some links between deterministic clustering methods and their stochastic variants. Thus, Fuzzy C-Means [2] can be viewed as the stochastic variant of K-means for which the optimized criterion uses weighting coefficients equals to the stochastics belongings defined by the method. Beside, the Generative Topographic Maps (GTM) [3] can be viewed as the SOM stochastic variant. However, while Fuzzy C-Means only added the stochastic belongings to the initial model, GTM uses the concept of latent space to define the prototype map, map which is then mapped into the feature space using a transformation which parameters are trained using the EM method (cf. Figure 1.1). A detailed description of the FCM algorithm can be found in Algorithm 3

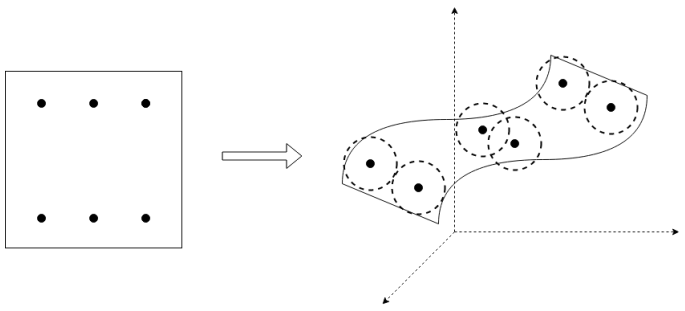

Figure 1.1: Generative Topographic Maps: projection of a map from the latent space to the feature space

A specific emphasis has to be made on this subsection because the methods defined here are later used in Section 1.4 when Collaborative Clustering is

Algorithm 3: Fuzzy C-Means algorithm

Initialization: Choose a value for C and m

Randomly initialize $s_{n,i}^0$ such that $\forall i,\ s_{n,i} \in [0,1], \quad \forall n \sum_{i=1}^{C} s_{n,i} = 1$ r = 0

do

  forall μ_i **do**

    $\mu_i = \frac{\sum_n x_n \left(s_{n,i}^{r-1}\right)^m}{\sum_n x_n \left(s_{n,i}^{r-1}\right)^m}$

  end

  for $n \in [1..N]$ **do**

    for $i \in [1..C]$ **do**

      $s_{n,i}^r = \sum_{j=1}^{C} \left(\frac{d(x_n,\mu_i)^2}{d(x_n,\mu_j)^2}\right)^{-\frac{2}{m-1}}$

    end

  end

  $r = r + 1$

while *a stopping criterion is not met*;

presented.

1.3.3 Density based methods

Even though hierarchical and vector quantization methods have been investigated since the emergence of clustering, the problem of the initialy defined number of cluster remains because it has to be defined manually by the user. The density-based methods presented in this section present alternative solutions to this problem by using the density of data points in the feature space to automatically determine the optimal number of clusters during the learning process (w.r.t. the optimized criterion).

The most commonly met density based method is called Density-based spatial clustering of applications with noise (DBSCAN) [12]. Its core idea is to consider the density around each data point, namely their numbers of neigbors, because a cluster can also be seen as a densely populated point in the feature space. To do so, the method requires two parameters: a distance *varepsilon* which determines if two points are neighbors, and *minPts* which determine if a data point should be considered as a core sample (intuitively, a sample in the near the cluster center) or a non-core sample (at the fringe of the cluster, where the space is less densely populated). Starting from a random point, then iteratively: if it has at least *minPts* neighbors, it is considered as a core point and all data points such as their distances to the original point is less than *varepsilon* are analyzed. If it has less than *minPts* neighbors: either it has a core sample in its neighborhood, in which case it is considered a non-core sample belonging to the same cluster as its core neighbor, or in the opposite case, it is considered as an outlier belonging to no cluster. By doing so, the method defines its own number of cluster depending on *varepsilon* and *minPts*. A description of the DBSCAN algorithm can be found in Algorithm 4.

The second most commonly met density based algorithm is the Mean-shift method defined in [5]. The core idea of the method is to consider data points

Algorithm 4: DBSCAN algorithm

Function `DBSCAN(`X,ε,$minPts$`):`
- $C = 0$
- **forall** $x_n \in X$ **do**
  - **if** x_n *has not been visited yet* **then**
    - mark x_n as *visited*
  - **end**
  - V_n = regionQuery (x_n,ε)
  - **if** *sizeOf* (V_n) $\geq$ *minPts* **then**
    - $C = C + 1$
    - expandCluster (x_n, V_n, C, ε, minPts)
  - **end**
  - **else**
    - mark x_n as noise
  - **end**
- **end**

Function `expandCluster(`x_n, V_n, C, ε, $minPts$`):`
- Add x_n to cluster C
- **forall** $x_i \in V_n$ **do**
  - **if** x_i *has not been visited* **then**
    - V_i = regionQuery (x_i, ε)
    - **if** *sizeOf* ($V_i > minPts$) **then**
      - $V_n = V_n \cup V_i$
    - **end**
  - **end**
  - **if** x_i *does not belong to any cluster yet* **then**
    - Add x_i to cluster C
  - **end**
- **end**

Function `regionQuery(`x_n, ε`):`
- list = $\emptyset$ **forall** $x_i \in X$ **do**
  - **if** $i \neq n$ *and* $d(x_n, x_i) \leq \varepsilon$ **then**
    - list = list $\cup \{x_i\}$
  - **end**
- **end**
- **return** *list*

has mobile points which will then be attracted iteratively by high-density zones in the feature space. It is based on a kernel function K which will define the attraction of a point on its neighborhood. At each iteration, each data point will be update as the mean of its neighbors weighted by their respective attractions (defined by K). By definition, the method is entirely defined by the kernel function and by its parameters.

1.3.4 Special case: Spectral clustering

In this subsection we present the method called Spectral clustering and defined in [29]. It is not presented in one of the previous section because its core idea does not fit with these already defined, but also because it is made of two steps, one to reduce the problem dimensionality, and the second one to effectively cluster the dataset. The second step can be done using by any algorithm defined previously. The core idea of this method is to use the eigenvalues of a similarity matrix computed on the dataset to reduce the dimensionality of the future space using the associated eigen-vectors.

Now that the most commonly met clustering methods have been presented, a specific subfield of the clustering paradigm, namely the Collaborative Clustering, is presented in the next section.

1.4 Collaborative Clustering

1.4.1 Multi-view learning

So far have been presented methods which relies on the learning of a whole dataset. However, it is possible to find cases in real life for which one does not have access to the totality of the dataset, because of privacy constraint for example (see Section 1.1 for further explanation). In some other cases, data may come from different heterogeneous sources. In both cases, it is at least hard, at most impossible to train a single algorithm on the totality of available data. This acknowledgment makes it possible to define a new clustering context, called the Multi-View Learning. In this latter, many methods have to be trained together before being combined in order to achieve the goal initially set.

As presented in [42], Multi-View Learning is a vast domain with many related subfields such as dimensionality reduction, semi-supervised and supervised learning and active learning. However, the following section is only focused on the application of Multi-View Learning in a clustering context, and more precisely on the collaborative clustering paradigm.

1.4.2 Cooperative versus Collaborative Learning

In the context of multi-view clustering, the methods proposed in the litterature present solutions to two main problems:

1. Improve a global clustering knowing the clusterings performed on each view.

2. Making the views collaborate to improve each local clustering.

As defined in [6], the first subfields corresponds to Cooperative Clustering while the second one is called Collaborative Clustering. While these fields may seem closely related, their fundamentals differ in some points.

Cooperative Clustering aims at making several clusterings of the same dataset by several different clustering methods. When each training is finished, the results are sent to a supervisor which task is to find a consensus between all the possible clusterings. Each training is done without inter-views communication, and the results are shared only during the final consensus phase performed by the supervisor. This consensus is found using different combination methods, as presented in [24, 9]. Several methods have been presented in the litterature, the most known and used being Bayesian averaging [24], Bagging [4] and Boosting [14].

Knowing this, Collaborative Clustering differs in many points from its Cooperative counterpart, the most important one being the its final goal: while Cooperative Clustering aims at find the best consensus among a set of clustering, Collaborative clustering aims at improving each local clustering knowing the results currently achieved by each other view. Moreover, the datasets used in each views does not have to be the same, this way Collaborative Clustering can work on heterogeneous data. Another difference lies in the inter-views communication: while each traning was conducted independently with Cooperative Clustering, here each view has to communicate its intermediate results to its pairs in order to improve each local result all along the training.

From now, the continuation of this manuscript is focused only on the Collaborative Clustering.

1.4.3 Horizontal and Vertical

As previously presented, Collaborative Clustering makes possible to make several views to learn different datasets while making them collaborate. This general idea has two different declinations, depending on the similarities that the datasets have to share despise all. These paradigms have been defined in [32] and in [17].

If all the datasets contain different individuals represented by the same set of features, the collaborative clustering is called vertical. On the opposite, if all the datasets contain the same set of individuals described by different sets of features, the clustering is called horizontal (cf Figure 1.2). A third paradigm, called hybrid, is a combination of the two previously mentioned. In the work presented here, we have been interested in the horizontal version of Collaborative Clustering because it makes possible to tackle the problem of heterogeneous data coming from different sources. The related works presented in the next Section are all related to horizontal clustering, unless the opposite is explicitly mentionned.

1.4.4 Theory

Collaborative Clustering methods are based on two distinct phases introduced in [30]: a first phase of local clustering, during which each view produces a model of its data independently of the other views, and a second called collaborative

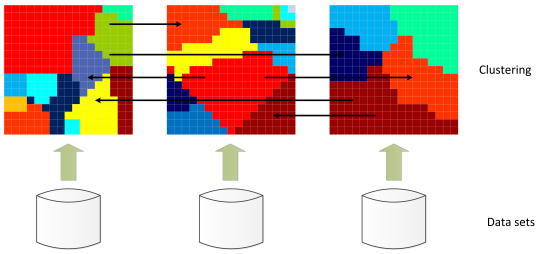

Figure 1.2: Horizontal Collaborative Clustering

during which the views exchange the information they have acquired during the first phase in order to continue to learn from what their pairs have found. These two steps are the first theoretical point defininf a Collaborative Clustering method.

Moreover, in Machine Learning, a method is usually designed around a function to optimize, called either cost function or criterion or even score. Collaborative Clustering consists in several algorithms collaborating, so each one has its own score to optimize. To formalize this idea mathematically and conjointly with the definition of the two steps mentioned above, two cost functions are computed per view.

The first one is the usual cost function of the local methods used in each view. During the first local phase, each method produces a model according to this function. The second one takes into consideration the information learned by each view to formalize the idea of concensus between the views. Formally, the criterion of the i-th view can be written as follow:

$$Q^i = Q^i_{local}(V_i) + Q^i_{collab}(V_{j \neq i}) \tag{1.13}$$

With Q_{local} being the local criterion used to trained the local model of the i-th view, and Q_{collab} being a term appended to the local criterion to formalize the collaborative part of the training. V_i stands for i-th view, and from now, we define N as the total number of views. In most of the literature, both local and collaborative terms have to be defined accordingly to the local methods which are used, even if some researches have been done to make possible to use different local methods. A summary is given in the next Section. When the cost function is defined, its differenciate is computed and then used to generate rules to update the parameters of the system using gradient descent:

$$\omega_{new} = \omega_{old} - \varepsilon \frac{\partial Q}{\partial \omega} \tag{1.14}$$

With ω the parameter to update and ε a fixed parameter defining the learning rate.

1.4.5 Methods presented in the literature

Fuzzy C-Means

Historically, the first version of Collaborative Clustering has been presented in [17] and in [31] and is based on Fuzzy C-Means [2]. During the local training phase, the methods is using the following cost function:

$$\forall j \in [1..N], \quad Q^i_{local} = \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} \left(s^i_{n,k}\right)^2 |x_n - \mu^i_k|^2 \tag{1.15}$$

With $s^i_{n,k}$ being the responsibility of the k-th cluster for the sample n, μ^i_k the center of the k-th cluster, and $|\cdot|$ being the distance between two data points. After the local training, the collaborative phase is performed: each view exchanges its responsibility matrix $S = (s_{n,k})$ with its pairs. During this phase, each view has to minimize the following function still using the FCM algorithm:

$$\forall j \in [1..N], \quad Q^i = Q^i_{local} + Q^i_{collab} \tag{1.16}$$

$$= Q^i_{local} + \sum_{j \neq i}^{N} \alpha_{i,j} \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} \left(s^i_{n,k} - s^j_{n,k}\right)^2 |x_n - \mu^i_k|^2 \tag{1.17}$$

with $\alpha_{i,j}$ being a weight defining the importance of the collaboration between the views i and j. The point of the collaborative term of this function is to formalize the distance between the allocation vector of each sample. To better understand the use of this term, it is necessary to observe its behaviour in two extreme cases: when the responsibilities are roughly equal and when they are not. If all the responsibilities are approximately equal for two views, their respective clusterings are approximately equivalent, meaning that a concensus has been found, thus the collaborative term and the cost function are low. On the opposite, if the responsibilities are totally different, the sum of each absolute difference makes the term important, and so improvable. By considering these two cases, one can understand that the final goal of the horizontal collaborative clustering is to find the best concensus as possible between all the views.

Self Organizing Maps

A second version of collaborative clustering has been proposed using self organizing maps [25]. It is based on the same idea than previously defined, and has been presented in [17]. An online version, based on the same cost function but with a different temperature function, has also been proposed in [27]. A SOM is defined by a temperature function λ which itself defines a neighborhood function K. To define the local and collaborative terms of the cost function, K is used instead of the responsibilities s, which gives the following equations for the i-th view:

$$\lambda(t) = \lambda_{\min}\Big(\frac{\lambda_{\min}}{\lambda_{\max}}\Big)^{\frac{1}{t}} \tag{1.18}$$

With $\lambda_{\min}$ and $\lambda_{\max}$ two parameters of the SOM algorithm. This function is then used to define the following neighborhood function and the corresponding local and collaborative terms:

$$K_{k,l} = \exp\Big(-\frac{d^2(k,l)}{\lambda(t)}\Big) \tag{1.19}$$

$$Q^i_{local} = \alpha_i \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} K^i_{k,\chi(x_n)} \|x^i_n - \omega^k\|^2 \tag{1.20}$$

$$Q^i_{collab} = \sum_{j \neq i} \beta_i^j \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} \left(K^i_{k,\chi(x_n)} - K^j_{k,\chi(x_n)}\right) \|x^i_n - \omega^k\|^2 \tag{1.21}$$

With $d(k,l)$ being the topological distance (number of hops) between the neurons k and l in the SOM, α_i being the weighting coefficient for the local view i, $\chi(x)$ being the function returning the best matching unit (MBU) of a sample, namely the neuron of the map which is the nearest from the data point x, and ω^k is the k-th neuron of the SOM.

The learning process is the same than for the FCM version: first, each local view learns a model of its data using the standard SOM algorithm, then the neurons are updated during the collaborative phase following the cost function defined by 1.20 and 1.21.

The collaborative clustering has also been applied using the Generative Topographic Mapping [3] presented is Section 1.3.2.

Generative Topographic Mapping

The GTM being considered as the stochastic evolution of the SOM, the use of the former as the local clustering method for Collaborative clustering instead of the latter has naturaly been studied and has been presented in [15], [38], [39] and [40]. The learning algorithm relies on the same principle than the SOM-based one, but this time using the Expectation-Maximization algorithm [8]. During the Expectation phase each datum is assigned to a neuron. Then during the Maximization phase, the neurons are updated using a penalized likelihood as described in [16].

1.4.6 Limitations and related works

The methods presented so far all share several limitations. First of all, the same algorithms have to be used in each view to make the collaboration possible. Moreover, because these algorithms are prototype based and because the collaboration phase relies on a comparison between the prototypes, the number of these latter has to be the same among all the views. An even more important restriction is that the maps defined by the SOM and the GTM algorithms have to be closed to each other in order to be compared. Additionally, these methods rely on the weighting coefficients α and β in order to select what collaborator should be favored or not during the learning process. These parameters have to be chosen closely, and this may become a tricky task if many views have to be considered.

Hopefully, several works have been conducted in order to remedy this situation. In [18] and in [34], the impact of the diversity of the solution found by each local view on the collaboration is studied. Even if these work do not provide a way to automatically update α and β, it gives an intuition of what is important for a specific view when considering its external pairs. An automatic update of the collaborative weights is presented in [17] and in [19], also based on the gradient descent. However, this method still presents a significant constraint ($\beta = \alpha^2$), which considerably simplify the update of the weight to the detriment of the genericity of the method. In a recent article, a new entropy-based

method is presented [41] has the double advantage to present a generic method to update the weighting coefficients β (α not being used in this method) while being usable for any combination of local clustering method. This has been made possible because of the sharing of the results of each clustering (either deterministic or stochastic) instead of an intermediary information such as the neighborhood functions for the SOM. This work is based on the researches previously presented in [37].

Bibliography

[1] Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A working set of axioms for clustering. In *Advances in neural information processing systems*, pages 121–128, 2009.

[2] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[3] Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Gtm: The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998.

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[5] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[6] Antoine Cornuejols, Cédric Wemmert, Pierre Gançarski, and Younès Bennani. Collaborative clustering: Why, when, what and how. *Information Fusion*, 39:81–95, 2018.

[7] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[9] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[10] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[11] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[13] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing*, 2(3):267–279, 2014.

[14] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[15] Mohamad Ghassany, Nistor Grozavu, and Younès Bennani. Collaborative generative topographic mapping. In *International Conference on Neural Information Processing*, pages 591–598. Springer, 2012.

[16] Peter J Green. On use of the em for penalized likelihood estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 443–452, 1990.

[17] Nistor Grozavu and Younes Bennani. Topological collaborative clustering. *Australian Journal of Intelligent Information Processing Systems*, 12(2), 2010.

[18] Nistor Grozavu, Guenael Cabanes, and Younes Bennani. Diversity analysis in collaborative clustering. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 1754–1761. IEEE, 2014.

[19] Nistor Grozavu, Mohamad Ghassany, and Younes Bennani. Learning confidence exchange in collaborative clustering. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 872–879. IEEE, 2011.

[20] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[21] Anil K Jain and Richard C Dubes. Algorithms for clustering data. 1988.

[22] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[23] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[24] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998.

[25] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.

[26] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[27] Denis Maurel, Jérémie Sublime, and Sylvain Lefebvre. Incremental self-organizing maps for collaborative clustering. In *International Conference on Neural Information Processing*, pages 497–504. Springer, 2017.

[28] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[29] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[30] Witold Pedrycz. Collaborative fuzzy clustering. *Pattern Recognition Letters*, 23(14):1675–1686, 2002.

[31] Witold Pedrycz. Fuzzy clustering with a knowledge-based guidance. *Pattern Recognition Letters*, 25(4):469–480, 2004.

[32] Witold Pedrycz. *Knowledge-based clustering: from data to information granules*. John Wiley & Sons, 2005.

[33] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[34] Parisa Rastin, Guénaël Cabanes, Nistor Grozavu, and Younes Bennani. Collaborative clustering: How to select the optimal collaborators? In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 787–794. IEEE, 2015.

[35] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[36] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[37] Jérémie Sublime. *Contributions au clustering collaboratif et à ses potentielles applications en imagerie à très haute résolution*. PhD thesis, Paris Saclay, 2016.

[38] Jérémie Sublime, Nistor Grozavu, Younes Bennani, and Antoine Cornuéjols. Vertical collaborative clustering using generative topographic maps. In *Soft Computing and Pattern Recognition (SoCPaR), 2015 7th International Conference of*, pages 199–204. IEEE, 2015.

[39] Jérémie Sublime, Nistor Grozavu, Guénaël Cabanes, Younès Bennani, and Antoine Cornuéjols. From horizontal to vertical collaborative clustering using generative topographic maps. *International Journal of Hybrid Intelligent Systems*, 12(4):245–256, 2015.

[40] Jérémie Sublime, Nistor Grozavu, Guénaêl Cabanes, Younès Bennani, and Antoine Cornuéjols. Collaborative learning using topographic maps. In *AAFD and SFC'16 Conférence Internationale Francophone" Science des données. Défis Mathématiques et algorithmiques"*, page np, 2016.

[41] Jérémie Sublime, Denis Maurel, Nistor Grozavu, Basarab Matei, and Younès Bennani. Optimizing exchange confidence during collaborative clustering. In *2018 International Joint Conference on Neural Networks, IJCNN 2018*, 2018.

[42] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[43] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[44] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[45] Cédric Wemmert. *Classification hybride distribuée par collaboration de méthodes non supervisées*. PhD thesis, Université Louis Pasteur (Strasbourg), 2000.

[46] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[47] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[48] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.