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 [31], 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 [69] and K-means [38] 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 [14].

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 [67]. 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 [72] and Locally Linear Embedding [52] 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 [15], the Davies-Bouldin index [9], the Silhouette index [51] the Wemmert-Gancarski Index [70]. Some measures even allow to compare two clusterings, namely the Rand index [47] and its Adjusted version [29]. 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:

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

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:

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

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 [31], in [73] and most recently in [18].

1.3.1 Hierarchical methods

This kind of algorithm is the first to appear in 1963 with the method presented in [69]. 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 [69] 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 [69]) 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 [41] 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 [74]. 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 [38]. 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 [30], k-medoids clustering [32] depending on the prototype update rule. An extension to this algorithm has been proposed in [34] 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 [12] 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 $\mu_i$
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 $\mu_i$ 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 [3] 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) [4] 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

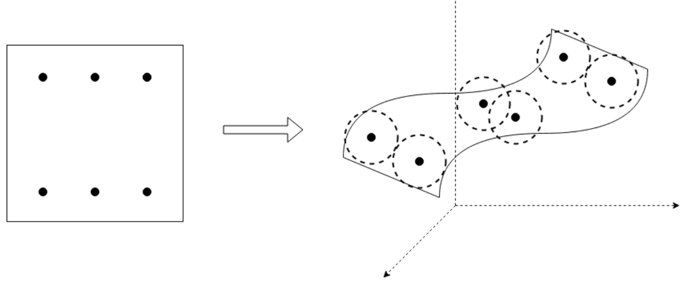


Figure 1.1: Generative Topographic Maps: projection of a map from the latent space to the feature space

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 $s^0_{n,i}$ 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 $\mu_i$ do
        $\mu_i = \frac{\sum_n x_n \left(s^{r-1}_{n,i}\right)^m}{\sum_n x_n \left(s^{r-1}_{n,i}\right)^m}$
    end
    for $n \in [1..N]$ do
        for $i \in [1..C]$ do
            $s^r_{n,i} = \sum_{j=1}^{C} \left(\frac{d(x_n,\mu_i)^2}{d(x_n,\mu_j)^2}\right)^{-\frac{2}{m-1}}$
        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) [17]. 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 [6]. The core idea of the method is to consider data points

Algorithm 4: DBSCAN algorithm

```
Function DBSCAN($X$,$\varepsilon$,$minPts$):
    $C = 0$
    forall $x_n \in X$ do
        if $x_n$ has not been visited yet then
            mark $x_n$ as visited
        end
        $V_n$ = regionQuery ($x_n$,$\varepsilon$)
        if sizeOf ($V_n$) $\geq$ minPts then
            $C = C + 1$
            expandCluster ($x_n$, $V_n$, $C$, $\varepsilon$, minPts)
        end
        else
            mark $x_n$ as noise
        end
    end
Function expandCluster($x_n$, $V_n$, $C$, $\varepsilon$, $minPts$):
    Add $x_n$ to cluster $C$
    forall $x_i \in V_n$ do
        if $x_i$ has not been visited then
            $V_i$ = regionQuery ($x_i$, $\varepsilon$)
            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($x_n$, $\varepsilon$):
    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 [42]. 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 3.1 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 [65], 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 [7], 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 [33, 13]. Several methods have been presented in the litterature, the most known and used being Bayesian averaging [33], Bagging [5] and Boosting [19].

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 [45] and in [23].

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 [43]: 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

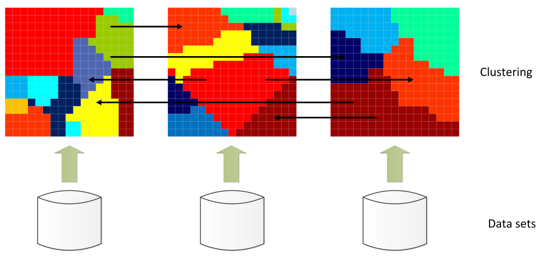


Figure 1.2: Horizontal Collaborative Clustering

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 [23] and in [44] and is based on Fuzzy C-Means [3]. During the local training phase, the methods is using the following cost function:

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

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:

$$\begin{aligned} \forall j \in [1..N], \quad Q^i &= Q^i_{local} + Q^i_{collab} &(1.16)\\ &= Q^i_{local} + \sum_{j \neq i}^{N} \alpha_{i,j} \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} (s^i_{n,k} - s^j_{n,k})^2 |x_n - \mu^i_k|^2 &(1.17) \end{aligned}$$

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 [34]. It is based on the same idea than previously defined, and has been presented in [23]. An online version, based on the same cost function but with a different temperature function, has also been proposed in [39]. 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:

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

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:

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

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

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

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 [4] 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 [21], [55], [56] and [57]. The learning algorithm relies on the same principle than the SOM-based one, but this time using the Expectation-Maximization algorithm [12]. 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 [22].

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 [24] and in [48], 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 [23] and in [26], 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 [64] 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 [54].

Chapter 2

Collaborative Clustering

2.1 Introduction

This work has been presented and accepted for publication in WCCI2018 [64].

Within the context of Collaborative Clustering, this chapter is concerned with proposing alternative methods to ensure that the collaboration of unsupervised algorithms leads to improved results. To do so, we will focus on a specific family of unsupervised collaborative frameworks based on topological maps: the collaborative version of the SOM algorithm for which several variations exist based on either SOM or its GTM evolution [23, 20, 58].

The contributions presented in this chapter are 3-folds:

- We propose an entirely automated and unsupervised optimization method to adjust the strength of the collaboration between algorithms collaborating together via the SOM algorithm.
- We experimentally demonstrate that our optimization method can efficiently be used to detect noisy views that would otherwise deteriorate the quality of a collaboration between algorithms.
- We give the theoretical properties of our proposed model. In particular, we show that our optimization method results in applying a meta-clustering on the different views, thus grouping them according to their similarities.

To do so, we propose an optimization method of the collaborative process likelihood function using Karush-Kuhn-Tucker optimization [37], and we interpret the results found for the algorithms weights in term of how they evolve based on criterion such as the stability and diversity of the partitions. This work can be compared with earlier studies on the influence of quality and diversity in collaborative clustering [26, 49, 25, 60]. It is an improvement upon these works in the sense that we give a mathematical justification and the theoretical properties of our weighting method, and we remove an user input parameter from the optimization constraints [60] which makes this chapter's results more generic. Furthermore, our experimental section goes deeper into the analysis of the effects of weighting views, and shows the ability of our method to detect noisy views.

The remainder of this chapter is organized as follows: Section 2.2 is devoted to the description of our proposed optimization method, as well as the interpretation of the proposed weights formulas, in Section 2.3, some numerical experiments are proposed and analyzed. Finally, Section 2.4 presents a conclusion on the work presented in this chapter.

2.2 Optimized weights for the horizontal collaborative SOM algorithm

2.2.1 Optimization problem

In this chapter we propose a different collaborative approach in which we modify the objective function (**??**) using a weighting strategy to reduce the risk of negative collaboration: we study how to optimize the weights of the combination function in Equation (**??**) can lead to an optimal value of the global function and reduce the risk of negative collaboration by further optimizing weight factors between the algorithms.

We begin by changing the equations of the α by $\alpha_j = 1$. We believe that it makes a lot more sense than the proposed square root which is never justified in the related works [23, 26]. This helps us to properly evaluate the balance between the local and the collaborative term, for which 1 variable is enough. We are therefore mainly interested in finding the positive weights $\beta_{i,j}$ that will determine the strength of the collaborative term. Moreover, as we restrict our study to the case of horizontal clustering, we would like to mention that in our theoretical model all data sets describe the same observations and all these collaborative data sets have the same number of observations but a different number of variables.

For fixed local maps w, our strategy to minimize Equation (**??**) is to minimize the second term. Indeed, since the collaboration weights are only in the collaborative term and because the local likelihood are fixed, we can ignore the local term in Equation (**??**).

The minimization of the collaborative term is based on the dual form of the problem. We do so under the Karush-Kuhn-Tucker conditions (KKT) [37] assuming that the weights $\beta_{i,j}$ respect the following conditions:

$$\forall j \quad \prod_{i \neq j}^{J} \beta_{i,j} = 1$$

$$\forall (i,j) \quad \beta_{i,j} > 0$$

Note that we use a product constraint instead of a sum. While it may seem unusual, it has already been used in related works on multi-view clustering [10]. Furthermore, in [60] it has been demonstrated that the constraint $\sum_{i \neq j}^{J} \beta_{i,j} = 1$ would lead to an unsatisfying result and that an extra parameter p (that needs to be learned) is needed to make it work with collaborative clustering. To simplify the presentation, in what follows we omit the dependency of $C_{i,j}$ to w in our notations.

The results of the optimization under the Karush-Kuhn-Tucker conditions are shown below in Equations (2.1). For all $i \neq j$ we have:

$$\beta_{i,j} = \frac{\left(\prod_{k \neq j} C_{k,j}\right)^{\frac{1}{J-1}}}{C_{i,j}} \tag{2.1}$$

The interpretation of these results is the following: in the context of horizontal collaborative clustering, the global results should be better if each individual algorithm gives higher weights to algorithms that have the most similar solutions compared with the local one (high $\beta_{i,j}$ value for a given $\mathcal{A}^j$).

In other words: from Equation (2.1), each algorithm would mostly collaborate with the algorithms that have the most similar solutions (small $C_{i,j}$). If several algorithms have the same most similar solution, they would be given the same weight. The algorithms with the most similar solutions would still be favored to optimize the cost function of the global collaborative framework. But algorithms whose solutions have a lesser degree of similarity would still be taken into consideration locally.

Let us detail the calculus for the KKT optimization problem. Given the $C_{i,j}$, we optimize the matrix $\beta = {\beta_{i,j}}_{J \times J}$ as shown in the system below:

$$\begin{cases} \beta^* = \operatorname{argmin}_\beta \sum_{i \neq j} \beta_{i,j} C_{i,j}, \\ \forall j \quad \prod_{i \neq j}^{J} \beta_{i,j} = 1. \\ \forall (i,j) \quad \beta_{i,j} > 0 \end{cases} \tag{2.2}$$

This is optimization under constraints problem. To solve this problem we use Lagrange multipliers. From $\forall j \quad \prod_{i \neq j}^{J} \beta_{i,j} = 1$, we obtain $\forall j \quad \sum_{i \neq j}^{J} \ln \beta_{i,j} = 0$. The Lagrangian then writes:

$$L(\beta, \nu, \lambda) = \sum_{j=1}^{J} \sum_{i \neq j}^{J} (\beta_{i,j} C_{i,j} - \nu_j \ln \beta_{i,j} - \lambda_{i,j} \beta_{i,j}). \tag{2.3}$$

From the definition of the Lagrangian, we get the following KKT conditions:

$$\forall (i,j), i \neq j \begin{cases} (1) \quad \beta_{i,j} > 0, \\ (2) \quad \prod_{i \neq j}^{J} \beta_{i,j} = 1, \\ (3) \quad \lambda_{i,j} \geq 0, \\ (4) \quad \beta_{i,j} \lambda_{i,j} = 0, \\ (5) \quad C_{i,j} - \frac{\nu_j}{\beta_{i,j}} - \lambda_{i,j} = 0. \end{cases} \tag{2.4}$$

Let's begin by considering the case where $\lambda_{i,j} > 0$ in (2.4)-4. Then, we would have $\beta_{i,j} = 0$ this case is not possible due to (2.4)-1, therefore we will only consider the case $\beta_{i,j} \neq 0$ and $\lambda_{i,j} = 0$. Then, with (2.4)-5, we have:

$$\beta_{i,j} = \frac{\nu_j}{C_{i,j}} \geq 0. \tag{2.5}$$

From Equation (2.4)-2 and (2.5), we have:

$$\prod_{i \neq j}^{J} \beta_{i,j} = \prod_{i \neq j}^{J} \left(\frac{\nu_j}{C_{i,j}} \right) = \frac{\prod_{i \neq j}^{J} \nu_j}{\prod_{i \neq j}^{J} (C_{i,j})} = 1. \tag{2.6}$$

It follows that:

$$(\nu_j)^{J-1} = \prod_{i \neq j}^{J} C_{i,j}.$$

Then by re-injecting the expression of ν_j into Equation (2.5), we get our claim shown in Equation (2.1).

Our modified version of the SOM algorithm for horizontal collaboration with the optimized weights is shown in Algorithm 5 below. The computational complexity for N data in J views is in $O(NJ)$ since it uses J times the SOM algorithm which is in $O(N)$.

Algorithm 5: Topological horizontal collaboration Algorithm

```
Initialization: Initialize all the map prototypes $W$ randomly.
Local step: Initilization of the maps
forall algorithms $\mathcal{A}^j$ do
    Minimize the objective function of the classical SOM
end
Collaborative step:
forall algorithms $\mathcal{A}^j$ do
    For fixed $w$, compute: $\beta$ using Equation (2.1)
    Update the prototypes of all maps by: $w^* = \text{argmin}_w \mathcal{C}(w, \alpha, \beta)$
end
```

2.2.2 Interpretation

From Equation (2.1), we can infer the following property: For two SOM algorithms $\mathcal{A}^j$ and $\mathcal{A}^i$, when the pairwise collaborative term $C_{i,j}$ is small (i.e. the maps are similar) comparatively with the other collaborative terms $C_{k,j}$, then the associated collaborative weight $\beta_{i,j}$ is large compared with the other $\beta_{k,j}$. The interpretation for this is that any SOM algorithm $\mathcal{A}^j$ should give a stronger collaborative weight to other self-organizing maps with a similar topology, and a weaker weight to the local term. As such, the Equation for the weights $\beta_{i,j}$ is an inverse geometric mean based on the similarity between two maps. Furthermore, with Equation (2.1), we can further interpret, that algorithms with relatively weak collaboration links $\beta_{i,j}$ -with maps very different from all others- whould give a stronger weight to their local term L_i from Equation (**??**) and would not collaborate much with the other algorithms.

These properties are an improvement from an earlier result [60] in a sense that our current model better balances between local and collaborative terms, and requires no extra parameter.

The first conclusion of these results is that in the context of horizontal collaboration between several SOM algorithms, the most efficient way for a given algorithm $\mathcal{A}^i$ to collaborate is to favor exchanges with other SOM algorithms $\mathcal{A}^j$ that have similar topologies and are stable. These results are echoing recent works on clustering stability [2, 68] stating that a clustering is stable if the partitioning remains similar when the data set or the clustering process is perturbed. In our case and within the context of collaborative clustering, if several self-organizing maps have a similar topology despite being drawn from different views, it is a proof of stability. As such, our weighting method favors collaboration between stable maps and marginalizes maps that are too different and may disturb the collaborative process.

The second conclusion of these results is that our proposed optimization method results in a meta-clustering of the views, in which SOM with similar topological maps are grouped into clusters of views with a strong intra-collaboration and a weak inter-collaboration, and in which noisy views are mostly discarded. This last property is the most interesting one because of noisy views being a recurring problem in multi-view and collaborative clustering [8].

2.3 Numerical Results

To evaluate our proposed optimization approach we used several datasets of different size and complexity in a collaborative clustering setting: Waveform, Wisconsin Diagnostic Breast Cancer (wdbc), Isolet, Spambase and VHR Strasbourg.

2.3.1 Data sets

The datasets used in our experiments are from the UCI website: *Waveform data set* (5000×40), *Wisconsin Diagnostic Breast Cancer (WDBC)* (569×30), *Isolet* (1559×617), *Spam Base* (4601×57) and VHR Strasbourg ($187,057 \times 27$). Their respective descriptions can be found in Appendix .1.

2.3.2 Validation criteria

The two main criteria used here were the quantization error (or distorsion, one of the most used criteria to evaluate the quality of a Kohonen's topological map) and the purity (accuracy index).

The quantization error is computed using the following expression:

$$q_e = \frac{1}{N} \sum_{i=1}^{N} \|x^i - w_{i_c}\|^2 \tag{2.7}$$

where N is the dataset size and w_{i_c} is the nearest prototype to the vector x^i. The values of the quantization error depends on the size of dataset and the size of built maps. Strong differencies may therefore arise when dealing with different datasets or Kohonen map sizes.

The purity (accuracy) of the map is equal to the average purity of all the neurons. A good SOM map should have a high degree of the purity index. The purity of a neuron is the percentage of data belonging to its majority class. Knowing the data labels set $L = \{l_1, l_2, \dots, l_{|L|}\}$ and the prototypes set $C = \{c_1, c_2, \dots, c_{|C|}\}$, the formula for the purity of a map is the following:

$$purity = \frac{1}{N} \sum_{k=1}^{|C|} c_k \times \frac{\max_{i=1}^{|L|} |c_{ik}|}{|c_k|} \tag{2.8}$$

where $|c_k|$ is the total number of data associated with the neuron c_k, $|c_{ik}|$ is the number of observations of class l_i which are associated to the neuron c_k and N - the total number of observations (data).

2.3.3 Experimental protocol

To test the validity of our method and to compare it with other state of the art methods, several points have been analyzed.

In a first experiment, we will investigate the evolution of the fitness function (Eq. **??**) with and without our proposed beta-optimization method. For comparison purposes, both criteria have been normalized as follows:

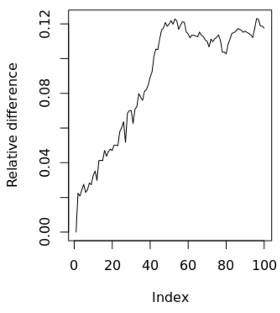
$$\mathcal{C}(w, \beta) = L_j(w) + \sum_{i \neq j} \Big(\frac{\beta_{i,j}}{\sum_{i \neq j} \beta_{i,j}} \cdot C_{i,j}(w) \Big) \tag{2.9}$$

The point of this criterion is to make the $\beta_{i,j}$ act as weighting coefficients summing to 1, allowing to compare both versions of CC with and without β optimization.

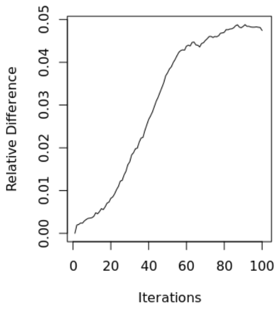
In the second experiment, the values of betas depending on the quality of the view is investigated: in order to analyze the capacity of the method to define which collaborations are useful, a view only made of uniform noise was added to each dataset (except for waveform which already has several features only made of noise). This noisy view was added only for this experiment. For each dataset, we split the data into three (WDBC, Spambase, VHR Strasbourg) or four (Isolet, Waveform) views of equal size, and we added a view of uniform noise. We then learn a SOM map for each database.
The goal was to analyze if the method was able to limit the impact of the noisy view on the learning process of the other views. Because we put the constraint $\prod_{i \neq j}^{J} \beta_{i,j} = 1$ for every view j, the previous assertion would lead to $\beta_{noisy,i} < 1$ for every other view i. In this first experiment, we show that our optimization method is able to detect the noisy view and to mitigate its impact during the learning process by properly weakening the values of the weights linked to it, i.e $\beta_{noisy,j}$ low compared to the others.

Finally, in the last experiment, we analyze the impact of the method on the learning itself, several criteria introduced earlier are presented in Table 2.2. The point of this analysis was to check that the collaborative constraint added during the collaborative phase did not impact the results of the model itself.

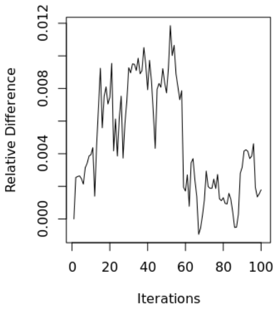
All the experiments were conducted with a 5×5 map. The choice of this size was made heuristically, based on the most appropriate number of neurons which optimize the quantization and topological errors during the local step [23].



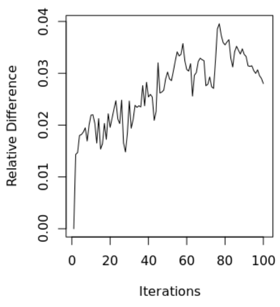
(a) WDBC



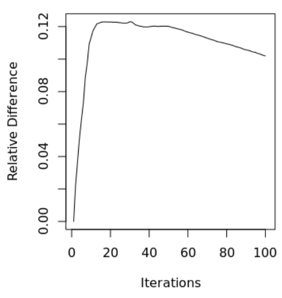
(b) Waveform



(c) Spambase



(d) Isolet



(e) VHR Strasbourg

Figure 2.1: Relative differences of the weighted criterion with and without β optimization all along the learning process

2.3.4 Results

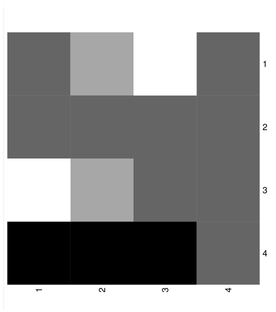
Relative Difference of the criterion The first experiment is about the evolution of the modified criterion presented in Eq. 2.9. Figure 2.1 shows the relative difference (RD) of this criterion between the original version of the collaborative SOM algorithm and our proposed version with the smart weights β. It appears that the relative difference between the criterion is always positive, meaning that the version with the β-weighting always improves the learning compared to the standard version. This can be understood knowing the interpretation in Sec. 2.2.2: the algorithm tends to make views that agree with each others collaborate, so the β-weighting favors lower values of $C_{i,j}$ (better collaboration), improving the global criterion.

However, it also appears that all datasets are not treated equally by this method: the best mean RD goes from 0.4

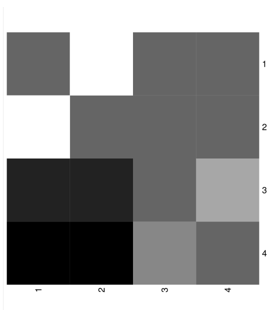
β analysis Now considering β coefficients themselves, Figure 2.2 presents the different β values obtained at the end of the collaboration process for each dataset. Table 2.1 gives their corresponding minimum and maximum values.

To recall, the value $\beta_{i,j}$ can be read as "how much does view j exchanges with view i compared to the others". The values on the diagonal -which are of no importance- are fixed to 1 to make the comparison easier. The last row of each matrix corresponds to the collaboration between each view and the artificially added noisy view of each data set (except for Waveform, for which the two last views were already noisy).

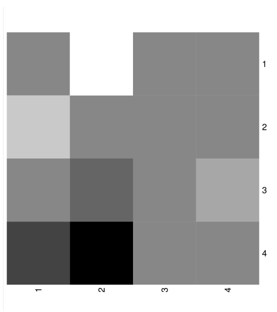
Several points can be mentioned concerning these images. First, as one can see collaborations with noisy views are mostly weak: the method presented here tends to minimize the impact of the collaboration between a useful view and a noisy one. This is particularly clear for the WDBC and VHR datasets where all β on the last row are below 1, while all other factors are at least around 1. Secondly, one can see that the strong collaborations are mostly symmetrical. To continue with the WDBC example, we got $\beta_{1,3} \approx \beta_{3,1} > 1$. However this phenomenon is not true for less strong collaborations: for WDBC and VHR, we got $\beta_{1,2} \neq \beta_{2,1}$ and $\beta_{2,3} \neq \beta_{3,2}$. It appears that the algorithm leads to the creation of subgroups of views: when two views tends to collaborate, their other β are approximately identical. This property can be seen as the continuation of the interpretation given in Sec. 2.2.2: our method will favor the collaboration between agreeing views, leading to the creation of subgroups of views which have the same common behavior towards views outside of their group.



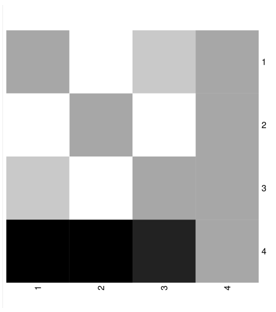
(a) WDBC



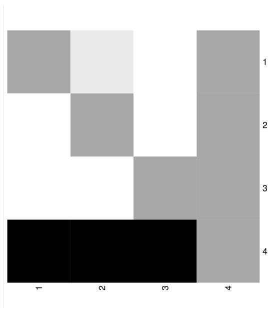
(b) Waveform



(c) Spambase



(d) Isolet



(e) VHR Strasbourg

Figure 2.2: Heatmap of the β matrices for each dataset. Colors go from white (strong collaboration) to black (weak collaboration). The gray color on the diagonal stands for $\beta = 1$.

Table 2.1: Minimum and Maximum values of β got for each dataset.

Dataset	Minimum	Maximum	Difference
WDBC	0.51	2.04	1.53
Waveform	0.75	1.79	1.04
Spambase	0.77	1.37	0.60
Isolet	0.67	1.48	0.81
VHR Strasbourg	0.48	1.63	1.15

Purity and QE analysis The last experiment consisted in the analysis of two criterion commonly met in the Kohonen's map literature, namely the purity and the quantization error. This analysis has been conducted in order to make sure that the collaborative phase and the β weighting did not damage the final result of the learning. Table 2.2 displays the mean values of each criterion for all the views, except the added noisy one. The results shows that our proposed weighting method, while succesful with unsupervised indexes has little to no impact on supervised criterions such as purity or quantization error. This result was to be expected since our proposed optimization does not bring any extra supervision compared to the original one, and it is therefore good already that it does not negatively impact supervised results.

Table 2.2: Experimental results on different datasets

Dataset	Method	Mean Purity	q_e
Wdbc	standard	86.86	4.01
	β-weighting	88.09	3.97
Isolet	standard	55.37	125.32
	β-weighting	54.98	125.2
Waveform	standard	64.27	6.15
	β-weighting	65.40	6.16
SpamBase	standard	80.34	14.56
	β-weighting	80.24	14.54
VHR Stransbourg	standard	48.94	3.37
	β-weighting	48.93	3.38

2.4 Conclusion

In this chapter, we have proposed an optimization method for the case of horizontal collaborative clustering between SOM algorithms. Our proposed method answers several questions regarding the tuning of the collaborative parameters between local and collaborative terms when using topological based collaborative clustering methods. Furthermore, we have also demonstrated how it can be used to make groups of similar maps, and to detect and discard noisy views.

Using our optimization model we have also found interesting properties, and in particular we have shown how diversity can be used to avoid bad influences from noisy or low quality view, and ultimately to improve the results of unsupervised collaborative learning. The conclusion from the theoretical part of this chapter is that a lower diversity is a good criterion to choose collaborators because it tends to favor stable solutions, which is a good thing since stability is a well known good quality criterion to find the intrinsic structures of the data set in unsupervised learning. However, one should keep in mind that the low diversity criterion has its limits and may hinder improvements in the collaborative process due to the lack of risk taking, or lead to no improvement at all if the diversity is too low. These later 2 issues are tackled in another paper co-published with this one and where an alternative bandit optimization scheme is proposed for a similar collaborative clustering problem [59].

Other possible extensions for this work could include similar studies on the case of vertical collaboration where the collaborating SOM algorithms handles different data sharing the same features, as well as the application of the same optimization technique for collaborative Generative Topographic Maps in a first time, and a further extension to non-topological collaborative methods in a second time.

Chapter 3

Collaborative Reconstruction Network

The work presented in this chapter has been submitted to KAIS2018.

The Collaborative Clustering paradigm is based on the prerequisite that each view has to contain a set of common individuals (described by different sets of features for the horizontal case) as big as possible to allow information exchange. However, this paradigm does not consider the case for which the views do not share the same set of individuals. Intuitively, if a view misses an individual in its database, it might be possible to use the information contained in all the other views to get a first approximation of the missing individual. This idea is developped in this chapter with the description of a system capable of recontructing an approximation of a missing individual in a multi-view context.

3.1 Context

Current proliferation of multi-view data in various domains such as marketing, bank administration or even survey analysis, has recently been accompanied by a global security awareness that questions which data should –or more often shouldn't– be made available and shared. This awareness is based on the question of the link between one's intimacy and the processing that is made of one's data. This topic being beyond the scope of this thesis, it will not be further detailed, however it will be used as the fundation of the security constraint which is detailed hereafter. This security problem is particularly relevant in the case of multi-view learning, a speciality of Machine Learning in which algorithms are trained using databases distributed among several independant (but communicating) views. Some multi-view paradigms such as Collaborative Clustering are based on the hypothesis that different views share the same set of individuals, point which makes possible the inter-view results comparison, and so the training and the improvement of each local model. However, this hypothesis does not have to be verified in practice: the presence of an individual in a database does not guarantee its presence in all the other views. In real life cases, it is even more likely that an individual is present in a limited number of views, considering all these available. The question of the use of the available information to infer missing data on an individual may be asked.

Because of the security concern mentionned abose, a solution to the missing data problem should at least be able to reconstruct missing data in the concerned views without sharing of the original data available in each local view. Within this context, this chapter presents a solution to fill in missing pieces of information in a given view by using the data contained in the other views but without any data transfer that may breach security issues. While data reconstruction as already been studied using method such as Collaborative Filtering [35], the work presented here presents the extension of this problem to the multi-view context while also considering the problem of data security.

At this point, it is useful to note that the work presented in this chapter take into account the difference between data privacy and data security. Data privacy is a whole research field which considers the problem of data sharing as well as the use which is done of this data. Currently, the specific field of differential privacy is the subfield of algorithmic which define a theoretical context to estimate the privacy level of a semi-randomized mechanism [16]. On the other side, data security in the context of this thesis is defined as the constraint of not being able to access original data if it is not from its original view. The point of this constraint is to make sure that the reconstruction algorithm which is presented here does not rely either on the transfer of original information, or on the possibility for an external view to have access to the original data from another view. To sum up the difference between these two fields, data privacy ensures that no information on the original data can be retrieved, would it be the original data or labels that may be attached to it, while data security in the context of this thesis just ensures that the original data is available only in its local view.

This point being clarified, the main difficulties of the solution lie in two points: how to transfer usable information in the local view without transferring the original external data, and how to reconstruct more or less reliable information from different sources to get the final result.

To solve these problems, we present a system called the Cooperative Reconstruction System. After encoding the original data using Autoencoders [28] to respect the security issues, the combination of external information is performed using Multi-Layer Perceptrons [53] (called Links in this article) and a smart weighting method presented in this chapter and called Masked Weighting Method. The goal of this latter is to (1) combine the information from different views, (2) reduce the weight of views with information which could hinder the cooperative reconstruction process [62], and (3) reduce the impact of missing data during the unsupervised learning process [11].

3.2 Neural Networks

3.2.1 Introduction

Neural Networks are a specific kind of Machine Learning algorithm based on an analogy of the interaction of the neurons in a human brain. Their history has known many steps the most known being the presentation of the perceptron (a.k.a. a neuron) by Rosenblatt in 1958 [50], the use of the backpropagation algorithm by Werbos in 1975 [71] and the presentation of the deep beliefs networks by Hinton in 2006 [27]. The original version has been modified to produce several types of neural networks, depending on the aim to achieve. The two most famous being Convolutional Neural Network [36] to perform image analysis and the Recurrent Neural Network [40] which are used to analyze temporal data. In this thesis, we are only interested in the Multi Layer Perceptron. The following sections briefly sum up the principal components of a Multi Layer Perceptron (MLP).

3.2.2 A neuron

A MLP is made of several layers of several neurons (see Figure 3.1), each having a set a parameters which are trained during the MLP learning. To get the ouput of a neuron, each feature of the input vector is weighted by a parameter of the neuron, before being summed and put in an activation function to get the final output. Regarding the activation function, the first one which has been used is the sigmoid function, which definition can be found at Equation 3.1. There are many different activation functions which can be used, however the one which tends to be the most commonly met in recent research work is the Rectified Linear Unit (ReLU), which definition is simply $ReLU(x) = \max(0, x)$. The activation function being known. The backpropagation algorithm is applied to train the weighting coefficients of the neuron.

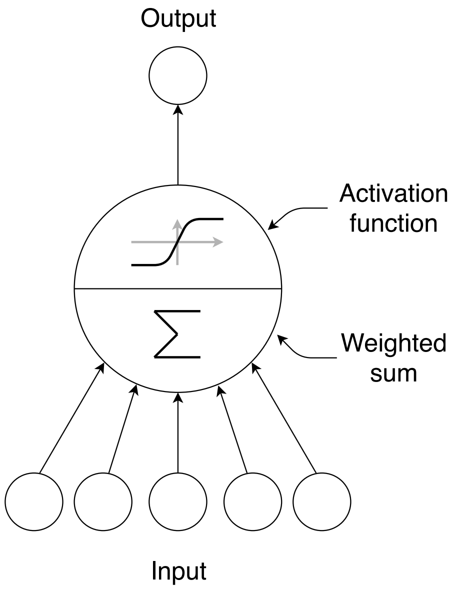


Figure 3.1: A single neuron. Each input x_i is weighted by a parameter w_i before being summed and put in an activation function. The output from this funtion corresponds to the output of the neuron.

$$sigmoid(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)} \tag{3.1}$$

3.2.3 The backpropagation method

The backpropagation method consists in the the propagation of the gradient of the error between the ouput of the MLP and what is expected from the output to the input of the system, hence the term backpropagation. The main equation of the gradient descent is the one presented in Equation 3.2, with w being the parameter to optimize, and E the error function depending on w. The minus symbol represents the idea that, when using this method, one tries to achieve the lowest point of the error function, as graphically represented on Figure 3.2.

$$w_{new} = w_{old} - \varepsilon \times \frac{\partial E}{\partial w} \tag{3.2}$$

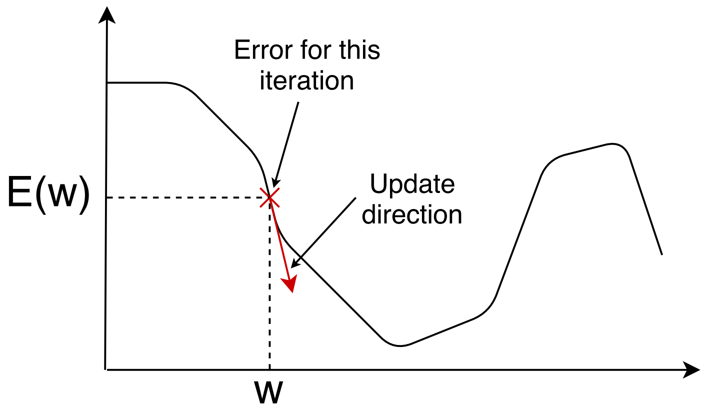


Figure 3.2: One step of gradient descent. The red cross is the current point, while the red arrow represents the direction in which the parameter has to be updated to lower the error.

The main difficulty of the update of the parameter using Equation 3.2 is to compute the value of the partial derivative of the error function E. This is achieved using the partial derivative composition property considering that the E function can be written as follows:

$$E\left(x_{target}, x_{output}, W\right) = l\left(x_{target}, f\left(\sum_{i=1}^{I} w_i x_i\right)\right) \tag{3.3}$$

With l a loss function such as the l_2-norm, f the activation function of the neuron, x_i the i-th value of the input (with a total of I input) and w_i the corresponding weight of the neuron. For clarity of the equation, the following notation will be used:

$$a_i = \sum_{i=1}^{I} w_i x_i \tag{3.4}$$

This allows to express the partial derivative of E in the following way:

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial f\left(a_i\right)} \frac{\partial f(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_i} = \frac{\partial E}{\partial f\left(a_i\right)} \frac{\partial f(a_i)}{\partial a_i} x_i \tag{3.5}$$

Equation 3.5 being generic, it can be used with any combination of loss and activation functions. The same composition rule is also used when dealing with several layers of neurons, but in this case the partial derivate of a_i by w_i has to be composed again in order to "reach" the parameter w_i in the following layer.

Knowing this method, the learning of a MLP is performed by iteratively applying Equation 3.2 to all the parameters of the network until the norm of the gradient is small enough to be considered negligeable. In the following Section are presented the two kinds of Neural Networks which are trained by this method and which are used in the CRS.

3.2.4 MLP and Autoencoder

The term MLP designates the supervised Neural Network algorithm which makes possible to learn a regression between a given input and output. This definition implicitly implies that the input and the output have to be different. A special kind of Neural Networks, presented in [28] and developped in [66], uses the input of the system as its output. They are called Autoencoders, because the intermediate layers of the networks, and more specifically their activations, can be used as codes to represent the input individuals. They are basically used to get a new representation of input data. They can also be used as a compression method if the encoding layer length is set to be smaller than the number of features describing the original data [28]. Formally, an Autoencoder is trained by minimizing a loss function, in our case, the Mean Square Error (MSE). With our notations, the MSE for a view V_i would be defined as follows:

$$\frac{1}{|V_i|} \sum_{x \in V_i} (x - \hat{x})^2 \tag{3.6}$$

With $\hat{x}$ being the output of the Autoencoder used in the i-th view for the input vector x and $|V_i|$ being the number of elements of V_i.

A graphical representation of both the MLP and the Autoencoder are presented on Figure 3.3, and their uses in the Collaborative Reconstruction system are detailed in Section 3.3.

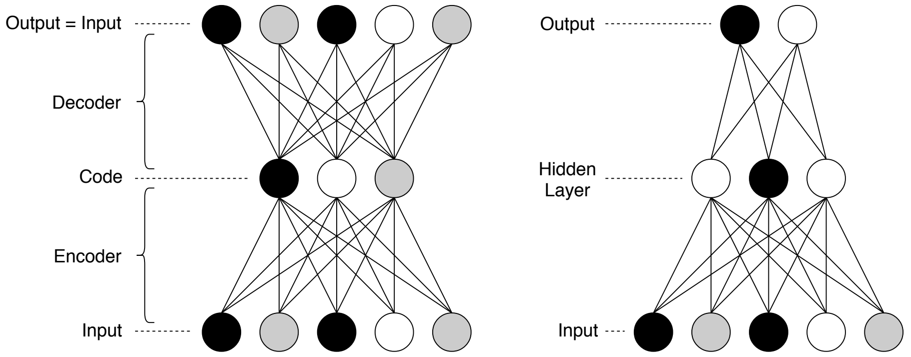


Figure 3.3: An Autoencoder (left) and a Multi-Layer Perceptron (right). The intensity of the grey in each neuron symbolizes its activation.

3.3 Cooperative Reconstruction System

In this section, we describe the architecture of our proposed Cooperative Reconstruction System. A representation of this system can be found on Figure 3.4. Our system is based on several modules: first, to solve the problem of security-friendly information transfer, the system uses a set of N Autoencoders [28] –with N being the number of views–, to locally encode data to make them impossible to read from outside of their views.

Then, when an individual is missing in a view, each external view sends its locally encoded version of the individual to the incomplete view, resulting in the transfer of $N-1$ encoded vectors. Then for each external view, a first approximation of the local values of the individual is inferred using a Neural Network (one per external view), in this case a Multi-Layer Perceptron. The role of this Neural Network is to make the link between the values of the external codes, and the features of the local view. After this step, the local views has access to $N-1$ versions of the missing individuals.

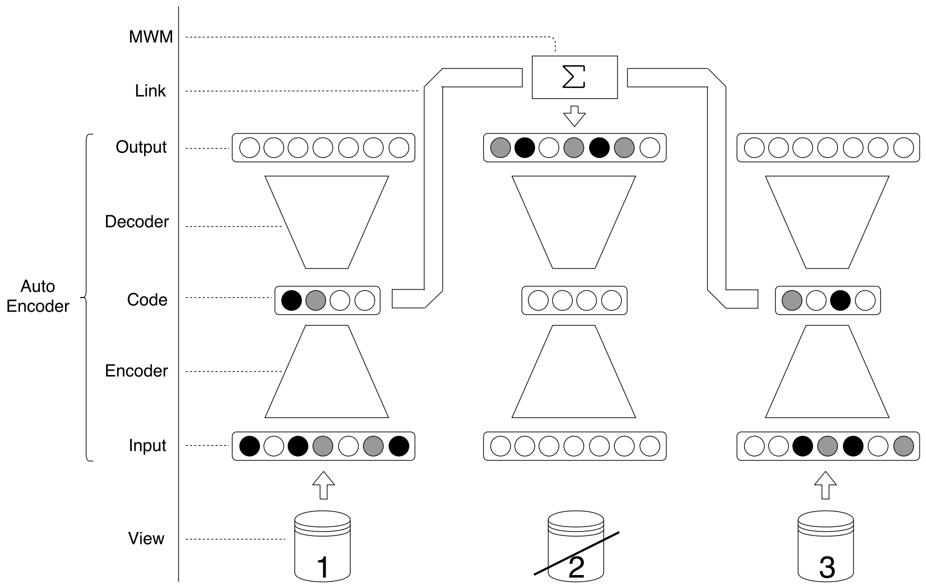


Figure 3.4: Cooperative Reconstruction System. In this example, Views 1 and 3 are sending their coded version of the individual to View 2.

The combination of the inferred individuals can then be used to reconstruct an accurate representation of the missing individual. However, since disagreement may occur between the different sources of information, the inferred data from each view need to be weighted to ensure an optimal reconstruction. This is solved using a weighting method we introduce in Section 3.3.5 and called the Masked Weighting Method. The basic idea of this method is to learn a set of $N-1$ scalar vectors, called masks, to weight each approximation generated locally (cf. Fig. 3.6). These masks can be trained using either Gradient Descent or using an iterative update rule. The description of both methods can be found in Section 3.3.5.

The global system is designed to be modular: when a new view is available, the system just has to learn its auto-encoder and the neural networks responsible for the links between this new view and the existing ones. However, due to the nature of the weighting methods between the views, all masks have to be learned again. This modularity is important because of the usually long learning time of a Deep Neural Network: learning the masks again does not take long, while having to re-train all neural networks would take a lot of time. It is therefore a huge gain of time that the already trained auto-encoders and links can be kept when a new view is added. This point has to be considered together with the fact that for a system made of N views, approximately N^2 networks have to be trained.

Our system has been tested on two points: how good are the individuals reconstructed compared to their original versions, and what are the classification scores of these latter compared to the original ones. Thus, we tested both its efficiency at reconstruction and whether or not reconstructed data could be used for further Machine Learning.

3.3.1 Notations

Formally, V_i and V_j are the datasets of the i-th and j-th views respectively. We note $V_{i|j}$ the subset of V_i (in the feature space of V_i) which individuals are also present in V_j. The size of this set is important because it will define the quantity of information available to train the inter-views Links (cf Section 3.3.3)

3.3.2 Autoencoders

We have selected Autoencoders to transfer information from a view to another because they offer the advantages of encoding data as scalar values, which can be used as input for further analysis, and they make it difficult to retrieve the original data without their decoding part, thus limiting possibilities of security breach. Moreover, the Autoencoders used in each view do not need to have the same architecture nor code lengths. This flexibility allows each view to use the best autoencoding architecture to describe their data.

When all the Autoencoders are trained, each view j is able to encode the subset $V_{j|i}$ of its dataset V_j, before sending the result to every other view i it has to collaborate with.

3.3.3 Links

A Link is a Neural Network in charge of infering the values of missing individuals based on the encoded data it received from an external view. In our case, a Link is more specifically a Multi Layer Perceptron: to reconstruct data in a local view i given information from view j, the Link will be trained using the version of $V_{j|i}$ encoded by the j-th Autoencoder as its input, and $V_{i|j}$ the original data as its output. The training process of a link is summed up on Fig. 3.5. We remind that $V_{i|j}$ and $V_{j|i}$ are the sets of shared individuals described in V_i and V_j feature spaces respectively, so they necessarily represent the exact same set of individuals.

It has to be noted that the receiving view j never tries to decode the encoded version of $V_{i|j}$, it only tries to infer the individuals features used in its local view.

This latter point is important because it is the one that ensure the security provided by the system.

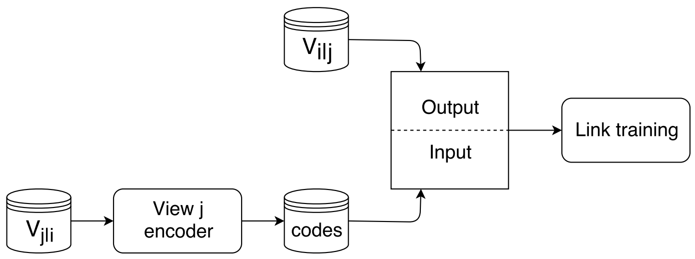


Figure 3.5: A Link training. The dataset $V_{j|i}$ is encoded, then sent to the local view. It it used as input for the link, while $V_{i|j}$ is used as output.

3.3.4 Missing Information

In some cases, it may happen that $V_{i|j} = \{\emptyset\}$, or is not big enough to learn the link between views i and j. The modularity of the method presented here implies that in this case, the information coming from the external view j is not taken into account, and the local view i will reconstruct its missing individuals based on the information from the other external views.

As this case does not change the global method, for the rest of the chapter we will only consider the case in which the individuals used in the training sets are present in all views. This simplification only aims at clarifying future algebra presented in Section 3.3.5. When all the Links have been trained, each view has access to (at most) $N-1$ Links allowing it to infer (at most) $N-1$ version of the missing individual values.

3.3.5 Masked Weighting Method

When a local view i has access to the $N-1$ infered versions of its missing data, it is necessary to find an efficient way to combine them to get the final version of the individual. We present a method based on a set of scalar vectors $W_i = \{w_{i|j},\, j \in [1..N] \setminus i\}$ such that $w_{i|j}$ is of same dimension as V_i. To get the final output $\widetilde{x_i}$ of the system, we use the following formula:

$$\widetilde{x_i} = \sum_{j \in [1..N] \setminus i} x_{i|j} \otimes w_{i|j} \tag{3.7}$$

where $\otimes$ is the pointwise vector product.

The coefficients can be learned using two methods: Gradient Descent on the reconstruction error, or iterative update using the zero of the derivate of this latter error. The analytical description and the characteristics of each method are described in the following section.

Gradient Descent:

When the reconstruction is done, it becomes possible to perform a Gradient Descent on the parameter contained in W_i. The error considered here is the

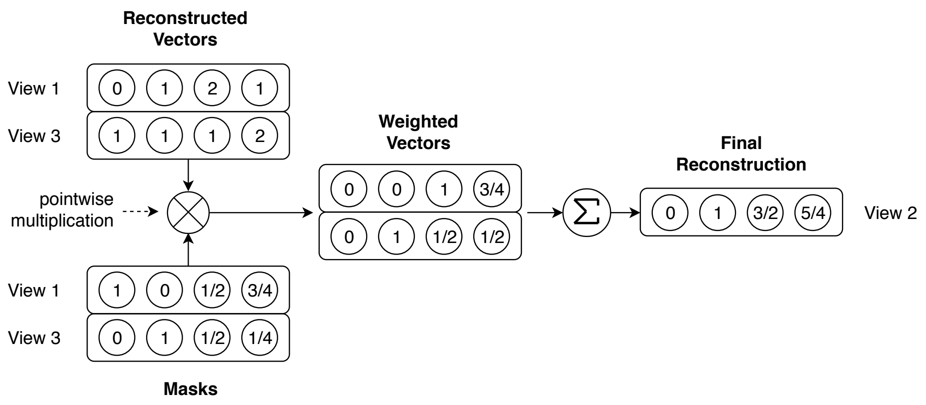


Figure 3.6: The Masked Weighting Method. View 2 has got the reconstructed individuals from Views 1 and 3, and it uses the masks previously trained to get the final weighted result.

MSE between target data and reconstructed ones. The computation of the error E_i for the view i can be written as follows:

$$
\begin{aligned}
E_i &= \frac{1}{|V_i|} \sum_{x_i \in V_i} ||x_i - \widetilde{x}_i||^2 \\
&= \frac{1}{|V_i|} \sum_{x_i \in V_i} \sum_{k=1}^{\dim(V_i)} (x_i^k - \widetilde{x}_i^k)^2 \\
&= \frac{1}{|V_i|} \sum_{x_i \in V_i} \sum_{k=1}^{\dim(V_i)} (x_i^k - \sum_{j \in [1..N] \setminus i} w_{i|j}^k x_{i|j}^k)^2
\end{aligned}
$$

where x_i^k is the k-th coordinate of the individual x_i. The differentiation of E w.r.t. the parameters $w_{i|j}^k$ of W_i can then be written:

$$
\begin{aligned}
\frac{\partial E}{\partial w_{i|j}^k} &= \frac{2}{|V_i|} \sum_{x_i \in V_i} x_{i|j}^k \, (\sum_{j \in [1..N] \setminus i} w_{i|j}^k x_{i|j}^k - x_i^k) \\
&= \frac{2}{|V_i|} \sum_{x_i \in V_i} x_{i|j}^k \, (\widetilde{x}_i^k - x_i^k) \qquad (3.8)
\end{aligned}
$$

This latter formula makes possible to update the weight $w_{i|j}^k$ using the usual gradient formula

$$
(w_{i|j}^k)^{new} = (w_{i|j}^k)^{old} - \epsilon \frac{\partial E}{\partial w_{i|j}^k} \qquad (3.9)
$$

where $\epsilon > 0$ is the parameter defining the learning rate of the process. This update process is performed on every weight until convergence. In practice, the learning is stopped when the norm of the update value defined in Eq. 3.8 goes under a threshold fixed by the user.

Iterative update

: It is also possible to update weights based on the minimum of E_i found using Eq.3.8.

$$\begin{aligned}
&\frac{\partial E_i}{\partial w^k_{i|j}} = 0 \\
\Rightarrow\quad & \frac{2}{|V_i|} \sum_{x_i \in V_i} x^k_{i|j} \left(\widetilde{x}^k_i - x^k_i \right) = 0 \\
\Rightarrow\quad & \sum_{x_i \in V_i} \Big((x^k_{i|j})^2 w^k_{i|j} + x^k_{i|j} \big(\sum_{j' \in [1..N] \backslash \{i,j\}} w^k_{i|j'} x^k_{i|j'} - x^k_i \big) \Big) = 0 \\
\Rightarrow\quad & w^k_{i|j} \sum_{x_i \in V_i} (x^k_{i|j})^2 = \sum_{x_i \in V_i} x^k_{i|j} \big(x^k_i - \sum_{j' \in [1..N] \backslash \{i,j\}} w^k_{i|j'} x^k_{i|j'} \big) \\
\Rightarrow\quad & w^k_{i|j} = \frac{\sum_{x_i \in V_i} x^k_{i|j} \big(x^k_i - \sum_{j' \in [1..N] \backslash \{i,j\}} w^k_{i|j'} x^k_{i|j'} \big)}{\sum_{x_i \in V_i} (x^k_{i|j})^2} \qquad (3.10)
\end{aligned}$$

Eq.3.10 shows that the update of $w^k_{i|j}$ requires the values of $\{w^k_{i|j'}, j' \in [1..N] \backslash \{i,j\}\}$. Thus it is possible to define an iterative update for which the values of $\{w^{k,t}_{i|j'}, j' \in [1..N] \backslash \{i,j\}\}$ at time t are used to obtained $w^{k,t+1}_{i|j}$ at time $t+1$. This problem being convex, the iterative process is performed until convergence of the weights.

This weighting method is used because it offers several advantages:

1. In the case where an external view is too noisy, or if the Link between this external view and the local one is not good enough to infer local individuals, the weighting coefficients for this view will converge to a value under $\frac{1}{N-1}$, lowering the impact of the bad reconstruction on the result. This latter value is the default value used to weights individuals, which corresponds to the mean of the incoming values to aggregate.

2. On the opposite, if a small subgroup of views is highly linked to the local one, this weighting method will favor these views to maximize the quality of the reconstructed individuals [61, 63].

3. Contrary to a weighted mean which would assign a single scalar to each view, this method allows to favor only a subpart of the reconstructed vector. Indeed, one can easily imagine that sometimes, the information contained in a view would only allow to recover part of the information contained in the local view, which entails a better reconstruction score on specific features of the local individual. Our weighting method makes possible to automatically identify these parts during parameters training.

When W_i has been trained for all the views, the system is ready to be used on missing data. An abstraction of the reconstruction process can be found on Figure 3.7, and a summary of the system architecture can be found on Figure 3.4. This latter can be used to see that there is no original data crossing the line between the local view and the external ones.

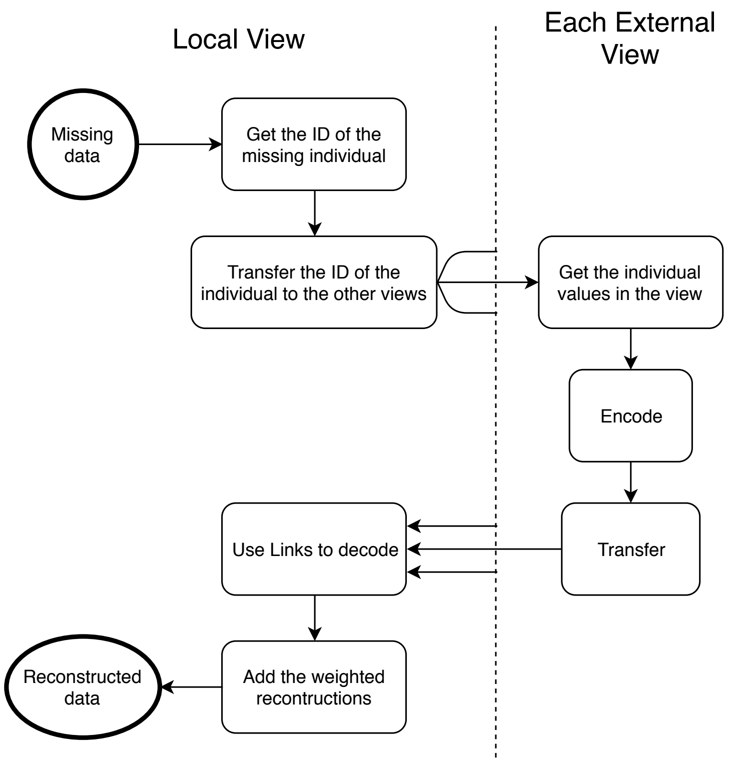

Local View
Each External View
Missing data
Get the ID of the missing individual
Transfer the ID of the individual to the other views
Get the individual values in the view
Encode
Transfer
Use Links to decode
Add the weighted recontructions
Reconstructed data

Figure 3.7: Reconstruction process going from the identification of a missing data to the getting of its reconstructed version.

3.4 Experimental setting

This Section presents the experiments that have been conducted to test our proposed method. First are presented in Section 3.4.1 the datasets which have been used during the experiments, then the global methodology used to analyze the system behavior is described in Section 3.4.2. The measures used to quantify the results are presented in Section 3.4.3, and finally numeric results are presented in Section 3.5.

3.4.1 Datasets

To get empirical results of the Collaborative Reconstruction System, we use it on three different datasets usable in a multi-view context, namely Wisconsin Diagnostic Breast Cancer (WDBC), Multi-Features Digital Dataset (MFDD), Madelon and Cube. While the description of the first three datasets can be found in Appendix .1, Cube is presented here because it is related to this chapter only.

1. *Wisconsin Diagnostic Breast Cancer (WDBC)*: This dataset has 569 instances with 32 variables (ID, diagnosis, 30 real-valued input variables). Each data observation is labeled as benign (357) or malignant (212). Variables are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present

in the image. Since each data contains the characteristics of 3 nuclei, we have 3 natural views here.

2. *Multi-Features Digital Dataset (MFDD)*: This dataset consists of features of handwritten numerals (from 0 to 9) extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. These digits are represented in terms of the following six feature sets, each set being here used as a view: 76 Fourier coefficients of the character shapes, 216 profile correlations, 64 Karhunen-Love coefficients, 240 pixel averages in 2×3 windows and 47 Zernike moments morphological features. Each set of coefficient stands for a view.

3. *Madelon*: This dataset is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the two classes (corresponding to the +-1 labels). Finally 480 features called 'probes' having no predictive power were added by the authors. The order of the features and patterns is random. This dataset is the most challenging among these used here. It is used to test the ability of our sytem to ignore noise (it should not reconstruct it) and to show that despite the large number of noisy features, we still have good classification results regardless of the poor reconstruction. Because no further information in available on this dataset, the views are randomly generated by picking a random set of 125 features for each.

4. *Cube*: In addition of the three previously described datasets, we have created a toy example which we mainly use to test the effectiveness of the Masked Weighting Method. This dataset, which we will refer to as the Cube dataset, is made of 1000 3-dimensional points divided in 4 classes of 250 members each. The points of each class are generated using a normal law with a standard deviation of 0.1 and centered either on the center of the feature space $(0, 0, 0)$, or at the extremity of one on the three unit vectors $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$. A graphical representation of the Cube dataset can be seen on Figure 3.8. The 3 views are obtained by projecting the whole dataset according to one of the three previous unit vectors. The point of this segmentation is explained in Section 3.5.

3.4.2 Methodology

In order to analyze each aspect of the system, several sets of experiments have been conducted. The first set consists in training a system with and without the Masked Weighting Method and to analyze the results in terms of reconstruction quality (Section 3.5.1). When it comes to combining the results from each external views, the system without our combination method simply uses a normalized equi-weighted sum of the reconstructed external vectors. This first experiment has been conducted in order to both test the viability of the method

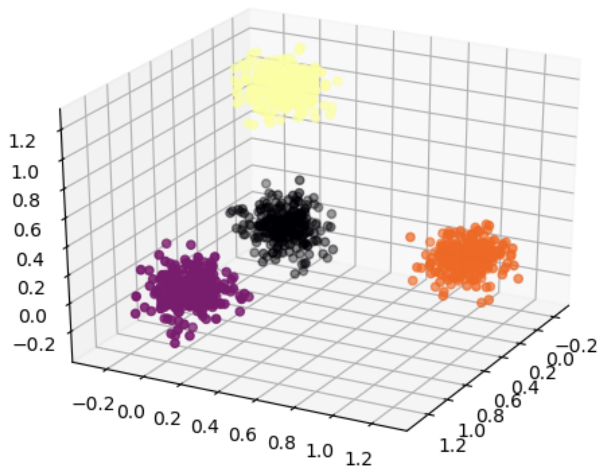


Figure 3.8: The Cube dataset

and determine the impact of our new combination method. An intermediary result is presented in Section 3.5.2: the reconstruction of images from the MFDD dataset is graphically presented. The second set of experiments consists in the analysis of the results obtained during the first set, but this time considering the impact of the Masked Weightinh Method on a classification task performed on the reconstructed data (Section 3.5.3). Finally, the third and last set of experiments consists in the analysis of the masks values for the toy example (Section 3.5.4). This is done to ensure the method is able to determine which reconstructor is better for which part of the reconstructed individuals.

For all sets of experiments, the global methodology remains the same: each view is split in a training set (90%) and a test set (10%), then all neural networks (Autoencoders and Links) are trained using the required training set. To test the system, the process described in Figure 3.7 has been conducted on the test dataset of each view, with the results being compared to the original data.

As there might be some variabilities in the results depending on the initialization of each neural network, the experiments have been conducted several times and the results have been averaged. Experiments on the WDBC dataset were repeated 50 times, while these performed on MFDD 20 times, these on Madelon 10 times and these on Cube 50 times. This difference is due to different dataset sizes, which increases the training time necessary for each neural network.

3.4.3 Measures

To determine the performance of our system, we used three measures. The first one is the Mean Squared Error (MSE) between the reconstructed vector and its target. Given two K-dimensional vectors x and y with respective coordinates sets $\{x_i\}_{i\in[1..K]}$ and $\{y_i\}_{i\in[1..K]}$, their MSE can be computed as follow:

$$MSE(x,y) = \frac{1}{K}\sum_{i=1}^{K}(x_i - y_i)^2 \tag{3.11}$$

The global error is then the average of the MSE of all the reconstructed vectors compared with their target values. The point of this measure is to get a global idea of the distance between the reconstructed vectors and the target ones.

The second error we use is the Mean Relative Difference (MRD) between the feature values of the reconstructed vector and these of the target vector. Given the same x and y than above, their MRD is computed as follow:

$$MRD(X,Y) = \frac{1}{K}\sum_{i=1}^{K}\left|\frac{x_i - y_i}{y_i}\right| \tag{3.12}$$

Here again, the global error is the average of the MRD of all the reconstructed vectors compared to their target values. This measure is used pairwise with the MSE in order to get more precise information about the difference between the reconstructed vector and the target one. Because of the security constraint and because of the difficulties the system may have to link the views, we do not expect these errors to be as good as these obtained by reconstruction and inference systems with less constraints such as standard Multi-Layer Peceptron [67].

To test the usability of the reconstructed vectors, they have been tested in a classification task: Random Forest classifiers were trained on the original data (one for each view), then we tested whether or not the data reconstructed using our proposed method were classified correctly by these trained Random Forest classifiers. The results were compared with performances on a test set with complete non reconstructed data.

The error considered here is the mean difference between the classification scores obtained in each view on their test datasets with the original data and the ones obtained with the reconstructed individuals. For the remainder of the thesis, we will name this error the Classification Difference. Contrary to the two previous ones, this error is not intended to determine the difference between a vector and its reconstruction, but rather to look at the impact of the reconstruction process on later data processing (such as a classification task). Even with mitigated reconstruction scores (MSE and MRD), a low Classification Difference would mean that the reconstructed individuals can be used in further applications. This score is presented along with the classification scores of each view. The Random Forest classifiers have been trained using the entropy cost function, with 50 estimators and with a max depth of 5.

Finally, to ensure the efficiency of the Masked Weighting Method when it comes to favor subparts of reconstructed vectors depending on the source external view, we simply have analyzed the vectors values of these masks for the Cube dataset. This dataset is particular because the projection performed to obtain a view entails the overlap of 2 clusters around the point $(0,0)$. Moreover, projecting according to a specific axe, which is equivalent to supress a column in the original 3-dimensional dataset, prevents the local view to have any information on this axe, while its pairs will need this information to reconstruct their local individuals. If the Masked Weighting Method works as intended, a huge difference between the values of the mask should be observed. This process is illustrated in Figure 3.9.

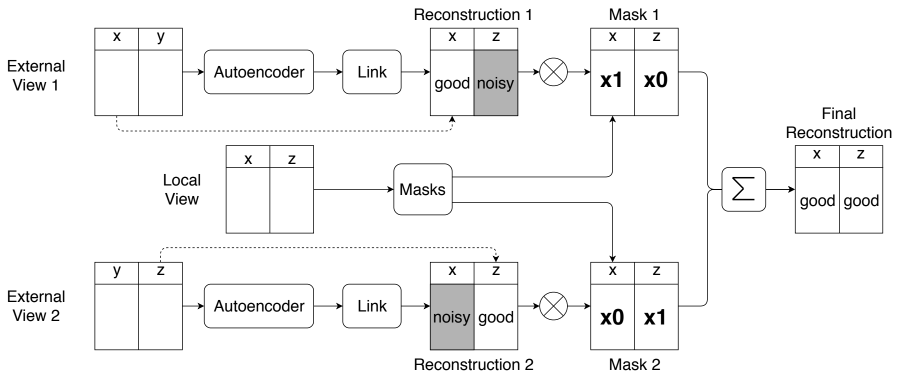


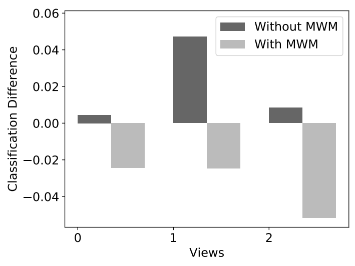
Figure 3.9: The combination of two partially good reconstructions into a good one. In this example, each view has enough information to reconstruct only one feature out of the two in the local view (doted lines). The Masked Weighted Method is designed to favor the best reconstructed part of each partial reconstruction, hence the $\times 0$ and $\times 1$ in the masks.

3.5 Results

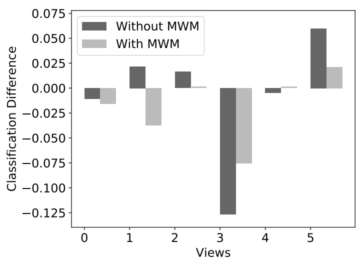
This Section is divided following the different kind of experiments that have been conducted. In Section 3.5.1 are presented the numeric results of the reconstruction process, then in Section 3.5.2 are presented some visual results presented the quality of the reconstructed individuals using the MFDD database. Section 3.5.3 presents the results obtained on the classification process performed on the reconstructed individuals, and finally Section 3.5.4 presents the analyzis conducted on the evolution of the masks coefficients depending on the information shared by views.

3.5.1 Basic reconstruction with and without the Masked Weighting Method

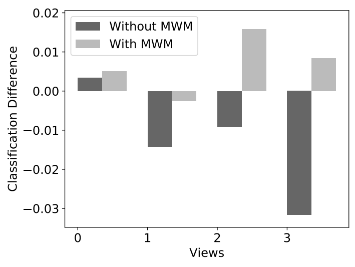
During this experiment, we were interested in the impact of our combination method on the results of the system. A summary of the results on WDBC, MFDD, Madelon and Cube can be found in Appendix 3.5.1 and 3.5.1.



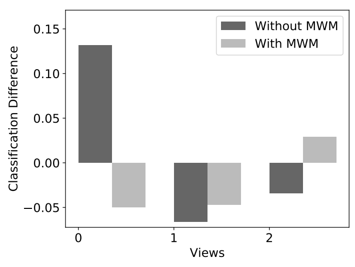
(a) WDBC



(b) MFDD



(c) Madelon



(d) Cube

Figure 3.11: Classification Scores and Difference for WDBC, MFDD and Madelon: original scores from the original data, classification score from the reconstructed data without and with the Masked Weighting Method and their difference.

For WDBC, MFDD and Cube, the Masked Weighting Method significantly reduces the MSE for almost every view (Figures 3.10a, 3.10c and 3.10g). This was expected because the use of this method implies the optimization of parameters w.r.t. this error. Moreover, the MRD is reduced for all the views in WDBC, MFDD and Cube (Figures 3.10b, 3.10d and 3.10h): the reconstructed individuals are closest to their original versions. The exceptionnal results obtained for the MSE on the Cube dataset (Figure 3.10g) can be explained by the fact that this dataset has been created as a perfect example for our weighting method. Further results can be found in Section 3.5.4.

Considering the reconstruction results on the Madelon dataset (Figure 3.10e and Figure 3.10f), the high MSE and MRD values were expected because of the numerous noisy features present in every view (480 out of 500): the Links could not reconstruct noise based on some more noise. The values around 1 for the MSE and MRD (Figure 3.10e and 3.10f) can be explained by the fact that during the training, the trained Links were only returning values around 10^{-2} (the noise could not be reconstructed as expected), while the scaled dataset mostly consists of values around 1. This case presents an extreme situation for which our system does not work as intended: the fact that it tries to reconstruct every feature of the local view implicitly implies that these features are not too noisy and can also be explained using the information available in the external views, which is not the case for the Madelon dataset.

3.5.2 Graphical Reconstruction of Handwritten Digits:

To better analyze the quality of the reconstructed individuals, we have used a specific view of the MFDD dataset, namely the one with the 240 pixel averages in 2×3 windows. The individuals of the tests datasets have been reconstructed and plotted. A sample of the individuals in the original dataset is presented on Figure 3.12. The contrast difference is explained by the normalization of the descriptive vectors before plotting.

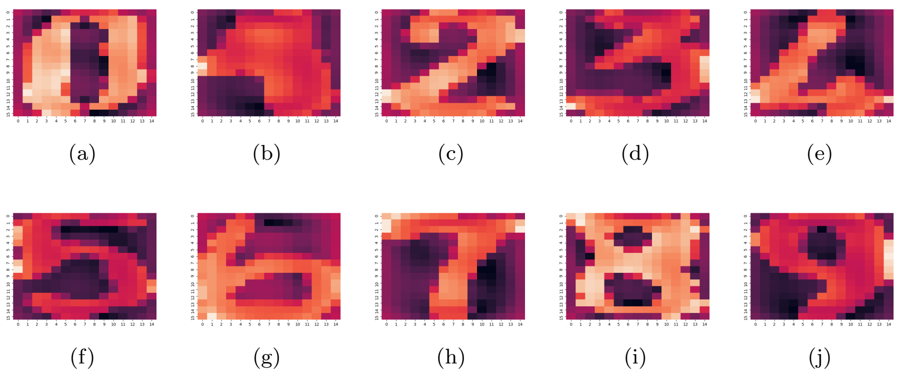


Figure 3.12: Sample of the original images available in the MFDD dataset

A sample of the individuals reconstructed is presented on Figure 3.13. While the MSE and the MRD are high for this view (Figure 3.10c and Figure 3.10d), it appears visually that the reconstructed individuals can be easily recognized.

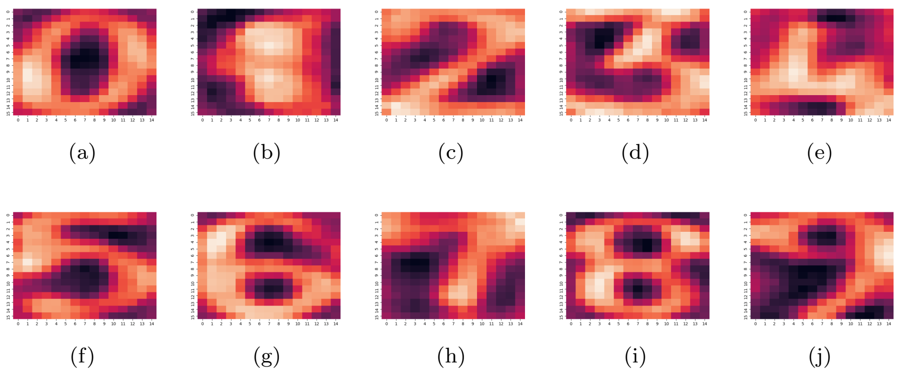


Figure 3.13: Sample of the reconstructed images available in the MFDD dataset. Some good examples.

However, while it is true for most of the reconstructed images, some examples do not work as well, as presented in Figure 3.14. Moreover, even if the numbers can be recognized, a blurring effect can be observed even on the best reconstructed examples. This puts forward the fact that the system approximation can be improved, because while it does not damages the recognition in the case of the MFDD dataset, we can imagine that it can damage it for some other use cases.

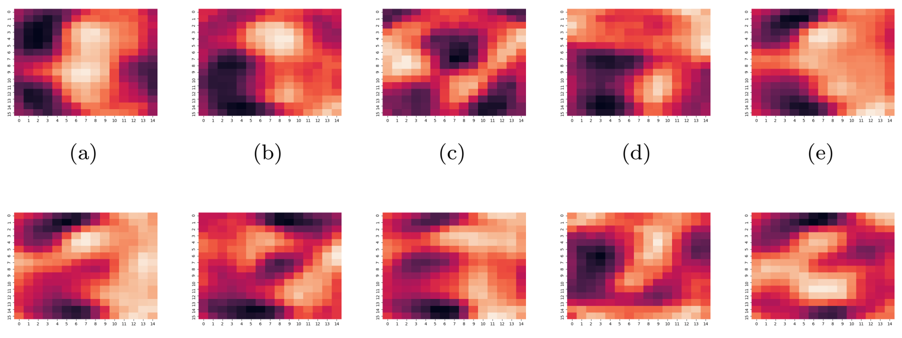


Figure 3.14: Sample of the reconstructed images available in the MFDD dataset. Some bad examples.

3.5.3 Impact on the Classification Score:

For the second experiment, when looking at Table 3.1 and Appendix 3.5.1, we notice that the Collaborative Reconstruction System gives classification scores comparable to these obtained on the original data: for WDBC, MFDD and Cube, the maximum absolute value of the Classification Difference is 7.5% (for the version using the Masked Weighting Method) when the mean original scores are respectively 90.9%, 88.4% and 75,35%. Secondly, our new combination method damages the Classification Score for WDBC and MFDD while improving it for Madelon and Cube. This point has to be considered conjointly with the fact that for almost every views, our combination method tends to lower the absolute Classification Difference of each dataset.

Table 3.1: Mean classification rate per database on the original data.

Dataset	WDBC	MFDD	Madelon	Cube
Mean rate	0.909	0.884	0.606	0.733

Even if we do not have clearly identified the source of this phenomenon, we suggest the following explanation: the quality of the output of a reconstruction system which does not use our combination method is highly dependent on the quality of the Links which make the inter-view reconstruction possible. Even if many tests have been performed for each database, the results depends on both manageable (hyperparameters of all the neural networks) and unmanageable (local minimum, initialization) points, both being very sensitive for the system training. That being said, it is very likely that the system results are very sensitive, which would explain the higher variability of the results obtained without the combination method compared to these obtained with it. This latter probably tends to mitigate the variability of the results because it depends far less on sensitive points: it only requires a learning step if the gradient descent method is used to update the weights and the initialization in the same every time.

3.5.4 Adaptation of the masks coefficients:

The point of this last set of experiments was to analyze the evolution of the masks coefficients to ensure that the method was able to determine which part of each reconstructed vector was the most useful to reconstruct the final individual. To make that possible, the Cube dataset has been generated as explained in Section 3.4.1, leading to the creation of 3 views each defined by 2 features. For each view, one of its feature is shared by one of the external views and the other feature is shared by the other external view. The point of this structure is to limit the mutual informations that two views can share. If the mutual information is limited to a specific set of features (the set being composed of only one feature in this example), the quality of the partial reconstructions should vary depending on the reconstructed feature, as presented in Figure 3.9.

In the Cube example, the information is either totally shared (same values if the feature is present in both views) or not at all (the feature not being present in the external view). Thus, we expect to obtain mask values around respectively 1 and 0. The results obtained empirically are described in Table 3.2. It clearly appears that the masks values adapt depending on the feature they are weighting: while these linked to the shared features are above 0.9, the ones linked to the other features never exceed 0.15. This validates the efficiency of the masks adaptation depending on the mutual information.

Table 3.2: Mean and standard deviation of the values of the masks coefficients depending on the feature they are weighting

	Mean	Standard deviation
Shared feature	0.920	0.026
Non shared feature	0.143	0.034

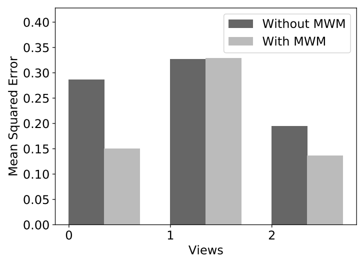
3.6 Conclusion & Perspectives

In this chapter, in a global context of multiplication of multi-view data, we have presented a new system called the Cooperative Reconstruction System. The purpose of this system is to reconstruct missing data using information contained in each view, without sharing the original data, thus avoiding security issues. To do this, the system relies on three modules: Autoencoders to cipher data under a scalar vector form, Multi-Layer Perceptrons -called Links- to decipher an external code in a local view, and the Masked Weighting Method, a new weighting method to combine all external reconstructions, thus obtaining the final reconstruction.

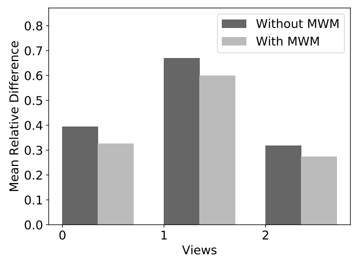
The Masked Weighting Method has three functions: combining external informations, reducing the influence of views with information which could hinder the reconstruction process, and reducing the impact of missing data during the system training process.

The efficiency of both our reconstruction system and our combination method has been tested on four different datasets: WDBC, MFDD, Madelon and Cube. To this end, two criterion have been considered: the adequation of the reconstructed individuals to their original versions considering using the Mean Squared Error and the Mean Relative Difference, and the impact of the use of reconstructed individuals instead of the original ones for classification purposes (tested with Random Forests). These experiments have demonstrated the main strengths and weaknesses of the system. Its main strengths are its ability to reconstruct an individual usable in a classification task without sharing data between views as well as its ability to weight views in such a way that it improves the final result compared to a standard meaning of the external reconstructions. On the opposite, its main weaknesses are its relatively weak reconstruction scores because of the training of the Links which may be difficult depending on the original datasets, and the number of hyperparameters to set, considering that a system composed of N views requires N^2 different neural networks to be trained.

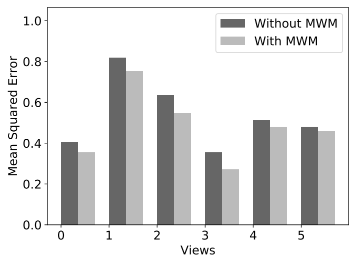
As future works, we plan on improving the reconstructions acquired from the external views through the modification of the inter-view Links. As well, because of the potentially high data dimensionality, the use of another error than the MSE should be considered. A feature selection process may be added to the system, thus limiting the impact of the noise features in the original dataset. Another possible future extension of this work would be to work on a lighter architecture that would scale better with large datasets.



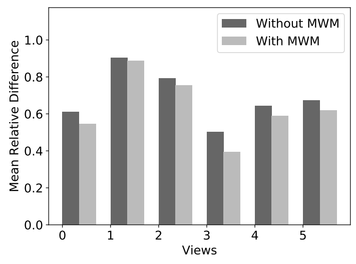
(a) WDBC



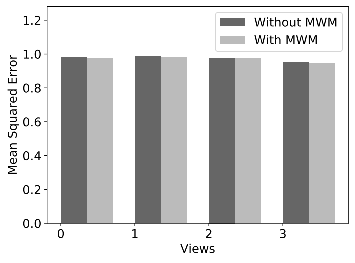
(b) WDBC



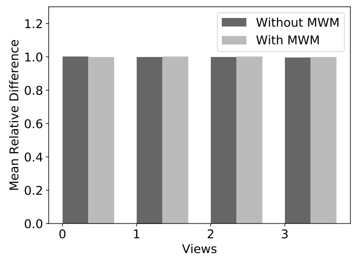
(c) MFDD



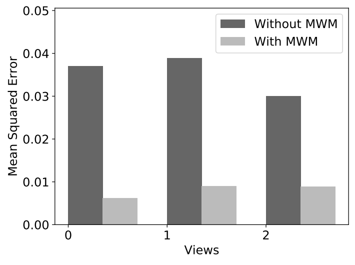
(d) MFDD



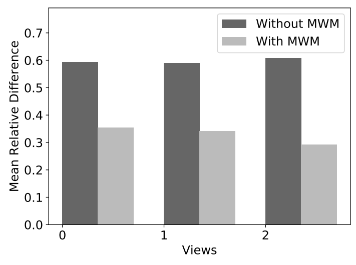
(e) Madelon



(f) Madelon



(g) Cube



(h) Cube

Figure 3.10: Mean Squared Error and Mean Relative Difference for all the datasets. A lower value corresponds to a better result.

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] Shai Ben-David, Ulrike von Luxburg, and Dávid Pál. A sober look at clustering stability. In Gábor Lugosi and HansUlrich Simon, editors, *Learning Theory*, volume 4005 of *Lecture Notes in Computer Science*, pages 5–19. Springer Berlin Heidelberg, 2006.

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

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

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

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

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

[8] Antoine Cornuejols, Cedric Wemmert, Pierre Gancarski, and Younes Bennani. Collaborative clustering: Why, when, what and how. *Information Fusion*, 39:81 – 95, 2018.

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

[10] Francisco de Carvalho, Filipe M. de Melo, and Yves Lechevallier. A multi-view relational fuzzy c-medoid vectors clustering algorithm. *Neurocomputing*, 163:115–123, 2015.

[11] Marcílio Carlos Pereira de Souto, Pablo A. Jaskowiak, and Ivan G. Costa. Impact of missing data imputation methods on gene expression clustering and classification. *BMC Bioinformatics*, 16:64:1–64:9, 2015.

[12] 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.

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

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

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

[16] Cynthia Dwork and Frank D McSherry. Differential data privacy, April 13 2010. US Patent 7,698,250.

[17] 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.

[18] 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.

[19] 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.

[20] Mohamad Ghassany, Nistor Grozavu, and Younès Bennani. Collaborative clustering using prototype-based techniques. *International Journal of Computational Intelligence and Applications*, 11(3), 2012.

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

[22] 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.

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

[24] 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.

[25] Nistor Grozavu, Guénaël Cabanes, and Younès Bennani. Diversity analysis in collaborative clustering. In *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pages 1754–1761, 2014.

[26] 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.

[27] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[28] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

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

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

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

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

[33] 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.

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

[35] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[37] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In Berkeley University of California Press, editor, *Proceedings of 2nd Berkeley Symposium*, pages 481–492, 1951.

[38] 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.

[39] 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.

[40] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

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

[42] 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.

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

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

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

[46] A. Puissant, A. Troya-Galvis, and J. Sublime. Vhr strasbourg data.

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

[48] 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.

[49] Parisa Rastin, Guénaël Cabanes, Nistor Grozavu, and Younès Bennani. Collaborative clustering: How to select the optimal collaborators? In *IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7-10, 2015*, pages 787–794. IEEE, 2015.

[50] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[51] 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.

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

[53] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

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

[55] 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.

[56] 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.

[57] 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.

[58] 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), 2016.

[59] Jérémie Sublime and Sylvain Lefebvre. Collaborative clustering through constrained networks using bandit optimization. In *2018 International Joint Conference on Neural Networks, IJCNN 2018*, 2018.

[60] Jérémie Sublime, Basarab Matei, and Pierre-Alexandre Murena. Analysis of the influence of diversity in collaborative and multi-view clustering. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, USA, May 14-19, 2017*, 2017.

[61] Jérémie Sublime, Basarab Matei, and Pierre-Alexandre Murena. Analysis of the influence of diversity in collaborative and multi-view clustering. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 4126–4133. IEEE, 2017.

[62] Jérémie Sublime, Denis Maurel, Nistor Grozavu, Basarab Matei, and Younes Bennani. Optimizing exchange confidence during collaborative clustering. In *The 2018 International Joint Conference on*. IEEE, 2018.

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

[64] 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.

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

[66] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[67] 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.

[68] Ulrike von Luxburg. Clustering stability: An overview. *Foundations and Trends in Machine Learning*, 2(3):235–274, March 2010.

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

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

[71] Paul Werbos. Beyond regression: new fools for prediction and analysis in the behavioral sciences. *PhD thesis, Harvard University*, 1974.

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

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

[74] 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.

Appendices

.1 Datasets

1. *Wisconsin Diagnostic Breast Cancer (WDBC)* — This dataset has 569 instances with 32 variables (ID, diagnosis, 30 real-valued input variables). Each data observation is labeled as benign (357) or malignant (212). Variables are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. Since each data contains the characteristics of 3 nuclei, we have 3 natural views here.

2. *Multi-Features Digital Dataset (MFDD)* — This dataset consists of features of handwritten numerals (from 0 to 9) extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. These digits are represented in terms of the following six feature sets, each set being here used as a view: 76 Fourier coefficients of the character shapes, 216 profile correlations, 64 Karhunen-Love coefficients, 240 pixel averages in 2×3 windows and 47 Zernike moments morphological features. Each set of coefficient stands for a view.

3. *Madelon* — This dataset is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the two classes (corresponding to the +-1 labels). Finally 480 features called 'probes' having no predictive power were added by the authors. The order of the features and patterns is random. This dataset is the most challenging among these used here. It is used to test the ability of the tested methods to tackle noise. Because no further information in available on this dataset, the views are randomly generated by picking a random set of 125 features for each.

4. *Isolet* — This data set was generated as follows: 150 subjects spoke the name of each letter of the alphabet twice. Hence, we have 52 training examples from each speaker. The speakers are grouped into sets of 30 speakers each. The data consists of 1559 instances and 617 variables. All variables are continuous, real-valued scaled variables.

5. *Spam Base* — The SpamBase data set is composed from 4601 observations described by 57 variables. Every variable describes an e-mail and its category: spam or not-spam. Most of the attributes indicate whether a particular word or character is frequently occurring in the e-mail. The run-length attributes (55-57) measure the length of sequences of consecutive capital letters. Views can be created using types of attributes.

6. *VHR Strasbourg* — This dataset [46] is based on a very high spatial resolution image (Pleiades) of the city of Strasbourg. The image was processed using the Multi-Resolution image segmentation (MRIS) algorithm implemented in the eCognition software (c) Definens (2014). A wide range of features available in eCognition are then computed for each segment, including spectral, textural and shape features that were exported in the CSV file. Views can be created according to the type of feature: spectral, texture, and shapes.