THÈSE DE DOCTORAT DE
l'UNIVERSITÉ SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Denis MAUREL

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ SORBONNE UNIVERSITÉ

Sujet de la thèse :

Contributions aux communications inter-vues pour l'apprentissage collaboratif

soutenue le 10 Décembre 2018

devant le jury composé de :

Mme. Raja Chiky	Directrice de thèse
M. Antoine Cornuéjols	Examinateur
Mme. Pascale Kuntz	Rapportrice
M. Sylvain Lefebvre	Encadrant
M. Christophe Marsala	Président du jury
M. Marcilio de Souto	Examinateur
M. Jérémie Sublime	Encadrant
Mme. Rosanna Verde	Rapportrice

DOCTORAL THESIS BY
l'UNIVERSITÉ SORBONNE UNIVERSITÉ

Speciality

Computer Science

École doctorale Informatique, Télécommunications et Électronique (Paris)

Presented by

Denis MAUREL

To obtain the grade of

Doctor of the UNIVERSITÉ SORBONNE UNIVERSITÉ

Thesis subject :

Contributions to inter-views communications applied to collaborative learning

defended on 10/12/2018

Jury :

Mme. Raja CHIKY	Directrice de thèse
M. Antoine CORNUÉJOLS	Examinateur
Mme. Pascale KUNTZ	Rapportrice
M. Sylvain LEFEBVRE	Encadrant
M. Christophe MARSALA	Président du jury
M. Marcilio DE SOUTO	Examinateur
M. Jérémie SUBLIME	Encadrant
Mme. Rosanna VERDE	Rapportrice

Contents

List of Figures

Abstract

This thesis presents several unsupervised algorithms used in a collaborative context. Its main axis is the improvement of communications between several distributed datasets and the developments of these communications, either in terms of quality or in terms of use case. This axis is then divided into two sub problems, each one defined by the goal of the algorithm.

The first one is Collaborative Clustering, which aims at refining local results through a consensus between several distributed datasets called views. Given a local view, this consensus is achieved through the prioritization of the information got from all the external views. The method presented in this thesis is generic and can be applied without consideration of the clustering algorithms used on each view.

The second use case is the reconstruction of missing data. Given several views, and considering a set of individuals described in more than one view, the goal of this use case is to transfer information from a view to another in order to infer an approximation of the missing description of an individual. While Collaborative Clustering has already been extensively studied in the literature, Collaborative Reconstruction is introduced in this thesis. This new use case is applied on several datasets containing either images or numerical vectors, the majority of which being commonly found in Collaborative Clustering literature. However, our method differs from what has previously been presented in the literature because the inter-views communication is performed both by a scalar based prioritization of the information and by a neural network based inference system. Moreover, while Collaborative Clustering ensures a minimum security by transfering informations different from the original data, our Collaborative Reconstruction method uses autoencoders, a specific kind of neural networks, to encode informations before sending them to ensure a minimum security regarding data transfer.

Chapter 1

Introduction

Contents

1.1 Thesis Scope

Each day, data grows in terms of quantity and complexity. This has the advantage of making possible a wide range of applications on different data of all natures, but it also has the disadvantage of making each application more specific, and their possible solutions more difficult to design. Moreover, because of this huge volumetry, one is likely to face a problem for which data are distributed among several datasets (also called views). It is in this context that Multi-View Learning has been created. This latter can be divided into several paradigms focused on the collaboration between different actors to fulfill different goals. Among them, two paradigms are specificaly focused on clustering: Collaborative and Cooperative Clustering.

Clustering is the process of organizing individuals in groups such as the similarity between the individuals of a same group is maximum while the similarity between individuals from two different groups is minimum. However, it is an ill-posed problem because there is no universal definition of what a similarity measure is and the determination of the right number of clusters might be difficult. Multi-view Clustering aims at mitigating the problem inherent to each local clustering by establishing inter-views communications.

Cooperative Clustering aims at finding a global consensus knowing groups identified locally obtained by each view, while Collaborative Clustering aims at updating local results to take into account locally groups that have been identified in the other views. As part of this thesis, a consensus is defined as the modification of each local solution to maximize the inter-views agreement while not hindering local results. Collaborative Clustering is sometimes considered as a specific phase of Cooperative Clustering, just before merging the results of each view to get the global consensus. These two paradigms being vast research fields in themselves, in this thesis we focus on horizontal Collaborative Clustering, the subfield considering several views describing the same set of individuals using different feature sets. In this context, the goal of Collaborative Clustering is to consider groups identified from different point of views (hence the designation of view) to update each local clustering. In that sense, one could consider Collaborative Clustering to look for a consensus. However, it is important to note that, while Cooperative Clustering aims at finding a unique global consensus, Collaborative Clustering rather aims at improving each local clustering to ensure that local results take into consideration the information extracted by all the other clusterings.

Considering the fundamental idea of Collaborative Clustering, namely the collaboration of several independent views to get local results, we have explored the application of inter-views communications to Collaborative Reconstruction. Having several datasets describing the same set of individuals but using different sets of features might sometimes implies that some individuals may be missing in some datasets while being present in some other. It might be because data have been gathered using various methods during different sessions maybe separated in time, or because individuals did not provide all the required data during a survey for example.

In this context, the information regarding a specific individual and spread among several external views could be used to infer a local approximation of this individual. This thesis present a system able to perform such inferences. An interest in this use case is to consider inter-views communications from an other point of view than the clustering one. To do so, the initial hypothesis of the shared set of individuals is used to infer links between the representations of an individual in two different feature spaces. In other words, we want to make a system learn the correspondences between two different features spaces using the individuals that these features spaces have in common.

To sum up, the improvement of inter-views communications in a collaborative context is developped either based on existing methods in the case of Collaborative Clustering, or on the definition of a new paradigm in the case of Collaborative Reconstruction.

1.2 Thesis Overview

This thesis is structured into four chapters (Introduction, Conclusion and Appendices excluded) and is organized as follows:

Chapter 2, State of the Art: This chapter introduces a state of the art divided in four main parts. First, a definition of clustering and its current research challenges are provided. Then the most commonly found methods are presented, to finally introduce Collaborative Clustering as topic in itself. The aim of this chapter is to bring the reader from a high level definition of what the clustering is, to a more specific collaborative context.

Chapter 3, Incremental Self Organizing Maps based Collaborative Clustering: This chapter present our early work on multi-view communications. The objective was to define a method train Self Organizing Maps in both a collaborative and incremental context. It is related to the problem of communications in a collaborative context because it focuses on how these communications could be performed all along the training process, and not just during a fixed segment in time. Thus the chapter first introduces the problem of online training in a collaborative context. Based on the limitations identified, we present an online version of Self Organizing Maps. This later method is then adapted to be used in a Collaborative Context. Experiments are presented to determine the efficiency of our new Collaborative Clustering method. A discussion regarding the limitations of this method as well as a conclusion ends the chapter.

Chapter 4, Optimized weights for the horizontal collaborative SOM algorithm: Based on the challenges and on the state of the art presented previously, this chapter presents a new weighting method used to define the relative confidence a view has in the information coming from all its external peers. We first define our problem as an optimization problem, leading to the creation of a constrained system of equation defining the values of the best collaboration coefficients. This system is then solved mathematically using the Karush-Kuhn-Tucker method, followed by an interpretation of the results. The method is finally numerically against existing approaches in the literature.

Chapter 5, Collaborative Reconstruction System: This chapter presents our method to achieve Collaborative Reconstruction. To the best of our knowledge, there is no existing work on Collaborative Reconstruction in the literature, thus we first give some definitions. Then, we introduce the reader to the main components used by the system, a quick overview of Neural Networks is then given with a focus on two of the many kinds of existing Neural Networks: Multi-Layer Perceptrons (MLP) and Autoencoders. After this introduction, a global definition of the architecture of what we call the Collaborative Reconstruction System is given. The main components are the Autoencoders that encode the information, the MLP which transfer the information from a view to an other, and the Masked Weighting Method which combines all the external informations. This later component being a contribution in itself, its definition and its training are detailed using a mathematical analysis of the learning process using either a Gradient Descent or an iterative process. Because of the complexity of the system, several sets of experiments have been conducted to test both the efficiency of the global system and the impact of this method on the results of the system. A graphical representation of what can be achieved in terms of reconstruction is presented using a dataset of handwritten digits. Finally, a discussion of the advantages and on the limits of the system is presented at the end of the chapter.

1.3 Main Contributions

International Journal

- (currently submited) Denis Maurel, Sylvain Lefebvre and Jérémie Sublime, *Deep Cooperative Reconstruction with Security Constraints*, Knowledge And Information System (KAIS), 2019.

International Conferences

- Denis Maurel, Jérémie Sublime and Sylvain Lefebvre, *Incremental Self-Organizing Maps for Collaborative Clustering*, International Conference on Neural Information Processing, 2017.
- Jérémie Sublime, Denis Maurel, Nistor Grozavu, Basarab Matei and Younès Bennani, *Optimizing exchange confidence during collaborative clustering*, International Joint Conference on Neural Networks (IJCNN), 2018.

National Conference

- Denis Maurel, Jérémie Sublime and Sylvain Lefebvre, *Cartes Auto-Organisatrices Incrémentales appliquées au Clustering Collaboratif*, Conference internationale sur l'extraction et la gestion des connaissances, 2017.

Chapter 2

State of the Art

This section first gives the definition of clustering along with its use cases and its main difficulties. Then several commonly found 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.

2.1 Definition of clustering

Every day, exabytes of data of various format are generated: 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 applications such as market study, advertisement targeting, product improvement and recommendation.

In [33], 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 labels (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 [81] and K-means [42] 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.

2.2 Results analysis

Clustering in itself is a very broad field, and it might be difficult to analyze the results of a training process depending on its context. Criterions usually found to analyze a clustering are detailed in this section: the similarity measures along with how they are related to data typology, then quality measures.

2.2.1 Similarity measures

The second problem related to clustering lies in the similarity measures used for each algorithm. Clustering relies on the idea that similar individuals must belong to the same cluster while dissimilar ones should not. Therefore, one of clustering challenges is to define the criterion that allows 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:

$$d(x, y) = \sum_{k=1}^{D} (x_k - y_k)^2 \tag{2.1}$$

Where D is the dimension of x and y, and 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 has its limitation when the number of dimensions increases. The choice of the inter-individuals distance must consider points such as the space dimensionality, the type of features used in the dataset, and even the goal of the clustering. A guide on this problem can be found in [17].

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 criterion 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 meth-

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

ods may or may not adapt to some topologies. For example, methods such as K-means based on $l2$-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 in this thesis, it will implicitly entail that a distance between individuals has been defined. The definition of this distance can be viewed as an additional parameter for every method using it.

2.2.2 Quality measures

Another problem appears when one has applied the selected clustering method on its dataset: how can one measures the quality of a clustering? With a 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 deduce update rules as in [79]. 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 created clusters 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 [84] and Locally Linear Embedding [60] 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, like in [3].

However, even if it is hard to give an intuitive measure of the quality of a clustering, two criteria are intuitively 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 found are the Dunn index [18], the Davies-Bouldin index [11], the Silhouette index [59] and the Wemmert-Gancarski Index [82]. Some measures even allow to compare two clusterings, namely the Rand index [54] and its Adjusted version [31]. 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.

There also exists two quality measures used especially for prototype based methods such as SOM and K-Means, namely the purity index and the Quantization Error. These measures take advantage on the defined prototypes to define measures based on these prototypes.

Each quality measure is detailed in the following paragraphs.

Dunn index

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

$$DU = \frac{\min\limits_{i \neq j} D(c_i, c_j)}{\max\limits_{i \in [1..C]} \Delta_i} \tag{2.2}$$

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 2.1: 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:

$$\mu_i = \frac{1}{|c_i|} \sum_{x \in c_i} x \tag{2.3}$$

Davies-Bouldin index

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

$$DB = \frac{1}{|C|} \sum_{i=1}^{|C|} \max_{j \neq i} \frac{\Delta_i + \Delta_j}{D(c_i, c_j)} \tag{2.4}$$

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 found version of the Davies Bouldin index in the one based on the third line of Table 2.1.

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 individual x, or for a cluster c_i, or for the whole clustering. For an individual x, a_x is defined as the mean distance between x and the elements membership 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:

$$SI(x) = \frac{b_x - a_x}{\max(a_x, b_x)} \tag{2.5}$$

From which it follows that

$$SI(x) = \begin{cases} \frac{b_x}{a_x} - 1, & \text{if } a_x > b_x \\ 0, & \text{if } a_x = b_x \\ 1 - \frac{a_x}{b_x}, & \text{if } a_x < b_x \end{cases} \tag{2.6}$$

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:

$$SI(c_i) = \frac{1}{|c_i|} \sum_{x \in c_i} SI(x) \tag{2.7}$$

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

$$SI(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} SI(c_i) \tag{2.8}$$

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

$$WG(c_i) = \begin{cases} 0 \text{ if } \frac{1}{|c_i|} \sum_{x \in c_i} \frac{d(x,\mu_i)}{d(x,\mu_j)} > 1 \\ 1 - \frac{1}{|c_i|} \sum_{x \in c_i} \frac{d(x,\mu_i)}{d(x,\mu_j)} \text{ otherwise} \end{cases} \tag{2.9}$$

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:

$$WG(C) = \frac{\sum_{i=1}^{|C|} |c_i| WG(c_i)}{\sum_{i=1}^{|C|} |c_i|} \tag{2.10}$$

Rand index

To compare two clustering, the Rand index considers the peers 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 peers of elements of the dataset studied. The membership of a pair to a set follows Table 2.2.

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

Table 2.2: Sets of peers of elements used for the definition of the Rand index

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

$$RI(C_1, C_2) = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} = \frac{2(a+b)}{n(n-1)} \tag{2.11}$$

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:

$$ARI(C_1, C_2) = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{2.12}$$

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 criteria brings to light that there are several challenges to address 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.

Purity index

The purity of prototype based model is the average purity of its prototypes, with the purity of a prototype being the proportion of the most represented class among the individuals belonging to this prototype. It has to be noted that, unlike the previously presented index, the purity requires the original labels of the considered individuals to be computed.

It can therefore be defined using the following formula:

$$purity = \frac{1}{N_p}\sum_{k=1}^{N_p} \max_{i\in[1..C]} (P_k(i)) \tag{2.13}$$

With N_p the number of prototypes, C the number of classes and $P_k(i)$ the proportion of the class i among the individuals belonging to the prototype k.

Quantization Error

The Quantization Error (QE) is defined as the Mean Squared Error between the prototypes of the models and their respectives individuals. It can be defined using the following formula:

$$qe = \frac{1}{N}\sum_{i=1}^{N} \|x_i - w_{\chi(x_i)}\|^2 \tag{2.14}$$

With N the number of individuals, x_i the i-th individual, w_k the k-th prototype of the model, and $\chi(x_i)$ the function returning the index of the prototype to which x_i belongs.

2.3 Popular clustering methods

This section presents a non-exhaustive selection of clustering algorithms which are the most commonly found 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 found clustering algorithms. Some other categorization may be found in the literature, as the one presented in [33], in [85] and most recently in [21].

2.3.1 Hierarchical methods

This kind of algorithm is the first to appear in 1963 with the method presented in [81]. Its core idea is to build a hierarchy of clusters by joining existing clusters (agglomerative methods, each individual starts in its own cluster) or by separating them (divise methods, all the dataset belongs to the same initial cluster). The principle 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 applies the method until all points belong to the same cluster and graphically determining the best number of clusters, or applies the method until a specific criterion is reached (number of clusters or inter-cluster distance for example).

The first most commonly found hierarchical clustering method is the Agglomerative method presented in [81] which pairs clusters by considering a criterion to optimize. 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 [81]) 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 [46] 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 found Hierarchical method is the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) one [86]. 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 pass through the dataset to create the tree.

2.3.2 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 membership of an individual 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 memberships for which an individual belong to one and only one cluster, and these based on stochastic memberships for which memberships 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 [42]. 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 individual is associated with its closest prototype (in terms of a distance criterion). When all the dataset has been processed, the prototypes are updated as the means of their associated individuals: they become the “center”, also called “centroid”, 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 k-means algorithm is illustrated in Algorithm 2.

The initial method has allowed the emergence of many variations such as k-medians clustering [32], k-medoids clustering [34] depending on the prototype update rule. An extension to this algorithm has been proposed in [37] 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 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. The Manhattan distance between two neurons in a SOM is defined as the minimum number of hops separating the 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 method also makes possible to get a two or three dimensional representation

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
```

of a possibly high feature space. Indeed in a supervised case, by assigning to each node the class of the majority of its associated individuals, it it possible to graphically see the underlying proximity structure in the dataset.

A SOM is defined by a temperature function λ which itself defines a neighborhood function K. The temperature function is usually defined as follow:

$$\lambda(t) = \lambda_{\text{min}} \Big(\frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} \Big)^{\frac{1}{t}} \tag{2.15}$$

With λ_{min} and λ_{max} two parameters of the SOM algorithm defining the minimum and maximum temperature used during the training. One can notice that the function goes from λ_{max} for $t = 1$ to λ_{min} for $t \to \infty$. This function is then used to define the following neighborhood function:

$$K_{k,l} = \exp\Big(- \frac{d^2(k,l)}{\lambda(t)} \Big) \tag{2.16}$$

The idea behind this function is to determine the width of the neighborhood affected by the modification of a single neuron. The higher $\lambda(t)$, the hotter is said the map, and the wider is the neighborhood impacted by the modification of a neuron. However, this wider area comes with a limitation in the modification that can be performed on each neuron. This phenomenon is related to the exponential function used for K, if the map is hot, the map performs global modifications of its structure in order to adapt to the data typology. On the opposite, when the map is cold, the map only performs local modifications in order to adapt its neuron to its immediate neighborhood.

A description of the SOM algorithm is presented in Algorithm 3.

Algorithm 3: SOM algorithm

```
Initialization: Choose the dimensions $n \times m$ of the map
Randomly initialize the $n \times m$ neurons $w_i$
while a stopping criterion is not met do
    forall $x \in X$ do
        Assign $x$ to its closest neuron $w_i$
        $X_i \triangleq$ set of all $x$ assigned to $w_i$
    end
    Compute $\lambda$ using Equation 2.15
    forall $w_i$ do
        forall $w_j$ do
            Compute $K_{i,j}$ using Equation 2.16
            $w_j^{new} = w_j^{old} + \varepsilon K_{i,j} \sum_{x \in X_i}(x - w_i)$
        end
    end
end
```

Deterministic Vector Quantization ideas have been adapted in a stochastic context: now, an individual does not belong to a single cluster, but rather is defined by a set of K membership probabilities, with K the total number of clusters. The memberships are told to be “fuzzy”. A comparision can be made between the two learning steps used by K-means and the Expectation-Maximization (EM) algorithm defined in [14] 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. In this method, data have been generated by a set of Gaussian functions. The EM method here make it possible to learn the mean, the variance and, in the multidimensional case, the covariance matrix of the functions.

Considering fuzzy memberships, it is also possible to draw some links between deterministic clustering methods and their stochastic variants. Thus, Fuzzy C-Means [5] can be viewed as the stochastic variant of K-means for which the optimized criterion uses weighting coefficients equals to the fuzzy memberships defined by the method. Beside, the Generative Topographic Maps (GTM) [6] can be viewed as the SOM stochastic variant. However, while Fuzzy C-Means (FCM) only added the stochastic memberships 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 2.1). A detailed description of the FCM algorithm can be found in Algorithm 4

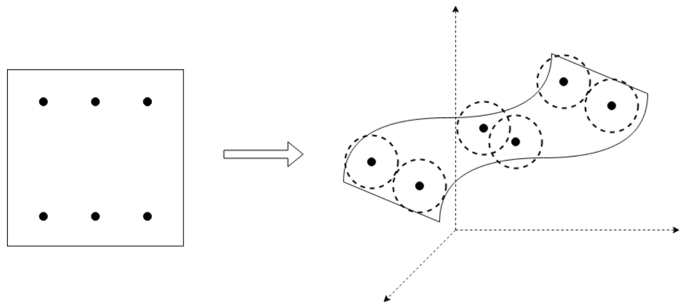


Figure 2.1: Generative Topographic Maps: projection of a map from the latent space to the feature space

These methods are further developed in Section 2.4 when Collaborative Clustering is presented.

2.3.3 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 individuals in the feature space to automatically determine the optimal number of clusters during the learning process (w.r.t. the optimized criterion). This kind of method can also be seen as an intuitive answer to the clustering problem given the definition of a cluster: these methods are based on the finding of dense zones in the middle of less dense zones.

The most commonly found density based method is called Density-based

Algorithm 4: 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;
```

spatial clustering of applications with noise (DBSCAN) [20]. Its core idea is to consider the density around each individual, namely their numbers of neighbors, 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 determines if an individual should be considered as a core sample (intuitively, a sample 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 individuals 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 clusters depending on *varepsilon* and *minPts*. A description of the DBSCAN algorithm can be found in Algorithm 5. The OPTICS algorithm [1] can be seen as an improvement of the DBSCAN algorithm. This algorithm is based on both the ordering of the individuals in such a way that spatially close points become neighbors in the ordering and on the storage in each point of a distance used to define same cluster memberships. These points are used to address one of DBSCAN's weaknesses: the problem of detecting meaningful clusters in data of varying density.

The second most commonly found density based algorithm is the Mean-shift method defined in [8]. The core idea of the method is to consider individuals as 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 individual will be updated as the mean of its neighbors weighted by their respective attractions (defined by K). The method is entirely defined by the kernel function and by its parameters.

2.3.4 Special case: Spectral clustering

In this subsection we present the method called Spectral clustering and defined in [47]. It is not presented in one of the previous sections 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.

2.4 Collaborative Clustering

2.4.1 Multi-view learning

Methods presented so far rely 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 5.1 for further explanation). It is also possible that data may come from different heterogeneous sources. Both cases make the training of a single algorithm on the whole dataset difficult. This acknowledgment had led to the definition of a new clustering context, called the Multi-View Learning. In this context, many methods have to be trained together before being combined in order to achieve the goal initially set.

As presented in [75], Multi-View Learning is a vast domain with many related subfields such as dimensionality reduction, semi-supervised, supervised learning and active learning. However, the following section is only

Algorithm 5: 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
```

focused on the application of Multi-View Learning in a clustering context, namely the collaborative clustering paradigm.

2.4.2 Cooperative versus Collaborative Learning

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

1. Improve a global clustering knowing the clusterings performed on each view.
2. Making the views collaborate to achieve a global inter-views consensus.

As defined in [9], the first subfield 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 [35, 16]. Several methods have been presented in the literature, the most known and used being Bayesian averaging [35], Bagging [7] and Boosting [22].

Knowing this, Collaborative Clustering differs in many points from its Cooperative counterpart, the most important one being its final goal: while Cooperative Clustering aims at finding 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 do 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 training was conducted independently with Cooperative Clustering, here each view has to communicate its intermediate results to its peers in order to improve each local result all along the training.

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

2.4.3 Horizontal and Vertical

As previously presented, the goal of Collaborative Clustering is to make several views collaborate using their local results in order to achieve a better consensus. This general idea has two different declinations, depending on the similarities that the datasets have to share. These paradigms have been defined in [52] and in [25].

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 2.2).

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, a problem that more and more applications are trying to deal with. The related works presented in the next Section are all related to horizontal clustering, unless the opposite is explicitly mentionned.

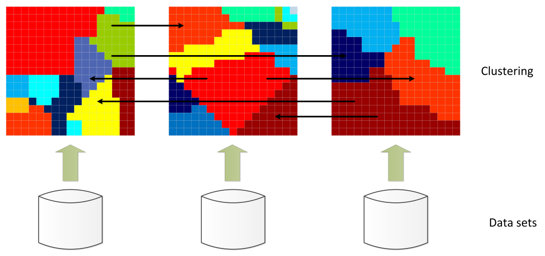


Figure 2.2: Horizontal Collaborative Clustering
/

2.4.4 Theory and Gradient Descent

Collaborative Clustering methods are based on two distinct phases introduced in [50]: 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 during which the views exchange the information they have acquired during the first phase in order to continue to learn from what their peers have found. These two steps are the first theoretical parts defining 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 consensus between the views. Formally, the criterion of the i-th view can be written as follow:

$$\begin{aligned} Q^i &= \alpha_i Q^i_{local}(V_i) + Q^i_{collab}(V_i, V_{j\neq i}) && (2.17) \\ &= \alpha_i Q^i_{local}(V_i) + \sum_{j\neq i} \beta^i_j C^i_j(V_i, V_j) && (2.18) \end{aligned}$$

With Q^i_{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. This latter term is made of the $N-1$ terms C^i_j corresponding to a dissimilarity measure between the local views and its peers. V_i stands for i-th view, and from now, we define N as the total number of views. The coefficients α and β are used to weight the relative importance of each term. Their definition is discussed later in this thesis.

A summary on collaborative methods is given in the next Section. When the cost function is defined, its differentiate is computed and then used to generate rules to update the parameters of the system using gradient descent which generic formula can be written:

$$w^{new} = w^{old} - \epsilon \frac{\partial E}{\partial w} \tag{2.19}$$

where $\epsilon > 0$ is the parameter defining the learning rate of the process, w the parameter to update, and E the criterion to optimize. This update process is performed on every parameter w until convergence. In practice, the learning is stopped when the norm of the update goes under a threshold fixed by the user.

2.4.5 Existing approaches

Fuzzy C-Means

Historically, the first version of Collaborative Clustering has been presented in [25] and in [51] and is based on Fuzzy C-Means [5]. During the local training phase, the method is using the following cost function:

$$\forall j \in [1..N], \quad Q^i_{local} = \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} \left(s^i_{n,k}\right)^2 |x_n - \mu^i_k|^2 \tag{2.20}$$

With $s^i_{n,k}$ being the probability the sample n belongs to the k-th cluster, also called responsibility, μ^i_k the center of the k-th cluster, and $|\cdot|$ being the distance between two individuals. After the local training, the collaborative phase is performed: each view exchanges its responsibility matrix $S = (s_{n,k})$ with its peers. During this phase, each view has to minimize the following function:

$$\forall j \in [1..N], \quad Q^i = Q^i_{local} + Q^i_{collab} \tag{2.21}$$

$$= Q^i_{local} + \sum_{j \neq i}^{N} \beta^i_j \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} \left(s^i_{n,k} - s^j_{n,k}\right)^2 |x_n - \mu^i_k|^2 \tag{2.22}$$

with β^i_j being a weight defining the importance that view i gives to the information coming from the external view j. The objective 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 consensus 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 consensus as possible between all the views.

Self Organizing Maps

A second version of collaborative clustering has been proposed using self organizing maps [37]. It is based on the same idea as previously defined

(FCM case), and has been presented in [25]. Both collaborative and local terms are defined using $K_{k,l}$:

$$Q^i_{local} = \alpha_i \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} K^i_{k,\chi(x_n)} \|x^i_n - w^k\|^2 \tag{2.23}$$

$$Q^i_{collab} = \sum_{j \neq i} \beta^j_i \sum_{n=1}^{|V_i|} \sum_{k=1}^{C} \left(K^i_{k,\chi(x_n)} - K^j_{k,\chi(x_n)}\right) \|x^i_n - w^k\|^2 = \sum_{j \neq i} \beta^j_i C^j_i \tag{2.24}$$

With $d(k,l)$ being the Manhattan 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 closest neuron of x, and w^k is the k-th neuron of the SOM.

The learning process is the same as the one of 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 2.23 and 2.24.

The collaborative clustering has also been applied using the Generative Topographic Mapping [6] presented is Section 2.3.2.

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 [23], [66], [67] and [68]. The learning algorithm relies on the same principle as the SOM-based one, but this time using the Expectation-Maximization algorithm [14]. During the Expectation phase each individual is assigned to a neuron. Then during the Maximization phase, the neurons are updated using a penalized likelihood as described in [24].

2.4.6 Genericity of Collaborative Clustering

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 prototypes 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 close 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 carefully, 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 [26] and in [55], 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 peers. An automatic update of the collaborative weights is presented in [25] and in [28], also based on the gradient descent. However, this method still presents a significant constraint ($\beta = \alpha^2$), which considerably simplifies 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 [74] having 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 [64].

2.5 Conclusion

Through this state of the art, several improvable points have been identified. These points establish the guiding lines of the work presented in this thesis.

First, one can notice the absence of incremental collaborative clustering methods among these previously presented. Now that Collaborative Clustering has a solid basis considering its definition and the methods it presents, it might be interesting to consider the case of ever coming information. In such a case, a continuous update of the clusters may avoid the previously obtained result to get obsolete.

A second point lies in the collaboration coefficients (α and β) used to weight the importance of the local and collaborative part in the collaborative clustering criterion. While some methods to automatically update these coefficients exist, they either rely on a simplification of the problem through the constraint $\beta = \alpha^2$ as seen in Section 2.4.6, or they imply the use of an additionnal parameter to fine tune the coefficient values.

Moreover, so far we have presented only clustering algorithms. While multi-view collaboration intuitively seem to be a generic concept, its application in the unsupervised Machine Learning literature seem to be mostly centered around clustering. It might be interesting to apply the fundamental idea of Collaborative Clustering, namely the collaboration of several views to improve local results, to a different application. Collaborative Clustering being sensible to missing data (if the views do not share the same set of individuals), missing data reconstruction could be a useful extension of the collaborative paradigm.

Each of the three points previously mentioned have been developed in this thesis. In the next chapter, a solution to the limitation regarding incremental learning is presented through the incremental adaptation of the Self-Organizing Maps as well as their adaptation to Collaborative Clustering.

Chapter 3

Incremental Self Organizing Maps based Collaborative Clustering

Contents

3.1 Introduction

Collaborative Clustering makes it possible for several independent views to collaborate. An incremental method for this problem, in which data are continuously added to each view through time, would make possible for a set of views to collaborate all along their existence, and not only at a fixed point in time. Indeed, so far, Collaborative Clustering methods perform their training on several *fixed* datasets. While the results obtained are usable as soon as they are obtained, they do not take into account the potential arrival of new data. In some cases, these arrivals may bring useful informations regarding the evolution of data distribution. For example, when looking at cyclical phenomena, if training data do not cover a whole cycle, the resulting clusters may not represent all the possible cases, leading to biased results. This latter point is one of the motivations that led us to study the specific case of online Collaborative Clustering. Moreover, to clarify a detail which differentiate online and incremental methods, unlike the former, incremental methods are allowed to store a batch of data when they arrive, making it possible to work with batches instead of just singletons.

The elaboration of such a method presents a challenge linked to the adaptation of the update rules of the method. As presented in 2.4, Collaborative Clustering relies the adaptation of the collaborative update rules depending on the chosen clustering method. This adaptation has now to take into account the incremental property of the method. In this chapter we try to address this challenge by focusing on the case of SOM.

The work presented in this chapter consists in an incremental SOM-based Collaborative Clustering method without topological modification of the SOM and which is robust to data distribution evolution. The key component of this approach lies in the modification of the temperature function of the SOM to make it time independent.

The rest of this chapter is organized as follows: our approach on incremental SOM and its application to Collaborative Clustering is presented in Section 3, followed by the experimental results presented in Section 3.5. Finally, a further analysis of our method is presented (easymotion-prefix) in Section 3.6.

3.2 Incremental Collaborative Clustering

Incremental SOM have already been studied in the litterature. However, all the presented solutions are based on topological modifications of the map [15, 49], and this kind of modifications are not compatible with the Collaborative Clustering update rules. Indeed, the Collaborative Clustering paradigm supposes that each dataset describes its data by the same number of prototypes to allow comparisons between several views and to keep topological mapping between each pair of views. In our case, the prototypes correspond to the neurons of the SOM, and as such without modification of the algorithm, the topology of each SOM has to be fixed during the initialization of the algorithm. Another problem encountered in general with incremental clustering is the possibility for the algorithms to answer changes in the data distribution through time. If the data distribution evolves, one has to be sure that the prototypes will follow the distribution of the most recent batches.

3.3 Incremental SOM

In our incremental version of SOM, we consider that the data are arriving all along the experiment. Therefore we assume that at each moment the model only knows the batch B of the N_{batch} last samples that have appeared during the learning. Our method here presents a variation of the original temperature function described in 2.4.5 which aims at avoiding the dependence between the temperature and the time. This is motivated by the incremental aspect of the subject, for which it is not possible to define a time limit $t_{\max}$ at which the algorithm will end. In order to make the SOM responsive to the arrival of new data, a new temperature function $\widetilde{\lambda}$ is defined by:

$$\widetilde{\lambda}(B, W) = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \|x_i - \chi(x_i)\|_2 \tag{3.1}$$

With $B \subset X$ the batch currently used and with $|B| = N_{batch}$. This $\widetilde{\lambda}$ function is then capped between $\lambda_{\min}$ and $\lambda_{\max}$ in order to avoid extreme modifications of the SOM. This definition of the temperature function allows the SOM to be more responsive to novelty encountered in the batch. If the elements of a batch are far from the current neurons, the whole map would need to be adjusted to the new distribution of the sample, and this case is empirically achieved for high values of $\widetilde{\lambda}$. On the opposite, if the samples are near the current centroids, the map only needs some adjustments in order to better match the sample distribution. This case is achieved for low values of $\widetilde{\lambda}$. To clarify notations, the neighboring function which is defined by $\widetilde{\lambda}$ will be named $\widetilde{K}$.

3.4 Adaptation to Collaborative Clustering

In this chapter, we only consider the case of horizontal Collaborative Clustering as defined in [23]. Thereafter, we consider datasets $\{X[i]|\ i \in 1..P\}$ containing the same set of objects described in different spaces, with P models (in our case SOM) being trained to represent each view separately. To clarify notation, $W^{m\in\{1..P\}}$ names the m-th model created using the m-th dataset. The point of Collaborative Clustering is to make those models collaborate in order to reveal common structures among them. The main hypothesis that is made here is if an observation from the i-th dataset belongs to the j-th neuron of the i-th model, then the same observation in the i'-th model will also belong to its j-th neuron or to its neighborhood. In other words, *equivalent neurons from different maps should capture the same observations* [23].

In order to adapt the original criterion to the incremental version of Collaborative Clustering, we use an approximation of this criterion using the kernel function $\widetilde{K}$ defined in Section 2.4.5 and where distances are summed over the current batch instead of over the whole dataset:

$$\widetilde{Q}^m(\chi, w) = \widetilde{Q}^m_{Local}(W) + \widetilde{Q}^m_{Collab}(W) \tag{3.2}$$

$$\widetilde{Q}^m_{Local}(W) = \alpha_m \sum_{i=1}^{N_{batch}} \sum_{j=1}^{|W|} \widetilde{K}^m_{j,\chi(x_i)} \|x^k_i - w^k_j\|^2 \tag{3.3}$$

$$\widetilde{Q}^m_{Collab} = \sum_{m'=1, m'\neq m}^{P} \beta^{m'}_m \sum_{i=1}^{N_{batch}} \sum_{j=1}^{|W|} \left(\widetilde{K}^m_{j,\chi(x_i)} - \widetilde{K}^{m'}_{j,\chi(x_i)}\right)^2 \|x^m_i - w^m_j\|^2 \tag{3.4}$$

Algorithm 6: Incremental horizontal Collaborative Clustering

```
Set the collaboration matrix $\{\alpha_{i,j}\}$
while Experiment going on do
    if new sample appears then
        Update batch as a FIFO stack of samples
        1. Local Step
        forall $m \in 1..P$ do
            Update prototypes of $W^m$ with one pass of incremental
             SOM
        end
        2. Collaboration Step
        for $m \in 1..P$ do
            for $w \in W^m$ do
                $w = w + \frac{\alpha_m \sum_{i=1}^{N_{batch}} \widetilde{K}^m_{j,\chi(x_i^m)} \Delta_i^m + \sum_{m'=1,m'\neq m}^{P} \sum_{i=1}^{N} \beta_m^{m'} L_{ij} \Delta_i^m}{\alpha_m \sum_{i=1}^{N_{batch}} \widetilde{K}^m_{j,\chi(x_i^m)} + \sum_{m'=1,m'\neq m}^{P} \sum_{i=1}^{N} \beta_m^{m'} L_{ij}}$
                with $L_{ij} = (\widetilde{K}^m_{j,\chi x_i} - \widetilde{K}^{m'}_{j,\chi x_i})^2$ and $\Delta_i^m = (x_i^m - w)$
            end
        end
    end
end
```

With α_m and $\beta_m^{m'}$ being defined as the collaboration coefficients. More precisely, α_m is defined as the relative weight of the local view in the training process compared to the relative weights of all the other views m' weighted by $\beta_m^{m'}$. In the next chapter, we get interested in the automatic definition of these weights to optimize the training process, but in this example they are fixed by the user at the beginning of the training process with the usual simplification $\beta = \alpha^2$ [23, 55].

A summary of the incremental horizontal Collaborative Clustering can be found in Alg. 6.

It is interesting to note that a lot of computation time may be avoided by performing the local step only during the first few steps of the algorithm. The first local steps help to improve the final quality of the clustering in terms of mean neurons to samples distance, whereas additional steps do not change much the final results.

3.5 Experimental Results

3.5.1 Datasets and quality measures

To evaluate the method presented in this chapter, we have tested them on four different datasets found on the UCI website [2]: Spam Base, Waveform, Wisconsin Diagnostic Breast Cancer (WDBC) and Isolet. Their respective descriptions can be found in Appendix B.

During our experiments, each of those datasets has been normalized and divided in 3 views each containing a third of the original variables. We suppose here that one has enough information on each variable to allow its normalization at the time it appears, for example by knowing its bounds. Each view will stand for an individual "site" which will collaborate with its peers.

The quality measures used during those experiments are the quantization error 2.2.2 and the purity index 2.4.5 commonly used to analyze SOM.

3.5.2 Experiments

In this section we present the results obtained on the four datasets. For these experiments, a 10×10 SOM has been used, with $\lambda_{\min} = 0.3$, $\lambda_{\max} = 3$, $\epsilon = 0.5$ (while it is usually time-dependent for classical SOM), $N_{batch} = 10$ and the local step of Collaborative Clustering has been performed for the first 10 batches. The models have been trained using only the 30 first batches in order to test the early convergence of the model and because it appears that the results do not change a lot on the long term. The methods have been coded in R v3.2.3.

The purities of the maps can be seen on Fig. 3.1a, Fig. 3.1b and Fig. 3.1c. For the sake of clarity, only the results for the Isolet dataset are presented here. It appears that Collaborative Clustering improves the purity of the maps even if it makes it less stable than the incremental SOM. The terms stable and unstable refers here to the standard deviation of the purities through time, which is higher in the case of Collaborative Clustering. This instability could be caused by the batch learning. The results of the learning phase depends on the incoming data: if one specific class is more represented than the others in a batch (which is more prone to happen if the batch is small), the neurons updates for this step will focus on this class specifically, and in the end it might hurt the next phases because of the bias acquired by the model for this specific class. An increase of the instability of the purities proportionally to the decrease of the batch size can be seen by comparing Fig. 3.1a, 3.1b and 3.1c to Fig. 3.1d, 3.1e and 3.1f, where we have set $N_{batch} =$

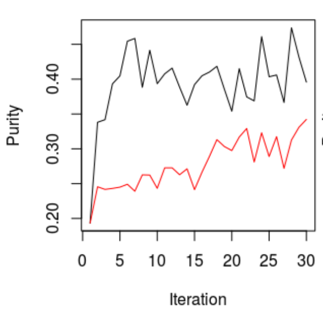
Table 3.1: Mean quantization errors on each database. Bold numbers are the lowest ones for each line.

	View	Incremental SOM	Incremental Collaborative Clustering
Spam Base	1	0.31	**0.26**
	2	**0.18**	0.19
	3	0.18	**0.16**
Waveform	1	**0.18**	0.23
	2	**0.17**	0.19
	3	**0.24**	0.30
WDBC	1	**0.19**	0.19
	2	**0.16**	0.19
	3	0.20	**0.16**
Isolet	1	2.15	**1.27**
	2	2.84	**1.38**
	3	2.85	**1.37**

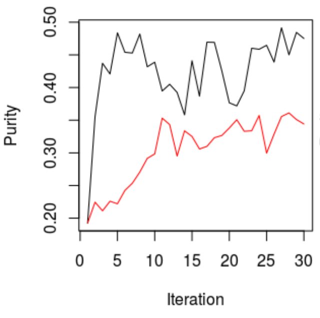
3. It is possible that the collaborative part of Eq. 3.2 makes the centroids move from the local solution minimizing Eq. 3.3: the collaborative SOM makes the centroid move toward a global solution rather than toward a local one. Concerning the quantization errors presented in Table 3.1, they all stay in an acceptable range considering that the data are scaled before the training. The case of the Isolet dataset is special because there are many more features than in the other datasets, and the database is sparser, point which may lower the performances of each local model depending on the distribution of features. Otherwise, it appears that the errors are always close to each other with a small advantage for the incremental SOM.

3.6 Conclusion

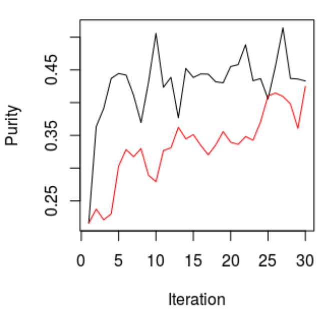
In this chapter, we have presented a method to perform incremental SOM without topological modifications of the map as well as the application of this method to adapt horizontal Collaborative Clustering according to the incremental constraint. Its application to vertical Collaborative Clustering is possible but has not been described in this thesis. With these methods, the



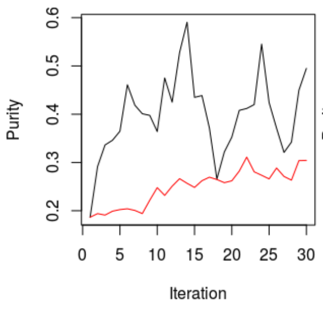
(a) $N_{batch} = 10$, View 1



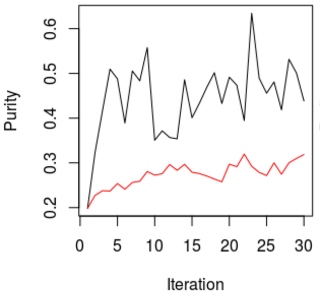
(b) $N_{batch} = 10$, View 2



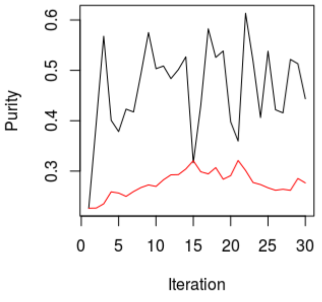
(c) $N_{batch} = 10$, View 3



(d) $N_{batch} = 3$, View 1



(e) $N_{batch} = 3$, View 2



(f) $N_{batch} = 3$, View 3

Figure 3.1: Evolution of the purities for the Isolet database with 2 different N_{batch}. The red lines represent the incremental SOM whereas the black lines represent the collaborative SOM. Each iteration corresponds to the arrival of a new sample

temperature function $\widetilde{\lambda}$, and so the neighboring function $\widetilde{K}$ of the generated maps are no longer time-dependent, and now only depend on incoming data. Knowing that, the map can be adapted to continuously incoming data. The presented methods have been tested on 4 different datasets, and the results show that our version of incremental SOM can be adapted to perform incremental Collaborative Clustering. The influence of the parameter N_{batch}, namely its impact on the stability of the learning, has also been investigated.

To pursue this work, we plan to consider methods which would make it possible to adapt topological modifications on SOM in the context of Collaborative Clustering. Furthermore, we plan to adapt what has been presented in this research work to the GTM, which are by nature similar to SOM.

The next chapter presents a contribution to multi-views communications based on the automatic training of the collaboration coefficients α and β, a limitation identified in 2.4.6. So far, they have been fixed by the user using the simplification $\alpha = \beta^2$, however, our solution makes it possible to get rid of this hypothesis by automaticaly update theses coefficients in order to find the best possible concensus, avoiding the handmade definition of these parameters.

Chapter 4

Optimized weights for the horizontal collaborative SOM algorithm

Contents

4.1 Introduction

In this chapter, we present an optimization method for the case of horizontal collaborative clustering between SOM algorithms. Our method aims at answering several questions regarding the tuning of the collaborative parameters between local and collaborative terms when using topological based collaborative clustering methods.

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 [40], 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 [28, 56, 27, 70]. 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 [70] 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 4.1 is devoted to the description of our proposed optimization method, as well as the interpretation of the proposed weights formulas, in Section 4.4, some numerical experiments are proposed and analyzed. Finally, Section 4.5 presents a conclusion on the work presented in this chapter.

4.2 Optimization problem

In this chapter we propose a different collaborative approach in which we modify the objective function (2.17) using a weighting strategy to reduce the risk of negative collaboration: we study how the optimization of the weights of the combination function in Equation (2.17) 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 values of the α by $\alpha_i = 1$. We believe that it makes a lot more sense than the proposed square root which is never justified in the related works [25, 28]. 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 β_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 (2.17) is to minimize the second term. Indeed, since the collaboration weights are only in the collaborative term and because the local likelihoods are fixed, we can ignore the local term in Equation (2.17).

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) [40] assuming that the weights β_i^j respect the following conditions:

$$\forall i \quad \prod_{j \neq i}^{N} \beta_i^j = 1$$

$$\forall (i,j) \quad \beta_i^j > 0$$

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 [12]. Furthermore, in [70] it has been demonstrated that the constraint $\sum_{j \neq i}^{N} \left(\beta_i^j\right)^p = 1$ 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.

We use the Karush-Kuhn-Tucker method to solve the created optimization problem. In this demonstration, we will only consider one local view

i because the weights β_i^j are computed and used locally. Given the C_i^j, we optimize the matrix $\beta = \left(\beta_i^j\right)_{N\times N}$ as shown in the system below:

$$\begin{cases} \beta^* = \operatorname{argmin}_\beta \sum_{j\neq i} \beta_i^j C_i^j, \\ \prod_{j\neq i}^N \beta_i^j = 1. \\ \forall j, \quad \beta_i^j > 0 \end{cases} \tag{4.1}$$

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

$$L(\beta,\nu,\lambda) = \sum_{j\neq i}^N (\beta_i^j C_i^j - \nu \ln \beta_i^j - \lambda_j \beta_i^j). \tag{4.2}$$

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

$$\forall j, j\neq i \begin{cases} (1) \quad \beta_i^j > 0, \\ (2) \quad \prod_{j\neq i}^N \beta_i^j = 1, \\ (3) \quad \lambda_j \geq 0, \\ (4) \quad \beta_i^j \lambda_j = 0, \\ (5) \quad C_i^j - \frac{\nu}{\beta_i^j} - \lambda_j = 0. \end{cases} \tag{4.3}$$

Let's begin by considering the case where $\lambda_j > 0$ in (4.3)-4. Then, we would have $\beta_i^j = 0$ this case is not possible due to (4.3)-1, therefore we will only consider the case $\beta_i^j \neq 0$ and $\lambda_j = 0$. Then, with (4.3)-5, we have:

$$\beta_i^j = \frac{\nu}{C_i^j} \geq 0. \tag{4.4}$$

From Equation (4.3)-2 and (4.4), we have:

$$\prod_{j\neq i}^N \beta_i^j = \prod_{j\neq i}^N \left(\frac{\nu}{C_i^j}\right) = \frac{\nu^{N-1}}{\prod_{j\neq i}^N \left(C_i^j\right)} = 1. \tag{4.5}$$

It follows that:

$$\nu^{N-1} = \prod_{j\neq i}^N C_i^j.$$

Then by re-injecting the expression of ν into Equation (4.4), we get for all $j \neq i$:

$$\beta_i^j = \frac{(\prod_{k \neq j} C_i^k)^{\frac{1}{N-1}}}{C_i^j} \tag{4.6}$$

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.

In other words: from Equation (4.6), 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.

Our modified version of the SOM algorithm for horizontal collaboration with the optimized weights is shown in Algorithm 7 below. The computational complexity for M data in N views is in $O(MN)$ since it uses N times the SOM algorithm which is in $O(M)$.

Algorithm 7: Topological horizontal collaboration Algorithm

```
**Initialization:** Initialize all the map prototypes $W$ randomly.
**Local step:** Initilization of the maps
**forall** *View i* **do**
    Minimize the objective function of the classical SOM
**end**
**Collaborative step:**
**forall** *View i* **do**
    For fixed $w$, compute: $\beta$ using Equation (4.6)
    Update the prototypes of all maps by: $w^* = \text{argmin}_w \mathcal{C}(w, \alpha, \beta)$
**end**
```

4.3 Interpretation

From Equation (4.6), we can infer the following property: For two SOM algorithms in the views i and j, when the pairwise collaborative term C_i^j is small (i.e. the maps are similar) comparatively with the other collaborative terms C_i^j, then the associated collaborative weight β_i^j is large compared with the other β_k^j. The interpretation for this is that any SOM algorithm 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 β_i^j is an inverse geometric mean based on the similarity between two maps. Furthermore, with Equation (4.6), we can further interpret, that algorithms with relatively weak collaboration links β_i^j –with maps very different from all others– whould give a stronger weight to their local term Q_{local}^i from Equation (2.17) and would not collaborate much with the other algorithms.

These properties are an improvement from an earlier result [70] 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 SOM to collaborate is to favor exchanges with other SOM that have similar topologies and are stable. These results are echoing recent works on clustering stability [4, 80] 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 [10].

4.4 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.

4.4.1 Datasets

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 B.

4.4.2 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 presented in the previous chapter in Equation 2.14. where N_{batch} is replaced by the dataset size. The values of the quantization error depend on the size of the dataset and the size of built maps. Strong differences 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 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 $L = \{l_1, l_2, \ldots, l_{|L|}\}$ and the prototypes set $C = \{c_1, c_2, \ldots, c_{|C|}\}$, the formula for the purity of a map is the following:

$$purity = \frac{1}{N} \sum_{k=1}^{|C|} c_k \times \frac{\max_{i=1}^{|L|} |c_{ik}|}{|c_k|} \tag{4.7}$$

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).

4.4.3 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. 2.17) with and without our proposed beta-optimization method.

For comparison purposes, both criteria have been normalized as follows:

$$\mathcal{C}(w,\beta) = L_j(w) + \sum_{j\neq i}\Big(\frac{\beta_i^j}{\sum_{j\neq i}\beta_i^j}\cdot C_i^j(w)\Big) \tag{4.8}$$

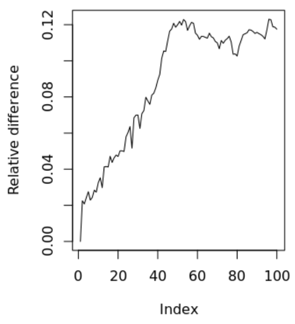
The point of this criterion is to make the β_i^j act as weighting coefficients summing to 1, allowing to compare both versions of CC with and without β optimization.

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 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 $\prod_{j\neq i}^{N}\beta_i^j = 1$ 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 β_{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 4.2. 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.

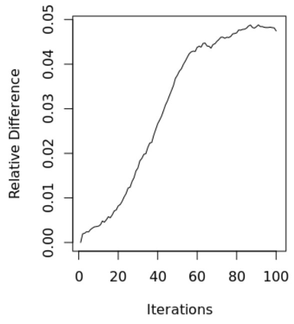
All the experiments were conducted with a 5×5 map. The choice of this size was made heuristically, based on the most appropriate number of neurons which optimize the quantization and topological errors during the local step [25].

4.4.4 Results

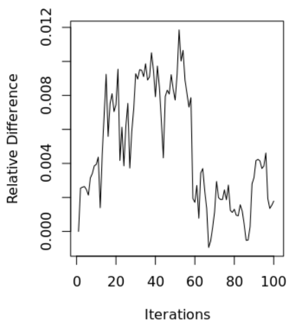
Relative Difference of the criterion: The first experiment is about the evolution of the modified criterion presented in Eq. 4.8. Figure 4.1 shows the relative difference (RD) of this criterion between the original version of



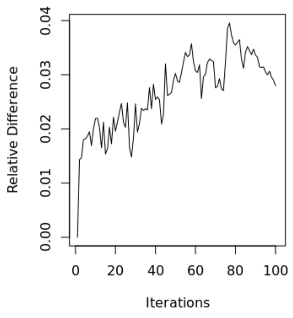
(a) WDBC



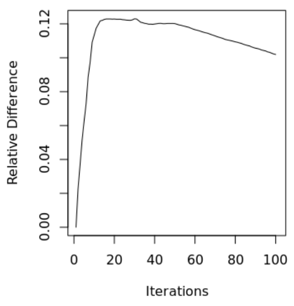
(b) Waveform



(c) Spambase



(d) Isolet



(e) VHR Strasbourg

Figure 4.1: Relative differences of the weighted criterion with and without β optimization all along the learning process

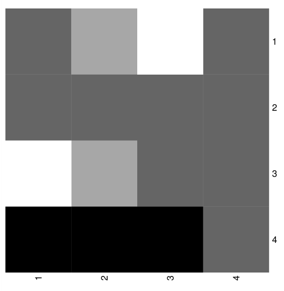
the collaborative SOM algorithm and our proposed version with the smart weights β. It appears that the relative difference between the criterion is always positive, meaning that the version with the β-weighting always improves the learning compared to the standard version. This can be understood knowing the interpretation in Sec. 4.3: 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% (Spambase) to 12% (WDBC and VHR). Moreover, the gap size does not evolve the same way for all data bases: for WDBC, Waveform, Isolet and VHR Strasbourg, the gap is growing during several steps, while for Spambase, the evolution seems to stop quickly. We think that this phenomenon is partly caused by the information contained in each view: in some cases, a β-weighting will be useful because it will favor a collaboration that will greatly improve the global results, while in some other cases the results will approximately be the same than with the standard method, and therefore the RD is caped more quickly.

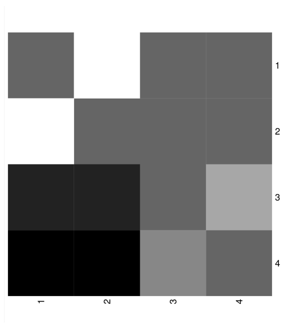
β analysis: Now considering β coefficients themselves, Figure 4.2 presents the different β values obtained at the end of the collaboration process for each dataset. Table 4.1 gives their corresponding minimum and maximum values. To recall, the value β_i^j can be read as "how much does view j exchanges 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 tend to collaborate, their other β are approximately identical. This property can be seen as the continuation of the interpretation given in Sec. 4.3: 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.

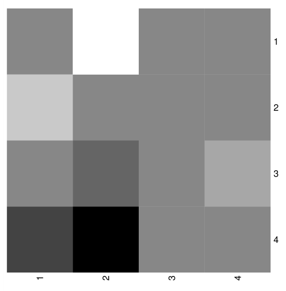
Purity and QE analysis: The last experiment consisted in the analysis of two criterion commonly found in the Kohonen's map literature, namely the purity and the quantization error. This analysis has been conducted in order to make sure that the collaborative phase and the β weighting did not damage the final result of the learning. Table 4.2 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 unsuper-



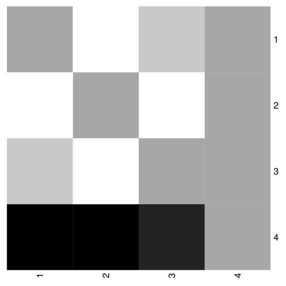
(a) WDBC



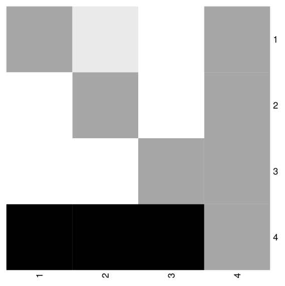
(b) Waveform



(c) Spambase



(d) Isolet



(e) VHR Strasbourg

Figure 4.2: 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 $\beta = 1$.

vised 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.

4.5 Conclusion

In this chapter, we have presented an optimization method for the case of horizontal collaborative clustering between SOM algorithms. Our 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

Table 4.1: 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

Table 4.2: Experimental results on different datasets

Dataset	Method	Mean Purity	qe
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 Strasbourg	standard	48.94	3.37
	β-weighting	48.93	3.38

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 a paper where an alternative bandit optimization scheme is proposed for a similar collaborative clustering problem [69].

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.

The method presented in this chapter can be considered as an improvement of the inter-views collaboration using a traditional scalar weighting method.

In the next chapter, we explore a collaboration method based on neural network and vectors rather than on scalar only. The complexity of this new method makes it possible to introduce a new use case to explore how an inter-view collaboration can be used to perform tasks different from clustering.

Chapter 5

Collaborative Reconstruction System

Contents

The work presented in this chapter has been submitted to the Knowledge and Information Systems journal.

As presented in Section 2.4, all the collaborative clustering methods presented in the litterature base their collaboration on scalar values which weight the importance that is given by a local view to the information provided by its peers. The intuition behind the work presented in this chapter is that a collaboration can be more complex than just the weighting of all the information coming from an external view. Instead, local information could be inferred based on an external information, and the combination of two views could be done on partial set of features from each view, which is not permited by the standard weighting method. These two ideas have been the starting point of what is presented in this chapter. In order to exploit these ideas, a new use case different from the clustering one has been created.

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 exact 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 able to recontruct an approximation of a missing individual in a multi-view context.

5.1 Context

5.1.1 Multi-view reconstruction

The 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 independent (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 mentioned above, 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 has already been studied using method such as Collaborative Filtering [38], the work presented here provides the extension of this problem to the multi-view context while also considering the problem of data security.

5.1.2 Data imputation and data reconstruction

Missing data is a problem which has already been extensively studied in the literature, and the field related to completing a partially sparse dataset is called data imputation. In this paradigm, several individuals are missing part of their features (not necessarily the same), and the presented methods aim at filling these missing features to be able to use the individuals a posteriori.

The main methods used to perform data imputation are the mean substitution (define the missing value of a feature as the mean of its values), the linear regression (with an optimized version presented in [57]), k Nearest Neighbors imputation [76], fuzzy k-Means imputation [41], Singular Values Decomposition imputation [76], bayesian Principal Components Analysis [48] and Multiple Imputation by Chained Equations [48]. Several surveys and comparisons can be found in the literature, as presented in [63], [61] and [77].

It has to be noted that the goal of data imputation is to mitigate the impact of missing data on the results obtained using the sparse database. This differs from what we try to achieve here in that we present a method to get the best reconstructed individuals possible, the reconstruction being an end in itself. Moreover, the imputation field does not consider the multi-view constraint: at every step of each method, the algorithm has access to the whole database to impute missing values.

5.1.3 Data security and data privacy

At this point, it is useful to note that the work presented in this chapter takes 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 algorithmics which defines a theoretical context to estimate the privacy level of a semi-randomized mechanism [19]. 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 [30] to respect the security issues, the combination of external information is performed using Multi-Layer Perceptrons [62] (called Links in this article) and a smart weighting method presented in this chapter and called Masked Weighting Method. This weighting method tackles two issues related to collaborative reconstruction: (1) combine the information from different views, (2) reduce the weight of views with information which could hinder the cooperative reconstruction process [72], and (3) reduce the impact of missing data during the unsupervised learning process [13].

5.2 Neural Networks

5.2.1 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 [58], the use of the back-propagation algorithm by Werbos in 1975 [83] and the presentation of the deep beliefs networks by Hinton in 2006 [29]. 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 [39] to perform image analysis and the Recurrent Neural Network [44] which are used to analyze temporal data. In this thesis, we are only interested in the Multi Layer Perceptron, a specific kind of supervised neural networks. The following sections briefly sum up the principal components of a Multi Layer Perceptron (MLP).

5.2.2 A neuron

A MLP is made of several layers of several neurons (see Figure 5.1 and 5.2), 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 set of parameters 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 5.1. There are many different activation functions which can be used, however the one which tends to be the most commonly found 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.

$$sigmoid(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)} \tag{5.1}$$

5.2.3 The backpropagation method

The backpropagation method consists in the propagation of the gradient of the error between the ouput of the MLP and the expected output to the input of the system, hence the term backpropagation. The main equation

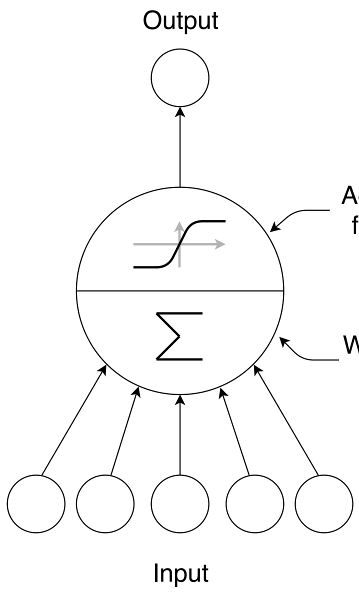


Figure 5.1: A single neuron. Each input x_i is weighted by a parameter w_i before being summed and put in an activation function. The output from this funtion corresponds to the output of the neuron.

of the gradient descent is the one presented in Equation 5.2, 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 5.3.

$$w_{new} = w_{old} - \varepsilon \times \frac{\partial E}{\partial w} \tag{5.2}$$

The main difficulty of the update of the parameter using Equation 5.2 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:

$$E\left(x_{target}, x_{output}, W\right) = l\left(x_{target}, f\left(\sum_{i=1}^{I} w_i x_i\right)\right) \tag{5.3}$$

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

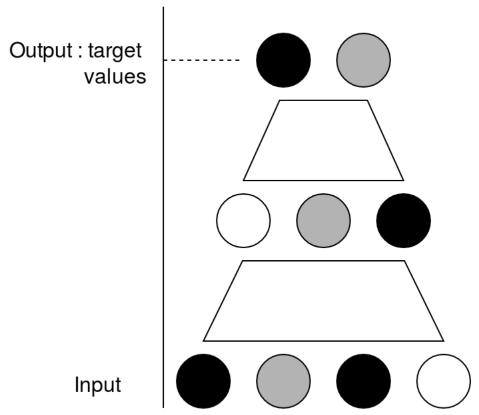


Figure 5.2: An example of Multi Layer Perceptron. In this case, the network has two hidden layers, the first one is made of 3 neurons, while the second one is made of 2 neurons. The smaller circles stand for the input (4 feature) and the output (2 features).

the corresponding weight of the neuron. For clarity of the equation, the following notation will be used:

$$a_i = \sum_{i=1}^{I} w_i x_i \tag{5.4}$$

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

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial f\left(a_i\right)} \frac{\partial f(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_i} = \frac{\partial E}{\partial f\left(a_i\right)} \frac{\partial f(a_i)}{\partial a_i} x_i \tag{5.5}$$

Equation 5.5 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 5.2 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.

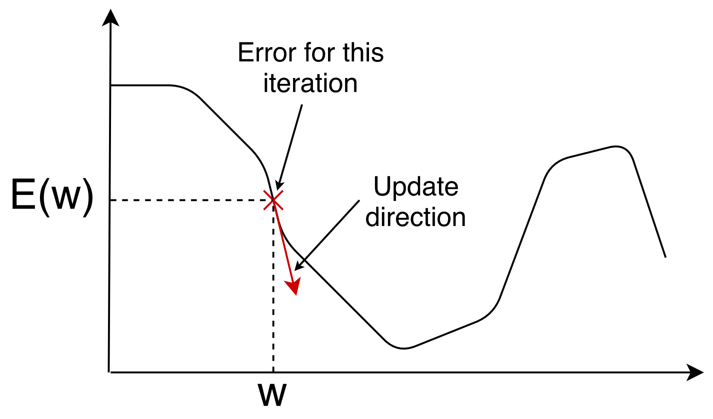


Figure 5.3: One step of gradient descent. The red cross is the current point, while the red arrow represents the direction in which the parameter has to be updated to lower the error.

5.2.4 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 implies that the input and the output have to be different. A special kind of Neural Networks, presented in [30] and developped in [78], 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 used for generating 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 [30]. 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 dataset V_i would be defined as follows:

$$\frac{1}{|V_i|}\sum_{x\in V_i}(x-\hat{x})^2 \tag{5.6}$$

With $\hat{x}$ being the output of the Autoencoder for the input vector x and $|V_i|$ being the number of elements of V_i. The MSE is simply the quantization error as presented in Equation 2.14, but using the target individual rather than the nearest prototype in a SOM.

A graphical representation of both the MLP and the Autoencoder are presented on Figure 5.4, and their uses in the Collaborative Reconstruction system are detailed in Section 5.3.

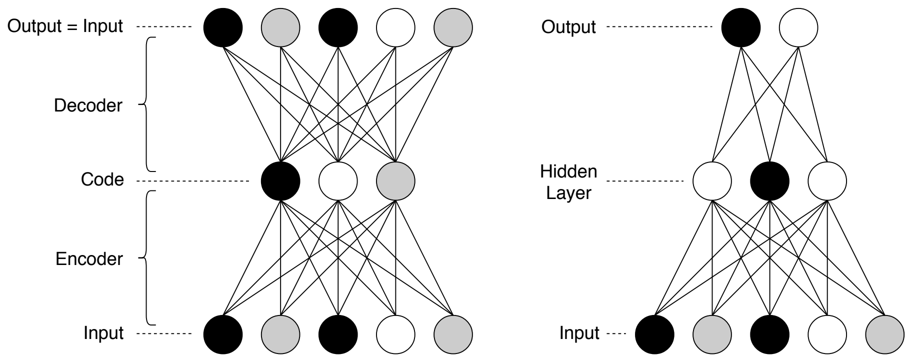


Figure 5.4: An Autoencoder (left) and a Multi-Layer Perceptron (right). The intensity of the grey in each neuron symbolizes its activation.

5.3 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 5.5. 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 [30] –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 view has access to $N-1$ versions of the missing individuals.

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. On Section 5.3.5 we introduce a weighting method called the Masked Weighting Method to tackle this issue. The basic idea of this method

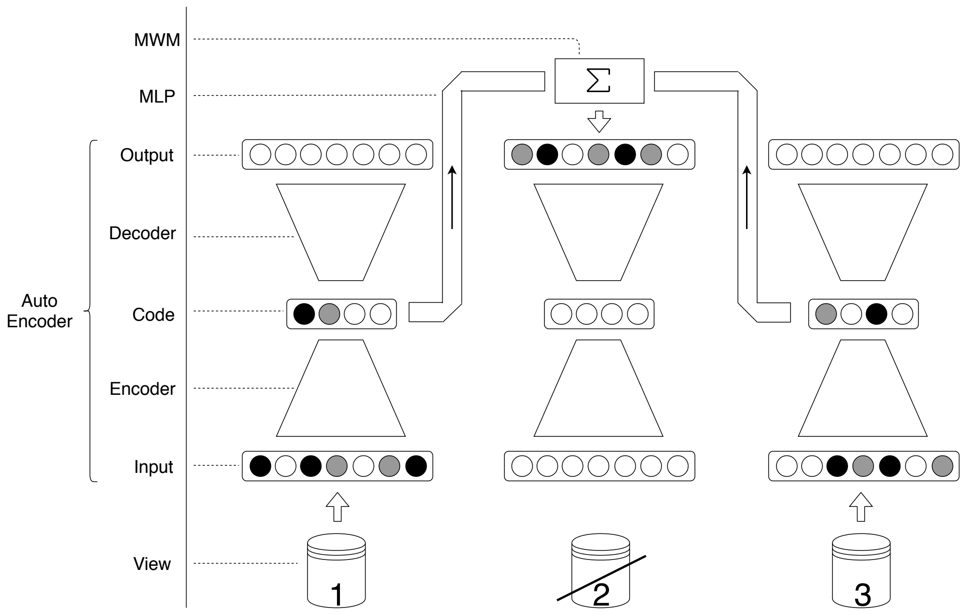


Figure 5.5: Cooperative Reconstruction System. In this example, Views 1 and 3 are sending their coded version of the individual to View 2.

is to learn a set of $N-1$ scalar vectors, called masks, to weight each approximation generated locally (cf. Fig. 5.7). These masks can be trained using either Gradient Descent or using an iterative update rule.

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.

5.3.1 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 5.3.3)

5.3.2 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.

5.3.3 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. 5.6. 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.

5.3.4 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.

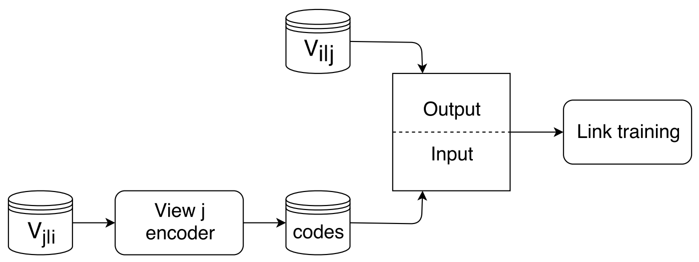


Figure 5.6: A Link training. The dataset $V_{j|i}$ is encoded, then sent to the local view. It it used as input for the link, while $V_{i|j}$ is used as output.

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 5.3.5. 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.

5.3.5 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:

$$\widetilde{x_i} = \sum_{j \in [1..N] \setminus i} x_{i|j} \otimes w_{i|j} \tag{5.7}$$

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

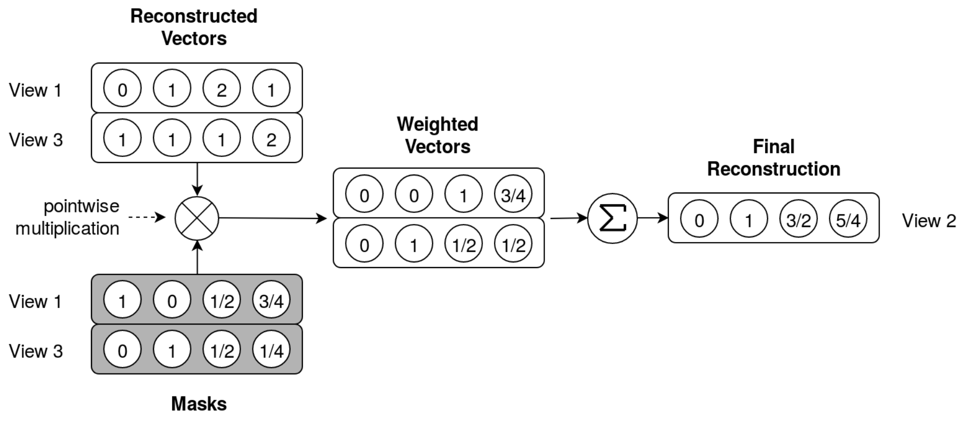


Figure 5.7: The Masked Weighting Method. View 2 has got the reconstructed individuals from Views 1 and 3, and it uses the masks previously trained to get the final weighted result.

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

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:

$$
\begin{aligned}
\frac{\partial E}{\partial w_{i|j}^k} &= \frac{2}{|V_i|} \sum_{x_i \in V_i} x_{i|j}^k \left(\sum_{j \in [1..N] \setminus i} w_{i|j}^k x_{i|j}^k - x_i^k \right) \\
&= \frac{2}{|V_i|} \sum_{x_i \in V_i} x_{i|j}^k \left(\widetilde{x}_i^k - x_i^k \right) \qquad (5.8)
\end{aligned}
$$

This latter formula makes possible to update the weight $w_{i|j}^k$ using the usual gradient formula as described in Section 2.4.4.

Iterative update :

It is also possible to update weights based on the minimum of E_i found using Eq.5.8.

$$\begin{aligned}
&\frac{\partial E_i}{\partial w^k_{i|j}} = 0 \\
\Rightarrow\ &\frac{2}{|V_i|} \sum_{x_i \in V_i} x^k_{i|j} \left(\widetilde{x}^k_i - x^k_i \right) = 0 \\
\Rightarrow\ &\sum_{x_i \in V_i} \Big(\big(x^k_{i|j}\big)^2 w^k_{i|j} + x^k_{i|j} \big(\sum_{j' \in [1..N] \backslash \{i,j\}} w^k_{i|j'} x^k_{i|j'} - x^k_i \big) \Big) = 0 \\
\Rightarrow\ &w^k_{i|j} \sum_{x_i \in V_i} \big(x^k_{i|j}\big)^2 = \sum_{x_i \in V_i} x^k_{i|j} \big(x^k_i - \sum_{j' \in [1..N] \backslash \{i,j\}} w^k_{i|j'} x^k_{i|j'} \big) \\
\Rightarrow\ &w^k_{i|j} = \frac{\sum_{x_i \in V_i} x^k_{i|j} \big(x^k_i - \sum_{j' \in [1..N] \backslash \{i,j\}} w^k_{i|j'} x^k_{i|j'} \big)}{\sum_{x_i \in V_i} \big(x^k_{i|j}\big)^2}
\end{aligned} \tag{5.9}$$

Eq.5.9 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 $\frac{1}{N-1}$ (default value when individuals are just averaged), lowering the impact of bad reconstruction on the result.

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 [71, 73].

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 5.8, and a summary of the system architecture can be found on Figure 5.5. These two diagrams can be used to see that there is no original data crossing the line between the local view and the external ones.

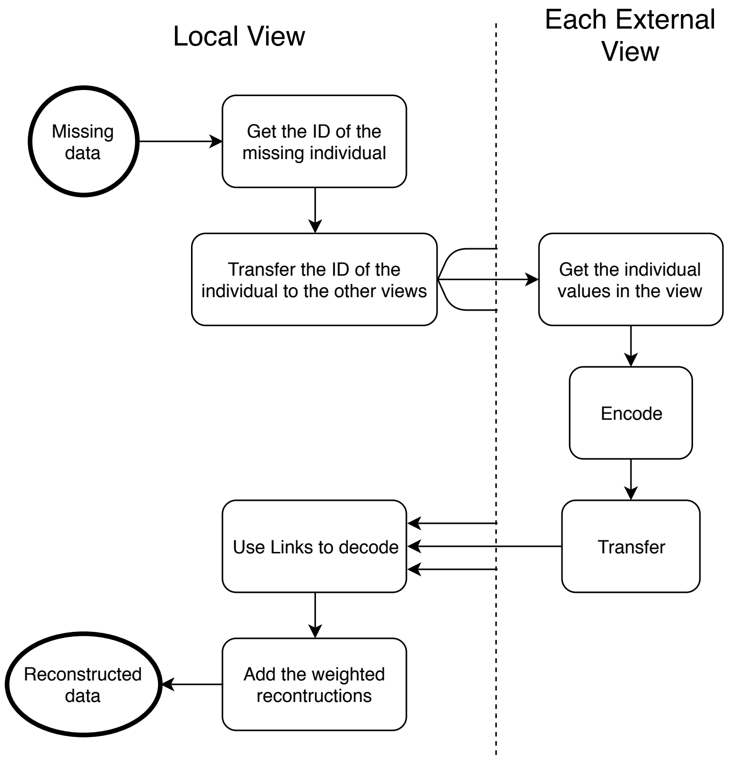

Local View
Each External View
Missing data
Get the ID of the missing individual
Transfer the ID of the individual to the other views
Get the individual values in the view
Encode
Transfer
Use Links to decode
Add the weighted recontructions
Reconstructed data

Figure 5.8: Reconstruction process going from the identification of a missing data to the getting of its reconstructed version.

5.4 Experimental setting

This Section presents the experiments that have been conducted to test our proposed method. First are presented in Section 5.4.1 the datasets which have been used during the experiments, then the global methodology used to analyze the system behavior is described in Section 5.4.2. The measures used to quantify the results are presented in Section 5.4.3, and finally numeric results are presented in Section 5.5.

5.4.1 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 B, Cube is presented here because it is related to this chapter only. Cube is a toy example which we mainly use to test the effectiveness of the Masked Weighting Method. This 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 5.9. 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 5.5.

5.4.2 Methodology

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.

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 5.5.1). 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 and determine the impact of our new combination method.

An intermediary result is presented in Section 5.5.2: the reconstruction

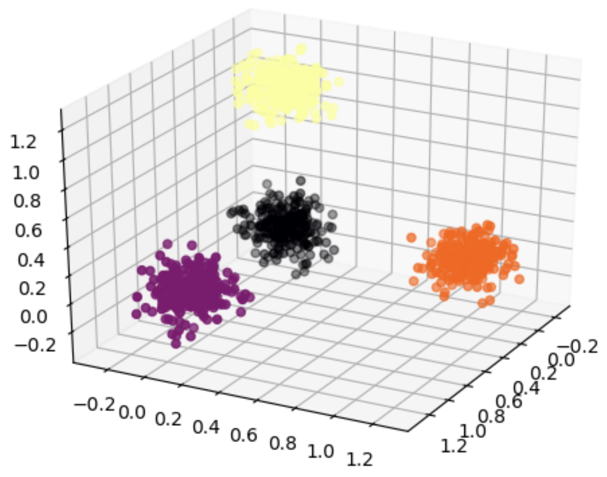


Figure 5.9: The Cube dataset

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 Weighting Method on a classification task performed on the reconstructed data (Section 5.5.3). Finally, the third and last set of experiments consists in the analysis of the masks values for the toy example (Section 5.5.4). 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 5.8 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 the various datasets sizes, which increases the training time necessary for each neural network.

5.4.3 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:

$$MSE(x,y) = \frac{1}{K}\sum_{i=1}^{K}(x_i - y_i)^2 \tag{5.10}$$

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:

$$MRD(X,Y) = \frac{1}{K}\sum_{i=1}^{K}\left|\frac{x_i - y_i}{y_i}\right| \tag{5.11}$$

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 [79].

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 axis, which is equivalent to supress a column in the original 3-dimensional dataset, prevents the local view to have any information on this axis, 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 5.10.

5.5 Results

This Section is divided following the different kind of experiments that have been conducted. In Section 5.5.1 are presented the numeric results of the reconstruction process, then in Section 5.5.2 are presented some visual results on the quality of the reconstructed individuals using the MFDD database. Section 5.5.3 presents the results obtained on the classification process performed on the reconstructed individuals, and finally Section 5.5.4 presents the analyzis conducted on the evolution of the masks coefficients depending on the information shared by views.

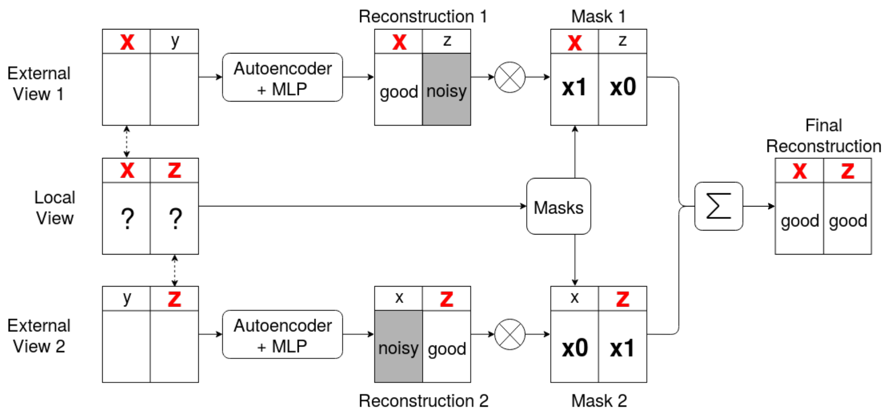


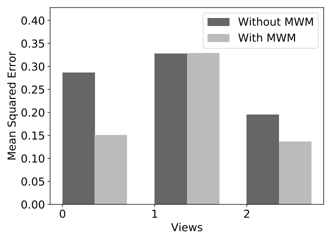
Figure 5.10: The combination of two partially good reconstructions into a good one. In this example, each view has enough information to reconstruct only one feature out of the two in the local view (doted lines). The Masked Weighted Method is designed to favor the best reconstructed part of each partial reconstruction, hence the $\times 0$ and $\times 1$ in the masks.

5.5.1 Basic reconstruction with and without the Masked Weighting Method

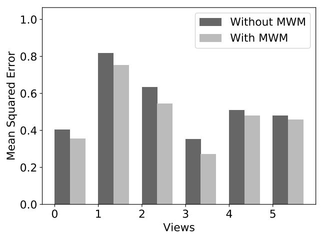
During this experiment, we were interested in the impact of our combination method on the results of the system. A summary of the results on WDBC, MFDD, Madelon and Cube can be found in Figure 5.5.1, Figure 5.12 and 5.5.1.

For WDBC, MFDD and Cube, the Masked Weighting Method significantly reduces the MSE for almost every view (Figures 5.11a, 5.11b and 5.11d). 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 5.12a, 5.12b and 5.12d): the reconstructed individuals are closest to their original versions. The exceptionnal results obtained for the MSE on the Cube dataset (Figure 5.11d) 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 5.5.4.

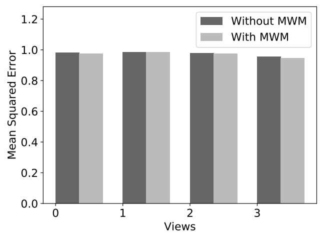
Considering the reconstruction results on the Madelon dataset (Figure 5.11c and Figure 5.12c), the high MSE and MRD values were expected because of the numerous noisy features present in every view (480 out of 500): the Links could not reconstruct noise based on some more noise. The values around 1 for the MSE and MRD (Figure 5.11c and 5.12c) can be



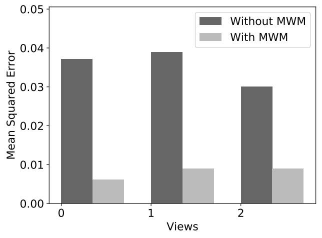
(a) WDBC



(b) MFDD



(c) Madelon



(d) Cube

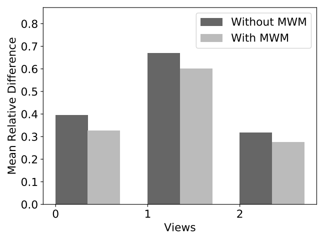
Figure 5.11: Mean Squared Error for all the datasets. A lower value corresponds to a better result.

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 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.

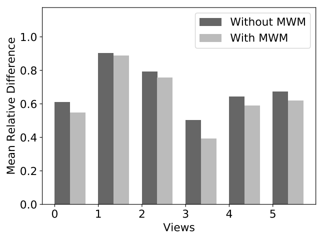
5.5.2 Graphical Reconstruction of Handwritten Digits

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 5.14. The contrast difference is explained by the normalization of the descriptive vectors before plotting.

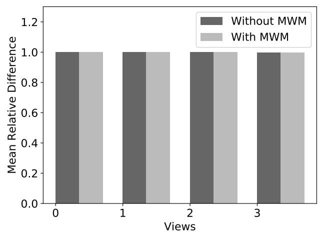
A sample of the individuals reconstructed is presented on Figure 5.15.



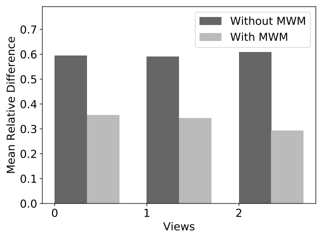
(a) WDBC



(b) MFDD



(c) Madelon



(d) Cube

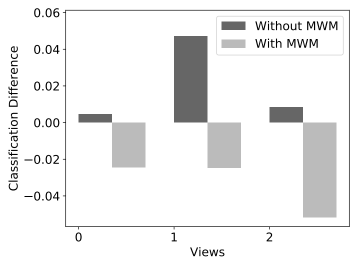
Figure 5.12: Mean Relative Difference for all the datasets. A lower value corresponds to a better result.

While the MSE and the MRD are high for this view (Figure 5.11b and Figure 5.12b), it appears visually that the reconstructed individuals can be easily recognized.

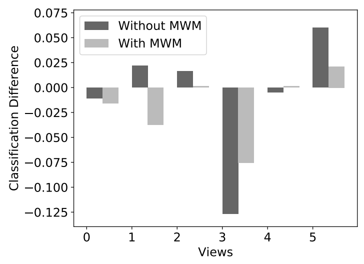
However, while it is true for most of the reconstructed images, some examples do not work as well, as presented in Figure 5.16. Moreover, even if the numbers can be recognized, a blurring effect can be observed even on the best reconstructed examples. This puts forward the fact that the system approximation 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.

5.5.3 Impact on the Classification Score:

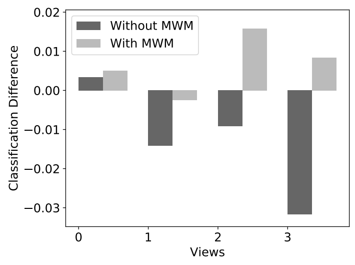
For the second experiment, when looking at Table 5.1 and Appendix 5.5.1, we notice that the Collaborative Reconstruction System gives classification scores comparable to these obtained on the original data: for WDBC, MFDD and Cube, the maximum absolute value of the Classification Difference is 7.5% (for the version using the Masked Weighting Method) when the mean



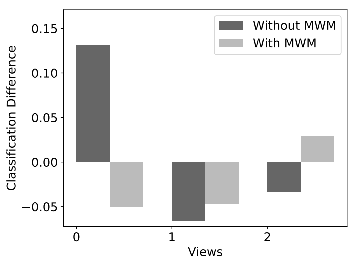
(a) WDBC



(b) MFDD



(c) Madelon



(d) Cube

Figure 5.13: 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. Values above 0 indicates that our classifications are better on reconstructed individuals than on original ones.

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 5.1: 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 re-

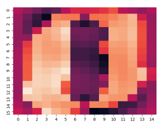
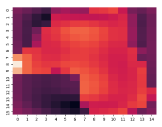
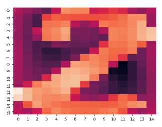
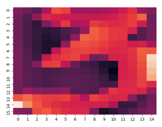
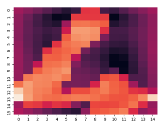
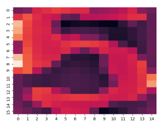
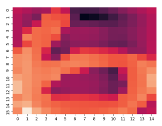
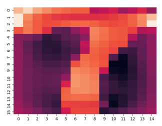
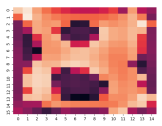
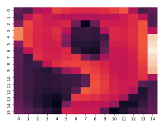

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

Figure 5.14: Sample of the original images available in the MFDD dataset

construction 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 method 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.

5.5.4 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 5.4.1, 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

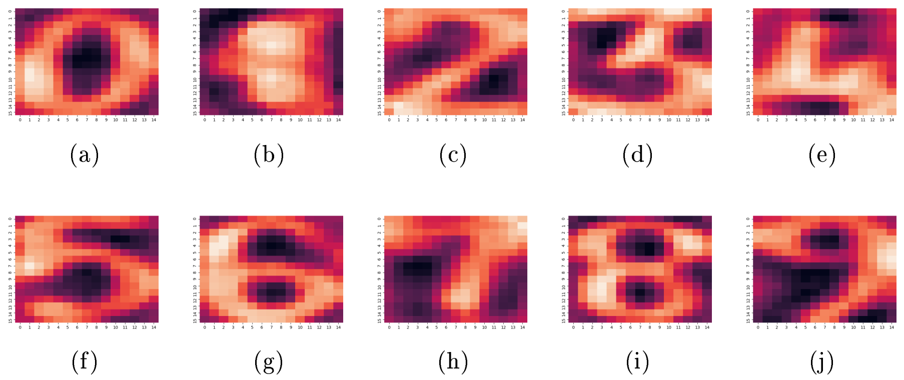


Figure 5.15: Sample of the reconstructed images available in the MFDD dataset. Some good examples.

feature, as presented in Figure 5.10.

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 5.2. 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 5.2: 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

5.6 Conclusion

In this chapter, in a global context of multiplication of multi-view data, we have presented a new Cooperative Reconstruction System. The purpose of

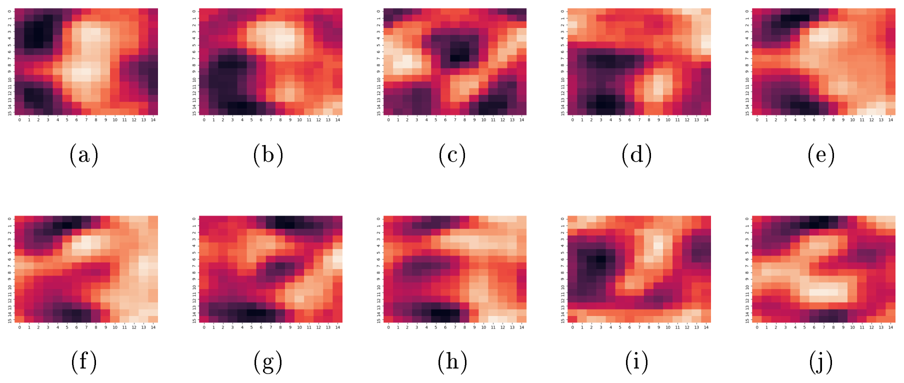


Figure 5.16: Sample of the reconstructed images available in the MFDD dataset. Some bad examples.

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.

To further develop the work presented in this chapter, it could be useful to study the link between the autoencoders and the MLP in charge of transfering this information. During the above experiments, the hyperparameters of each neural networks hadto be set manually. The automatic definition of such parameters, or at least the identification of a relation between a code and its use to infer information, may improve the quality of the results obtained with the Collaborative Reconstruction System.

Chapter 6

Conclusion and Perspectives

Contents

6.1 Summary of the contributions

The aim of this thesis is to explore the possible improvements that could be done regarding communications in a collaborative multi-view context using unsupervised methods.

6.1.1 Contributions applied to incremental training of Collaborating Clutering

In most of literature about clustering, the problem is defined for a specific moment in time, without any possibility to modify the results in case of a change in data distribution through time. While incremental training is a specific field in Machine Learning, it has never been studied in the particular case of Collaborative Clustering. The adaption of the inter-views communications in order to perform incremental training of a set of collaborative views has been studied in this thesis.

Because Collaborative Clustering is based on the results achieved locally by each clustering method, the definition of an incremental Collaborative Clustering method implies the use of an incremental clustering method locally. Self Organizing Maps have been choosen for this purpose because they have already been extensively used for Collaborative Clustering. However, the work already available in the literature on incremental Self Organizing Maps was incompatible with the requirement of Collaborative Clustering: while the incremental adaptation of maps always required modification of its topology, the paradigm of Collaborative Clustering requires that topologies remain constant all along the training in order to be compared. Thus, we present an incremental version of Self Organizing Maps based on the adaptation of the temperature function of this later. By doing so, the training of a map only depends on the distribution of last arriving data, making the adaptation to Collaborative Clustering possible.

Experimental results are provided on four different datasets to attest to the efficiency of our method.

Our contributions regarding the incremental training of Collaborative Clustering are:

- The definition of an incremental version of Self Organizing Maps not based on topological modification of the maps.
- The adaptation of Collaborative Clustering score function to enable incremental training.
- The experimental tests attesting of the efficiency of our method.

6.1.2 Contributions applied to inter-view communications for Collaborative Clustering

In the context of Collaborative Clustering, a local view receives information from all the other external ones. These informations are used to modify the results obtained locally in order to find the best possible concensus among all the existing views. To find the best concensus implies to find the best way for views to exchange information, but also to know how to combine these informations in order to achieve the desired concensus.

This information combination has already been studied in the literature about Collaborative Clustering and is based on a set of collaboration weights defining the importance a view gives to the information provided by another view. To define this importance is equivalent to modify the relative value of the collaborative weight corresponding to this view. In this thesis, we present a method to automaticaly update these values through a training process. This method is based on a problem under constraint defined both by a cost function representing the current concensus score of the system (knowing each local result and the pairwise importance weights) and by a constraint on the values of the collaboration weights.

The analytical results show that the algorithm tends to create clusters of views mutually agreeing on the results they got on their local individuals. This interpretation is coherent with the original goal of the method: by combining similar views and by lowering the impact of the dissimilar views on the score, the achieved concensus is more likely to be better than if all the importance weights were equal. These theoretical results have been tested on five different datasets which were voluntary choosen to be dissimilar in order to test the generecity of our method.

Thus, our contributions regarding the improvement of communications in Collaborative Clustering are:

- The definition of a new weighting method defining the importance a view has to give to the information provided by of one of its peers. This method has the advatange to get rid of the parameter p used coinjointly with a sum rather than a product in [71] and also to not require the use of the simplification $\beta = \alpha^2$.

- The presentation of the analytical fundation of this method as well as its interpretation.

- The experimental tests presenting the results our method can achieve on datasets varying in terms of nature, size and complexity.

6.1.3 Contributions applied to Collaborative Reconstruction

After developing the two axis presented above, it appeared that when gathering data in a multi-view context, it is very likely that data are not gathered neither through the same process nor at the same moment, and this even if it is performed on the same set of individuals. Thus, data about an individual may be missing in a specific view while its other descriptions may be available in all the other views. The intuition behind the idea developed here is that it is possible to use all the available remote informations about an individual in order to get a first approximation of its missing local description. We refer to this new paradigm as Collaborative Reconstruction.

This time, information transfer from a view to another one is performed by two different components instead of the usual importance weights. First, a set of neural networks (one per external view) is used to infer a first approximation of the individual knowing the information coming from a single view. Then, a weighting method based on vectors rather than scalar is used to combine all the external inferences. We call this method Masked Weighting Method. To ensure a minimum security on data transfer, an autoencoder is first used locally before transfering any information to prevent the receiving view from accessing original data which are not its.

For this contribution again, experimental results are provided to attest of the efficiency of our method. The experimental set up is more important than for the other contributions because the efficiency of the method could not be defined as easily as for the other methods. Because there is no comparable work in the literature as far as we know, we have suggested to analyze the following points in order to attest to the quality of a reconstruction:

- The reconstructed individual should be as near as possible from the original sample (considering the RMSE as the reference distance in our experiments)
- Even if the reconstructed individual is relatively far from its original version, it may be considered as good if a classification method can still predict its correct label only based on the reconstructed features (the Random Forests algorithm has been used during these experiments).

Thus, our method has been tested following these two points, and the results are presented and analyzed in this thesis. A set of graphical reconstructions of handwritten digits is also displayed to enable the visual validation of the reconstruction efficiency.

To sum up our contributions regarding Collaborative Reconstruction, we present:

- The definition of the new paradigm of Collaborative Reconstruction.
- The definition of a system enabling to reconstruct missing data in a collaborative context.
- The definition of a new combination method which weights are automatically trained in a collaborative context.
- The experimental results of our method on various datasets.

6.1.4 Implementation

The different algorithms presented in this thesis have been implemented using either R or Python language. Here is a list of the main components developed all along the previously mentioned experiments:

- The original Self Organizing Maps as well as our incremental version in R.
- Collaborative Clustering methods, the original one based on Self Organzing Maps and the incremental version have also been developed in R. This has been done conjointly with the Self Organizing Maps development.
- Autoencoders, Multi-Layer Perceptron and our Masked Weighting Method have been coded in Python using the Pytorch library. They correspond to standard components of our Collaborative Reconstruction System.
- A reusable version of our Collaborative Reconstruction System has been developed in Python.

6.2 Short term perspectives

Perspectives of the work presented in this thesis depend on the use case considered.

Perspectives regarding Collaborative Clustering: Regarding automatic training of collaborative weights in a collaborative context, possible extensions 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. Furthermore, the application of our weighting method could be applied to incremental Collaborative Clustering.

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

6.3 Long term limitations and perspectives

The most interesting perspectives for this thesis would be to continue the research initiated on Collaborative Reconstruction of missing individuals. As presented previously, there are points which can still be technically improved regarding each component of the system. Moreover, while the tests performed on the system have presented consistent results regarding its efficiency, a theoretical analysis of the whole system might be useful. Similarly, research on the interaction between MLP trainings and the Masked Weighting Method training may improve reconstruction results.

It may also be interesting to consider the information given by a view to another one and the use that can be made of it. Nowadays, data privacy and security are hot topics which have to be adressed regarding the evolution of technology and the use that is made of it. To put constraints on the information transfered while allowing to still improve local result in any way may globally benefit fields such as Collaborative Clustering or Collaborative Learning. So far in our method, the security put on data tranfer is done using an Autoencoder, preventing the external view to have access to original data while still allowing it to infer something linked to its local data representation. While it is a first step toward data security, we are still far from satisfying to rules such as these defined by Differential Privacy [19].

More generally, even if it intuitivelly appears to be a vast domain, building a theoretical framework describing collaborative processes in Machine Learning may make possible the extension of the paradigm to a wider range of applications than just clustering and reconstruction. The exchange of information between two views linked both by interests (to get new information to improve local results) and by constraints (original data should not be shared) is a concept which could be used as a basis for further researches.

Bibliography

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[2] A. Asuncion and D.J. Newman. UCI Machine Learning Repository, 2007.

[3] 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.

[4] 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.

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

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

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

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

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

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

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

[12] 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.

[13] 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.

[14] 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.

[15] Da Deng and Nikola Kasabov. Esom: An algorithm to evolve self-organizing maps from online data streams. In *Neural Networks*, volume 6, pages 3–8. IEEE, 2000.

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

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

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

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

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

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

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

[35] 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.

[36] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cyb.*, 43:59–69, 1982.

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

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

[39] 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.

[40] 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.

[41] Dan Li, Jitender Deogun, William Spaulding, and Bill Shuart. Towards missing data imputation: a study of fuzzy k-means clustering method. In *International conference on rough sets and current trends in computing*, pages 573–579. Springer, 2004.

[42] 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.

[43] 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.

[44] 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.

[45] Pierre-Alexandre Murena, Jeremie Sublime, Basarab Matei, and Antoine Cornuéjols. An information theory based approach to multisource clustering. 07 2018.

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

[47] 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.

[48] Shigeyuki Oba, Masa-aki Sato, Ichiro Takemasa, Morito Monden, Kenichi Matsubara, and Shin Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

[49] Andrew P Papliński. Incremental self-organizing map (isom) in categorization of visual objects. In *ICONIP*, pages 125–132. Springer, 2012.

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

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

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

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

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

[55] 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.

[56] 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.

[57] Yehezkel S Resheff and Daphna Weinshall. Optimized linear imputation. *arXiv preprint arXiv:1511.05309*, 2015.

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

[59] 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.

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

[61] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

[62] 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.

[63] Peter Schmitt, Jonas Mandel, and Mickael Guedj. A comparison of six methods for missing data imputation. *Journal of Biometrics & Biostatistics*, 6(1):1, 2015.

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

[65] Jérémie Sublime, Nistor Grozavu, Younès Bennani, and Antoine Cornuéjols. Collaborative clustering with heterogeneous algorithms. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-18, 2015*, 2015.

[66] 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.

[67] 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.

[68] 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.

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

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

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

[76] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Miss-

ing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[77] Stef Van Buuren. *Flexible imputation of missing data*. Chapman and Hall/CRC, 2018.

[78] 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.

[79] 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.

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

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

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

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

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

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

[86] 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

Appendix A

Résumé en français

A.1 Contexte

Cette thèse portant sur l'étude des communications au sein de modèles d'apprentisage automatique collaboratif a été dirigée par Raja Chiky (ISEP) et co-encadrée par Jérémie Sublime (ISEP) et Sylvain Lefebvre (ISEP).

L'objectif principal de cette thèse était d'étudier les communications inter-vues au sein de modèles d'apprentissage collaboratifs afin d'améliorer la transmission d'informations entre les vues. Cette idée a été déclinée suivant deux axes en fonction du type d'application concerné:

- Le clustering collaboratif pour lequel chaque vue disposera initialement d'un clustering local qui sera ensuite modifié afin d'arriver à une série de concensus entre les vues. La modification de chaque clustering local se base sur l'échange d'informations entre les vues, afin que les resultats obtenus localement puissent être utilisés par les vues externes. Le clustering collaboratif s'attache à trouver un concensus aussi global que possible plutôt qu'à améliorer chaque clustering local. Le clustering collaboratif est à distinguer de l'Ensemble Learning qui lui cherche à trouver un concensus unique entre toutes les vues à l'aide d'une fusion au sein d'un modèle global de l'ensemble des clusterings locaux. Au sein de cette thèse, nous parlerons majoritairement de la version horizontale du clustering collaboratif, pour laquelle chaque vue dispose du même ensemble d'individus décrits au sein de chaque vue par un ensemble différent de caractéristiques. Deux sous-axes ont été explorés concernant les communications inter-vues:

1. L'optimisation des coefficients définissant l'importance que chaque vue accorde à l'information reçue de ses paires. Pour cela nous proposons une nouvelle méthode d'apprentissage permettant d'adapter dynamiquement ces poids à l'aide d'un apprentissage.
2. La proposition d'une méthode d'apprentissage au cours du temps (que nous qualifierons d'en ligne) permettant à des vues de communiquer au fil du temps afin de faire évoluer les résultats obtenus localement à d'éventuels changements de distribution.

- La reconstruction collaborative dont le but est de reconstruire localement des données manquantes à l'aide d'informations présentes dans les vues externes. Cette application est développée dans cette thèse avec la proposition d'un système permettant d'inférer l'approximation d'un individu à l'aide entre autre de réseaux de neurones. Ces réseaux seront utilisés soit pour coder l'information à transférer afin d'assurer une sécurité minimum, soit pour inférer les valeurs locales d'un individu en se basant sur l'information reçue de la vue externe.

A.2 Clustering collaboratif

Le clustering collaboratif est défini par un ensemble de base de données (appelées vues) ayant chacune opéré un clustering sur leurs données locales. Le but du clustering collaboratif va être de faire s'échanger de l'information entre les vues afin de modifier chaque clustering local pour finalement se rapprocher d'un concensus entre les vues.

Se pose alors le problème du recoupement d'information lorsqu'une vue locale reçoit des informations provenant de plus d'une source externe. Les méthodes existantes dans l'état de l'art se basent sur une pondération de ces informations à l'aide de coefficients scalaires [9, 50, 43, 23, 68, 55]. Cependant, la méthode de définition de ces coefficients est à chaque fois empirique, et c'est sur ce constat que se basent les travaux présentés dans la première partie de cette thèse. Un second constat après étude de l'état de l'art a été qu'il n'existait actuellement pas de méthode permettant d'effectuer un apprentissage en ligne (au cours du temps) de modèles collaboratifs. Un second sous axe d'exploration a ainsi consisté en la modification d'une méthode existante de clustering collaboratif [23] basée sur des cartes auto adaptatrices [36] afin de l'adapter à l'apprentissage en ligne. Ces travaux étaient de plus motivés par la volonté de rendre les modèles collaboratifs réactifs aux éventuels changement au cours du temps dans la distribution des données.

L'ensemble des travaux présentés dans cette section se base sur la fonction de coût définissant le score du modèle à chaque instant:

$$Q^i = \alpha_i Q^i_{local}(V_i) + Q^i_{collab}(V_i, V_{j\neq i}) \tag{A.1}$$

$$= \alpha_i Q^i_{local}(V_i) + \sum_{j\neq i} \beta^j_i C^i_j(V_i, V_j) \tag{A.2}$$

Ces formules contiennent l'ensemble des éléments nécessaires pour définir un problème de clustering collaboratif. Q représente à chaque fois un critère d'évaluation, Q^i représente la valeur de ce critère pour la i-ème vue V_i, avec une distinction entre Q^i_{local} et Q^i_{collab} qui définissent les critères sur respectivement les résultats locaux du clustering ainsi que sur l'état du concensus entre les vues. C^i_j définit la dissimilarité entre les vues V_i et V_j. La pondération entre le critère local et les différentes mesures de similiraités est assurée par l'ensemble de coefficients α_i et β^j_i donc la fonction et la définition sont donnés dans la section suivante.

À noter que les définitions des critères Q_{local} et Q_{collab} sont propres soit à l'algorithme de clustering local utilisé dans chaque vue [23], soit à la définition même du problème de clustering [45]. Dans les deux cas, le critère doit être redéfini pour chaque problème.

A.2.1 Optimisation des poids pour du clustering collaboratif basé sur des cartes auto adaptatrices

Nos contributions presentées dans cette section sont les suivantes:

- Nous proposons une méthode d'optimisation automatisée et non-supervisée afin d'ajuster la valeur des coefficients définissant l'importance que les vues doivent mutuellement s'accorder lors de leur apprentissage collaboratif.

- Nous démontrons expérimentalement que notre méthode d'optimisation est capable de détecter les vues bruitées qui auraient pu détériorer les apprentissages finaux.

- Nous fournissons les propriétés théoriques de notre méthode. En particulier, nous montrons que notre méthode d'optimisation définit un méta-clustering sur les vues, en les regroupant suivant leurs similarités.

La définition de notre méthode d'optimisation s'est faite grâce à l'ajout de deux contraintes au problème initial. La première a été que, quelque soit la vue V_i, la valeur associée à Q^i_{local} devrait être égale à $\alpha_i = 1$. Cette contrainte traduit le fait qu'en divisant l'ensemble des β^j_i par α_i dans l'équation A.1, on obtient un ensemble de coefficients se trouvant uniquement sur les collaborations. Le but des α et des β est d'établir une pondération *relative* des uns par rapport aux autres. L'aspect relatif de cette pondération nous permet de fixer artificiellement la valeur de l'un des coefficient (α_i) à 1.

La second contrainte a été posée sur l'ensemble des β:

$$\forall j \quad \prod_{j\neq i}^{N} \beta^j_i = 1, \quad \forall (i,j) \quad \beta^j_i > 0 \tag{A.3}$$

Ce type de contrainte a déjà pu être rencontrée dans des travaux relatifs au clustering multi-vues [12]. De plus, il a été montré dans [70] que la contrainte de prime abord plus intuitive $\sum_{j\neq i}^{N} \beta^j_i = 1$ mène à des résultat non satisfaisants et qu'un paramètre supplémentaire p devait être défini et appris afin de parvenir à un résultat exploitable.

Le problème d'optimisation obtenu étant maintenant sous contrainte, nous avons utilisé la méthode de Karush-Kuhn-Tucker afin de déterminer les valeurs optimales des coefficients β. Pour tout $j \neq i$, nous obtenons:

$$\beta^j_i = \frac{(\prod_{k\neq j} C^i_k)^{\frac{1}{N-1}}}{C^i_j} \tag{A.4}$$

Si l'on essaie d'interpréter ce résultat, on constate que pour une vue donnée, notre méthode octroie plus d'importance aux vues qui ont des coefficients de dissimilarités C^i_j faibles, avec des valeurs de $\beta > 0$ si la dissimilarité est inférieure à la moyenne géométrique des similarités avec les autre vues, et des valeurs de $\beta < 0$ dans le cas contraire. Notre méthode définit donc l'importance d'une collaboration suivant la similarité des résultats obtenus pour chaque vue. Ce point peut se comprendre intuitivement: si l'on cherche à obtenir le meilleur score de concensus possible, il faut privilégier les collaborations de vues similaires et limiter les collaborations de vue en désaccord.

Notre méthode d'optimisation s'incrit dans le cadre d'un apprentissage collaboratif standard. L'algorithme détaillé peut être trouvé dans l'Algorithme 8.

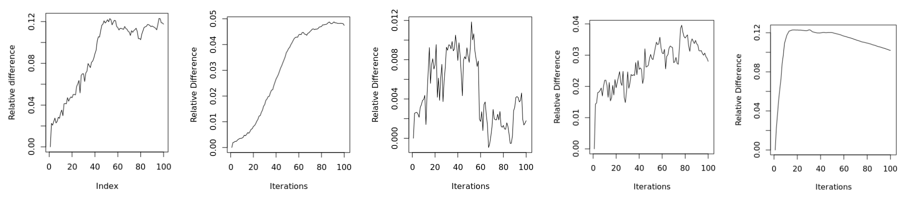
La méthode précédente a été testée sur plusieurs jeux de données de tailles et de complexités variées: WDBC, Waveform, Spambase, Isolet et VHR Strasbourg. La comparaison avec une méthode sans adaptation de poids a été effectuée afin d'attester de l'efficacité de notre méthode. Cette

Algorithm 8: Algorithme topologique de collaboration horizontale

```
Initialisation: Initialiser toutes les cartes de prototypes $W$
  aléatoirement.
Étape locale: Initialisation des cartes
forall Vue $i$ do
    Minimize the objective function of the classical SOM Minimiser la
      fonction objectif des cartes auto-adaptatrices standards.
end
Étape collaborative:
forall Vue $i$ do
    For $w$ fixé, calculer: $\beta$ en à l'aide de l'Équation A.4 Mettre à jour
      les prototypes de toutes les cartes: $w^* = \text{argmin}_w \mathcal{C}(w, \alpha, \beta)$
end
```

comparaison s'est faite en étudiant la différence relative entre les critères A.1 respectifs des deux méthodes.

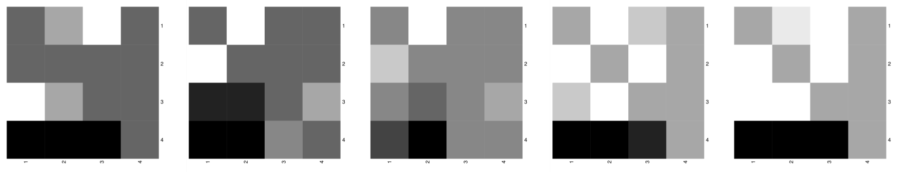
Les résultats (Figure A.1) mettent en avant que la différences est toujours positive, démontrant que le critère pour la méthode avec optimisation des β améliore le score du modèle (pour rappel, plus un score est faible, plus les dissimilarités sont faibles, et plus on est proche du concensus). Les valeurs des β sont présentées graphiquement sur la figure 4.2.



(a) WDBC (b) Waveform (c) Spambase (d) Isolet (e) VHR Strasbourg

Figure A.1: Différences relatives des critères pondérés avec et sans optimisation des β tout au long du processus d'apprentissage

Ces cartes font clairement apparaître l'identification des vues bruitées par notre méthode. Tandis que toutes les vues arrivent à identifier les vues bruitées afin de ne pas prendre en compte leurs résultats, les vues bruitées considèrent les vues non bruitées indépendamment de leurs résultats. On



(a) WDBC (b) Waveform (c) Spambase (d) Isolet (e) VHR Strasbourg

Figure A.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 $\beta = 1$.

peut aussi noter que les vues non bruitées ne coopèrent pas toujours entre elles. Ainsi pour WDBC, les vues 1 et 3 collaborent exclusivement entre elles, tandis que la vue 2 tire son information des deux vues précédentes.

De plus, comme on peut le voir pour Waveform, les vues similaires ont tendance à se regrouper entre elles. Les deux vues bruitées collaborent exclusivement entre elles, de même que les deux vues non bruitées.

En conclusion, notre méthode permet d'adapter dynamiquement les communications inter-vues pour du clustering collaboratif à l'aide de coefficients scalaires représentant l'importance qu'une vue accorde à l'information d'une de ses paires. L'efficacité de la méthode ainsi que sa capacité à regrouper les vues similaires sont démontrées par les expériences. Dans la section suivante, nous présentons les résultats obtenus sur l'adaptation du clustering collaboratif afin de permettre son apprentissage en ligne. Cet axe a été étudié afin d'explorer l'impact qu'aurait un tel contexte sur les communications inter-vues.

A.2.2 Cartes auto adaptatrices incrémentales appliquées au clustering collaboratif

Dans cette section, nous présentons les contributions suivantes:

- La définition d'une méthode permettant d'apprendre des Cartes Auto Adaptatrices (CAA) en ligne (au cours du temps).

- L'adaptation d'une méthode de clustering collaboratif permettant de tenir compte des modifications apportées aux Cartes Auto Adaptatrices.

- Le développement et la présentations de résultats empiriques montrant l'efficacité de notre méthode.

L'adaptation du clustering collaboratif à l'apprentissage en ligne a nécessité l'adaptation du modèle utilisé localement pour obtenir une première version des clusterings locaux. Nous avons choisi les Cartes Auto Adaptatrices car elles constituent un modèle couramment rencontré dans la littérature sur le clustering collaboratif [25, 23, 55].

Bien que plusieurs versions en ligne des Cartes Auto Adaptatrices ont été proposées dans la littérature [15, 49], toutes se basent sur des modifications topologiques des cartes originales afin de les adapter à l'évolution des données. Ce type de changement n'est pas permi initialement par le clustering collaboratif, du fait des comparaisons qui sont susceptibles d'être faites neurones à neurones entre les cartes. Plutôt que d'adapter les règles du clustering collaboratif afin de permettre ce genre de modifications, nous avons choisi de définir une nouvelle version en ligne de ces cartes pour ensuite l'adapter au clustering collaboratif.

La modification de ces cartes se base sur la modification de la fonction de température permettant de définir le voisinage influencé par la modification de chaque neurone. Cette fonction est normalement dépendante du temps, comme présenté dans la formule suivante:

$$\lambda(t) = \lambda_{\max}\left(\frac{\lambda_{\min}}{\lambda_{\max}}\right)^{\frac{1}{t}} \tag{A.5}$$

avec $\lambda_{\max}$ et $\lambda_{\min}$ deux constantes définissant respectivement les températures initiale et finale du modèle. Lorsque la carte est dite “chaude”, la modification d'un neurone va impacter un large voisinage, c'est l'étape initiale durant laquelle la carte s'adapte grossièrement aux données. Plus l'apprentissage va avancé, plus la carte va se “refroidir”, pour arriver à de petites valeurs de λ. Durant cette phase, la carte adaptera plus localement l'emplacement de ces neurones. L'avantage de cette méthode par rapport à une méthode telle que K-means est que l'on conserve une dimension topologique entre les clusters, alors que les centroids de K-means sont indépendants les uns par rapport aux autres.

Afin de s'affranchir de la dépendance temporelle de la fonction de température et afin de la rendre réactive aux éventuels changement dans la distribution des données, nous avons défini la fonction de température $\widetilde{\lambda}$ suivante:

$$\widetilde{\lambda}(B,W) = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \|x_i - \omega_{\chi(x_i)}\|_2 \tag{A.6}$$

avec B le batch des N_{batch} dernières données arrivées, W l'ensemble des neurones de la carte et χ la fonction qui a un point associe l'indice du neurone ω le plus proche de la carte. Cette fonction présente le double avantage de ne pas être dépendante du temps tout en s'adaptant à l'état actuel des données: si en moyenne les données sont loins de la carte, la température sera élevée car l'ensemble de la carte aura besoin d'être adaptée. À l'inverse, si les données sont proches de leurs neurones respectifs, la température sera faible car seules des modifications locales des neurones seront nécessaires.

Cette modification a été inclue dans les équations régissant le comportement du clustering collaboratif afin de le rendre utilisable en ligne. Dans un soucis de concision, le détail des formules n'est pas précisé ici.

Afin d'attester de l'efficacité de notre méthode, nous avons effectuer des apprentissages sur plusieurs jeux de données: Spam base, Waveform, WDBC et Isolet. Pour chaque jeu de donnée, nous avons regardé quelle était l'erreur de quantification moyenne par vue avec et sans utilisation du clustering collaboratif online. L'erreur de quantification est définie par l'erreur quadratique moyenne entre les individus du batch et leurs neurones les plus proches.

Les résultats obtenus sont présentés dans le tableau A.1.

Ces résultats indiquent que pour toutes les bases de données sauf Isolet, la carte auto adaptatrice en ligne que nous avons proposée obtient des scores avoisinnant ceux de la version avec clustering collaboratif. C'est un point utile car l'utilisation du clustering collaboratif peut éventuellement réduirel a qualité des résultats obtenus localement du fait de la recherche d'un concensus global. Pour la cas particulier d'Isolet, les meilleurs résultats pour la méthode collaborative peuvent être expliqués par la limitiation de l'impact des données bruitées (96% des données) grâce au clustering collaboratif.

Nous avons de même étudié l'impact de notre méthode sur l'apprentissage au cours du temps d'un modèle collaboratif. Pour se faire, nous avons comparé les valeurs des puretés obtenus d'une part par notre méthode de clustering collaboratif online, et d'autre part par une méthode de clustering classique pour laquelle nous prenions chaque itération comme une unité de temps. Les résultats obtenus sont présentés sur la Figure A.3. La pureté d'un neurone est égale à la fraction d'individus qui lui sont rattachés et qui appartiennent à la classe la plus représentée sur ce noeud. Par extension, la pureté d'une carte est égale à la pureté moyenne de ses noeuds.

Ces figures font apparaître une meilleure pureté pour notre méthode par

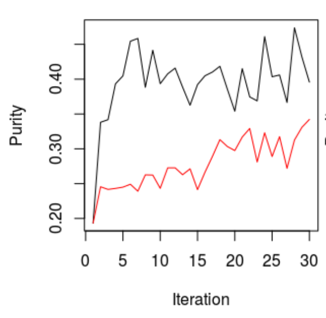
Table A.1: Erreur de quantification moyenne pour chaque base de donnée. Les nombres en gras sont les plus petits pour chaque ligne

	Vue	CAA Incrémentales	Clustering Collaboratif Incrementale
Spam Base	1	0.31	**0.26**
	2	**0.18**	0.19
	3	0.18	**0.16**
Waveform	1	**0.18**	0.23
	2	**0.17**	0.19
	3	**0.24**	0.30
WDBC	1	**0.19**	0.19
	2	**0.16**	0.19
	3	0.20	**0.16**
Isolet	1	2.15	**1.27**
	2	2.84	**1.38**
	3	2.85	**1.37**

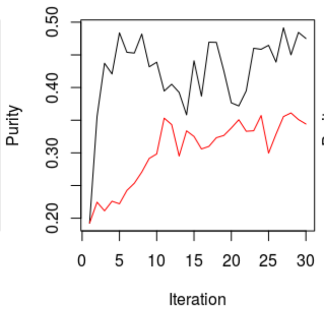
rapport à la méthode classique dans la première phase de l'apprentissage. À l'adaptation en temps réel qui est faite sur la fonction de température, permettant d'obtenir de meilleurs résultats plus rapidement qu'avec une méthode classique. On peut de plus remarquer l'influence du paramère N_{batch} sur l'apprentissage: une valeur plus faible implique une variance plus importante de la pureté au cours du temps. Ce point se comprend intuitivement par le fait que lorsque N_{batch} est faible, le système dispose de peu d'informations pour adapter ses neurones, ce qui implique nécessairement une grande variabilité suivant l'échantillon de données en cours de traitement.

En conclusion, nous avons présenté dans cette section une méthode permettant d'effectuer un apprentissage en ligne des cartes auto adaptatrices sans utiliser de modification topologique. Cette méthode a ensuite été adaptée au clustering collaboratif, puis son efficacité a été présenté sur différents jeux de données. L'influence du nombre de données par échantillon a été étudiée et reliée à la variance des scores obtenus lors de l'apprentissage.

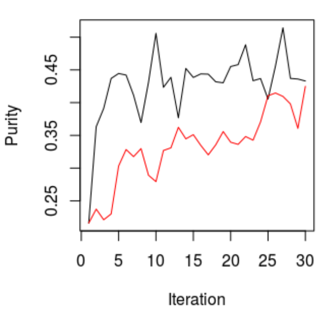
La section suivante présente un use case différent de celui du clustering traité jusqu'à présent. L'objectif principal de cette thèse étant d'explorer les possibilités offertes par les communications inter-vues dans un contexte col-



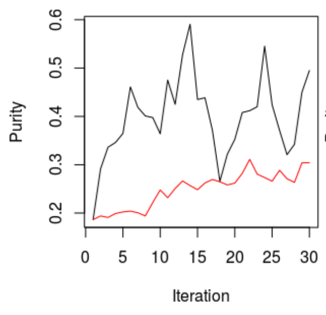
(a) $N_{batch} = 10$, View 1



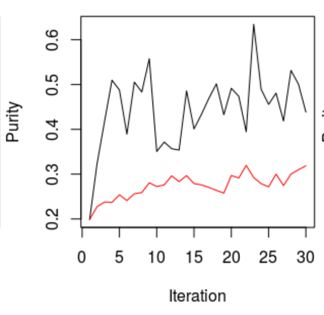
(b) $N_{batch} = 10$, View 2



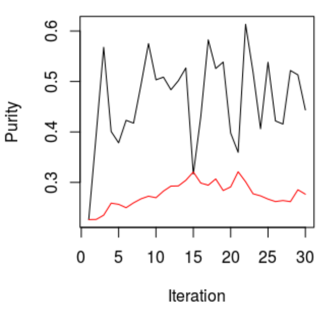
(c) $N_{batch} = 10$, View 3



(d) $N_{batch} = 3$, View 1



(e) $N_{batch} = 3$, View 2



(f) $N_{batch} = 3$, View 3

Figure A.3: Évolution des puretés for le jeu de données Isolet. Les lignes rouges représentes les CAA incrémentales tandis que les lignes noires représentent les CAA collaboratives. Chaque itération correspond à l'arrivée d'une nouvelle donnée

laboratif, nous nous sommes intéressé au problème des données manquantes et aux manières d'y pallier.

A.3 Système de reconstruction collaborative

Les contributions présentées dans cette section sont les suivantes:

- Définition d'un nouveau cas d'utilisation dans un contexte d'apprentissage collaboratif: la reconstruction collaborative.
- Définition d'un modèle pouvant répondre au problème posé.
- Définition d'une nouvelle méthode de pondération permettant de combiner des vecteurs point à point.

- Attestation de l'efficacité du modèle au travers d'expériences menées sur divers jeux de données avec identification des limites du modèle et de pistes d'améliorations.

Le clustering collaboratif se base sur l'hypothèse que les vues communicantes disposent de suffisamment d'individus en commun pour échanger leurs résultats et les comparer. Cependant en pratique, la récupération de plusieurs bases de données sur un même ensemble d'individus est une chose difficile à mettre en place, et la récupération des données peut entraîner l'apparition de données manquantes. L'idée initiale développée dans cette section a été qu'il était possible d'utiliser l'ensemble des informations disponibles sur le sous ensemble d'individus en commun entre les vues pour inférer les valeurs des individus manquants.

Une représentation de l'architecture de notre méthode est présentée sur la Figure A.4. Alors que le clustering collaboratif se base sur une correspondance cluster à cluster, la reconstruction collaborative s'appuie elle sur l'inférence de données cibles à partir de données initiales. La difficulté de la mise en pratique de cette idée a motivé l'utilisation de réseaux de neurones, et plus particulièrement de Perceptrons Multi-Couches (aussi appelés Liens, ou Link en anglais), comme liens entre les vues afin de donner une première approximation de l'individu manquant. L'apprentissage de tels réseaux est possible du fait de la présence d'un ensemble d'individus en commun à une paire de vue. En utilisant les descriptions de la vue externe comme entrée et celles de la vue locale comme sortie, il était possible d'effectuer un apprentissage supervisé du réseau.

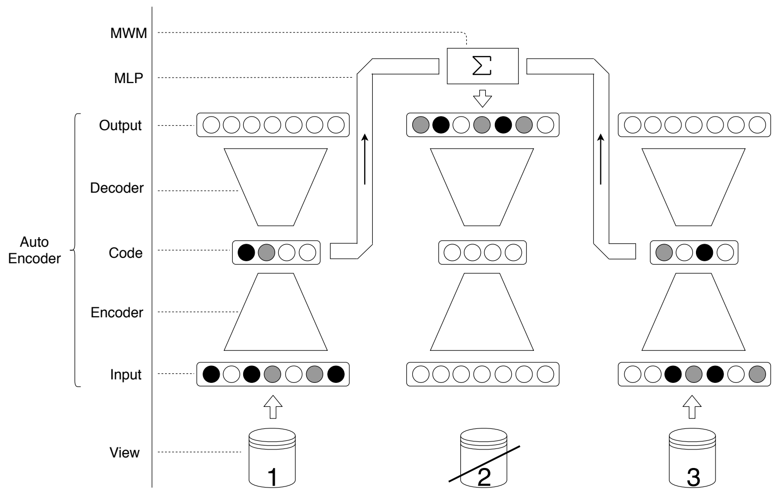


Figure A.4: Système de recontruction coopérative. Dans cet example, les vues 1 et 3 envoient leur versions codées de l'individu à la vue 2.

Un autre aspect propre au clustering collaboratif constitue la sécurité qu'il met en place concernant les données en transit: les données originales ne sont jamais transférées d'une vue à une autre. À la place sont transférées soit les identifiants des clusters auxquels appartiennent les individus [68, 65], soit des informations utilisées au cours de l'apprentissage comme c'est le cas pour l'apprentissage à l'aide de cartes SOM [23, 43]. Dans le cas de la reconstruction collaborative, l'utilisation d'un réseau de neurone comme système d'inférence implique nécessairement un codage des données initiales sous la forme d'un vecteur scalaire qui sera utilisé à la place des données originales comme données d'entrée du réseau en charge de l'inférence. Ce codage sous forme de vecteur est assuré dans notre modèle par un auto-encodeur, une catégorie de réseaux de neurones ayant la particularité de reconstruire en sortie les données fournies en entrée [78].

Enfin, le dernier composant de notre méthode consiste en une méthode de pondération, que nous avons appelé méthode de pondération par masques, qui permet de combiner l'ensemble des approximations créées à partir des données reçues des vues externes ($N-1$ dans un système à N vues où chaque vue externe aurait une information sur l'individu manquant). L'idée fondamentale de cette méthode consiste en l'utilisation de vecteurs permettant de pondérer des individus point à point (descripteur par descripteur) plutôt que d'utiliser un unique coefficient pondérant l'ensemble de l'individu. En effet, il est facile d'imaginer que pour un individu manquant donné, chaque vue externe permette d'en reconstruire seulement une partie. L'utilisation d'un coeffcient de pondération unique ne permet pas de prendre en compte cette différenciation. Un schéma décrivant le processus de pondération par un ensemble de vecteurs, que nous appelerons désormais masques, est présenté sur la Figure A.5. Les valeurs des masques sont entraînées au préalable afin de mieux correspondre à chaque vue, et chaque vue possède $N-1$ masques, un par vue externe.

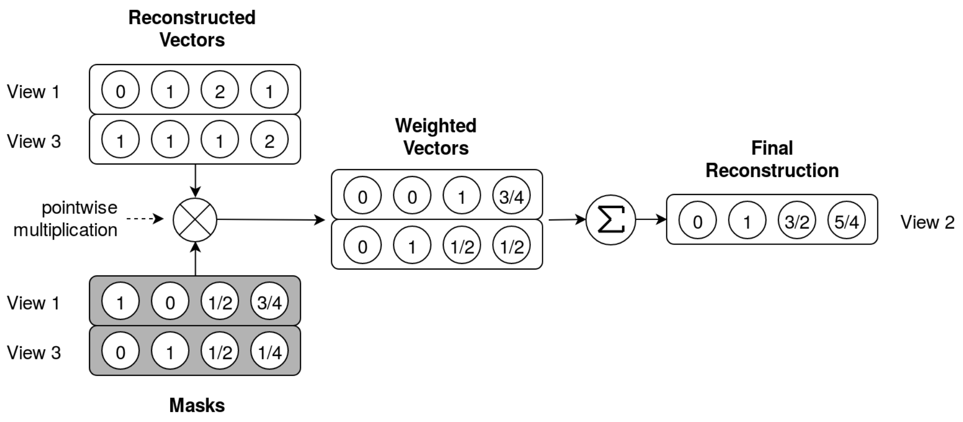


Figure A.5: La méthode de pondération par masques. La vue 2 possède les individus inférés à l'aide des informations des vues 1 et 3, et elle utilise ses masques entraînés au préalable afin d'obtenir le résultat pondéré final.

L'idée générale de la méthode d'apprentissage consiste simplement en une descente de gradient sur un critère défini comme la distance entre l'individu cible et sa version reconstruite à l'aide des réseaux de neurones et de notre méthode de pondération. Une seconde méthode itérative a été proposée et démontrée analytiquement en annulant le gradient précedemment obtenu et en mettant à jour les poids des masques de manières itérative.

Concernant l'apprentissage du système dans sa globalité, il peut être effectué séquentiellement:

1. Chaque vue entraîne de manière indépendante un auto-encodeur en charge de chiffrer les données originales.

2. Chaque vue encode l'ensemble de sa base de donnée d'apprentissage et envoie le résultat à ses paires

3. Chaque vue entraîne $N-1$ perceptrons multi-couches, 1 par vue externe, en mettant en correspondance les individus chiffrés et les individus cibles.

4. Les vues utilisent leurs perceptrons pour inférer les valeurs des individus présents dans la base d'apprentissage, chaque vue possède alors $N-1$ base de données d'individus inférés.

5. Les poids des masques sont entraînés à l'aide des $N-1$ bases d'individus inférés et des données originales.

Une fois l'ensemble de ces étapes effecutées pour toutes les vues du problème, la reconstruction d'un individu manquant devient possible.

Plusieurs tests ont été effectués à l'aide de notre système de reconstruction collaborative:

- Afin de tester l'efficacité du système global, des reconstructions ont été faites et comparées aux versions originales en utilisant l'erreur quadratique moyenne.

- Le test suivant a consisté en une classification des individus reconstruits à l'aide de l'algorithme de Random Forest pour tester si la classe prédite correspondait à la classe réelle.

- Enfin, la qualité des reconstructions a été testée en remplaçant la méthode de pondération par masque par une simple moyenne. Le but de ce test a été de vérifier l'efficacité de notre nouvelle méthode de pondération.

Ces tests ont été effectués sur 4 jeux de données différents: WDBC, Multi-Features Digital Dataset (MFDD), Madelon et Cube. Ce dernier est un jeu de donnée artificiel crée spécialement pour tester l'efficacité de notre méthode de pondération. Il est constitué d'un ensemble de 4 groupes de points répartis dans un espace en 3 dimensions. Trois clusters se trouvent à chacune des extrémités des vecteurs de base, le quatrième se trouvant à l'origine de l'espace. Trois vues sont ensuite créées en projetant l'ensemble du jeu de données suivant chacun des trois vecteurs de base. L'intérêt d'une vue ainsi créée est que l'information permettant de reconstruire ses individus est par définition répartie dans les deux vues restantes. Si notre méthode de pondération fonctionne, on peut s'attendre à ce que les masques privilégient une caractéristique spécifique tout en rejetant totalement l'autre.

Un schéma décrivant la pondération par masque et comment elle permet de s'affranchir de caractéristiques mal reconstruites et/ou bruitées peut être trouvé sur la Figure A.6.

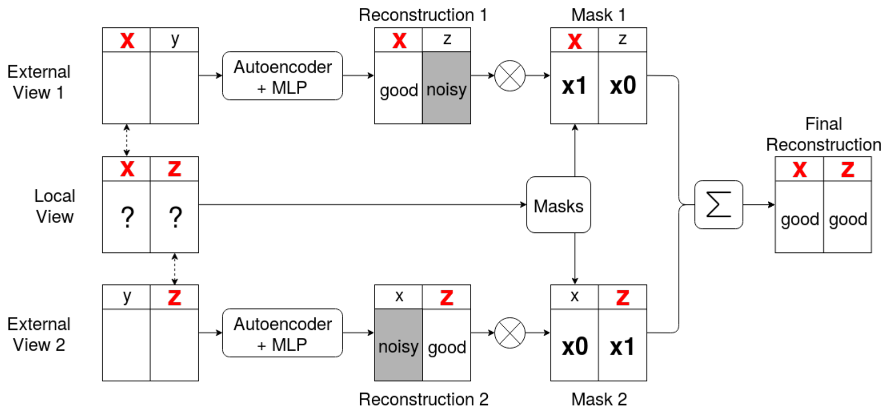


Figure A.6: Combinaison de deux reconstructions partiellement bonnes. Dans cet exemple, chaque vue dispose d'assez d'information pour reconstruire seulement une caractéristique sur les deux dans la vue locale (pointillés). La méthode de pondération par masques favorise les parties les mieux reconstruites de chaque résultat, d'où le $\times 0$ et $\times 1$ dans les masques.

L'utilisation du jeu de données MFDD permet d'obtenir des reconstructions visuelles, permettant d'appréhender plus facilement leurs qualités. La figure A.7 montre un échantillon de 10 images avant apprentissage. La figure 5.15 présente 10 reconstructions considérées comme de bonnes qualités. Bien qu'il ne s'agisse que d'un résultat purement visuel, la majorité des individus présente une reconstruction de qualité avoisinante à celle des images présentées ci-dessous. Cependant dans certains cas, le système n'a pas été capable d'inférer efficacement les chiffres à reconstruire, ce qui a mené à des reconstructions comme celles présentées en figure 5.16.

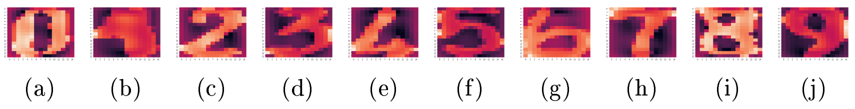


Figure A.7: Échantillon des images disponibles dans le jeu de données MFDD.

Expérimentalement, le système obtient des résultats en terme d'erreur

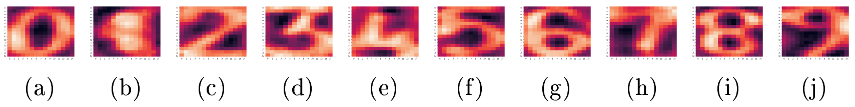


Figure A.8: Échantillon d'images bien reconstruites pour le jeu de données MFDD.

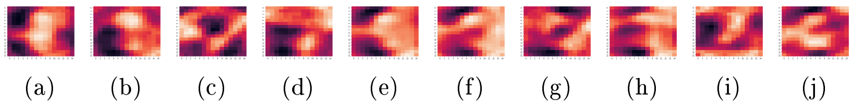
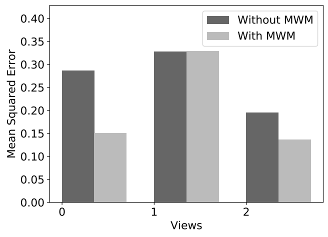


Figure A.9: Échantillon d'images mal reconstruites pour le jeu de données MFDD.

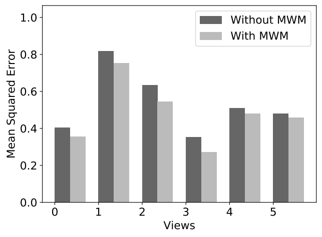
quadratique (relativement à chaque caractéristique) plutôt moyens (Figure A.3). Cependant on constate sur un échantillon graphique que la reconstruction globale des individus présente une qualité qui permet de reconnaître l'individu initial. C'est aussi ce que montre les résultats en terme de classification présentés sur la Figure 5.5.1. Bien que les reconstructions ne soient pas exactement fidèles aux originales, elles sont suffisamment précises pour obtenir des scores en classification proches de ceux obtenus sur les données originales. En résumé, le système arrive à récupérer certaines informations caractéristiques des individus et à les retranscrire, bien que la qualité de la reconstruction en elle même soit améliorable.

Comme présenté sur la Figure A.3 et 5.5.1, le test indiquant l'efficacité de la méthode de pondération par masque par rapport à une moyenne indique que notre méthode améliore sensiblement la qualité de reconstruction des individus. De plus, les tests conduits sur Cube montrent clairement que notre méthode arrive à détecter automatiquement quels caractéristiques sont à privilégier suivant la vue externe considérée. En effet, les valeurs des coefficient portant sur la caractéristique que les vues ont en commun avoisine toujours 0.92, tandis que les autres sont proches de 0.14, indiquant un favoritisme fort pour les caractéristiques en commun. Les résultats concernant la classification semblent cependant indiquer que notre méthode n'apporte pas d'amélioration significative de ce point de vue là.

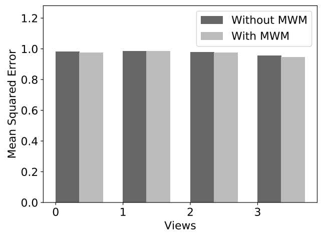
Pour conclure cette section, nous avons défini un nouveau contexte d'apprentissage collaboratif permettant de reconstruire des individus manquant à partir d'informations présentes dans d'autres vues. Une nouvelle méthode de pondération collaborative a été définie et testée. Le système, bien que présentant



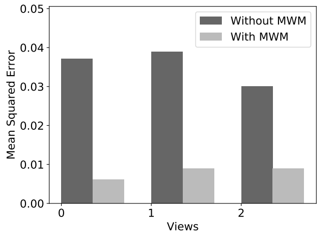
(a) WDBC



(b) MFDD



(c) Madelon



(d) Cube

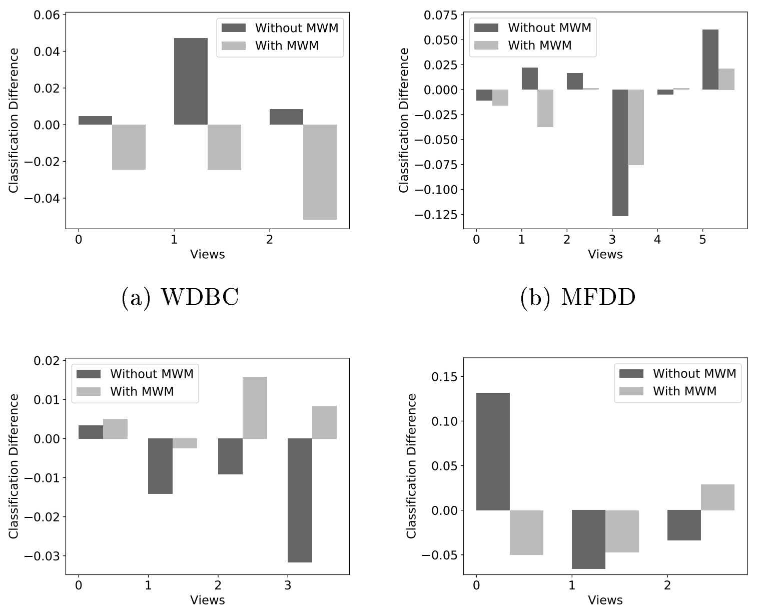
Figure A.10: Erreur quadratique moyenne pour tous les jeux de données. Une valeur plus faible correspond à un meilleur résultat.

des résultats améliorables en terme de reconstruction, permet de capturer des informations intrinsèques aux individus telles que leur classe, et la méthode de pondération arrive à identifier clairement quels caractéristiques sont à privilégier lorsque plusieurs sources sont disponibles pour la reconstruction.

A.4 Résumé des contributions scientifiques et perspectives

A.4.1 Contributions au clustering collaboratif

Durant cette thèse, nous avons pu définir une nouvelle méthode permettant d'apprendre automatiquement les coefficient définissant l'importance qu'une vue doit accorder à l'information fournie par une de ses paires. Le fondement théorique de cette méthode se base sur la définition d'un problème d'optimisation sous contrainte que nous avons résolu à l'aide de la méthode de Karush-Kuhn-Tucker. Notre seconde contribution a consisté en la définition d'une nouvelle méthode permettant d'apprendre des cartes auto adaptatrices au cours du temps. Cette méthode présente l'avantage de ne



(a) WDBC (b) MFDD

(c) Madelon (d) Cube

Figure A.11: Scores et Différences de Classification pour WDBC, MFDD et Madelon. Une valeur au dessus de 0 indique que les classifications effectués à l'aide des données reconstruites sont meilleures que celles basées sur les données originales.

pas dépendre de modifications topologiques, ce qui a permis son adaptation au clustering collaboratif. Cette méthode se base essentiellement sur la redéfinition de la fonction de châleur à la base des cartes auto adaptatrices pour faire en sorte qu'elle dépende de l'arrivée des nouvelles données plutôt que du temps.

A.4.2 Contribution à la reconstruction collaborative

L'étude des communications au sein des modèles collaboratifs nous a permis de définir un nouveau contexte d'apprentissage permettant de reconstruire des données manquantes localement à l'aide des informations contenues dans les vues externes. Afin de proposer une première approche, nous avons défini un système basé sur les réseaux de neurones et sur une nouvelle méthode de pondération. Nous avons pu tester son efficacité ainsi que l'efficacité de notre méthode de pondération sur des cas variés montrant que le système était capable de capturer et de reconstruire suffisamment d'informations pour permettre l'indentification graphique et le classement des individus reconstruits.

A.4.3 Perspectives

Plusieurs axes de recherches peuvent être développés à court et long terme en se basant sur les travaux eeffectués lors de cette thèse.

Perspectives à court termes

Concernant l'optimisation des pondérations dans un contexte multi-vues, des extensions possibles peuvent s'appliquer au clustering collaboratif vertical pour lequel les cartes auto adaptatrices doivent gérer des informations décrites par un même ensemble de caractéristiques appliquées à un ensemble d'individus différents. De plus, l'application des méthodes décrites dans cette thèse à des algorithmes de clustering tels que les Generative Topographic Maps peut être envisagée dans un premier temps, puis à des méthodes de clustering non topologiques dans un second temps.

Comme présenté dans la section ci-dessus, les résultats obtenus à l'aide du système de reconstruction collaborative sont encore perfectibles du point de vue du détail de reconstruction. Bien que les résultats préliminaires soient encourageant, des recherches approfondies seraient menées à court termes sur le lien entre la taille du code utilisé pour le transfert de données et les difficultées rencontrés par les perceptrons multi-couches lors de leur entraînement. De même, l'application de la méthode de pondération par masques sera étudiée dans des contextes collaboratifs autres que celui de la reconstruction.

Perspectives à long termes

À plus long terme, l'étude de l'information transférée d'une vue à une autre et l'utilisation qu'il est possible d'en faire permettra de faire avancer le domaine de l'apprentissage collaboratif. De nos jours, la sécurité des données sont des sujets vivement étudiés qui doivent prendre en considération les différentes utilisations qui sont faites des données, et le contexte collaboratif se prête particulièrement bien à ce genre de réflexions et de recherches. Nos travaux dans le domaine de la reconstruction se basent sur des Auto-encodeurs pour fournir un minimum de sécurité, mais cette solution n'est pas satisfaisante sur le long terme.

Enfin, l'étude théorique d'un framework collaboratif permettant son application à des cas d'utilisations autre que la reconstruction et le clustering pourrait s'avérer utile. L'échange d'informations entre deux vues liées à la fois par des intérêts (obtenir de nouvelles informations pour améliorer les résultats locaux) et par des contraintes (partage de données sensibles) est un concept qui sera utilisé comme base pour de futures recherches.

Appendix B

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 [53] 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.