N° d'Ordre : XXXX

EDSPIC :

Université Sorbonne Paris Nord

ÉCOLE DOCTORALE GALILÉE

Ph.D. Thesis

by

Yohan Foucade

for the degree of

Doctor of Computer Science

Learning from Data and Learners

defended in June 2022 in front the following jury :

Thesis supervisor :		
	Younès Bennani	Professor, Université Sorbonne Paris Nord
Reporters :		
	Yann Guermeur	Research Director, LORIA-CNRS
	Mathilde Mougeot	Professor, ENSIIE, ENS Paris-Saclay
Examiners :		
	Basarab Matei	Associate professor, HDR, Université Sorbonne Paris Nord
	Nicoleta Rogovschi	Associate professor, HDR, Université Paris Cité
	Dan Vodislav	Professor, Université de Cergy-Pontoise

Acknowledgements

Résumé

Cette thèse porte sur l'apprentissage collaboratif. Ce paradigme d'apprentissage est une des nombreuses méthodes ayant vu le jour au cours de ces dernières années afin de tenter d'exploiter le plus efficacement possible le volume croissant de données générées et stockées de par le monde.

Parmis ces méthodes, on trouve par exemple l'apprentissage fédéré et le clustering ensembliste. Ces différentes approches ont pour point commun de permettre un calcul distribué et préservant la confidentialité des données traitées. Leur objectif est d'obtenir un modèle entrainé sur des données dispersées sur différents noeuds d'un réseau. Ce qui est fait soit en entrainant directement un modèle partagé par les différents sites, soit en cherchant un consensus entre plusieurs apprenants entraînés séparément.
Ce dernier point constitue la principale différence avec l'apprentissage collaboratif. En effet, dans ce paradigme, on ne cherche pas à réaliser un consensus. L'objectif est que chaque apprenant bénéficie des résultats obtenus par ses homologues tout en préservant la confidentialité des données.

Un des enjeux du domaine est le choix des sites distants avec lesquels collaborer et de l'intensité de la collaboration. Sans précaution particulière, il est tout à fait possible que les performances locales soient détériorées suite à la collaboration. Nous proposons l'utilisation de l'entropie comme mesure de l'incertitude quant aux prédictions des différents modèles afin de pondérer, pour chaque modèle, l'information qu'il reçoit de ses homologues.

Nous proposons une implémentation dans le cadre de l'apprentissage supervisé en utilisant des arbres de décision comme modèle de base.

Un autre verrou scientifique réside dans les difficultés liées à l'apprentissage non-supervisé. Outre le fait qu'il n'existe pas de mesure objective de la performance, le paradigme d'apprentissage collaboratif souffre du fait qu'il n'y ait pas de correspondance évidente entre les classes formées par les différents modèles. Nous proposons une architecture de clustering collaboratif permettant à tous les modèles de s'améliorer. Nous proposons aussi une analyse détaillée de ce mode de collaboration en étudiant l'effet, au niveau individuel, de l'échange d'informations entre les différents apprenants.

Les approches proposées dans les deux premiers chapitres sont essentiellement heuristiques et ont été implémentées dans le but de répondre aux verrous scientifiques identifiés au début de ces travaux. Nous en fournissons néanmoins une analyse théorique portant sur la complexité en espace, sur certaines conditions suffisantes pour que l'algorithme termine, et sur les bornes de généralisation.

Abstract

This thesis focuses on collaborative learning. This machine learning paradigm is one of the many methods that have emerged in recent years in an attempt to efficiently mine the growing volume of data generated and stored around the world.

Federated learning and ensemble clustering are, e.g., belong to these numerous methods. These different approaches have in common that they allow distributed computation and preserve data privacy. Their purpose is to obtain a model trained on data scattered across different nodes of a network. This is done either by directly training a model that is shared by the data sites, or by seeking a consensus between several learners that are trained separately.
This last point is the main difference with collaborative learning. Indeed, in this paradigm, we do not try to reach a consensus. The objective is that each learner benefits from the results obtained by his counterparts while preserving data privacy.

One of the issues in this area is the choice of remote sites to collaborate with and the intensity of the collaboration. Without special care, it is quite possible that local performance will be degraded as a result of the collaboration. We propose the use of entropy as a measure of uncertainty in the predictions of different models in order to weight the sources of information. We also propose a detailed analysis

An experiment was carried out in the framework of supervised learning using decision trees as an underlying model.

Another scientific obstacle lies in the difficulties associated with unsupervised learning. In addition to the fact that there is no objective measure of performance, the collaborative learning paradigm suffers from the fact that there is no obvious correspondence between the classes formed by the different models. We propose a collaborative clustering architecture allowing all models to improve. We also propose a detailed analysis of this mode of collaboration by studying the effect, at the individual level, of the exchange of information between the different learners.

The approaches proposed in the first two chapters are essentially heuristic and have been implemented in order to address the scientific obstacles identified at the beginning of this work. Nevertheless, we provide a theoretical analysis on the space complexity, some sufficient conditions for the algorithm to finish, and the generalization bounds.

Contents

List of Figures

List of Tables

Forewords

This thesis deals with collaborative learning and was carried out at the Northern Paris Computer Science Lab (LIPN)[1], within the framework of a consortium agreement called "FACEBOOK grants" between *Université Paris-Est Marne la Vallée*, *CY Cergy Paris Université*, *Université Côte d'Azur*, and *Université Sorbonne Paris Nord*[2].

The LIPN has been associated with the French National Centre for Scientific Resear (CNRS) since 1992 and is developing around several axes, in particular in combinatorics, combinatorial optimization, algorithmics, logics, software engineering, natural languages and machine learning.

Professor Younès Bennani has directed several theses dealing with unsupervised learning and in particular with collaborative clustering. The present document is a continuation of this work and extends the collaborative learning paradigm to the supervised framework.

[1] Laboratoire d'Informatique de Paris Nord, LIPN, https://lipn.univ-paris13.fr/en/home/

[2] http://www.u-pem.fr/
https://www.cyu.fr/
https://univ-cotedazur.fr/
https://www.univ-paris13.fr/

Remerciements

Introduction

This thesis focuses on collaborative learning. This learning paradigm is one of the many methods that have emerged in recent years in an attempt to efficiently exploit the growing volume of data generated and stored around the world.

Among these numerous methods are, for example, federated learning and ensemble clustering. These different approaches have in common that they allow a distributed computation and preserve the confidentiality of the processed data. Their objective is to obtain a trained model on data scattered on different nodes of a network; this is done either by collaboratively training the model directly, or by seeking a consensus between several participants that have been trained in isolation.

In this last point lies the main difference with collaborative learning. Indeed, in collaborative learning, there is no attempt to reach a consensus. Instead, the aim is for each learner to benefit from the results obtained by its counterparts while preserving data confidentiality. To do so, the learners involved in the collaboration process have to exchange information about their local findings.

One of the issues in this area is the choice of remote sites to collaborate with and the intensity of the collaboration. Without special care, it is quite possible that local performance will be degraded as a result of the collaboration. We propose the use of entropy as a measure of uncertainty in the predictions of different models in order to weight the sources of information.

We propose an implementation in the supervised learning framework using decision trees as a an underlying model.

Another scientific obstacle lies in the difficulties associated with unsupervised learning. In addition to the fact that there is no objective measure of performance, the collaborative learning paradigm suffers from the fact that there is no obvious correspondence between the classes formed by the different models. We propose a collaborative clustering architecture that allows all models to improve.

This thesis is composed of four main parts listed below.

Overview of the thesis

Chapter 1 : Learning from data and learners

In this chapter, we explain the interest of using distributed learning. We review the different distributed learning methods and analyze their properties. We will see that the purpose of most of these methods is to obtain a unique model. This is done either by collaboratively training the model directly, or by seeking a consensus between several participants that have been trained in isolation.

Collaborative learning is an exception in that it does not seek to achieve consensus between local models. Rather, the exchange of information between these models aims at improving each of them. Among the advantages of this learning mode, we can mention the fact that we make fewer assumptions about the distribution (in the probabilistic meaning of the word) of the data. In particular, we do not hypothesize that the data are independent and identically distributed. Each model can still benefit from the information transmitted by the other sites to improve its performance on the local data. Thus, all algorithms, and in particular the best performing ones before the collaboration, can potentially benefit from the collaboration and improve.

Chapter 2: Collaborative random forests

In this chapter, we propose to apply our algorithm in the context of a collaborative learning of decision trees for a classification task. Having a set of views on a dataset, we wish to train a classification tree on each of these views, while trying to benefit from the results obtained by the remote trees.

Chapter 3 : Unsupervised collaborative learning

In this chapter, we propose a collaborative clustering algorithm. It is a collaborative learning architecture in which the trained models are clustering algorithms. After a brief history of work in the field, we present the scientific challenges we address.

As we said before, each algorithm can theoretically improve its performance by benefiting from the information sent by its counterparts. However, we observe that in practice, the algorithms with the best initial performance see their scores deteriorate. We therefore propose a method allowing all algorithms to improve.

Chapter 4: Theoretical guarantees

In this chapter, we provide a theoretical analysis of the collaborative architecture. We derive the complexity in space of the collaboration step. We then provide a termination analysis and show that under mild conditions on the quality criterion, our algorithm has *STOP property*. We finally provide some generalization bounds.

1 Learning from data and learners: State of the art

1.1 Introduction

Learning algorithms are trained on increasingly large volumes of data.

Two phenomena have made it possible to train on large volumes:

1. The computing power of computers
2. The ubiquity of data (IoT, social networks...)

Much of this data is stored on personal and portable devices, and often has a confidential nature, which is a barrier to its use for machine learning.

New obstacles have emerged with this ubiquity of data. For example, technical or financial constraints may prevent us from consolidating all the data on the same physical site. In addition, increasing attention is being paid to the protection of data privacy [53, 73].

In such circumstances, machine learning algorithms must either make do with locally available data, or exchange information in order to benefit at least partially from these remote data. The algorithms will then learn not only from locally available data, but also from results obtained by remote algorithms. Learning is thus realized from data and learners.

The type of information exchanged will depend, in this context, on the way the data is distributed. There are two main types of data partition:

- Horizontal partition : different individuals are described by the same variables on the different data sites (cf figure 1.1).
- Vertical partition : the same individuals are described by different variables on each site (cf figure 1.2).

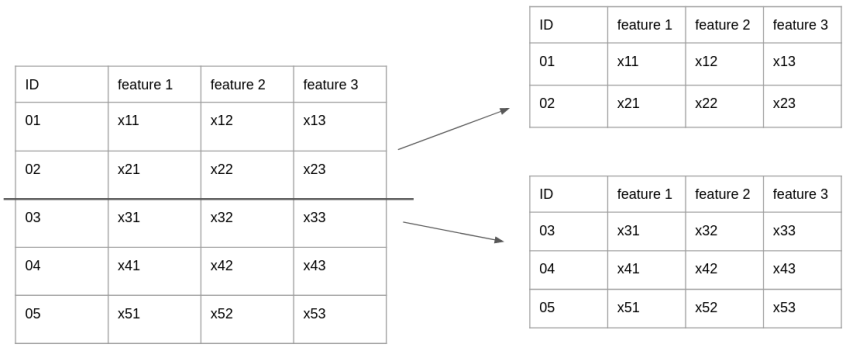


FIGURE (1.1) Horizontal data partition

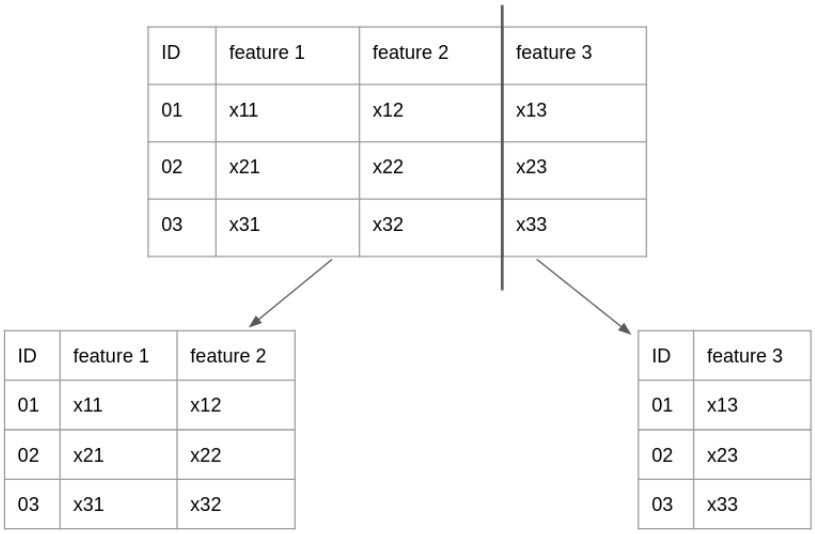


FIGURE (1.2) Vertical data partition

In the following sections, we focus on the main methods for learning from distributed data.

1.2 Federated learning

Federated learning, introduced in 2016 [45], is a decentralized machine learning architecture. Its main purpose is to train a global model using data from different devices, also called *clients*, without them having to disclose private data. This paradigm is particularly useful in a situation where:

- The number of participating clients is very large.
- The data for one client is not representative of the population distribution.
- The amount of local training data varies among clients.
- Bandwidth is limited.

In their paper, the authors attempt to train deep neural networks from decentralized data. They propose the FederatedAveraging algorithm. It is an adaptation of the stochastic gradient descent algorithm. It works according to the following rules. At each iteration, a random fraction of the clients is selected. Each of the selected clients then computes the gradient of the loss function on its local data. Finally, a central server aggregates the induced updates. Since all clients share the same model, they also share the feature space. This is a horizontal learning configuration.

The final model is shared and common to all participants. Moreover, since there are many participants, they have only a marginal influence on the learning process. This paradigm is therefore particularly sensitive to the free rider problem, and the question arises of how to encourage each of them to participate in the learning process.

Furthermore, this uniqueness of the final model implies that the final model will often perform worse than the best models that can be obtained using only local data from a client with large, qualitative data. Thus, customers with the most qualitative data have no incentive to participate in the learning process. Conversely, customers with less qualitative data are incentivized to participate [43].

Federated learning has been implemented with a wide variety of algorithms, e.g. neural networks, decision trees and random forests, logistic regression, support vector machines (SVM), *etc*.

1.3 Ensemble clustering

Ensemble clustering is also an approach to mining distributed data to improve the overall performance of a learning algorithm. Unlike federated learning, the process is not iterative. Here, the learning process is divided into two steps. First, each algorithm will use a local dataset and obtain a partition. Then, a central entity will aggregate these intermediate results and combine them to obtain a final clustering.

This approach is related to the ensemble learning methods developed for supervised learning. These methods consist in combining the results of several algorithms, for example by taking a weighted average of their predictions. The best known ensemble methods are Bagging and Boosting [21, 40, 56, 77]. Note that their also exist adaptive versions of this paradigm [13, 33, 74].

- Strehl and Ghosh [60] introduced in 2002 the problem of combining several partitions without accessing the original data. They called it *ensemble clustering*. The authors choose to represent each partition by a hypergraph whose edges are clusters. They then propose three heuristics to obtain a consensus partition from these hypergraphs

- Topchy, Jain and Punch [66] study the properties of the combination of several so called "weak" clusterings. These weak clusterings are either obtained by classical clustering algorithms, but after projection into a lower dimensional subspace, or by partitioning the data by random hyperplane.

- Wemmert and Gançarski [72] introduce an architecture based on a majority voting system. A confusion matrix is first built and allows to associate to each cluster of each algorithm a corresponding cluster in each distant algorithm. In a second step, for each object to be classified, each model will vote for the cluster in which it has classified it locally, and for each corresponding remote cluster. By taking the cluster that has received the most votes, we obtain a "best class" for each object if the cluster has obtained a majority of votes. In the case where no cluster has obtained a majority, the object is said to be *non-consensual*.

This approach to unsupervised learning is the subject of an important literature : [1, 3, 25, 27, 30, 35, 48, 78]. See [70] for a review of the said literature, and [39] for an experimental comparison of the different methods.

1.4 Collaborative learning

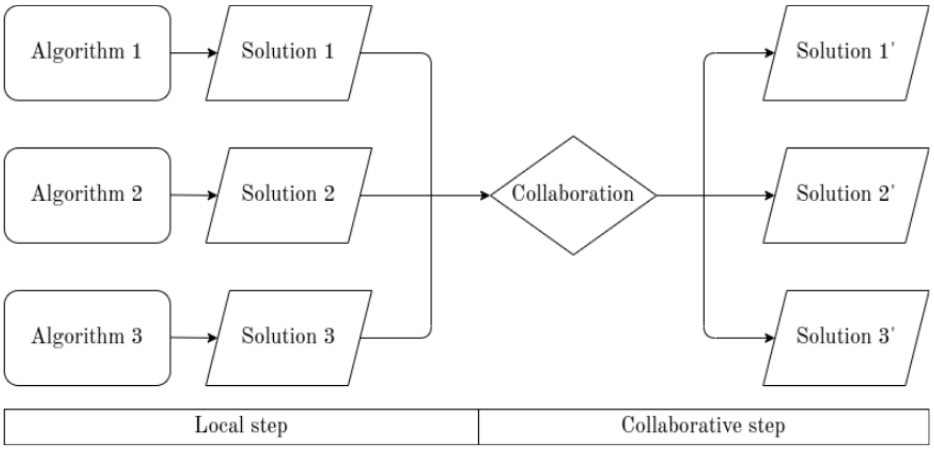


FIGURE (1.3) Representation of the collaborative learning process. The collaboration step is iterative.

1.4.1 Introduction

Most machine learning algorithms assume that data is stored on a single site. However, this paradigm is increasingly challenged by the development of massive data. In this

context, the volume, distribution and confidentiality of data are the main obstacles to the use of these algorithms. This constraint related to data storage has been the main driver for the development of Distributed data mining research [76].

In this thesis, we are particularly interested in collaborative clustering algorithms that seek to address the following problem:

> Having a collection of databases distributed on several different sites, how to partition each of them considering local data and remote classifications of other collaborating databases without sharing the data itself between the different sites?

On this topic, one of the first works was published by Pedrycz in 2002 : [50]. In this seminal paper, Pedrycz proposes a method based on the fuzzy C-means algorithm [7, 23]. Since then, many publications have improved this method or proposed new ones: [15, 28, 42, 48, 61].

The collaborative learning process is divided into two steps:

- Local step: learning algorithms learn on their local data as they would in a classical framework.
- Collaboration step: the collaboration step is often iterative. During this phase, information is exchanged between the different sites and then used by each of them to try to improve their respective models.

It is therefore possible to analyze the different existing collaborative clustering architectures according to the choices made during these different steps.

First of all, during the local step. The various implementations may require identical algorithms on each site ([28, 50]), or on the contrary allow heterogeneous algorithms

([26, 63]). Additionally, the type of clusters sought by the algorithms can be important. Indeed, one can have mutually exclusive clusters (hard clustering), overlapping clusters (soft clustering), or fuzzy clusters (fuzzy clustering).

The collaboration step starts with an information exchange step. The type of information exchanged depends on the type of data partition that is encountered. We will then see that the intensity of the collaboration between the different databases, once the necessary information has been transmitted by the remote sites, depends on a parameter named collaboration coefficient.

In the rest of this section, we will look at the type of architecture we can encounter in collaborative clustering, and then at the different possible implementations for collaboration.

1.4.2 How to partition the data present on the different nodes

One of the main goals of collaborative clustering is that each site has a local data partition. As mentioned before, these partitions can be constituted :

- of overlapping groups (soft clustering)
- of mutually exclusive groups (hard clustering)
- or fuzzy clustering

If we work with exclusive clusters, then we assign each observation to a single cluster. However, there are situations in which it is natural to assign an individual to several clusters simultaneously, in which case we say that the clusters are non-exclusive. As for fuzzy clustering, it consists in considering that each individual belongs to all the clusters with a certain degree of membership located between 0 (does not belong at all) and 1 (belongs completely). It is generally imposed that the sum of these weights is equal to 1 [65].

A second question arises: do the different sites work with the same algorithms or not? Besides the fact that collaborative clustering allows to work on a larger amount of data, it also allows the use of various algorithms. This diversity can be due to the use of radically different algorithms, but also to a difference in the parameters of the same algorithm, or to the initialization of the clusters.

Finally, if in clustering we always look for partitions such that similar individuals are in the same group, and that the groups are as dissimilar as possible, there are several types of clusters [32, 65]. One example is prototype-based clusters, where each cluster is a set of objects closer to the prototype representing that cluster than to any other prototype. In particular, the prototype can be the average of all the objects belonging to the cluster. There are also methods based on graphs, density, etc.

In [50], the author proposes that each site uses the fuzzy c-means algorithm locally [7, 23]. The local results are consequently fuzzy partitions. The clusters are based on prototypes, the latter being an average of the individuals belonging to the cluster, weighted by the degree of membership. Many contributions have since sought to improve this algorithm, initially intended for horizontal collaborative clustering and in which the bases had to agree on the number of clusters to search. As an example, we can mention Pedrycz and Rai who, in [51], propose a framework for collaborative clustering in the presence of a different number of clusters on each site.

In [28, 29], the authors propose an adaptation of Bishop's Generative Topographic Mapping (GTM) [8] for collaborative clustering. It is the probabilistic counterpart of Kohonen's self-organizing maps (SOM) [36], and is supposed to overcome most of their limitations. This generative model allows to associate to each point of a latent space, a point of a variety of the same dimension in the data space. This method allows in a second step, for a fixed observation, to find the *posterior* distribution in the latent space, the latter being a grid with a finite number of points, which are themselves

prototypes. In other words, we can calculate the contribution (or responsibility) of each prototype to a given observation. This is the probability that a given prototype has generated the observation.

In [4], the authors propose an architecture for horizontal collaboration based on the optimal transport theory. We remind that in the case of horizontal collaboration, the sites involved have access the same variables describing different individuals. In the local step, each observation is associated to the prototypes via the solution of the optimal transport problem of the data to the prototypes. At each iteration of the collaboration step, a remote model to cooperate with is chosen. Then the optimization of a new function including a term penalizing the distance between the local and remote prototypes allows the update of the parameters.

1.4.3 Organization of the collaboration

The exchange of information between the different databases is one of the main characteristics of characteristics of collaborative clustering. In this section, we will see which information should be exchanged depending on the context. Finally, it should be noted that each model is free to weight (or even ignore) information sent by the remote leaners. The way in which one can adjust the impact that the results of a remote site have on the local partition will be studied in a second step.

The information exchanged among the local models:

- information on the membership of the observations to the different clusters;
- information about the prototypes;
- other information on the distribution of the data and on the structures sought by the algorithms.

The information on the membership of individuals to the different clusters is often transmitted via the partition matrix. This of course supposes to agree on the identity of the different individuals. This is why this method is often used in horizontal collaborative clustering [14, 50, 60, 61]. The exchange of information about prototypes is only useful if the remote prototypes make sense locally. This is why this system is mostly used in vertical collaborative clustering [28, 29, 51].

1.4.4 Intensity of the collaboration

Once the information has been exchanged among the data sites, each one must decide how it will use that information in order to come up with a new classification. When this choice exists, it is often made by adjusting a parameter called "collaboration coefficient", which is generally a non-negative number. If the collaboration coefficient associated with a remote model equals zero, that means that the corresponding remote model has no effect on our future results. And the higher this coefficient, the higher the effect of the collaboration on our local results. The impact of this coefficient on the result of the collaboration has been studied by Pedrycz [50]. To do this, the author defines two indices. The first one measures the difference between the classifications obtained on two different sites after collaboration. It decreases with the increase of the collaboration coefficient. The second, on the other hand increases with this coefficient. It consists in comparing locally the difference between the classifications obtained with and without collaboration. Finally, if the collaboration coefficient is often calibrated by the user, some algorithms propose to automatize the tuning process [59, 62].

1.4.5 Types of collaborative clustering algorithms according to the nature of the data distribution

We assume that the data are distributed across different nodes in a network. Then each node can have a replica of the data set. This can be useful to minimize communication costs or to ensure continuity of service in case of failure of one of the sites. As we have seen previously, a database can be distributed in several ways (horizontal and vertical partition). We are interested here in the type of information exchanged according to the type of data partition.

Horizontal collaborative learning

In the case of a horizontal partition, the different sites hold information on different individuals. However, these individuals are described by the same variables. It is therefore possible to exchange information that can be described in the feature space.

Vertical collaborative clustering

Pedrycz's original paper describes a vertical collaborative clustering algorithm. The clustering algorithms operating on different subsets of data collaborate by exchanging information.

2 Collaborative random forests

In this chapter, we present a collaborative learning architecture for classification algorithms. We assume the existence of a vertically distributed dataset over several sites. We wish to train a learning algorithm on each site. Each algorithm will be trained in a classical way on the local data. Then, during the collaboration step, the models will exchange information about their results and adjust their states accordingly. The goal is that all algorithms will be able to improve their performance following this collaboration step.

The interest of our approach lies in the way it takes into account remote results. As we have seen in the previous chapter, in the case of a vertical collaboration, the algorithms exchange information about their predictions for each individual. This information is usually taken into account through a modification of the objective function, in order to penalize the prediction discrepancies. This has the effect of improving the performance of the algorithms that were the least efficient at the beginning, at the cost of deteriorating the performance of the best algorithms. To overcome this problem, we propose to weight the information received according to certain criteria.

In the following section, we explain in more detail the framework and what information will be exchanged between the different algorithms. We then discuss the criteria used to weight this information. Finally, we propose to test this architecture using decision trees as basic algorithms.

2.1 Problem set-up

Our goal is to train classification models on different sites, each with a view about a common dataset. The goal of each training algorithm is to obtain a classification of its local data using the results of its remote counterparts without exchanging the data themselves. We assume that each model has a probabilistic classification as output for each input data. In other words, the output of a model is not simply the predicted class for an observation, but a probability distribution over the different classes. For each data $\mathbf{x}_i,\ i = 1,\dots,N$, and each class $k \in \{1,\dots,K\}$, we call the responsibility of the class k for the individual i, and we note r_{ik} the following value :

$$r_{ik} \equiv p(y_i = k|\mathbf{x}_i, \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is the set of possible parameters of the model.

Then the matrix $R = [r_{i,k}]$ is such that for all $i = 1,\dots,N$ and all $k = 1,\dots,K$, we have

$$r_{i,k} \geq 0$$

and

$$\sum_{1 \leq k \leq K} r_{i,k} = 1.$$

It is the content of this matrix that will be exchanged between the different sites.

We propose to use entropy as a criterion to weight the information received.

2.2 Entropy as a measure of uncertainty

In this section, we describe the entropy associated with a probability distribution. To do so, we temporarily leave the framework of collaborative learning. In the rest of this section, X denotes a random variable with values in $\mathcal{X}$.

The entropy of a random variable X, denoted $H(X)$ is a measure of the uncertainty associated with this random variable.

It is defined as follows:

$$H(X) \equiv -\sum_{x \in \mathcal{X}} p(X = x) \cdot \log p(X = x) \tag{2.1}$$

If the log is expressed in base two, then the entropy is expressed in bits. It can be shown that the maximum entropy is reached for a uniform distribution:

$$p(X = x) = \frac{1}{|\mathcal{X}|}, \forall x \in \mathcal{X}, \tag{2.2}$$

where $|\mathcal{X}|$ is the cardinal of the set $\mathcal{X}$. And in this case, we have $H(X) = \log |\mathcal{X}|$.

2.2.1 Example : the entropy of a Bernoulli distribution.

A random variable X has a Bernoulli distribution of parameter $\theta \in [0,1]$ if :

$$p(X = x) = \theta^x \dot{(}1-\theta)^{1-x}, x \in \{0,1\} \tag{2.3}$$

In other words, X takes value 1 with probability θ and 0 with probability $1-\theta$. The entropy of a Bernoulli distribution of parameter θ according to θ is represented in the figure 2.1.

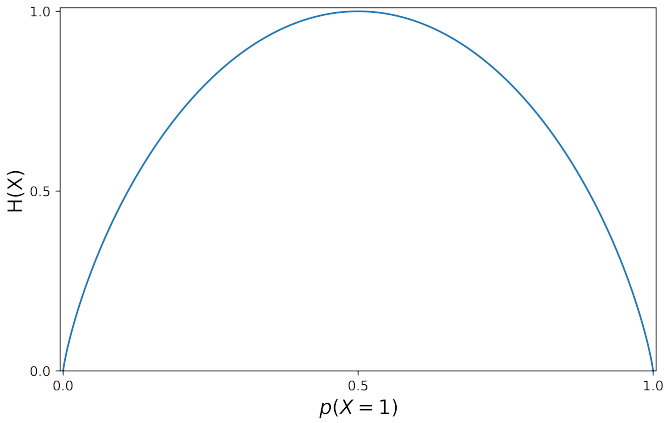


FIGURE (2.1) Entropy of a Bernoulli distribution of parameter θ, as a function of θ.

For this relatively simple probability law, the interpretation of entropy as a measure of uncertainty is rather straightforward. When $\theta = 0$ (respectively $\theta = 1$), the random variable X is takes value 0 (resp. 1) with probability 1. The uncertainty about the value taken by a random variable having such a distribution is then minimal. Conversely, when $\theta = 1/2$, then $p(X = 0) = p(X = 1) = \frac{1}{2}$. The random variable has therefore a uniform distribution on $\{0, 1\}$, the uncertainty about the value taken is maximal. All this is in agreement with what can be observed in the figure 2.1.

In the next section, we explain how we use this measure of uncertainty in the collaboration process.

2.3 Using entropy as a criterion for collaboration

As a means of measuring uncertainty, entropy provides the beginning of an answer to the question of the weighting of received information. In our framework, the classification of each individual is represented as a probability distribution $R_i = R_{i,k} : k = 1, \ldots, K$. We can then use the entropy of this distribution to calculate the uncertainty. We give some examples :

- If $R_i = (1/3, 1/3, 1/3)$, then $H(R_i) \approx 1.58$, which is the maximum possible value for three events.
- If $R_i = (0.01, 0.01, 0.98)$, then $H(R_i) \approx 0.16$, a relatively small value.
- If $R_i = (0.2, 0.3, 0.5)$, then $H(R_i) \approx 1.48$, so it is a relatively high uncertainty.

The use of entropy can serve several purposes:

- Weighting of the information received for each individual
- Change of classification according to the local certainty
- Change of classification according to the distant certainty

This will allow the best algorithms to potentially improve. Indeed, we can assume that the best performing algorithms are those that are the most successful in discriminating individuals. So on average their classifications will likely have a lower entropy. For the well classified individuals, they will keep the same prediction. On the other hand, for individuals where the decision is more difficult, these models will be able to benefit from the remote information to try to improve the classification. An illustration is provided in figure 2.2.

Example with a Gaussian mixture model

A mixture model is a probabilistic model allowing to represent the presence of subgroups in a population [47]. For that, we suppose that the distribution of the individuals is in fact a weighted sum of distributions (or components):

$$p(X) = \sum_{k=1}^{K} \pi_k \mathcal{L}(\theta_k) \tag{2.4}$$

where K is the number of components, π_k is the weight of component k, $\sum_k \pi_k = 1$, and θ_k is the set of possible parameters of the $\mathcal{L}$ distribution. We speak of a Gaussian mixture when each component $\mathcal{L}$ is a normal distribution.

For our example, we trained a Gaussian mixture model on the Iris dataset [22]. This is a dataset describing 150 of these plants, each one described by four attributes (sepal length, sepal width, petal length, petal width). The data are equally divided into three classes (setosa, versicolor, virginica). One of the classes (setosa) is linearly separable from the other two. The latter are not linearly separable.

To set ourselves in a collaborative context, we separate the 4 attributes into two views, represented in figure 2.2.

The first view has the sepal attributes. In this view, two classes overlap, while the third (setosa) is quite distinct from the others. The second view is composed of attributes related to petals. In this view, two classes are relatively close, while the third (setosa) is isolated.

On each view, we trained a three-component Gaussian mixture model. These components are represented by the elliptical contour lines in Figure 2.2. These components give, for each individual i, a probability distribution $(R_{ik})_k$ representing the probability of belonging to each class. Moreover, for each point of the respective spaces, it is possible to compute such a distribution. Indeed, for a given point, the responsibility of a component is proportional to the product of its weight and its density at this point [47]. It is therefore possible to calculate the entropy of our model at each point.

We plot the entropy of our two models in grayscale in Figure 2.2. We see that the data belonging to the setosa class are in areas of low entropy. In other words, both models have little uncertainty about the classification of these individuals. A possible collaboration would therefore probably not cause them to change their predictions for these individuals (if entropy is used as a criterion). The same is not true for the

first view for individuals located in the area where the versicolor and virginica classes overlap. All other things being equal, this model would use more information received during the collaboration step to adjust its predictions for the individuals concerned. This applies to fewer individuals for the second model, but it can still improve its predictions for the few individuals for which the classification is still unclear. All this without deteriorating its performance.

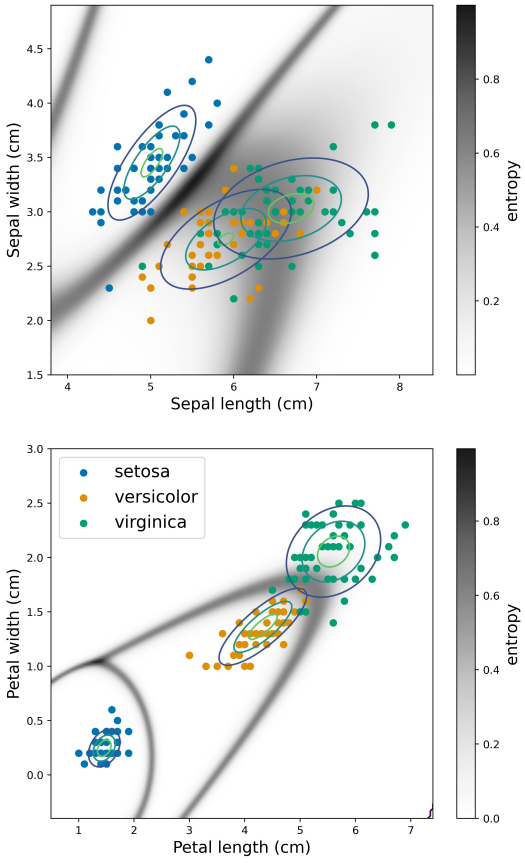


FIGURE (2.2) Entropy and Gaussian components on the Iris dataset

We have seen in this section how entropy can be used as a criterion to weight the information received during collaboration. Its interpretation in terms of uncertainty

related to a distribution allows a given model to give relatively more weight to incoming information regarding:

- Individuals whose local prediction is uncertain.
- Individuals whose distant predictions are certain.

The performance of a model can be expected to improve depending on how well the following conditions are met:

- Classification errors tend to occur at the edges of classes and where classes overlap. These regions correspond to regions of higher entropy, and for the observations found there, the model will use more remote information to fine-tune its predictions. The lower the entropy (and therefore the uncertainty) of the distant information, the more weight it will give to it. This will improve its performance if...
- ...The predictions regarding the observations located in regions of low entropy are generally correct (no systematic bias). These observations are generally located in the center of a cluster or in an isolated cluster.

The optimal way to weight this information is an interesting problem. Many implementations are possible and can be chosen according to the properties sought. Probabilistic opinion pooling is a field that studies how to combine probabilistic predictions on a set of events (these predictions being provided by experts), and to the study of their properties [20, 55]. Our topic differs from this field in that we do not seek consensus.

We have chosen a simple implementation respecting the properties listed above. We have also chosen to adopt the same implementation on all sites. However, the choice of the weighting function can be adapted to the objectives of each site. We tested our

approach using decision trees as a basic model. In the following section, we briefly recall some notions about decision trees.

2.4 Decisoin trees

A decision tree is a structure that, for each input, provides a decision resulting from a succession of choices. In machine learning, these models are trained by recursively partitioning the data space. Decision trees are among the most used algorithms because of their intelligibility and generalization capacity. The figures 2.3 and 2.4 are representations of the same decision tree trained on the Iris dataset.

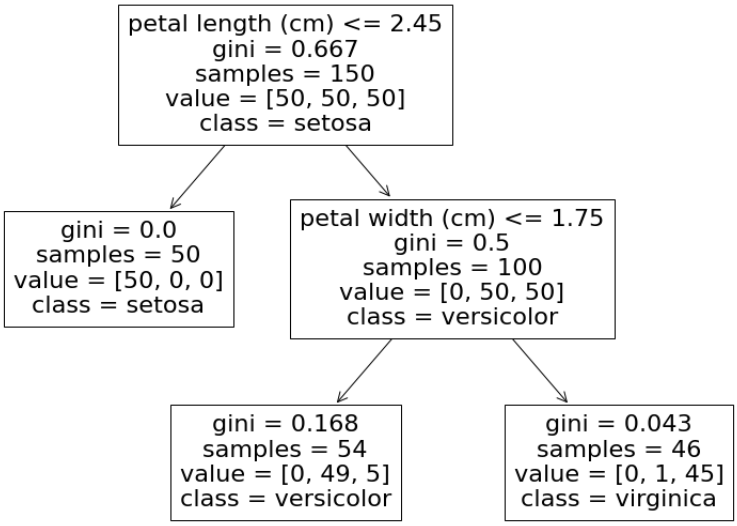


FIGURE (2.3) Representation of a classification tree trained on the Iris dataset

2.5 Contribution

One of the challenges of data mining is to exploit data distributed over different sites. Here, we propose to use collaborative learning to try to address this problem. This method can handle multi-source data, i.e., multiple sets that represent the same individuals in different attribute spaces. In other words, each database is a view

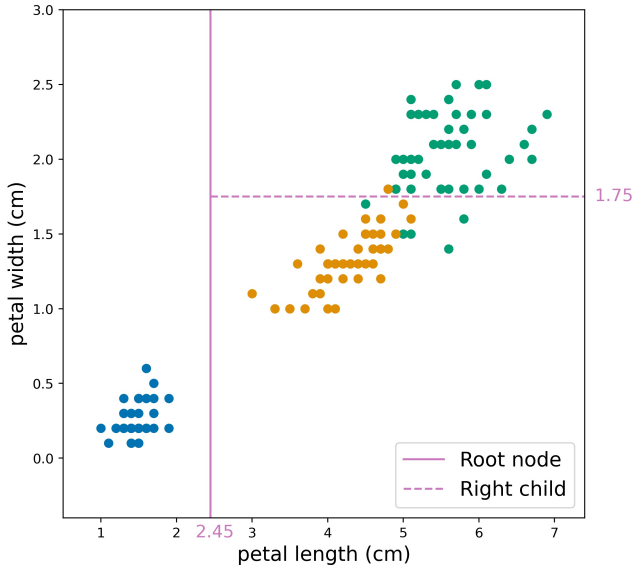


FIGURE (2.4) Partition of the Iris dataset obtained from the classification tree represented in figure 2.3

of a global set about the same individuals. In this section, we describe the general framework in which our method is applied and explain the method.

2.5.1 Configuration

The goal of collaborative learning is to learn from local data and a set of learners whose data is stored at different sites.

More formally, let the global data set X be distributed among P sites: $X^{[1]}, \ldots, X^{[p]}, \ldots, X^{[P]}$, where $X^{[p]} = \{x_i^{[p]}\}_{i=1}^{N}$ is a set of N objects and each object $x_i^{[p]}\mathbb{R}^d$ is characterized by d variables (i.e., it is a vector in a d dimensional feature space).

Each data site has access to different features $d^{[p]}$ describing the same individual samples of the overall data set. At each data site, we have a classification algorithm $\mathcal{A}^{[p]}$, $p = 1, \ldots, P$, and each algorithm will attempt to propose a classification of its data $X^{[p]}$. The process has two steps: a local step and a collaborative step.

2.5.2 Local step

During the local step, each $\mathcal{A}^{[p]}$ model is trained using its local dataset $X^{[p]}$ and adjusts its parameters as it would in a non-collaborative setting. Here, the models in question are decision trees. We therefore recursively partition the dataset by choosing at each iteration a variable and a threshold to split the data into two groups, as in figure 2.3. This variable and this threshold are chosen so as to minimize the Gini criterion:

$$G = 1 - \sum_k p_k^2 \tag{2.5}$$

Where p_j is the proportion of individuals in a node that belonging to class j.

If the average Gini index of the possible child nodes is higher than the Gini index of the considered node, then we stop partitioning the data and this last node becomes a leaf. The prediction of a leaf is the proportion of individuals belonging to each class on this leaf.

Our method requires that each algorithm produce a responsibility matrix $R^{[p]}$. This matrix contains the contributions of each component to each observation, $R_{i,k}^{[p]} = \mathbb{P}(Z_i = C|X_i, \theta)$, where C is the number of classes, $Z_i \in \{1, \ldots, C\}$ are the components of the model, and the elements of θ are the parameters of the distribution.

Once each model has been trained, we want them to exchange information to improve their performance. This is done in the collaboration step.

2.5.3 Collaboration step

In the collaborative steps, the different algorithms $\mathcal{A}^{[p]}$ will exchange information in an attempt to improve their respective classifications. During the learning step, in addition to $X^{[p]}$, the algorithms also receive additional information, $X^{[p]^*}$. This

additional information is presented in the form of partition matrices $R^{[p]}$. The local learner thus has access to $X^{[p]}$ and $X^{[p]^*}$, instead of just $X^{[p]}$, to perform its task.

In our case, for each algorithm $\mathcal{A}^{[p]}$, $X^{[p]^*} \equiv R^{-[p]}$, where:

$$R^{-[p]} = \{R^{[q]} : q \in \{1, \ldots, P\} \setminus \{p\}\} \tag{2.6}$$

The set $R^{-[p]}$ contains all the partition matrices of site p's counterparts. In the following, we show how $X^{[p]}$ and $X^{[p]^*}$ can be used to try to improve the prediction performance.

Algorithm 1: CoDT algorithm

Local step:
for *All algorithms* $\mathcal{T}^{[p]}$ **do**
    train $\mathcal{T}^{[p]}$ on the data $X^{[p]}$
    get local parameters $\theta^{[p]}$ and partition matrix $R^{[p]}$
    `// the array` $\mathcal{B}$ `saves the historical best state for each tree, along with its score`
    $\mathcal{B}^{[p]} \leftarrow \left(\mathcal{T}^{[p]}, \text{Quality}(R^{[p]})\right)$
Collaborative step:
for *All algorithms* $\mathcal{T}^{[p]}$ **do**
    $R^{[p]}(t+1) \leftarrow f\left(R^{[p]}(t), R^{-[p]}(t)\right)$
    train model starting from new partition $R^{[p]}(t+1)$ using data $X^{[p]}$
    get $\theta^{[p]}(t+1)$ and $R^{[p]}(t+1)$
    if *Quality* $\left(R^{[p]}(t+1)\right)$ *is better than* $\left(R^{[p]}(t)\right)$ **then**
        $\mathcal{B}^{[p]} \leftarrow \left(\mathcal{T}^{[p]}, \text{Quality}(R^{[p]}(t+1))\right)$

If the score increases and there is no overfitting problem, we then agree to the collaboration. On the other hand if it decreases, we dismiss the collaboration and use the initial results. When accepting the collaboration, a learner updates its internal state and improves its quality criterion; when a learner's quality is compromised by the collaboration, it can decide to skip this collaboration iteration and keep its state

unchanged. This does not mean that it cannot improve later in the process. Indeed, the system as a whole might keep evolving, leading to potentially fruitful collaboration in the following iterations.

2.5.4 Fine-tuning step

As mentioned above, each iteration of the collaboration step is followed by an update of the model parameters. In this case, we use decision trees as collaboration models. These models are defined by a variable and a critical threshold at each node. For the update of the parameters, we only adjust the thresholds according to the remote information received. Algorithm 2 shows how we adjusted the thresholds for each node in the tree using the collaboration results.

Algorithm 2: Update trees after collaboration

Input: $\{T_i : i = 1, \ldots, P\}$: Trees after collaboration $(T_i = (N_i))$;
$(N_i = Dictionary\ \{\ \mathcal{A},\ \delta\ \})$
Labelled data by decisions after collaboration

Output: $\{\tilde{T}_i : i = 1, \ldots, P\}$ tuned trees, $\tilde{T}_i = (\tilde{N}_i)$

for *i=1,...,P* **do**
  for $j = 1, \ldots, |X|$ **do**
    $dp(x_j) = N_i(x_j)$ `// decision path`
    $G(x, \delta) = e^{\frac{-(x-\delta)^2}{\sigma^2}}$
    for $k = 1, \ldots, |dp(x_j)|$ **do**
      if x_j *well classified AND* $\mathcal{A}_k(x_j) <= \delta_{x_j}$
      OR x_j *misclassified AND* $\mathcal{A}_k(x_j) > \delta_{x_j}$ **then**
        $\delta_{x_j}(t+1) \leftarrow \delta_{x_j}(t) + \epsilon \cdot d\left(td_{loc}(x_j), td_{coll}(x_j)\right) \cdot G(x_j, \delta_j)$
      else
        $\delta_{x_j}(t+1) \leftarrow \delta_{x_j}(t) - \epsilon \cdot d(td_{loc}(x_j), td_{coll}(x_j)) \cdot G(x_j, \delta_j)$

For each individual, we compare:

- the classification obtained by the tree after the local step,
- and the one obtained after the collaboration.

If they match, then we want to strengthen that decision. If they don't, we want to weaken it. To reinforce (resp. weaken) a decision, we look at the path of the individual in the tree and note all the nodes he passed through. Then, for each of these nodes, we increase (resp. decrease) the difference between the threshold and the value of the tested variable (by changing the threshold). To do this, we proceed as follows: for each node we test if the example in this node is well or badly classified, if it is well classified and the example associated with this node is less than or equal to the threshold of this same node then we modify the value of the threshold by adding the product of the Jensen - Shannon divergence between the local probability distribution of the example and its probability distribution after the collaboration, a learning rate ϵ and a function inversely depending on the distance between the value of the example and the threshold (for example a Gaussian kernel). This is done for all the individuals. We then update the other parameters of the tree (class proportion on each node), taking into account the new values of the threshold.

2.6 Experimental validation

In this section, we present the results obtained after running our algorithm on several datasets. The datasets used are described in Appendix A.

2.6.1 Results and analysis

The collaborative learning algorithm has been performed on a variety of datasets and the criterion for accepting the collaboration was accuracy improvement. Accuracy is defined as follows:

$$\text{precision} = \frac{1}{n} \sum_{k=1}^{C} \text{precision}_k \tag{2.7}$$

TABLE (2.1) Performance of random forests before and after collaboration. The score used is the accuracy.

	Random Forest	Collaborative RF
Adult	.785	.800
Amazon	.943	.943
Click prediciton	.848	.855
KDD Appetency	.901	.906
KDD Churn	.819	.825
KDD upselling	.840	.852
KDD 98	.884	.893
Kick prediction	.838	.850
Madelon	.663	.678
Spambase	.856	.856
Waveform	.696	.726
Wdbc	.934	.951

Where

$$\text{precision}_k = \frac{\text{\# individuals correctly assigned to class k}}{\text{\# individuals assigned to class k}} \tag{2.8}$$

We conducted experiments on several real-world datasets. All datasets were randomly shuffled and divided into several subsets (views) with the same number of features. Note that some datasets contain uninformative features. For example, the Waveform dataset has 19 attributes that are all noise attributes, with mean 0 and variance 1. During the local step, classification trees has been trained using the attributes of each view. Next, we proceeded to the collaboration step using the collaboration framework described in the previous chapter, with the goal of improving each local algorithm by exchanging information based on the prediction matrices found by the learners.

Finally, we measured the accuracy of each algorithm before and after the collaboration. We reported these results in table 2.1. In these experiments, the classification trees have been able to achieve better results after collaboration.

2.7 Comparison with the catboost benchmark

In this section, we compare the performance of our method with different gradient boosting methods on binary classification tasks.

We use the experimental setup used by the Catboost developers [52] to compare their framework with popular gradient boosting libraries: LightGBM, XGBoost, H2O. They implement two variations of each algorithm:

- **Default**: each of the algorithms has a set of specific parameters, for which the default values are proposed by their authors. The authors of CatBoost use these values and only decide on the number of trees to train on each dataset.
- **Tuned**: The hyperparameters are optimized using the Tree-structured Parzen Estimator [6].

The authors use cross-validation for model selection. The datasets used in the experiments are described in Appendix A.

2.7.1 Preprocessing

Since the objective of the study is to compare the performance of the algorithms themselves, we do not perform any complex preprocessing (elimination of unbalanced classes, feature selection, etc.).

- For categorical variables, missing values are replaced by a special value, i.e. we treat missing values as a modality in their own right.
- For numeric variables, missing values are replaced with zeros, and a dummy binary feature is added to indicate each individual involved.

Each dataset was divided into

- a training set (X_{train}, y_{train}) (60%)

- a test set (X_{test}, y_{test}) (20%)
- and a validation set (X_{val}, y_{val}) (20%)

2.7.2 Benchmark

In this section, we present experimental results using the logloss metric for all algorithms and compare them with our method.

In table 2.2, we compare the performance of collaborative classification trees to gradient boosting algorithms. The results were measured by the logloss and the percentage is the metric difference measured against the CatBoost results. We can see that, as with all other approaches, the logloss decrease after the collaborative step and the adjustment of the algorithms' parameters.

Collaborative classification trees have shown comparatively poor performance compared to Gradient Boosting algorithms, when measured in absolute terms. This can be explained by several factors. The first one is that CatBoost is designed for the classification of data represented by categorical features (whence its name). The datasets on which it is tested are mainly made up of categorical data. But our architecture is not well suited for categorical data. Indeed, it is based on small adjusments of the thresholds to try to fit the data distribution. But with e.g. One-hot encoded categorical data, if the threshold is between 0 and 1, making marginal adjustment to it would not change the classification of the observation unless it reaches 0 or 1, and it that case, all observations would fall on the same side of the threshold. In other words, with categorical features, we lack the ability to make small adjustments to the model.
Two other reasons why there is such a performance gap are the number of base models and the data available to each learner. In the proposed collaborative architecture, only a few models are involved in the learning process. While in CatBoost, for example, the default number of base models is 5000. And in the collaborative framework, each

base model has access only to a subset of features, whereas in the gradient boosting algorithms, each learner is proveded with the entire set of features.

Nevertheless, performance between the two steps in the collaboration process has been consistently improved. This indicates an increased ability to generalize.

TABLE (2.2) Comparison with Gradient Boosting algorithms. The index used is the *logloss*. The percentages correspond to the rate of change with respect to the best performing model.

	CatBoost		**LightGBM**		**Our method**	
	Tuned	Default	Tuned	Default	Default	Tuned
Adult	**0.270**	0.273 1.21%	0.276 2.33%	0.287 6.46%	0.387 43.33%	0.312 15.56%
Amazon	**0.138**	0.138 0.29%	0.164 18.80%	0.167 21.38%	.248 79.74%	.217 57.65%
Click prediction	**0.391**	0.391 0.06%	0.396 1.39%	0.397 1.69%	0.516 32.01%	0.460 17.74%
KDD appetency	0.072	**0.071** **-0.19%**	0.072 0.40%	0.075 4.63%	.104 45.56%	.085 18.76%
KDD churn	**0.231**	0.232 0.28%	0.232 0.33%	0.236 1.89%	0.343 48.25%	0.273 18.01%
KDD upselling	**0.166**	0.167 0.37%	0.167 0.42%	0.171 2.98%	0.298 79.33%	0.269 61.82%
KDD 98	**0.195**	0.195 0.07%	0.196 0.56%	0.198 1.91%	0.228 16.95%	0.202 3.84%
Kick prediction	**0.285**	0.285 0.05%	0.296 3.82%	0.299 4.91%	0.361 27.05%	0.352 23.55%

TABLE (2.3) Comparison with Gradient Boosting algorithms (continued).

	XGBoost		H2O		Our method	
	Tuned	Default	Tuned	Default	Default	Tuned
Adult	0.275 2.11%	0.280 3.84%	0.275 1.99%	0.276 2.35%	0.387 43.33%	0.312 15.56%
Amazon	0.163 18.56%	0.165 20.07%	0.163 18.10%	0.170 23.08%	.248 79.74%	.217 57.65%
Click prediction	0.396 1.37%	0.398 1.73%	0.398 1.72%	0.398 1.78%	0.516 32.01%	0.460 17.74%
KDD appetency	0.072 0.35%	0.075 4.41%	0.072 1.33%	0.074 2.86%	.104 45.56%	.085 18.76%
KDD churn	0.233 0.80%	0.234 1.04%	0.233 0.64%	0.233 0.69%	0.343 48.25%	0.273 18.01%
KDD upselling	0.166 0.12%	0.169 1.57%	0.168 1.28%	0.170 2.22%	0.298 79.33%	0.269 61.82%
KDD 98	0.196 0.52%	0.198 1.69%	0.195 0.37%	0.196 0.72%	0.228 16.95%	0.202 3.84%
Kick prediction	0.295 3.47%	0.298 4.70%	0.295 3.52%	0.296 4.06%	0.361 27.05%	0.352 23.55%

2.8 Conclusion

In this chapter, we have presented a new collaborative learning architecture. This architecture has several advantages.

It brings a reasonable answer to the question of the choice of the collaborator. Indeed, in most collaborative learning architectures, collaboration only takes place between two sites at a time. The selection of the appropriate collaborator is a complex problem to which it is difficult to provide an objective answer, especially in an unsupervised learning framework. We will see in the next chapter that some tracks exist however. In the architecture we propose, the collaboration is done simultaneously with all the remote sites. Note that this also increases the speed of information propagation between learners.

The incoming information is weighted at the level of each learner. This weighting method allows to simultaneously keep the most reliable predictions and to adapt the

other predictions. To do this, we use entropy as a measure of uncertainty.

Finally, we implemented this algorithm with decision trees as a base model. We used data sets intended for classification tasks, and observed an improvement in the performance of the algorithms between the local step and the end of the collaboration. These performances are nevertheless well below the state of the art of gradient boosting. This is explained by the task at hand and the number of trained decision trees.

3 Unsupervised collaborative learning

3.1 Introduction

Clustering is a common data analysis task. Its objective is to partition observations into homogeneous (low intra-group variability) and distinct (high inter-group variability) groups. There is a large variety of clustering algorithms and most of them require that all data be available simultaneously on the same site [11, 34]. However, technical, legal, confidentiality or performance reasons may prevent the gathering of data or the exchange of sensitive information.

In the collaborative clustering framework, we have a distributed dataset on different nodes of a network. The objective is to obtain on each node a partition of the local data by using the results obtained on the remote nodes, without exchanging the data themselves [15, 50]. This operation is often performed in two steps. First, the local step, during which a clustering algorithm is applied locally and independently on each site. Then the collaborative step, during which sites exchange knowledge in order to try to improve their own results.

3.2 Related work

Collaborative clustering research was introduced by Pedrycz in 2002 [50]. The author proposed a collaborative version of the Fuzzy C-Means algorithm [7]. The objective

function of the Fuzzy C-Means algorithm is extended with a second term that penalizes the discrepancy between the partitions.

The SAMARAH method can handle heterogeneous "hard" clustering algorithms [72]. This method aims at increasing the similarity between clusterings by evaluating and resolving conflicts. In order to compare partitions with possibly different numbers of clusters, a confusion matrix is computed between each pair of algorithms. A consensus is then reached by applying a voting algorithm to these results.

In 2015, Sublime *et al.* proposed an approach lifting some limitations of previous work: they introduce a framework for collaboration with heterogeneous algorithms. This method has the advantage of not requiring a global confidence coefficient, so the impact of a remote algorithm on local results can be different for each observation. However, using the hard clustering results to construct the confusion matrices results in some loss of information. Moreover, each algorithm has the same weight in this process. As a result, since no difference is made between a data site with well-separated clusters, and another with overlapping clusters, this method leaves the possibility for poorly performing algorithms to hamper the results of others.

After proposing collaborative versions of Self Organizing Maps (SOM) and Generative Topographic Mapping (GTM), Ghassany *et al.* combined the Fuzzy C-Means and GTM algorithms to obtain a collaborative clustering algorithm [29].

In [61] the authors describe a collaborative framework for model-based clustering algorithms. They define a global likelihood function and attempt to optimize a proxy for this function. This process requires setting a weight parameter between local and external information.

More recently, an automated method for optimizing trust in exchanges was proposed by Sublime *et al.*, and the collaboration involved four different views. The

authors achieved good results in detecting noisy views, but the method tends to favor collaboration between very similar views.

3.2.1 Performance metrics in clustering

The task of clustering itself is an ill-defined problem. We have discussed above the objective of clustering, namely to partition a set of elements into homogeneous and distinct groups. In other words, we want similar elements to share the same cluster and dissimilar elements to be in different clusters. But "being in the same cluster" is an equivalence relation and "being similar" is not [15]. So, if the clusters are not well separated, we will have to find a compromise. Moreover, different numbers of clusters can be considered as this is somehow subjective and it depends on the model chosen and on its parameters [64]. Under these conditions it is possible to imagine several equally acceptable solutions. Hence, any index measuring the quality of a partition will favor certain criteria.

There are many indices for measuring clustering quality [5, 18]. Traditionally, a distinction is made between internal and external indices. Internal validation indices use the data and the obtained partition to compute a score that will depend on the compactness and the separation of the clusters. The external validation indices are used to compare the adequacy between the obtained clustering and an external reference. We give here the most common internal and external validation indices. In what follows, we assume that the data are represented by a matrix X:

$$X = \begin{pmatrix} x_1^1 & \cdots & x_1^d \\ \vdots & & \vdots \\ x_N^1 & \cdots & x_N^d \end{pmatrix} \tag{3.1}$$

where d is the number of features, and N the number of individuals. We denote by

C_k cluster number k, with $k = 1, \cdots, K$. We denote by I_k the set of indices i such that the observation x_i is in cluster k, and $n_k = |I_k|$ the number of points assigned to cluster C_k.

$$I_k = \{i \mid x_i \in C_k\}$$

Internal validation indices

The Davies-Bouldin Index Let δ_k denote the average distance of the points belonging to the cluster C_k to the centroid G^k of this cluster:

$$\delta_k = \frac{1}{n_k} \sum_{i \in I_k} d(x_i, G^k) \tag{3.2}$$

Then δ_k is a measure of the compactness (or cohesion) of the cluster k. So we have $\delta_k \in [0, +\infty)$ and in general, the smaller the value of δ_k the better the clustering.

Let $\Delta_{kk'}$ denote the distance between the barycenters G_k and $G_{k'}$ of clusters k and k':

$$\Delta_{kk'} = d(G^k, G^{k'}) \tag{3.3}$$

Then $\Delta_{kk'}$ tells us how well the clusters are separated. In general, we want this indicator to be high.

For each cluster k, we compute the maximum of the quotient $\frac{\delta_k + \delta_{k'}}{\Delta_{kk'}}$ for $k \neq k'$. The Davies-Bouldin score S_{DB} is then given by:

$$S_{DB} = \frac{1}{K} \sum_{k=1}^{K} \max_{k' \neq k} \left(\frac{\delta_k + \delta_{k'}}{\Delta_{kk'}} \right) \tag{3.4}$$

The Davies-Bouldin index is therefore positive and the lower its value, the better the clustering.

The Silhouette Index calculates a "silhouette" coefficient for each point. The average of these coefficients gives the silhouette index of the clustering. Like the Davies-Bouldin index, the silhouette index realizes a comparison between the cohesion of clusters and their separation. The cohesion $a(i)$ for the individual i is given by the following equation:

$$a(i) = \frac{1}{|I_k| - 1} \sum_{j \in I_k} d(x_i, x_j) \tag{3.5}$$

It is the average distance of the observation x_i to the observations of the same cluster.

The separation is given by the following equation:

$$b(i) = \min_{k' \neq k} \frac{1}{|I_{k'}|} \sum_{i' \in I_{k'}} d(x_i, x_{i'}) \tag{3.6}$$

It is the average distance between the observation x_i and the individuals belonging to the closest cluster.

The *silhouette coefficient* $s(i)$ for individual i is given by:

$$s(i) = \frac{a(i) - b(i)}{\max(a(i), b(i))} \tag{3.7}$$

And the silhouette index by:

$$S_{sil} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|I_k|} \sum_{i \in I_k} s(i) \tag{3.8}$$

The silhouette index is between -1 and 1 and a higher value indicates a better clustering.

External validation indices

The Rand index measures the similarity between a partition U and a reference partition V. Each pair of elements is assigned to one of the following four categories:

- TP: the elements are assigned to the same clusters in U and in V.
- TN: the elements are assigned to different clusters both in U and V.
- FP: the elements are grouped in U and separated in V.
- FN: the elements are separated in U and grouped in V.

Then the rand index is given by :

$$S_{rand} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.9}$$

Mutual information is another way to measure the similarity between two partitions [67]. Let $p_U(k) = \frac{|Ik|}{N}$ the probability that an element chosen uniformly at random from partition U belongs to the cluster k, and $p_{UV}(k,k') = \frac{|C^k \cap C^{k'}|}{N}$ the probability that it belongs to the clusters k and k' respectively in the partitions U and V.

Then the mutual information $\mathbb{I}(U,V)$ is defined by :

$$\mathbb{I}(U,V) = \sum_{k=1}^{K_1} \sum_{k'=1}^{K_2} p_{UV}(k,k') \log \frac{p_{UV}(k,k')}{p_U(k) p_V(k')} \tag{3.10}$$

It lies between 0 and $\min(\mathbb{H}(U), \mathbb{H}(V))$, where $mathbbH(U)$ denotes the entropy induced by the U partition. This measure of similarity is sensitive to the number of clusters, it is possible to increase it by adding clusters. To compensate for this, we

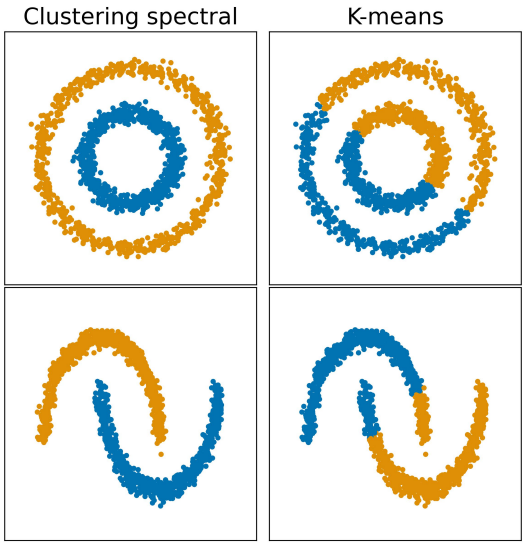


FIGURE (3.1) Examples of non-convex clusters. For such clusters, the most common validation indices are not adequate. For concentric circles, the clusters' centroids are very close (weak separation). For interleaving half circles, the observations are relatively far from the cluster centroid. Even worse, the centroid of a cluster is almost in the other cluster.

rather use its normalized version, between 0 and 1 :

$$\mathrm{NMI}(U,V) = \frac{2 \cdot \mathbb{I}(U,V)}{\mathbb{H}(U) + \mathbb{H}(V)} \tag{3.11}$$

The biases induced by these indices

Internal validation indices are often based on an assumption made about the data distribution. This is in particular the case for the Davies-Bouldin indices and the silhouette index that we discussed above. When describing these indices, we said that we want clusters with a low variance (the elements of the cluster are close to the barycenter) and a high separation (the barycenters are far away). However, as we see in the figure 3.1, natural clusters sometimes do not respect these properties. An option in this case is to use a validation index based on the density [46].

Choosing a validation index

In the collaboration step of our proposed algorithm, models can accept or reject collaboration depending on how it affects performance. In the case of clustering, this performance can be measured by a validation index. As we have seen, each index has its own biases and it is possible to choose different indices for each model. In our implementation, we have chosen the Davies-Bouldin index because it is often used in the literature. We could have chosen another internal validation index.

Note that an internal validation index would make less sense here. Indeed, this kind of index is used to measure the similarity between two partitions. However, if we want to take advantage of remote information to improve the local model, we do not want to approach the remote partitions, let alone a consensus. External validation indices can still be useful in collaborative clustering. In particular when we want to study the effect of the similarity of the initial partitions on the collaborative process. This work has been done in [54]. The authors of this paper are interested in the impact of the quality and diversity of learners on collaborative clustering. They empirically show that the best way to predict whether collaboration will be beneficial or not for a given site is to know its initial performance and the performance of the site with which it collaborates. This performance is measured by the purity of score [1]. The authors show that collaboration with a remote model having a higher purity is likely to improve the local purity. Conversely, collaboration with a remote model having a lower purity index is likely to deteriorate the local purity. In their experiments, Self Organizing Maps (SOMs) [36] have been used in both horizontal and vertical collaboration. Their results are reported in the tables 3.1 and 3.2.

[1] The purity index is in fact an external validation index, like the ones presented above. But with the difference that the reference partition is made of the "true classes" of observations. In other words, we need a labeled dataset (X, Y) where Y_i is the label associated to X_i. Suppose that the dataset is composed of K' *true* classes. Then the purity index is:

$$S_{pur} = \frac{1}{N} \sum_{k=1}^{K} \max_{1 \leq k' \leq K'} |C^k \cap C^{k'}| \tag{3.12}$$

Dataset	Map	Purity
Waveform	SOM_1	86.54
	SOM_2	39.5
	$SOM_{1\to2}$	73.24
	$SOM_{2\to1}$	58.34
Isolet	SOM_1	80.79
	SOM_2	93.27
	$SOM_{1\to2}$	81.46
	$SOM_{2\to1}$	92.87
Wdbc	SOM_1	94.02
	SOM_2	96.49
	$SOM_{1\to2}$	95.23
	$SOM_{2\to1}$	96.57
SpamBase	SOM_1	80.57
	SOM_2	84.95
	$SOM_{1\to2}$	82.84
	$SOM_{2\to1}$	83.79

TABLE (3.1) Experimental results of [54] for vertical collaboration. $SOM_{1\to2}$ denotes a collaboration in which the 1 model receives information from the 2 model.

Dataset	Map	Purity
Waveform	SOM_1	69.71
	SOM_2	69.87
	$SOM_{1\to2}$	74.57
	$SOM_{2\to1}$	70.71
Isolet	SOM_1	98.85
	SOM_2	98.46
	$SOM_{1\to2}$	79.54
	$SOM_{2\to1}$	98.30
Wdbc	SOM_1	96.71
	SOM_2	97.87
	$SOM_{1\to2}$	96.99
	$SOM_{2\to1}$	97.49
SpamBase	SOM_1	76.26
	SOM_2	70.43
	$SOM_{1\to2}$	72.28
	$SOM_{2\to1}$	69.78

TABLE (3.2) Experimental results of [54] for horizontal collaboration.

The calculation of the purity can be a good indicator of the performance of a clustering model on a dataset (provided that the purity index is not an objective function, *cf.* Goodhart's law). However, it is rarely possible to compute it in practice because very often in clustering we do not have labeled data.

The authors therefore choose to use the normalized Rand index as an indicator of diversity between partitions. They test the relation between the diversity of the collaborators and the evolution of the performance after collaboration (always measured by the purity index). They find that for models with a relatively high initial performance, collaboration with a model obtaining a similar score leads to little change in performance after collaboration. While high diversity often leads to performance degradation. An intermediate level of similarity is more likely to result in successful collaboration.

3.3 Theoretical framework and algorithm

In this section, we describe the general framework in which our method is applied, as well as an example of implementation.

3.3.1 General setting

The goal of collaborative learning is to learn from local data and a set of learners whose data is stored at separate sites.

More formally, suppose that the global dataset X is distributed over P sites: $X^{[1]}, \ldots, X^{[p]}, \ldots, X^{[P]}$, where $X^{[p]} = \{x_i^{[p]}\}_{i=1}^{N}$ is a set of N objects and each object $x_i^{[p]} \in \mathbb{R}^d$ is a d dimensional vector.

It is the average proportion of the majority class for each cluster. This index is sensitive to the number of clusters. In particular, it will be maximal (equal to 1) if we form N clusters composed of only one individual. Finally, note that the reference partition to compute this index must be the set of true classes of the data X. However, clustering algorithms are rarely applied to labeled data sets.

Our study focuses on the horizontal collaborative clustering problem. Thus, each site has access to different features $d^{[p]}$ describing the same individuals. On each site, an algorithm $\mathcal{A}^{[p]}$, $p = 1, \ldots, P$ will propose a partition of the local data $X^{[p]}$. The process has two steps: a local step and a collaborative step.

3.3.2 Collaborative process

Local step In the local step, each algorithm $\mathcal{A}^{[p]}$ works on its own dataset $X^{[p]}$ and adjusts its parameters as it would in a non-collaborative setting. In order to exchange their results, the algorithms must have some type of information in common. In our case, the partition computed by each algorithm will be represented as a responsibility matrix $R^{[p]}$. This matrix contains the contributions of each component to each observation, $R^{[p]}_{i,k} = \mathbb{P}(Z_i = k | X_i, \theta)$, where K is the number of clusters, $Z_i \in \{1, \ldots, K\}$ are the components of the model and the elements of θ are its parameters.

Once each model has been trained locally, we want them to exchange these partition matrices with each other to improve their performance. This exchange is done during the collaboration step.

Collaborative step During the collaboration step, the different algorithms $\mathcal{A}^{[p]}$ will exchange information in order to improve their respective classification. The resulting clustering will not only be based on the examples $X^{[p]}$, but also on the additional information $X^{[p]^*}$, which are in fact the remote partition matrices. The local learner thus has access to $X^{[p]}$ and $X^{[p]^*}$, instead of just $X^{[p]}$, to perform its clustering. In our case, for each algorithm $\mathcal{A}^{[p]}$, $X^{[p]^*} \equiv R^{-[p]}$, where:

$$R^{-[p]} = \{R^{[q]} \, : \, q \in \{1, \ldots, P\} \setminus \{p\}\} \tag{3.13}$$

The set $R^{-[p]}$ contains all the remote partition matrices, from the perspective of site p. In this case, how can we use $X^{[p]}$ and $X^{[p]^*}$ to improve each local learner $\mathcal{A}^{[p]}$?

For the sake of simplicity, we consider two learners: $P = 2$. At the local step, they produce two partition matrices: $R^{[1]}(t)$ and $R^{[2]}(t)$. In particular, for the sample x_i, we have $R^{[1]}_{i,\cdot}$ and $R^{[2]}_{i,\cdot}$. Thus, the update rule for learner 1 for sample x_i is:

$$R^{[1]}_{i,\cdot}(t+1) \longleftarrow f\left(R^{[1]}_{i,\cdot}(t), R^{[2]}_{i,\cdot}(t)\right) \tag{3.14}$$

Where f refers to a generic function.

We want this update rule to depend on the level of certainty of site 1 about its own classification. The higher this level of certainty, the smaller the difference between $R^{[1]}_{i,\cdot}(t+1)$ and $R^{[1]}_{i,\cdot}(t)$ should be. Conversely, the lower the level of certainty, the more weight will be given to $R^{[2]}_{i,\cdot}(t)$. Similarly, the level of certainty of site 2 will be taken into account for its classification: the higher (resp. lower) the certainty of site 2, the more (resp. less) it will influence $R^{[1]}_{i,\cdot}(t+1)$.

As mentioned in section 2.2, entropy can be used as a measure of the amount uncertainty associated with a probability distribution.

$$H(X) = -\sum_{i=1}^{d} p(x_i) \log_2 p(x_i) \tag{3.15}$$

Where X is a random variable with possible values $\{x_1, \ldots, x_d\}$. It follows that the uncertainty for a distribution R_i is:

$$H(R_{i,\cdot}) = -\sum_{k=1}^{K} R_{i,k} \log_2 R_{i,k} \tag{3.16}$$

And its normalized version, using the fact that it is positive and its maximum is achieved when X is uniformly distributed:

$$\mathcal{H}(R_{i,\cdot}) = \frac{H(R_{i,\cdot})}{log_2(K)} \tag{3.17}$$

$$\mathcal{H}(R_{i,\cdot}) = \frac{H(R_{i,\cdot})}{log_2(K)} \tag{3.18}$$

Then $0 \leq \mathcal{H}(R_{i,\cdot}) \leq 1$, and

- $\mathcal{H}(R_{i,\cdot}) \approx 0 \Rightarrow R_{i,\cdot} \approx \mathbb{1}_{Z_i=k}$ pour $k \in 1,\ldots,K$, this represents a high level of confidence in the classification of observation i.
- $\mathcal{H}(R_{i,\cdot}) \approx 1 \Rightarrow R_{i,\cdot} \approx \frac{1}{K}\ \forall k \in 1,\ldots,K$, this reflects a high uncertainty on the classification of the observation i.

The equation 3.19 gives the rule for updating $R^{[ii]}$ when P algorithms are involved in the collaboration.

$$R^{[p]}(t+1) \longleftarrow \alpha^{[p]} \cdot R^{[p]}(t) + \sum_{R^{[q]} \in R^{-[p]}} \beta^{[p]}_{[q]} \cdot R^{[q]}(t) \tag{3.19}$$

where

$$\begin{cases} \alpha^{[p]} = \left(\frac{1}{P-1} \sum_{R^{[q]} \in R^{-[p]}} \mathcal{H}\left(R^{[q]}(t)\right) \right) \cdot \left(1 - \mathcal{H}\left(R^{[p]}(t)\right)\right) \\ \\ \beta^{[p]}_{[q]} = \mathcal{H}\left(R^{[p]}(t)\right) \cdot \left(1 - \mathcal{H}\left(R^{[q]}(t)\right)\right) \end{cases} \tag{3.20}$$

In these equations, $\alpha^{[p]}$ is a vector of size N. Each element i of $\alpha^{[p]}$ is a weight associated with the local classification of the i-th observation. This weight depends

negatively on the amount of uncertainty carried by the local classification, and positively on the average uncertainty of the remote classifications.

Likewise, the $\beta_{[q]}^{[p]}$ are vectors of size N. Each element i of $\beta_{[q]}^{[p]}$ is a weight associated with the classification of the i-th observation by the $\mathcal{A}^{[q]}$ algorithm. This weight depends positively on the amount of uncertainty carried by the local classification, and negatively on the uncertainty of the remote classification.

Note that the sites only share their partition matrices. Therefore, regardless of the underlying algorithms, as long as they search for the same number of components, the collaboration algorithm remains relevant. However, the models must agree on the identity of the clusters. Our implementation uses the Hungarian algorithm to rearrange the clusters at each site [38]. In the remainder of this section, we describe the CoLUPI algorithm, and then show how the values of α and β can be used to visualize the flow of information in the collaboration process.

Collaborative learning algorithm Based on the theoretical formalism presented in the previous section, we can design an architecture to collaboratively train multiple models on a distributed dataset. This algorithm uses equation 3.19 to update the parameters of the sites in interaction.

3.3.3 Visualization of the collaboration process

In this section, we focus on the visualization of information flow during collaboration.

As mentioned above, each contributor assigns a weight to each source site (including himself), for each observation. The average of these weights for each remote site gives us the average weight assigned to that site and can be considered as a confidence coefficient, although it is not uniform across observations. If there are P sites, this

Algorithm 3: CoLUPI algorithm

Local step:
for all algorithms $\mathcal{A}^{[p]}$ **do**
  train $\mathcal{A}^{[p]}$ on the data $X^{[p]}$
  get local parameters $\theta^{[p]}$ and partition matrix $R^{[p]}$
end for
Collaborative step:
repeat
  for all algorithms $\mathcal{A}^{[p]}$ **do**
    $R^{[p]}(t+1) \leftarrow f\left(R^{[p]}(t), R^{-[p]}(t)\right)$
    train model starting from new partition $R^{[p]}(t+1)$ using data $X^{[p]}$
    get $\theta^{[p]}(t+1)$ and $R^{[p]}(t+1)$
    if quality$\left(R^{[p]}(t+1)\right)$ is better than quality$\left(R^{[p]}(t)\right)$ **then**
      Accept collaboration
    else
      $\theta^{[p]}(t+1) \leftarrow \theta^{[p]}(t)$
      $R^{[p]}(t+1) \leftarrow R^{[p]}(t)$
    end if
  end for
until no algorithm improves for its criterion

gives us a P matrix as defined in equation 3.21 which can be considered as a confidence matrix. We represent this matrix as a heatmap for the Wdbc dataset in section 3.5.

$$C_{(P\times P)} = \begin{pmatrix} \overline{\alpha^{[1]}} & \overline{\beta^{[1]}_{[2]}} & \cdots & \overline{\beta^{[1]}_{[P]}} \\ \overline{\beta^{[2]}_{[1]}} & \overline{\alpha^{[2]}} & \cdots & \overline{\beta^{[2]}_{[P]}} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\beta^{[P]}_{[1]}} & \overline{\beta^{[P]}_{[2]}} & \cdots & \overline{\alpha^{[P]}} \end{pmatrix} \tag{3.21}$$

Where

$$\overline{\alpha^{[p]}} = \frac{1}{N}\sum_{i=1}^{N} \alpha_i^{[p]}, \tag{3.22}$$

and

$$\overline{\beta^{[p]}_{[q]}} = \frac{1}{N}\sum_{i=1}^{N} \beta_i{}^{[p]}_{[q]}. \tag{3.23}$$

3.4 Implementation examples

The architecture we have presented is model-agnostic. The only condition is that the models provide a probability distribution of belonging to the different classes, in order to be able to use entropy as a criterion for collaboration. We present here two models that can be used as local models for collaboration. First, Gaussian mixture models because they are widely used. And the Generative Topography Mapping (GTM) [8,9]. It is the probabilistic counterpart of SOM maps [36].

3.4.1 Gaussian mixture models

In general, when we use a mixture model, we assume that the distribution of our data is in fact a combination of several underlying distributions. Each observation $\mathbf{x}_i, i \in \{1,\dots,N\}$ in the dataset comes from one of K basis distributions $p_k,\ k \in \{1,\dots,K\}$. The probability that the observation i was generated by the component k is $p(z_i = k) = \pi_k$. We write:

$$p(\mathbf{x}_i|\theta) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}_i|\theta) \tag{3.24}$$

Where θ is the set of parameters of the model, and the prior probabilities of the components π_k satisfy $\sum_{k=1}^{K} \pi_k = 1$.

A mixture model whose components are all Gaussian vectors is called a Gaussian mixture model. In this case :

$$p(\mathbf{x}_i|z_i = k, \theta) = \mathcal{N}(\mathbf{x}_i|\mu_k, \Sigma_k) \tag{3.25}$$

Where $\theta = (\theta_k)_{1\leq k\leq K}$ and $\theta_k = (\mu_k, \Sigma_k)$.

Training a Gaussian mixture model is done by looking for the parameters that achieve the maximum likelihood of the model. It can be shown that in the case where the component to which each individual belongs is known (complete-data), then the problem has an analytical solution and is easy to solve. But often only the observed data is known and in practice we use the Expectation Maximisation (EM) algorithm to approach the optimal parameters [16].

E-step

Let $\mathcal{R}_{ik}$ denote the probability that individual i belongs to class k:

$$\mathcal{R}_{ik} = p(z_i = k|\mathbf{x}_i, \theta) \tag{3.26}$$

Then applying Bayes' formula:

$$\begin{aligned} \mathcal{R}_{ik} &= \frac{p(z_i = k) \cdot p(\mathbf{x}_i|z_i = k, \theta^{(t-1)})}{\sum_{k'} p(z_i = k') \cdot p(\mathbf{x}_i|z_i = k', \theta^{(t-1)})} \\ &= \frac{\pi_k \cdot \mathcal{N}(\mathbf{x}_i|\theta_k^{(t-1)})}{\sum_{k'} \pi_{k'} \cdot \mathcal{N}(\mathbf{x}_i|\theta_{k'}^{(t-1)})} \end{aligned} \tag{3.27}$$

M-step

$$\begin{aligned} \pi_k &= \frac{1}{N} \sum_i \mathcal{R}_{ik} \\ \boldsymbol{\mu}_k &= \frac{\sum_i \mathcal{R}_{ik} \mathbf{x}_i}{\mathcal{R}_k} \\ \boldsymbol{\Sigma}_k &= \frac{\sum_i \mathcal{R}_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top}{\mathcal{R}_k} \end{aligned} \tag{3.28}$$

Where $\mathcal{R}_k = \sum_i \mathcal{R}_{ik}$.

3.4.2 Generative Topographic Mapping

Generative Topographic Mapping (GTM) is a probabilistic dimensionality reduction algorithm. Its goal is to learn a low-dimensional manifold that correctly fits the high-dimensional data, often for the purpose of visualization, after projection of the data on the 2-D latent space.

In practice, we fix M basis functions $\phi_j(\mathbf{x})$ and try to fit a $D \times M$ matrix $\mathbf{W}$ that maximizes the likelihood

$$\mathcal{L}(\mathbf{W}, \beta) = \sum_{n=1}^{N} \ln \left\{ \frac{1}{K} \sum_{i=1}^{K} p\left(t_n | \mathbf{x}_i, \mathbf{W}, \beta\right) \right\} \tag{3.29}$$

Where t_n are the data points, $\mathbf{x}_i$ the L-dimensional latent points and β the precision of the underlying Gaussian distribution. The parameters of this model are fitted using the EM algorithm. We give both EM steps:

E-step

$$R_{in}(\mathbf{W}_{\text{old}}, \beta_{\text{old}}) = \frac{p(\mathbf{t}_n | \mathbf{x}_i, \mathbf{W}_{\text{old}}, \beta_{\text{old}})}{\sum_{i'=1}^{K} p(\mathbf{t}_n | \mathbf{x}'_i, \mathbf{W}_{\text{old}}, \beta_{\text{old}})}$$

M-step

$$\mathbf{\Phi}^\top \mathbf{G}_{\text{old}} \mathbf{\Phi} \mathbf{W}_{\text{new}}^\top = \mathbf{\Phi}^\top \mathbf{R}_{\text{old}} \mathbf{T}$$

And

$$\frac{1}{\beta_{\text{new}}} = \frac{1}{ND} \sum_{n=1}^{N} \sum_{i=1}^{K} R_{in}(\mathbf{W}_{\text{old}}, \beta_{\text{old}}) \left\| \mathbf{W}_{\text{new}} \phi(\mathbf{x}_i) - \mathbf{t}_n \right\|^2 .$$

3.5 Experimental validation

In this section, we present the results obtained after running our algorithm on several datasets. First, we describe the datasets that were used in the experiments, and then

we present two types of results: performance evaluation and visualization of the collaboration process.

3.5.1 Datasets

- The Breast Cancer Wisconsin (Wdbc) dataset consists of 569 scanned images of a breast mass. It contains 30 real-valued variables describing the cell nuclei present in each image. Each observation is labeled as benign or malignant.
- The Spambase dataset is composed of 57 attributes describing a collection of 4601 spam and non-spam emails.
- The Battalia3 dataset is an artificial dataset describing 2000 exoplanets generated with 27 numerical attributes.
- The MV2 dataset consists of 2000 entries, each described by 6 features. They were randomly generated from the mixture of a noise and four Gaussian components.
- The Isolet (isolated letters) dataset contains 617 variables describing 7797 voice recordings of individuals who have pronounced the name of each letter of the alphabet.
- The Madelon dataset is an artificial dataset containing 4400 rows composing 32 clusters placed on the vertices of a five dimensional hypercube. Then 15 redundant features and 480 unnecessary features (random probes) were added, for a total of 500 attributes.

We split the datasets to achieve a vertical collaborative clustering framework - that is, each data site has access to different variables on the same set of observations.

3.5.2 Results

Co-LUPI based on GTM models

The Co-LUPI algorithm was applied to each of the 6 datasets mentioned in section 3.5.1 using generative topographic maps (GTM).

The criterion for accepting the collaboration was the improvement of the Davies-Bouldin index. This is an internal index, so no prior knowledge of the data structure is required. A second version of the algorithm, RCo-LUPI (Regulazed Collaborative Learning Using Privileged Information), has been implemented. It includes a new, random initialization of the responsibility matrix at each step, as well as of the collaboration matrix. This technique is often used in unsupervised learning and is intended to reduce the dependence on the initial parameters. The table 3.3 shows that in most cases, RCo-LUPI gave slightly better results than Co-LUPI.

To visualize the dynamics of this process, we can examine the successive confidence matrices of the collaboration step. Figure 3.3 represents such data. The Co-LUPI algorithm was applied to the Wdbc dataset, spread over 18 data sites. It can be seen that not all algorithms benefited from collaboration at every step. In particular, the model learning on data site number 1 did not improve its results until the fifth iteration of the collaboration phase. Furthermore, at iteration number 7, only the second algorithm improved its results. Although this could be interpreted as a sign of an imminent end to the process, three other algorithms benefited from these new findings in the next iteration. Thus, the process ended only after four more iterations.

3.5.3 Comparison with other collaborative approaches

The Co-LUPI and RCo-LUPI algorithms were empirically compared to four recent implementations of collaborative clustering algorithms.

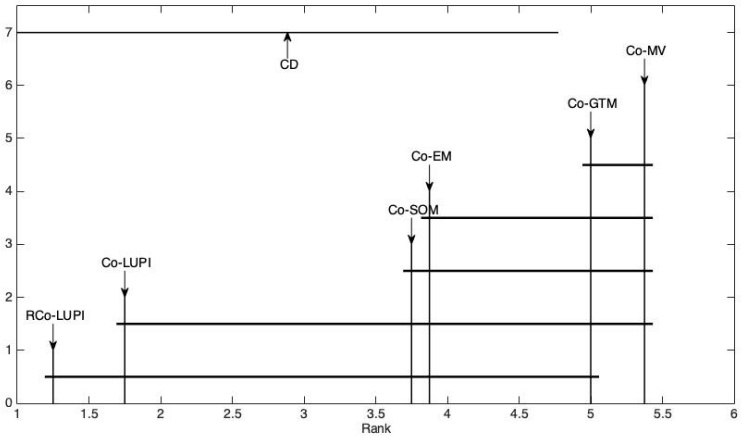
TABLE (3.3) Experimental results, Davies-Bouldin index - Co-EM: [61], Co-MV : [29], Co-GTM: [28], Co-SOM: [31], Co-LUPI, RCo-LUPI

Name	Co-EM	Co-MV	Co-GTM	Co-SOM	Co-LUPI	RCo-LUPI
Wdbc	0.85	0.97	0.9	0.84	0.78	**0.69**
Spam Base	0.94	1.27	0.92	0.87	**0.42**	0.59
Battalia3	2.43	2.83	2.68	2.51	1.47	**1.37**
MV2	1.34	1.34	1.61	1.44	0.86	**0.85**
Isolet	–	–	–	–	1.33	**1.31**
Madelon	–	–	–	–	0.87	**0.82**

To evaluate the performance of our approaches, we use the Friedman test and the Nemenyi test recommended in [17]. First, the algorithms are ranked according to their performance on each dataset. There are then as many rankings as there are datasets.

Next, the Friedman test is performed to test the null hypothesis that all approaches are equivalent, which assumes that their average rankings are equal. If the null hypothesis is rejected, then the Nemenyi test is performed. If the average ranks difference of two approaches exceeds a given threshold (namely the critical difference (CD)), then it can be concluded that their performances are significantly different. In the Friedman test, we set the significance level at $\alpha = 0.05$. Figure 3.2 shows a critical diagram representing the average ranks of the algorithms. The methods are ordered from left (best) to right (worst) and a horizontal line connects groups of algorithms that are not significantly different (for significance level $\alpha = 5\%$). As shown in figure 3.2, Co-LUPI and RCo-LUPI seem to obtain some improvement over the other proposed techniques. But the results are not sufficient to conclude that there is a statistically significant improvement. This can be explained by the small number of datasets and

FIGURE (3.2) Friedman and Nemenyi test to compare multiple approaches on multiple data sets: Approaches are ranked from left (best) to right (worst)



by the fact that the Nemenyi test only considers the performance of the algorithms through their rank, without taking into account the actual value of the performance index.

3.6 Analysis of the collaboration process

In this section, we analyze the impact of the weighting function on the evolution of classifications. The architecture we proposed requires that each learner has a way to weight the information received. We proposed to use entropy as a criterion in order to take into account the uncertainty related to each prediction. We used a simple function that respected our criteria, we presented it in the equations 3.19 and 3.20. However, there are many possible choices. We analyze the collaboration process using the implementation proposed in the latter equations and show an alternative using the Kullback Leibler divergence.

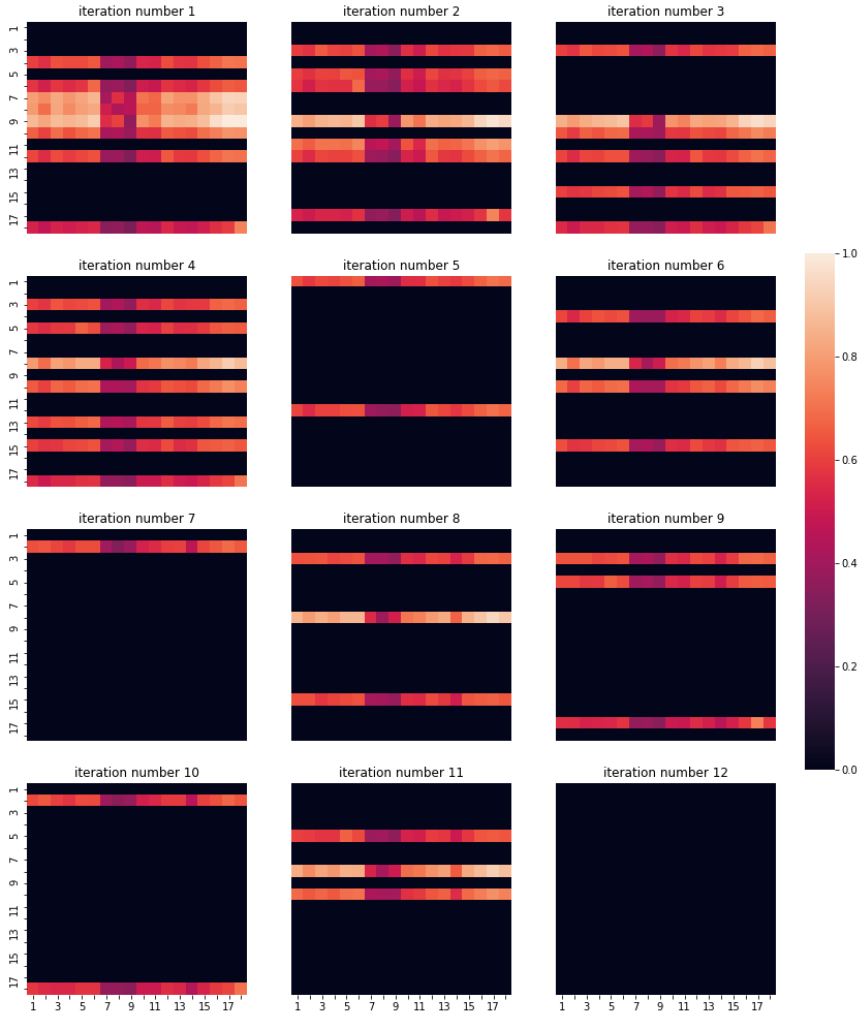


FIGURE (3.3) Information flow during collaboration for the Wdbc dataset. Each row represents a data site and each column the total weight assigned to that source.

3.6.1 Simple use of entropy

In order to analyze the impact of the weighting function, we consider four learners seeking to classify a single individual among three categories. We represent their predictions at each step in the process in Figure 3.4. Step 0 represents the initial predictions of each site for the individual in question. Each bar represents a probability distribution among the three possible classes. The first learner considers the observation to belong to the blue class with high probability. The second learner has a rather uniform distribution. The third learner predicts that the observation belongs to the orange class. Finally, the fourth learner considers equiprobable the green and orange classes.

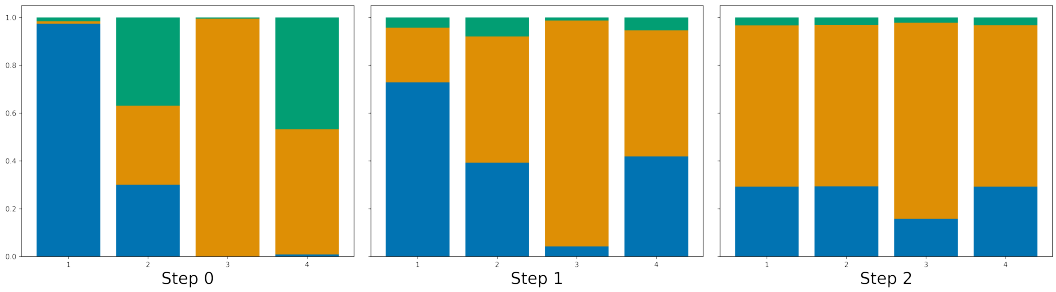


FIGURE (3.4) Evolution of the classification following a collaboration iteration

Step 1 represents learners' predictions after an application of the formula given in equation 3.20 which we report here:

$$R^{[p]}(t+1) \longleftarrow \alpha^{[p]} \cdot R^{[p]}(t) + \sum_{R^{[q]} \in R^{-[p]}} \beta^{[p]}_{[q]} \cdot R^{[q]}(t)$$

$$\begin{cases} \alpha^{[p]} = \left(\frac{1}{P-1} \sum_{R^{[q]} \in R^{-[p]}} \mathcal{H}\left(R^{[q]}(t)\right) \right) \cdot \left(1 - \mathcal{H}\left(R^{[p]}(t)\right)\right) \\ \\ \beta^{[p]}_{[q]} = \mathcal{H}\left(R^{[p]}(t)\right) \cdot \left(1 - \mathcal{H}\left(R^{[q]}(t)\right)\right) \end{cases}$$

Then each model adjusted its predictions according to its initial classification, its entropy, and the distant classifications and entropies. We see that the two models with low entropy at the beginning slightly changed their prediction in step 1. The models with less clear-cut predictions changed their predictions to a greater extent. In particular, they now put more weight on the blue and orange groups.

In step 2, all predictions are similar. Note that we just applied the collaboration formula mentioned above several times, thus ignoring the step of adjusting the model parameters after the predictions are adjusted. In practice, after the state represented in step 1, the models would have started from their new partition matrices to adjust the model parameters, and would have obtained new partition matrices before the next iteration.

Despite the above clarification, this fast convergence behavior is still a problem. In a situation were we are not looking for a consensus, having four quite different predictions achieving one in two steps is troublesome. Indeed, model number 4 excluded the blue group at the beginning. However, as of step 1, it considers it as the most likely group. This behavior seems to go against the objective of collaborative learning, i.e. to privilege a possible local structure. To remedy this, it is possible to penalize too much divergence between local and remote results.

3.6.2 Introduction of a similarity criterion

We used the Kullback-Leibler divergence, which measures the dissimilarity between two distributions. The Kullback-Leibler divergence between two distributions p and q is written as follows:

$$D_{KL}(p||q) = \sum_x p_x \log \frac{p_x}{q_x} \tag{3.30}$$

We therefore want the weight $\beta_{[q]}^{[p]}$ associated with site q to depend negatively on the Kullback Leibler divergence. This means that any proposal that is too far away from

the local structure of the data will be given less weight. We represent a collaboration process using the equation 3.31 to weight the received information.

$$R^{[p]}(t+1) \longleftarrow \alpha^{[p]} \cdot R^{[p]}(t) + \sum_{R^{[q]} \in R^{-[p]}} \frac{\beta^{[p]}_{[q]}}{D_{KL}\left(R^{[p]}(t) || R^{[q]}(t)\right)} \cdot R^{[q]}(t) \quad (3.31)$$

We see in figure 3.5 that models 2 and 4 fit their parameters with a bias towards distant classifications with a low divergence from their own. Model 2, which was rather agnostic to begin with, adjusts its prediction more towards the blue group (which is the majority among the distant classifications). Model 4, on the other hand, is closer to model 2.

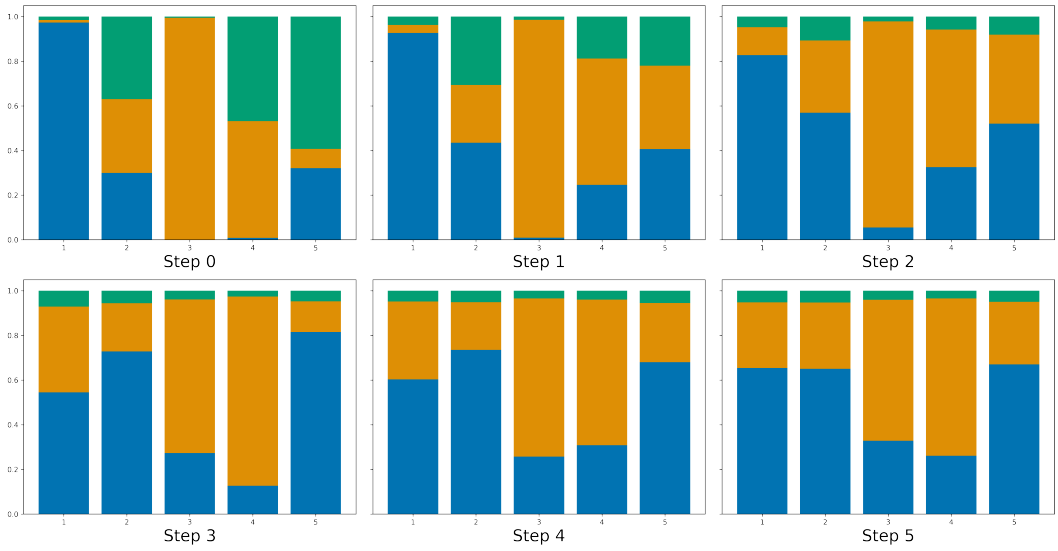


FIGURE (3.5) Evolution of the classification with Kullback Leibler divergence

This focus at the level of the classification of an individual allows us to better understand how the process can be affected by the choice of confidence functions. We then present an analysis of a step of collaboration between Gaussian mixture models.

Name	Input variables	Purity		Davies-Bouldin	
		Local	Collaboration	Local	Collaboration
Iris	[0,1]	.733	.92	.718	1.34
	[2,3]	.947	.92	.563	0.60
Waveform	[0-9]	.774	.810	1.81	1.98
	[10-19]	.806	.811	1.77	1.92
	[20-29]	.341	.780	2.54	37.12
	[30-39]	.347	.785	3.52	32.22
Wdbc	[0,10,20]	.919	.930	1.02	1.13
	[1,11,21]	.627	.868	1.40	2.73
	[2,12,22]	.898	.923	1.10	1.12
	[3,13,23]	.916	.930	1.09	1.11
	[4,14,24]	.627	.866	1.14	2.91
	[5,15,25]	.757	.920	1.13	2.24
	[6,16,26]	.798	.919	1.13	1.79
	[7,17,27]	.682	.902	1.15	1.97
	[8,18,28]	.633	.896	1.18	3.32
	[9,19,29]	.627	.779	1.19	3.76

TABLE (3.4) Experimental results of the Collaborative Gaussian Mixture Models (CoGMM) algorithm on a few data sets. For each site, we display the purity and the Davies-Bouldin index.

3.6.3 Collaboration between Gaussian mixture models

In this section, we present the results obtained after a simple collaboration step without adjustment of the models' parameters. This allows us to have a larger scale view of the changes induced by the simple exchange of information between the models.

We have collaborated Gaussian models on three data sets. We computed before and after collaboration: the purity index on one hand and the Davies-Bouldin index on the other hand. The results are reported in the table 3.4.

We find that the purity index has increased for almost all models. In particular, for the Waveform and Wdbc datasets, all sites have seen their purity index increase. However, the Davies-Bouldin index has increased, which corresponds to a deterioration of the clustering quality. It increases relatively less for the best performing sites after the

local step.

The increase in the Davies-Bouldin index means that models are moving away from a local optimum in the ability to explain the data. And the way in which this increase depends on initial performance may be a sign that poorer performing models are adapting their results more.
The increase in the purity index is a consequence of an exchange of information about the overall structure of the data.

The results for Waveform are rather interesting because sites 3 and 4 have only noise. Their purity scores are due to the presence of three classes. However, it turns out that the other two models manage to improve their performance after the collaboration. In other words, the presence of poor performers did not deteriorate the quality of the other learners.

We have seen in the section 3.3 how we can aggregate the weights given to a remote site to obtain a *confidence coefficient*. The set of these coefficients allows us to represent the information exchanges between learners. We represent these coefficients for the 10 learners trained on the wdbc dataset in the figure 3.6.

We found a correlation between the level of performance of a model and how it weights remote information. The three best performing sites (1, 3, and 4) place little weight on remote sites and thus essentially keep their results intact. In particular, site number 4 places very little weight on remote sites and has high confidence in its own results (*cf.* line 4 of the heatmap 3.6). The other models give more weight to distant results. Finally, all models benefit from collaboration.

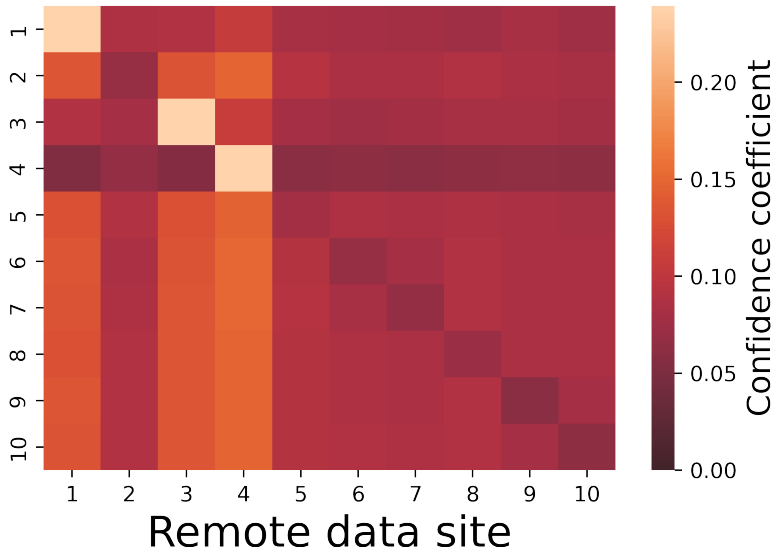


FIGURE (3.6) Confidence coefficients of the different models. Each line represents the confidence given by the model in question to all remote models and to itself.

3.7 Conclusion

We presented Co-LUPI and RCo-LUPI, algorithms based on the use of privileged information and probabilities for collaborative clustering. This allows the algorithms to refine the collaboration based on their - and their remote counterparts' - (in)certainty about the classification of each observation, measured by entropy. This updating rule has the advantage of being simple and the collaboration is done with all remote sites at the same time. This avoids the classic problem of choosing the site to collaborate with at each step.

We tested our approach on several datasets in the horizontal collaboration setting, but it is also applicable in the hybrid setting. The results showed an improvement over the state of the art. The framework also provides a way to visualize information flow during the process. It shows interesting behaviors, as algorithms with lower initial performance tend to use more incoming information than others. Moreover, even the algorithms with the best performance after the local step were able to improve during

the process. Indeed, the flexibility provided by Co-LUPI in weighting incoming information allows algorithms to benefit from globally less efficient counterparts, as the latter may perform better locally.

Several improvements can be made to the proposed algorithm. An optimal transport algorithm could be used to relax the assumption that each algorithm searches for the same number of clusters. It would also be interesting to test it in the hybrid setting, where different data sites have access to different individuals.

Finally, we provided a detailed analysis of the collaboration process. We concluded that the choice of the weighting function influences the way information flows between data sites. Each data site might have its own way of weighting incoming information, depending on what it expects from the collaboration.

4 Theoretical guarantees

So far in this work, we have adopted an empirical approach. We have identified several issues in collaborative clustering and proposed an architecture that addresses them. We then tested that architecture in several settings and the experiments exhibited some interesting properties. In this chapter we are interested in the theoretical results that can be derived from this collaborative learning architecture. We start with an analysis of the complexity of the proposed algorithm. Then we provide a termination analysis of the algorithm. Finally, we discuss the generalization bounds.

4.1 Reminder: the proposed architecture

Let S be a training sample distributed across different nodes of a network. The dataset S can either be labelled or unlabelled:

1. $S = \{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{n}$ in the labelled case. Then $\mathcal{D}$ denotes the joint distribution of $(\mathbf{X}, \mathrm{Y})$.
2. $S = \{(\mathbf{x}_k)\}_{i=1}^{n}$ in the unlabelled case. Then $\mathcal{D}$ denotes the distribution of $\mathbf{X}$.

In both cases, we can write $S \sim \mathcal{D}^n$. In a classical setting, all data in $\mathbf{X}$ are described by real-valued d-dimensional vectors, i.e. $\mathbf{X} \subseteq \mathbb{R}^d$. But in the collaborative learning setting, we suppose that the global dataset X is distributed over P data sites: $X^{[1]}, \ldots, X^{[p]}, \ldots, X^{[P]}$, where $X^{[p]} = \{x_i^{[p]}\}_{i=1}^{n}$ is a set of n objects. By abuse of notation, we say that each object $x_i^{[p]} \in \mathbb{R}^d$ is a d-dimensional vector, ignoring that the number of features depends on p.

Now our goal is to train a set of machine learning models $A^{[p]}$, one on each node of the network. Each model will be trained by optimizing a quality function that we denote by $Q^{[p]}$, be it in the supervised or in the unsupervised setting. This training takes place in two steps:

Local step In the local step, each algorithm $\mathcal{A}^{[p]}$ works on its own dataset $X^{[p]}$ and adjusts its parameters as it would in a non-collaborative setting. In order to exchange their results, the algorithms must have some type of information in common. In our case, the partition computed by each algorithm will be represented as a responsibility matrix $R^{[p]}$. This matrix contains the contributions of each component to each observation, $R^{[p]}_{i,k} = \mathbb{P}(Z_i = k|X_i, \theta)$, where K is the number of clusters, $Z_i \in \{1, \ldots, K\}$ are the components of the model and the elements of θ are its parameters.

Once each model has been trained locally, we want them to exchange these partition matrices with each other to improve their performance. This exchange is done during the collaboration step.

Collaborative step During the collaboration step, the different algorithms $\mathcal{A}^{[p]}$ will exchange information in order to improve their respective classification. The resulting clustering will not only be based on the examples $X^{[p]}$, but also on the additional information $X^{[p]^*}$, which are in fact the remote partition matrices. The local learner thus has access to $X^{[p]}$ and $X^{[p]^*}$, instead of just $X^{[p]}$, to perform its clustering. In our case, for each algorithm $\mathcal{A}^{[p]}$, $X^{[p]^*} \equiv R^{-[p]}$, where:

$$R^{-[p]} = \{R^{[q]} : q \in \{1, \ldots, P\} \setminus \{p\}\} \tag{4.1}$$

The set $R^{-[p]}$ contains all the remote partition matrices, from the perspective of site p.

The collaboration function to update a model's predictions is given in section 3.3.2. If the new predictions lead to a quality improvement ($Q^{[p]}(t+1) < Q^{[p]}(t)$), then the model's parameters are updated, otherwise they stay in the same state. The process stops on the first iteration where no model improves more than some threshold ε.

One might wonder what are the properties of this algorithm. First, we said that the different data site exchange information during the collaborative step. What is the volume of the information exchanged ? We also said that the process stops whenever no algorithm improve during an iteration of the collaborative step. Is there any guarantee that this criterion will ever be met ? Finally, what can we say about the performance of the models ?

4.2 Complexity

The architecture we have proposed is model-agnostic. The time complexity will depend on the models used. On the other hand, the information exchanged is the same whatever the models used. We can therefore calculate the additional space complexity brought by the collaboration.

Each model outputs a responsibility matrix of size $N \times K$. And sends it to the $P-1$ remote counterparts. This result in a space complexity for one collaboration iteration in $O(NKP^2)$, where K is the number of classes.

4.3 Termination analysis

At each collaboration iteration, each model shares its results and uses those of its counterparts to updadte its state. As is, there is no guarantee that this algorithm will eventually converge and stop. Indeed, if we consider the sequence of states taken by the entire system, it is easy to imagine it oscillating between some accumulation

points. Thus, a stopping criterion using, say, the convergence of the models parameters, cannot provide any guarantee that the algorithm stops. Unless the convergence is artificially enforced by some decreasing energy function scaling the updates. This way of ensuring can seem clumsy as the user is stopping the learning process without knowing whether there is still progress to be made or not.

Fortunately, we can still say something about the termination property in our setting. For this, will will use the fact that the quality functions $Q^{[p]}$ are asked to increase throughout the collaboration. Then if each quality function satisfies at least one of the following properties, our algorithm has stop property:

P_1: Q is computed using a hard clustering

P_2: Q is continuous with respect to the responsibility matrix R

P_3: Q is bounded

If Q is based on a hard clustering, then given a dataset of size n, it can only take a finite number of values. In particular, if we look for k clusters, the quality function Q takes no more values than the number of different possible partitions of n objects into k non-empty subsets. This number of partitions is given by the Stirling number of the second kind $S(n,k)$:

$$S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n \tag{4.2}$$

As a finite set, the image of the set of possible partitions by Q is bounded. Thus, for any given dataset, Q is bounded.

Now, we show that if Q is continuous with respect to its input, i.e. the responsibility matrix R, then it is also bounded. First, note that each line of R is a probability distribution of K events. It thus lives in the $(K-1)$-simplex Δ^{K-1} which is compact.

Then $R \in \{\Delta^{K-1}\}^N$ which is also a compact set. Thus, as a continuous function on a compact set, Q is bounded (and attains its lower and upper bounds).

It follows that $P_1 \Rightarrow P_3$ and $P_2 \Rightarrow P_3$.

We now assume that for all $p = 1, \ldots, P$, $Q^{[p]}$ is bounded. Whether model p tries to minimize or to maximize its quality function, $\left(Q^{[p]}(j)\right)_{j\in\mathbb{N}}$ is a monotone sequence. Then for all p, $\left(Q^{[p]}(j)\right)_{j\in\mathbb{N}}$ is a bounded, monotone sequence, and as such, it converges. Thus, $\forall \varepsilon > 0$, $\exists j_p$ such that, $\forall j \geqslant j_p$, $\left|Q^{[p]}(j) - Q^{[p]}(j+1)\right| < \varepsilon$.

Let ε be the threshold triggering the end of the algorithm, as described in section 4.1, using the fact that $\left(Q^{[p]}(j)\right)_{j\in\mathbb{N}}$ is convergent and the least integer principle, we can define for each data site p

$$j_p = \min\{j : \forall j' \geqslant j, \left|Q^{[p]}(j') - Q^{[p]}(j'+1)\right| < \varepsilon\},$$

and

$$j^* = \max_p\{j_p\}.$$

Then, $\forall p \in \{1, \ldots, P\}$, $\left|Q^{[p]}(j^*) - Q^{[p]}(j^*+1)\right| < \varepsilon$, which is our stopping criterion.

In summary, despite the absence of stop property in the most general case, we were able to prove that under mild hypothesis, our alogorithm eventually terminates. In a nutshell, the proof is based on the fact that the models try to improve their quality functions and that they cannot do so forever[1], provided that each function has any of the properties P_1, P_2, or P_3.

[1] or more precisely, if they can improve forever, then for any positive value, the magnitude of the improvements will eventually stay below this value.

4.4 Generalization bounds for supervised collaborative learning

In this section, we focus on the learners' ability to generalize to unseen data. This question is central in machine learning. Answering it allows us to know with a certain level of confidence, to what extent a model has succeeded in adapting to the (unknown) distribution of the data. We focus on one learner of the collaborative architecture and try to provide generalization bounds for that learner.

We consider an input space $\mathcal{X}$ and an output space $\mathcal{Y}$. We only consider the case where $\mathcal{X} = \mathbb{R}^d$, and $\mathcal{Y} = \{-1, 1\}$. We denote by $\mathcal{D}$ the (unknown) distribution of $\mathcal{X} * \mathcal{Y}$. We consider a sample S of independent and identically distributed (i.i.d.) pairs $(X_1, \mathrm{Y}_1), \ldots, (X_n, \mathrm{Y}_n)$. Then $S \sim \mathcal{D}^n$. We consider a set $\mathcal{H} = \{h \mid h : \mathcal{X} \to \mathcal{Y}\}$ of predictors. This set is also known as a hypothesis class. We consider a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, +\infty)$, with $\ell(y, y) = 0$. Common loss functions are:

- The 0-1 loss: $\ell(y', y) = \mathbb{1}(y \neq y')$
- The hinge loss: $\ell(y', y) = max(1 - yy', 0)$
- The absolute loss: $\ell(y', y) = |y' - y|$
- The quadratic loss: $\ell(y', y) = (y' - y)^2$

The true risk of a predictor $h \in \mathcal{H}$ is given by

$$R_{\mathcal{D}}(h) = \mathbb{E}_{\mathcal{D}}[\ell(h(\mathrm{X}), \mathrm{Y})]. \tag{4.3}$$

A good predictor must have a small risk. However, the definition of the true risk involves the data distribution $\mathcal{D}$, which is unknown. Therefore, we cannot compute

$R(h)$ and have to fall back to empirical risk:

$$R_{\hat{\mathcal{D}}}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i) \tag{4.4}$$

Given a hypothesis $h \in \mathcal{H}$, one can compute its empirical risk $R_{\hat{\mathcal{D}}}(h)$. But what we truly want to optimize is the true risk, $R_{\mathcal{D}}$. Then, given the empirical risk, what can we say about the value of the true risk? Interestingly enough, even though $\mathcal{D}$ is unknown, we can still say something about the difference between $R_{\hat{\mathcal{D}}}(h)$ and $R_{\mathcal{D}}(h)$. These results come in the form of a probability:

$$\mathbb{P}\left(|R_{\hat{\mathcal{D}}}(h) - R_{\mathcal{D}}(h)| \leq \varepsilon\right) \geq 1 - \delta \tag{4.5}$$

There are many papers proposing new bounds. These bounds can be of several types. For instance, the Vapnik-Chervonenkis bounds rely on the concept of VC dimension [68, 69]. There are also bounds based on the Rademacher complexity, the uniform stability, or the robustness of the algorithm at hand [10, 19, 37, 41, 75].

The kind of bounds that we are mostly interested in is the PAC-Bayes generalization bounds. These bounds have been introduced by Shawe-Taylor and Williamson in [58], and extended by McAllester in [44]. Instead on trying to pinpoint one best hypothesis $h^* \in \mathcal{H}$, the PAC-Bayes approach will consider that given a prior distribution π on $\mathcal{H}$, one can try to find a posterior distribtuion ρ such that the aggregated predictor with weights ρ:

$$h_\rho(x) = \mathbb{E}_{h\sim\rho}[h(x)] \tag{4.6}$$

performs well.

The PAC-Bayes generalization bounds can provide results for three kinds of risk:

- the risk of a single draw $\hat{h}$ from the posterior: $R_{\mathcal{D}}(\hat{h})$, where $\hat{h} \sim \rho$

- the expected risk of a random hypothesis: $\mathbb{E}_{h\sim\rho}R_{\mathcal{D}}(h)$
- the risk of the deterministic aggregated hypothesis: $R_{\mathcal{D}}(h_\rho)$

Bounds for all these types of risks can be found in [2]. Here, we start by giving a simple bound for $\mathbb{E}_{h\sim\rho}R_{\mathcal{D}}(h)$. This bounds is due to Catoni [12]:

Theorem 1 (Catoni's bound) *Let* $\lambda > 0$, *and* $\forall\varepsilon \in (0,1)$, *then*

$$\mathbb{P}_S\Bigg(\forall\rho\in\mathcal{P}(\mathcal{H}),\ \mathbb{E}_{h\sim\rho}[R_{\mathcal{D}}(h)] \leq \mathbb{E}_{h\sim\rho}[R_{\hat{\mathcal{D}}}(h)]+\frac{\lambda C^2}{8n}+\frac{KL(\rho\|\pi)+\log\frac{1}{\varepsilon}}{\lambda}\Bigg)\geq 1-\varepsilon$$

Where $\mathbb{P}_S$ is the probability with respect to the sample $[(\mathbf{X}_1,\mathrm{Y}_1),\ldots,(\mathbf{X}_n,\mathrm{Y}_n)]$, $\mathcal{P}(\mathcal{H})$ is the set of probability distributions on $\mathcal{H}$, C is such that $0 \leq \ell(y',y) \leq C$, and KL is the Kullback-Leibler divergence defined by:

$$KL(\mu\|\nu)=\sum_x \log\left(\frac{\mu(x)}{\nu(x)}\right)\cdot\mu(x) \tag{4.7}$$

A proof of the theorem is given in appendix C.

For a convex loss function (e.g. the hinge loss), the Jensen's inequality gives:

$$\mathbb{E}_{h\sim\rho}[R_{\mathcal{D}}(h)] \geq R_{\mathcal{D}}(\mathbb{E}_{h\sim\rho}[h]) \tag{4.8}$$

This gives a bound for the aggregated hypothesis $R_{\mathcal{D}}(h_\rho)$:

$$\mathbb{P}_S\Bigg(\forall \rho \in \mathcal{P}(\mathcal{H}), R_{\mathcal{D}}(\mathbb{E}_{h\sim\rho}[h]) \\ \leq R_{\hat{\mathcal{D}}}(\mathbb{E}_{h\sim\rho}[h]) + \frac{\lambda C^2}{8n} + \frac{KL(\rho\|\pi) + \log\frac{1}{\varepsilon}}{\lambda}\Bigg) \geq 1-\varepsilon \tag{4.9}$$

The equation gives a bound for the aggregated predictor, provided that the posterior ρ is constant with respect to the input. However, in the collaboration scheme we proposed, these weights vary across the input space, depending on the models expertise in the different regions of the input space. The prediction aggregation can be written as follows

$$h^*(\mathbf{x}) = \sum_{j=1}^{P} f^j(\mathbf{x})\, h^j(\mathbf{x})$$

where $f : \mathcal{X} \to [0,1] \in \mathcal{F}$ is such that $\sum_j f^j(\mathbf{x}) = 1, \forall \mathbf{x} \in \mathcal{X}$.

In the particular case of the heuristic we proposed, $f^j(\mathbf{x})$ depends on the entropy of the predictors for the classification of the datapoint $\mathbf{x}$.

We can now define a new hypothesis class $\mathcal{G}$ defined by:

$$\mathcal{G} = \{g = P \cdot f \cdot h : (f,h) \in \mathcal{F} \times \mathcal{H}\}$$

and write,

$$h^*(\mathbf{x}) = \frac{1}{P}\sum_{j=1}^{P} g^j(\mathbf{x}) \tag{4.10}$$

and what was a locally weighted aggregation in $\mathcal{H}$ is now uniformly weighted in $\mathcal{G}$. We cannot use the 0-1 loss in this context, because in general the prediction will be in $(0,1)$. But we can still use the hinge loss, which is convex, so equation 4.9 holds.

This workaround comes at a cost. Indeed, we increased the size of the hypothesis space. Thus, the term $KL(\rho\|\pi)$ in the bound will likely be increased. Suppose for example that the prior π is the uniform distribution on $\mathcal{G}$, and that the posterior is uniformly distributed over P hypothesis. Then

$$KL(\rho\|\pi) = \sum_{j=1}^{P} \frac{1}{P} \log\left(\frac{|\mathcal{G}|}{P}\right). \tag{4.11}$$

In this case, the Kullback-Leibler divergence increases with $|\mathcal{G}|$. In general, the larger $\mathcal{G}$, the more elements have low prior probability. This phenomenon increases $KL(\rho\|\pi)$.

A first solution would be to use *data-dependent priors*. The idea is to use part of the dataset to adjust the prior and have more weight on the relevant hypothesis [24,49,57]. A second one would be to look for bounds that apply to the specific case of locally weighted classifiers.

5 Conclusion and perspectives

We have identified several scientific obstacles and have proposed an architecture to address them. We have used for the first time the collaborative learning paradigm in the supervised framework. The reader will also find in appendix an analysis of some properties of the collaborative function we propose.

The first scientific obstacle to which we propose an answer is the choice of the collaborator. Most often, in the literature, a single remote model is selected at each iteration, and the collaboration then takes place with this model.

The second obstacle on which we have worked is the choice of the intensity of the collaboration. This is the coefficient associated with the penalty of the divergence between two models. Determining a good collaboration coefficient is a complex task, on the one hand because in unsupervised learning the performance of a model is difficult to measure objectively, and on the other hand because this coefficient is global. The implications in terms of the evolution of the classification for each individual are difficult to predict.

Another challenge, related to the two previous ones, is the deterioration of the results by the less performing models. It seems that the use of a global collaboration coefficient is the main reason for this deterioration. Even after selecting the collaborator according to some criterion, if we give the same weight to the distant classifications of all the individuals, then we risk modifying without distiction those who are easy to classify and those for whom the task is more complicated.

The approach we propose allows us to modify the classifications of only some individuals, depending on whether or not they are in a region of expertise. The higher the uncertainty about the classification of an individual, the more we will use external information to decide. In chapter 4, we provided a generalization bound for this algorithm. It required us to consider a wide hypothesis class, leading to potentially loose bounds. A way to tackle this problem would be to come up with *ad hoc* generalization bounds.

We have seen that one of the limitations of our algorithm is a tendency to homogenize the results. We proposed to introduce a similarity criterion to solve this problem. By using such a criterion, we ensure that the collaboration does not lead to an already excluded track. To further explore this method, we could test it in a configuration where there are two different structures, and where each model has a view highlighting one or the other of these structures. Then the models seeing the same structure should exchange more between them. In this case, we would see groups of sites with high levels of mutual trust, and relatively low levels of trust towards the other sites.

In our proposal, we made two assumptions that we could try to overcome. The first one is that the models trained on each site all have information about the same individuals. The second is that all algorithms search for the same number of clusters.

The former is a very strong hypothesis. Fortunately, it is also the easiest to overcome. Indeed, we can allow each site to have data on only a subset of the individuals in the database. Then at the time of collaboration, for each individual it would only consider the remote sites that have entries on these individuals. The sum in the equation 3.20, instead of covering all the remote sites, would then cover all the sites with the individuals in question. The normalization constants must be adjusted accordingly.

The second constraint can be droped by allowing each model to search for an arbitrary number of clusters. The next step is to find a correspondence between different

numbers of clusters. This can be done by using the optimal transport system [71].

A Datasets

Dataset name	Samples	Features	Description
adult	48842	15	Prediction task is to determine whether a person makes over 50K a year. Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: (AAGE>16) and (AGI>100) and (AFNLWGT>1) and (HRSWK>0)
amazon	32769	10	Data from the Kaggle Amazon Employee challenge.
click	399482	12	This data is derived from the 2012 KDD Cup. The data is subsampled to 1% of the original number of instances, downsampling the majority class (click=0) so that the target feature is reasonably balanced (5 to 1). The data is about advertisements shown alongside search results in a search engine, and whether or not people clicked on these ads. The task is to build the best possible model to predict whether a user will click on a given ad.

Dataset name	Samples	Features	Description
internet	10108	69	Binarized version of the original data set. The multi-class target feature is converted to a two-class nominal target feature by re-labeling the majority class as positive ('P') and all others as negative ('N'). Originally converted by Quan Sun.
iris	150	4	This dataset contains 150 instances of iris flowers described by 4 attributes (sepal length, sepal width, petal length, petal width). The data are equally divided into three classes (setosa, versicolor, virginica). One of the classes (setosa) is linearly separable from the other two. The latter are not linearly separable.
kdd98	191260	479	Dataset KDD98 challenge. The goal is to estimate the return from a direct mailing in order to maximize donation profits. This dataset represents problem of binary classification - whether there was a response to mailing. The data is subsampled to 50% of the original number of instances.
kddchurn	50000	231	Small version of KDD 2009 Cup data.
kick	72983	36	Data from "Don't Get Kicked!" Kaggle challenge.
spam base	4601	57	Spam and non-spam e-mails.

Dataset name	Samples	Features	Description
waveform	5000	40	5000 instances with 40 attributes each. 21 of them are informative and the latter 19 are all noise. There are 3 classes of waves and each class is a combination of 2 of 3 base waves.
wdbc	569	30	The Breast Cancer Wisconsin (Diagnostic) (WDBC) dataset consists in 569 digitized images of a breast mass. There are 30 real-valued input variables describing the cell nuclei present in each image. Each observation is labelled as benign or malignant.
wine	178	13	Wine data is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
appetency	50000	231	Small version of KDD 2009 Cup data.
upsel	50000	231	Small version of KDD 2009 Cup data.

B Publications

Preprint

Foucade Y., Bennani Y., (2021) Unsupervised collaborative learning using privileged information.

French Conference

Foucade Y., Bennani Y., (2021) Clustering collaboratif à partir de données et d'informations privilégiées, Conférence Internationale Francophone sur la science des données.

International Conferences

Foucade Y., Bennani Y., (2020) Unsupervised learning from data and learners, Soft Computing and Pattern Recognition.

Foucade Y., Bennani Y., Aabbou Z., (2021) Collaborative Random Forests Learning, 7th IEEE International Conference on Data Science and Systems.

C Proof of theorem 1

The proof of the theorem is given in [2] and [12]. We report it here with our notations. We first give two preliminary results:

Lemma 2 (Hoeffding's inequality) *Let $U_1, \dots, U_n$ be independent random variables taking values in an interval $[a, b]$. then $\forall t > 0$*

$$\mathbb{E}\left[e^{t\sum_{i=1}^{n}(U_i - \mathbb{E}(U_i))}\right] \leq e^{\frac{nt^2(b-a)^2}{8}} \tag{C.1}$$

Lemma 3 (Donsker-Varadhan representation of the KL divergence) *For any measurable, bounded function $g : \mathcal{H} \to \mathbb{R}$, we have:*

$$\log \mathbb{E}_{h\sim\pi}[e^{g(h)}] = \sup_{\rho\in\mathcal{P}(\mathcal{H})} \left[\mathbb{E}[g(h)] - KL(\rho\|\pi)\right] \tag{C.2}$$

Proof of theorem 1. For any $h \in \mathcal{H}$ and $t > 0$, the Hoeffding's inequality applied to $U_i = \mathbb{E}[\ell(h(\mathbf{x}_i), y_i)] - \ell(h(\mathbf{x}_i), y_i)$ yields:

$$\mathbb{E}_S[e^{tn[R(h)-\hat{R}(h)]}] \leq e^{\frac{nt^2C^2}{8}} \tag{C.3}$$

let $t = \frac{\lambda}{n}$

$$\mathbb{E}_S[e^{\lambda[R(h)-\hat{R}(h)]}] \leq e^{\frac{\lambda^2C^2}{8n}} \tag{C.4}$$

Integrate with respect to π and apply Fubini:

$$\mathbb{E}_S \mathbb{E}_{h\sim\pi}[e^{\lambda[R(h)-\hat{R}(h)]}] \leq e^{\frac{\lambda^2 C^2}{8n}} \tag{C.5}$$

Now we apply lemma 3:

$$\mathbb{E}_S[e^{\sup_{\rho\in\mathcal{P}(\mathcal{H})} \lambda \mathbb{E}_{h\sim\rho}[R(h)-\hat{R}(h)]-KL(\rho\|\pi)-\frac{\lambda^2 C^2}{8n}}] \leq 1 \tag{C.6}$$

And use the Chernoff bound with $s > 0$:

$$\mathbb{P}_S[\sup_{\rho\sim\mathcal{P}(\mathcal{H})} \lambda \mathbb{E}[R(h) - \hat{R}(h)] - KL(\rho\|\pi) - \frac{\lambda^2 C^2}{8n} > s] \leq e^{-s} \tag{C.7}$$

Finally, write $\epsilon = e^{-s}$, and rearrange the terms:

$$\mathbb{P}_S\left[\exists \rho \in \mathcal{P}(\mathcal{H}) : \mathbb{E}_{h\sim\rho}[R(h)] > \mathbb{E}_{h\sim\rho}[\hat{R}(h)] + \frac{\lambda C^2}{8n} + \frac{KL(\rho\|\pi) + \log\frac{1}{\epsilon}}{\lambda}\right] \leq \epsilon \tag{C.8}$$

This ends the proof.

Bibliography

[1] Muna Al-Razgan and Carlotta Domeniconi. Weighted clustering ensembles. In *SDM*, 04 2006.

[2] Pierre Alquier. User-friendly introduction to pac-bayes bounds, 2021.

[3] Hanan Ayad and Mohamed S. Kamel. Cumulative voting consensus method for partitions with variable number of clusters. *IEEE transactions on pattern analysis and machine intelligence*, 30:160–73, 02 2008.

[4] Fatima-Ezzahraa Ben Bouazza, Guenael Cabanes, Younès Bennani, and Abdelfettah Touzani. Collaborative clustering through optimal transport. In *International Conference on Artificial Neural Networks*, pages –. Springer, In press.

[5] Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A working set of axioms for clustering. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[7] James Bezdek. *Pattern Recognition With Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 01 1981.

[8] Christopher Bishop, Markus Svensen, and Christopher Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10:215–234, 05 1997.

[9] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: The Generative Topographic Mapping. *Neural Computation*, 10(1):215–234, 01 1998.

[10] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 06 2002.

[11] Guénaël Cabanes and Younès Bennani. A simultaneous two-level clustering algorithm for automatic model selection. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 316–321, 12 2007.

[12] Olivier Catoni. A pac-bayesian approach to adaptive classification, 2003.

[13] Hakan Cevikalp and Robi Polikar. Local classifier weighting by quadratic programming. *IEEE Transactions on Neural Networks*, 19:1832–1838, 2008.

[14] Guillaume Cleuziou, Matthieu Exbrayat, Lionel Martin, and Jacques-Henri Sublemontier. Cofkm : un modèle de clustering flou collaboratif pour les données multi-représentées. In *é-EGC Apprentissage Statistique et Fouille de données*, 05 2009.

[15] Antoine Cornuéjols, Cédric Wemmert, Pierre Gancarski, and Younès Bennani. Collaborative clustering: Why, when, what and how. *Information Fusion*, 39, 04 2017.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[17] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[18] Bernard Desgraupes. Clustering indices. *University of Paris Ouest-Lab Modal'X*, 1:34, 2017.

[19] L. Devroye and T. Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207, March 1979.

[20] Franz Dietrich and Christian List. Probabilistic opinion pooling. In *Oxford Handbook of Probability and Philosophy*, page forthcoming. Oxford University Press, 2016.

[21] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[22] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[23] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[24] Gintare Karolina Dziugaite and Daniel M. Roy. Data-dependent pac-bayes priors via differential privacy. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8440–8450, Red Hook, NY, USA, 2018. Curran Associates Inc.

[25] Xiaoli Zhang Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 36, New York, NY, USA, 2004. Association for Computing Machinery.

[26] Germain Forestier, Cédric Wemmert, and Pierre Gancarski. Collaborative multi-strategical clustering for object-oriented image analysis, 10 2018.

[27] Ana L.N. Fred and Anil K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):835–850, 2005.

[28] Mohamad Ghassany, Nistor Grozavu, and Younès Bennani. Collaborative generative topographic mapping. In *Neural Information Processing*, volume 7664, pages 591–598, 11 2012.

[29] Mohamad Ghassany, Nistor Grozavu, and Younès Bennani. Collaborative multi-view clustering. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 08 2013.

[30] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. *TKDD*, 1, 03 2007.

[31] Nistor Grozavu and Younès Bennani. Topological collaborative clustering. *Aust. J. Intell. Inf. Process. Syst.*, 12, 2010.

[32] Christian Hennig. What are the true clusters? *Pattern Recognition Letters*, 64:53–62, 2015. Philosophical Aspects of Pattern Recognition.

[33] Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. Adaptive mixture of local expert. *Neural Computation*, 3:78–88, 02 1991.

[34] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, sep 1999.

[35] Rasha Kashef and Mohamed S. Kamel. Cooperative clustering. *Pattern Recognition*, 43:2315–2329, 06 2010.

[36] Teuvo Kohonen. *Self-Organized Formation of Topologically Correct Feature Maps*, page 509–521. MIT Press, Cambridge, MA, USA, 1988.

[37] Vladimir Koltchinskii and Dmitry Panchenko. Rademacher processes and bounding the risk of function learning, 2004.

[38] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[39] L.I. Kuncheva, S.T. Hadjitodorov, and L.P. Todorova. Experimental comparison of cluster ensemble methods. In *2006 9th International Conference on Information Fusion*, pages 1–7, 2006.

[40] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, USA, 2004.

[41] Samuel Kutin and Partha Niyogi. Almost-everywhere algorithmic stability and generalization error. In *UAI*, 2002.

[42] Zhuwen Li, Jiaming Guo, Loong-Fah Cheong, and Steven Zhou. Perspective motion segmentation via collaborative clustering. In *2013 IEEE International Conference on Computer Vision*, pages 1369–1376, 12 2013.

[43] Yi Liu, Li Zhang, Ning Ge, and Guanghao Li. A Systematic Literature Review on Federated Learning: From A Model Quality Perspective. *arXiv:2012.01973 [cs]*, December 2020. arXiv: 2012.01973.

[44] David A. McAllester. Some pac-bayesian theorems. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 230–234, New York, NY, USA, 1998. Association for Computing Machinery.

[45] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks

from decentralized data, 2017.

[46] Davoud Moulavi, Pablo A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. Density-based clustering validation. In *SDM*, 2014.

[47] Kevin P. Murphy. Mixture models and the em algorithm. In *Machine learning: a probabilistic perspective*, chapter 11. MIT Press, Cambridge, Mass. [u.a.], 2013.

[48] Nam Nguyen and Rich Caruana. Consensus clusterings. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 607 – 612, 11 2007.

[49] Emilio Parrado-Hernández, Amiran Ambroladze, John Shawe-Taylor, and Shiliang Sun. Pac-bayes bounds with data dependent priors. *Journal of Machine Learning Research*, 13(112):3507–3531, 2012.

[50] Witold Pedrycz. Collaborative fuzzy clustering. *Pattern Recognit. Lett.*, 23:1675–1686, 2002.

[51] Witold Pedrycz and Partab Rai. Rai, p.: Collaborative clustering with the use of fuzzy c-means and its quantification. fuzzy sets and systems 159(18), 2399-2427. *Fuzzy Sets and Systems*, 159:2399–2427, 09 2008.

[52] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2019.

[53] Oliver Radley-Gardner, Hugh Beale, and Reinhard Zimmermann, editors. *Fundamental Texts On European Private Law*. Hart Publishing, 2016.

[54] Parisa Rastin, Basarab Matei, Guénaël Cabanes, Nistor Grozavu, and Younès Bennani. Impact of Learners' Quality and Diversity in Collaborative Clustering. *Journal of Artificial Intelligence and Soft Computing Research*, 9(2):149–165, April 2019.

[55] Jan-Willem Romeijn. An interpretation of weights in linear opinion pooling. *Episteme*, page 1–15, 2020.

[56] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[57] Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *J. Mach. Learn. Res.*, 3(null):233–269, mar 2003.

[58] John Shawe-Taylor and Robert C. Williamson. A pac analysis of a bayesian estimator. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, COLT '97, page 2–9, New York, NY, USA, 1997. Association for Computing Machinery.

[59] Yinghua Shen and Witold Pedrycz. Collaborative fuzzy clustering algorithm: Some refinements. *International Journal of Approximate Reasoning*, 86:41–61, 2017.

[60] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 01 2002.

[61] Jeremie Sublime, Basarab Matei, Guénaël Cabanes, Nistor Grozavu, Younès Bennani, and Antoine Cornuéjols. Entropy based probabilistic collaborative clustering. *Pattern Recognition*, 72:144–157, 12 2017.

[62] Jeremie 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)*, 07 2018.

[63] Jérémie Sublime, Nistor Grozavu, Younès Bennani, and Antoine Cornuéjols. Collaborative clustering with heterogeneous algorithms. *Proceedings of the International Joint Conference on Neural Networks*, 2015-Septe, 2015.

[64] Catherine A. Sugar and Gareth M. James. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98(463):750–763, 2003.

[65] P.N. Tan, Michael Steinbach, and Vipin Kumar. Cluster analysis: Basic concepts and algorithms. *Introduction to Data Mining*, pages 487–568, 01 2005.

[66] A. Topchy, Arjun Jain, and William Punch. Combining multiple weak clusterings. In *Third IEEE International Conference on Data Mining*, pages 331– 338, 12 2003.

[67] Shivakumar Vaithyanathan and Byron Dom. Model selection in unsupervised learning with applications to document clustering. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 433–443, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[68] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[69] Vladimir N. Vapnik. *Controlling the Generalization Ability of Learning Processes*, pages 93–122. Springer New York, New York, NY, 2000.

[70] Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *IJPRAI*, 25:337–372, 05 2011.

[71] Cédric Villani. *Optimal transport – Old and new*, volume 338, pages xxii+973. Springer Berlin Heidelberg, 01 2008.

[72] Cédric Wemmert and Pierre Gançarski. A multi-view voting method to combine unsupervised classifications. In *2nd IASTED International Conference on Artificial Intelligence and Applications*, pages 447–452, 09 2002.

[73] White House Report. Consumer Data Privacy in a Networked World: A Framework for Protecting Privacy and Promoting Innovation in the Global Digital Economy. *Journal of Privacy and Confidentiality*, March 2013.

[74] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[75] Huan Xu and Shie Mannor. Robustness and Generalization. *arXiv:1005.2243 [cs]*, May 2010. arXiv: 1005.2243.

[76] Li Zeng, Ling Li, Lian Duan, Kevin Lu, Zhongzhi Shi, Maoguang Wang, Wenjuan Wu, and Ping Luo. Distributed data mining: A survey. *Information Technology and Management*, 13, 12 2012.

[77] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman &; Hall/CRC, 1st edition, 2012.

[78] Zhi-Hua Zhou and Wei Tang. Clusterer ensemble. *Knowl. Based Syst.*, 19:77–83, 2006.