Pattern Recognition 72 (2017) 144–157

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Entropy based probabilistic collaborative clustering

Jérémie Sublime[a,b,*], Basarab Matei[b], Guénaël Cabanes[b], Nistor Grozavu[b], Younès Bennani[b], Antoine Cornuéjols[c]

[a] LISITE Laboratory, RDI Team - ISEP 10 rue de Vanves, 92130 Issy Les Moulineaux, France
[b] Université Paris 13, Sorbonne Paris Cité, LIPN - CNRS UMR 7030 99 av. J-B Clément, 93430 Villetaneuse, France
[c] UMR MIA-Paris, AgroParisTech, INRA Université Paris-Saclay, 75005 Paris, France

ARTICLE INFO

Article history:
Received 17 December 2016
Revised 24 April 2017
Accepted 8 July 2017
Available online 10 July 2017

Keywords:
Collaborative clustering
EM algorithms
Entropy based methods

ABSTRACT

Unsupervised machine learning approaches involving several clustering algorithms working together to tackle difficult data sets are a recent area of research with a large number of applications such as clustering of distributed data, multi-expert clustering, multi-scale clustering analysis or multi-view clustering. Most of these frameworks can be regrouped under the umbrella of collaborative clustering, the aim of which is to reveal the common underlying structures found by the different algorithms while analyzing the data.

Within this context, the purpose of this article is to propose a collaborative framework lifting the limitations of many of the previously proposed methods: Our proposed collaborative learning method makes possible for a wide range of clustering algorithms from different families to work together based solely on their clustering solutions, thus lifting previous limitation requiring identical prototypes between the different collaborators. Our proposed framework uses a variational EM as its theoretical basis for the collaboration process and can be applied to any of the previously mentioned collaborative contexts.

In this article, we give the main ideas and theoretical foundations of our method, and we demonstrate its effectiveness in a series of experiments on real data sets as well as data sets from the literature.

1. Introduction

Data Clustering is a fundamental task in the process of knowledge extraction from databases that aims to discover the intrinsic structures in a set of objects by forming clusters that share similar features. This task is more difficult than supervised classification as the number of clusters to be found is generally unknown and consequently it is difficult to rate the quality of a clustering partition. Over the past two decades, this task has become even more challenging when the available data sets became more complex with the introduction of multi-view data sets, distributed data, and data set having different scales of structures of interest (e.g. hierarchical clusters). This increased complexity in an already hard problem makes it difficult for lone clustering algorithms to give competitive results with a high degree of confidence. However, very much like in the real world, such problems can be tackled more easily by having several algorithms working together in order to increase both the quality of the results and their reliability.

Approaches based on this idea of several algorithms working together have been widely studied in the case of supervised learning [1–4] where they gave birth to the field of Ensemble Learning.

Ensemble methods are easy to implement in supervised learning for two reasons: First, it is straightforward to define a combination of predictive functions to get an aggregated prediction function (for instance, a linear combination is used in boosting). Second, it is simple to measure both the performance of individual prediction functions and the diversity of the set of the functions that are candidate for being part of the combined global decision function. Things are not so straightforward in unsupervised learning. Here, each individual solution is a soft or hard partition of the data set. How to combine these partitions has no obvious answer.

In cooperative clustering, each clustering algorithm produces its result independently. The final clustering is computed in a post-processing step, and the only exchange of information is about when the individual processes are completed, so that post-processing can start. In this case, a set of clustering algorithms are used in parallel on a given data set. Once all local computations

* Corresponding author.
E-mail addresses: jeremie.sublime@isep.fr, jeremie.sublime@gmail.com (J. Sublime), matei@lipn.univ-paris13.fr (B. Matei), guenael.cabanes@lipn.univ-paris13.fr (G. Cabanes), nistor.grozavu@lipn.univ-paris13.fr (N. Grozavu), younes.bennani@lipn.univ-paris13.fr (Y. Bennani), antoine.cornuejols@agroparistech.fr (A. Cornuéjols).

http://dx.doi.org/10.1016/j.patcog.2017.07.014
0031-3203/

are completed, a master algorithm takes control and combines the local results to get a hopefully better overall clustering. The resolution of the possible conflicts between the local solutions requires an algorithm that is able to compare results that may differ in their format (e.g. different numbers of clusters, different degrees of belief associated with the results, ...) and to find a consensus solution that minimizes the overall violation to the local results. The cooperative framework is closely related to the ensemble methods developed for supervised learning. In these approaches, a set of (diverse) classifiers is learned and the classification of new data points is obtained by taking a (weighted) vote of their predictions. Bayesian averaging can be considered as a precursor method. Numerous new ones have been developed, from error-correcting output coding to Bagging, and Boosting and their application in various domains have become routine with often good results.

In collaborative clustering (The sequel of this paper), the group solves together problems defined and imposed by the central controller, affecting an individual task to each learner. Interactions are recurrent between team members, responsibility is collective, the action of each teammate is geared to the performance of the group and vice versa. By contrast to the cooperative clustering model, the collaborative model does not seek an overall hopefully better clustering of a given data set through the combination of individual solutions. In the collaborative framework, the goal is that each local computation, quite possibly applied to distinct data sets, benefits from the work done by the other collaborators. This can be done through the exchange of information about the local data, or the current hypothesized local clustering, or the value of one algorithm's parameters. The validity of the approach rests on the assumption that useful information can be shared among the local tasks. This scheme leads naturally to distributed implementations of the computations, but unlike in the cooperative framework, it generally entails several iterations at each local node because convergence of the consensus solution requires several passes of the algorithm. Indeed, in addition to the problem of what information to exchange between collaborators, one question is how to measure the evolution at each node and on a global level.

There are many applications in unsupervised learning for which collaborative clustering can prove useful:

- *Multi-scale analysis*: In this case several algorithms would be analyzing the same objects, all looking at the same features, but searching for a different number of clusters. That kind of analysis can be beneficial for data sets that have intrinsic multi-scale structures such as satellite images for which a lower level analysis of global landscape areas (urban areas, water bodies, forests) often helps to improve a higher level analysis of smaller details (trees, cars, houses, gardens, streets, etc.).
- *Multi-expert analysis*: In this case, all algorithms would be working on the same objects and features of a difficult data set. Given the very high number of existing clustering algorithms, all more or less specialized and that may or may not give good results depending on the problem, trying several of them on a data set and having them exchanging their information could be justified: merging the informations on clusters found only by some clustering algorithms, refining the results based on clusters that are more or less well identified depending on the method, etc.
- *Multi-view clustering* [5,6]: Different algorithms process different types of attributes for the same objects. For example one algorithm for geometric attributes, one for text attributes, one for colors, one for numerical attributes, etc. The goal of the collaboration in this case would be to have each attribute type processed by a specialized algorithm while giving these algorithms a more global picture of the data set by enabling some exchanges between them.
- *Clustering of distributed data* [7]: The same objects have their attributes split on several databases that can't exchange their data because of privacy issues. While the name is different, this is in fact very much equivalent to multi-view clustering.
- *Big Data Clustering* [8]: Data sets that are too large or have too many attributes to be processed efficiently by a single algorithm may be easier to tackle once their attributes are split and processed by several algorithms. This type of clustering is useful in the area of Big Data analysis and would require a high degree of cooperation between the algorithms to get the global picture.

As one can see, all these applications have a lot of similarities: we have several algorithms working on the same data or subsets of the same data, and that will or could at some point try to aggregate or to mutually exploit their respective results. While some of these applications could be considered a field of their own such as multi-view clustering or distributed clustering [5], all of them can be classified as horizontal collaborative clustering frameworks [9–12]: several algorithms working on the same data eventually looking for a different number of clusters, and not necessarily having access to the same features.

We generally distinguish between two types of collaborative methods [9,11]: Vertical collaboration encompasses all cases where several algorithms are working on different data that have similar clusters or distributions. And Horizontal collaboration deals with cases where several algorithms are collaborating on the same objects, eventually described from different views. In this article, we are mostly interested in horizontal collaboration.

Collaborative methods usually follow a two-step procedure [13]:

1. *Local step*: Each algorithm will individually process the data it has access to and produce a local clustering partition.
2. *Collaborative step*: The algorithms share their results and try to confirm or improve their models with the goal of achieving better clustering results.

These two steps are sometimes followed by an aggregation step which aims at reaching a consensus with the final results after collaboration. In this work we will not address the aggregation step because it is a field of its own, and that depending on the application it may not always be advisable to aggregate, for instance when the different views, sites or scales have conflicting partitions [14]. We will instead focus on the collaborative step where the algorithms exchange bits of information with a goal of mutual improvement.

From there, the main difference between what is traditionally referred as "clustering ensemble learning" [15] and collaborative clustering is that clustering ensemble learning methods aim at finding a single consensus partition, while collaborative clustering does not have this final goal. In short, the field of collaborative clustering is concerned with finding algorithms and functions that allow algorithms to share information and to improve their results based on each other similarities, while the field of ensemble learning is more concerned with finding algorithms and methods to merge the solutions or find a consensus between them. Collaborative clustering can therefore be a task of its own (e.g. multi-view clustering where consensus is not always possible nor advisable), or a preliminary step to an ensemble learning task. The methods and techniques used by both fields are therefore naturally overlapping, and a good collaborative algorithm must respect properties that are very similar to these of a good ensemble learning method:

- Robustness: The collaborative process must lead on average to partitions that are better than the local clustering results.
- Consistency: The updated results must be somehow similar to the original local results.

- Novelty: Collaborative clustering must make it possible to find solutions that would have been otherwise unattainable locally.
- Stability: Results that have a lower sensitivity to noise.

Within this context, in this article we introduce a new and original framework for collaborative clustering that can be applied to the various types of unsupervised collaborative learning tasks that we have previously discussed. Our proposed method lifts off several limitations of previous ensemble learning and collaborative frameworks: the data need not be shared between the different algorithms, the number of cluster can be different between the algorithms, and very different types of algorithms can collaborate together.

The theoretical basis of our work is close from the work of Bickel and Scheffer on the estimation of Mixture Models using Co-EM [16,17]. Our proposed method differs from theirs in the following points: in our case we are treating a broader context than multi-view clustering. Our method makes it possible for algorithms from different families to work together, and once again we do not have the limitation that all algorithms should be searching for the same number of clusters. We propose a variational version of their work for multi-view clustering based on the optimization of a different objective function. The core of our proposed approach is a different discretization process based on a particular class of a posteriori distributions called "combination functions" presented in Section 3.4.1.

The remainder of this article is organized as follows:

In Section 2, we propose a state of the art in which we introduce some of the pioneer and earlier proposed methods and frameworks for collaborative learning with their strengths and weaknesses.

In Section 3, we introduce our proposed method for horizontal collaborative clustering. As stated previously, the method that we propose aims at being more generic than the previously proposed frameworks. We begin by explaining the principle of our method and its theoretical basis. Then we study the stopping criterion and parameters tuning of our algorithm. And finally, we demonstrate that our proposed method has good convergence properties similar to these of a *EM* algorithm.

In Section 4, we show some experimental results. We are mostly interested in showing some potential applications of our proposed method applied to multi-scale clustering and multi-view clustering.

Finally, this work ends with a conclusion and perspectives on future works.

2. State of the art in collaborative clustering

One of the first collaborative clustering algorithm was introduced in 2002 by Pedrycz [13,18] under the name "*Collaborative Fuzzy Clustering*" (CoFC). This method was designed for the specific case of distributed data where the information cannot be shared between the different sites. This method was based on a modified version of the Fuzzy C-Means algorithm [19].

The main limitation of this approach is that it only enables Fuzzy C-Means algorithms to collaborate together, and furthermore some methods even require that all of them be looking for the same number of clusters.

Similar approaches were used to develop several other collaborative-like methods *CoEM* [17], *CoFKM*, [20], and another collaborative EM-like algorithm [21] based on Markov Random Fields.

All these algorithms display similar limitations: the objective functions and sometimes the number of clusters must be identical for all exchanged information. This is due to the fact that they all try to optimize an objective function the form of which is:

$$(\boldsymbol{S}_{opt}, \boldsymbol{\Theta}_{opt}) = \underset{(\boldsymbol{S},\boldsymbol{\Theta})}{\text{Argmax}}\, L_g(\boldsymbol{S}, \boldsymbol{\Theta}) = \underset{(\boldsymbol{S},\boldsymbol{\Theta})}{\text{Argmax}} \sum_{i=1}^{J} \left(L(X^i|S^i, \Theta^i) - \sum_{j\neq i} \tau_{j,i} \cdot \Delta(\Theta^i, \Theta^j) \right) \quad (1)$$

where J is the number of collaborators, $\mathbf{S}$ contains all algorithm's partitions, $\boldsymbol{\Theta}$ their distributions parameters, $L_g(\boldsymbol{S}, \boldsymbol{\Theta})$ is the global likelihood of the system, each $L(X^i|S^i, \Theta^i)$ is the local log-likelihood of a collaborating algorithm, each $\Delta(\Theta^i, \Theta^j)$ the "collaborative term" is a custom pairwise penalty that compares the difference between the parameters or prototypes of two algorithms, and the $\tau_{j,i}$ which do not exist in all methods are weights given to the collaborative penalties. The definition of the local term $L(X^i|S^i, \Theta^i)$ based on which algorithms collaborate together makes the main difference between all these methods, while definition of the penalty $\Delta(\Theta^i, \Theta^j)$ only slightly differs depending on the collaborative method. This later parameter is the limiting one since comparing prototypes and parameters requires that the algorithms have the same types of prototypes and some kind of mapping between the clusters of the different algorithms.

The work of Pedrycz on the CoFC algorithm was also extended to be adapted to the Self-Organizing Maps (SOM) [11,22,23] and to the Generative Topographic Maps (GTM) [24].

In [23], the classical SOM objective function is modified by adding a specific extra term for horizontal collaboration and a different one for vertical collaboration. For the collaborative version of the GTM algorithm [24], the principle is the same with the M-Step of the *EM* algorithm mapping the neurons to the final clusters being modified.

One problem with these two methods is that they do not really solve the main issue of collaboration between different types of algorithms since their model in once again analog to the one in Eq. (1). Furthermore, while the number of clusters does not matter in the case of the collaborative SOM and collaborative GTM, in both cases the maps must have the same number of neurons and be topologically similar to each other. This is actually even more restraining than a requirement on the number of clusters.

The SAMARAH method [25,26] is another type of collaborative framework the strength of which is that it can deal with any kind of hard clustering algorithm and is not concerned with issues such as fitness functions, number of clusters, or prototypes. Unlike the previously introduced method, SAMARAH only handles horizontal collaboration due to the lack of prototypes, and was designed mostly for clustering applied to image data. Its goal is very simple: given J clustering results for the same data, the idea is to modify these results in an iterative and collaborative way with the aim of reducing their diversity in order to make the finding of a consensus solution easier.

Once the results have been generated during the local step, the SAMARAH method maps the clusters of the different algorithms using probabilistic confusion matrices (PCM). Let S^i and S^j be two clustering results from two algorithms $\mathcal{A}^i$ and $\mathcal{A}^j$ looking for K_i and K_j clusters respectively.

Then, the probabilistic confusion matrix (PCM) $\Omega^{i,j}$ that maps the clusters from $\mathcal{A}^i$ to $\mathcal{A}^j$ is defined as shown below:

$$\Omega^{i,j} = \begin{pmatrix} \omega_{1,1}^{i,j} & \cdots & \omega_{1,K_j}^{i,j} \\ \vdots & \ddots & \vdots \\ \omega_{K_i,1}^{i,j} & \cdots & \omega_{K_i,K_j}^{i,j} \end{pmatrix} \text{ where } \omega_{a,b}^{i,j} = \frac{|S_a^i \cap S_b^j|}{|S_a^i|} \quad (2)$$

In Eq. (2), S_a^i denotes the ath cluster of algorithm $\mathcal{A}^i$ i.e., $S_a^i = \{x; x \in X^i, x \in a \text{ by } \mathcal{A}^i\}$ and $|S_a^i|$ is the number of data in this clus-

ter, and $|S_a^i \cap S_b^j|$ is the number of data linked to the ath cluster of $\mathcal{A}^i$ and the bth cluster of $\mathcal{A}^j$ at the same time. The PCM $\Omega^{i,j}$ makes it possible to know whether or not the objects of two results have been grouped in a similar way, or if the two clustering results are dissimilar. The matrix has a key role in the comparison of two clustering results -such as detecting agreements and conflicts-, and has the major advantage of being independent from the clustering algorithm used to generate the results.

The SAMARAH method uses this matrix to detect pairwise conflicts between the different partitions and reduces them by order of perceived importance based on a conflict metric criterion [25] by splitting, merging, or removing clusters. Once the solutions have all been refined, and are consequently quite similar to each other, it proceeds with aggregating them using a process similar to a majority vote [27]. It is therefore a very complete framework that covers all 3 steps of local learning, collaborative learning and result aggregation and does not rely on users parameter.

However, its conflict resolution system certainly is a weak point: it relies on a pairwise conflict criterion, and solves the conflicts one by one by order of perceived importance, and it can lead to sub-optimal results. Finally, while it is also a strong point of the method, the fact that the algorithms parameters or prototypes do not play any role once the local step is over may constitute a weakness, in the sense that the local model is never rebuilt using the new partitions and does not play any active role in either the collaborative step or the consensus step.

3. Horizontal collaborative clustering guided by diversity

3.1. Formalism

In horizontal collaborative clustering we consider a finite group of algorithms $\boldsymbol{\mathcal{A}} = \{\mathcal{A}^1, \ldots, \mathcal{A}^J\}$ that are working on the same data elements, albeit possibly with access to different features, and also possibly looking for a different number of clusters. No assumptions are made on the algorithms themselves. Let $X = \{x_1, \ldots, x_N\}, x_n \in \mathbb{R}^d$ be a data set containing N elements, each of them with d real number features.

Each clustering algorithm $\mathcal{A}^i$ has its own parameters to describe either the clusters or its model, and produces its own clustering solution S^i made of K_i clusters, based on the features of the data set $X^i \subseteq X$ it has access to. In the case of hard clustering, S^i can be translated into a solution vector of size N, and for fuzzy clustering into a matrix of size $N \times K_i$. We denote this later matrix $S^i = (s^i_{n,c})$, where $1 \leq n \leq N$ and $1 \leq c \leq K_i$. The solutions S^i output by the algorithms are therefore two-dimensional matrices of size $N \times K_i$ where each element $s^i_{n,c}$ expresses the responsibility (probability) given by algorithm $\mathcal{A}^i$ to a cluster c for the data element x_n.

Each algorithm $\mathcal{A}^i$ computes the solutions S^i, as usual by introducing a latent discrete random vector Z^i defined on some latent space with the range $[1, \ldots, K_i]$, hence computing the *a posteriori* distribution of the variable Z^i conditionally on X^i and S^i.

Finally, in order to quantify the degree of information coming from the collaboration, for a given algorithm $\mathcal{A}^i$, we will assume the existence of some weight $\tau_{j,i} \in (0, 1)$, which measure the relative external information from the algorithm $j \neq i$ accepted by $\mathcal{A}^i$. All weights $\tau_{j,i}$ are stored in a square matrix of size $J \times J$ which therefore contains the strength of all collaboration links. Most notations used in this article are summed up in Table 1 below.

3.2. Problem formulation

Within the context of horizontal collaboration that we have presented before, the method that we propose takes many advantages of both prototype-based collaborative methods and the SAMARAH method, without their issues.

Our goal in this section is to find a way to modify Eq. (1) so that the collaborative term will not depend on the prototypes. Therefore, we propose a likelihood function based on Eq. (3) which uses a global consensus term $C(\boldsymbol{S})$ based on the partitions. The main differences with Eq. (1) are that we used a model based on partitions rather than prototypes, our proposed model is consensus based instead of divergence based, and we use a global term instead of a pairwise one. We chose this global model because unlike the pairwise version, it does not require to assume that the algorithms are independent from each other (which is of course not true).

In this model, $\lambda \in [0, 1]$ is a weight parameter to balance between the local and collaborative term. The left term $\sum_{i=1}^{J} L(X^i|S^i, \Theta^i)$ is called the *local term*, and the right term $\lambda \cdot C(\boldsymbol{S})$ is the *collaborative* term. Note that the $C(\cdot)$ here stands for "consensus": we have a collaborative term based on a consensus function.

$$(\boldsymbol{S}_{opt}, \boldsymbol{\Theta}_{opt}) = \underset{(\boldsymbol{S},\boldsymbol{\Theta})}{\text{Argmax}}\ L_g(\boldsymbol{S}, \boldsymbol{\Theta}) = \underset{(\boldsymbol{S},\boldsymbol{\Theta})}{\text{Argmax}} \sum_{i=1}^{J} L(X^i|S^i, \Theta^i) + \lambda \cdot C(\boldsymbol{S}) \tag{3}$$

With this model, and using a collaborative term based on different a posteriori distributions instead of a collaborative term based on distributions parameters, our proposed model lifts off the limitation that only identical algorithms looking for the same number of clusters can work together. Furthermore, using our model even non-parametric algorithms -for which the distributions parameter Θ^i can not be explicitly formulated- can be used in a collaborative setting since our model is based on the partitions (solution matrices or vectors) which are explicit for any clustering algorithm. The penalty factor $\lambda > 0$ regularizes the collaboration part. Please note that in [28], the authors have demonstrated that there is a direct relation between reducing the divergences and maximizing the consensus under mild assumptions. Therefore, both strategies are equivalent.

Analogously to Eq. (3), our idea is to optimize a modified fitness of the log-likelihood function that considers both the local partitions and the information coming from the other algorithms' solutions. By considering only the partitions S^i and not the parameters, very much like in the SAMARAH method [25,26], we ensure that our model is both generic.

As we will demonstrate in the next subsection, this change from Θ^i to S^i is made possible because we use an alternate maximization procedure in which the partitions are computed from the prototypes and then the prototypes are updated based on the partitions and the data. In short, the partitions can be seen as a discretization of the distributions described by the prototypes.

While this improvement will result in a more generic paradigm when it comes to horizontal collaboration, it is worth mentioning that removing the prototypes also makes vertical collaboration (algorithms collaborating on different data sets with similar clusters) impossible whereas some of the earlier methods covered this case of knowledge transfer between similar data sets [11,13,24], albeit only between identical algorithms.

To optimize (3) we use the Expectation Maximization (*EM*) strategy. The workflow in Algorithm (1) highlights how our algorithm can indeed be considered as an *EM* algorithm. During the E-Step, the partitions $\boldsymbol{S}$ are updated using fixed values for the distributions parameters $\boldsymbol{\Theta}$. Then, during the M-Step, these parameters $\boldsymbol{\Theta}$ are updated based on the new partitions.

The exact form of the functional L_g is explained in the next section, while the sopping criterion is detailed in Section 3.5.

Table 1
Notations.

Notation	Development	Comment
X^i	$X^i = \{x_1^i, \dots, x_N^i\}, x_n^i \in \mathbb{R}^d$	The subset of the data observed by algorithm $\mathcal{A}^i$
$\boldsymbol{X}$	$\boldsymbol{X} = \{X^1, \dots, X^J\}$	The full data with all views
Θ^i		The parameters describing the distributions observed by algorithm $\mathcal{A}^i$
$\boldsymbol{\Theta}$	$\boldsymbol{\Theta} = \{\Theta^1, \dots, \Theta^J\}$	The set of distributions parameters for all algorithms
$\mathcal{A}^i$	$\mathcal{A}^i = \{X^i, S^i, \Theta^i, K_i\}$	An algorithm looking for K_i clusters of distribution parameters Θ^i in the subset X^i and finding a partition S^i
$\tau_{j,\,i}$	$\tau_{j,\,i} \in [0,\,1]$	The weight of the collaboration from $\mathcal{A}^j$ to $\mathcal{A}^i$
$s_{n,c}^i$	$s_{n,c}^i \in (0,\,1), \quad \sum_{c=1}^{K_i} s_{n,c}^i = 1$	The responsibility given by algorithm $\mathcal{A}^i$ to the cluster $c \in [1..K_i]$ for the data x_n^i
S^i	$S^i = (s_{n,c}^i)_{K_i \times K_i}$	The partition found by algorithm $\mathcal{A}^i$. For fuzzy clusters, S^i is a matrix.
Z^i	$Z^i\colon \Omega \to [1..K_i]$	The latent random vector linked to the solutions of algorithm $\mathcal{A}^i$
$P(Z^i \mid X^i,\,\Theta^i)$		the a posteriori distribution of Z^i conditionnally to X^i and Θ^i
$\mathcal{H}$	See Eq. (16)	The global entropy of the collaborative system for all algorithms
$\omega_{a,b}^{i,j}$	$\omega_{a,b}^{i,j} = P(Z_n^j = b \mid Z_n^i = a, \boldsymbol{S}, \boldsymbol{X}, \boldsymbol{\Theta})$	The percentage of data associated to cluster a by $\mathcal{A}^i$ that belong in the cluster b of $\mathcal{A}^j$
$\boldsymbol{q}$	$\boldsymbol{q} = \{q_1, \cdots, q_J\}, \forall i \quad q_i \in [1..K_i]$	A combination of clusters (see Section 3.4)
$g^i(\boldsymbol{q},\,c)$	$g^i(\boldsymbol{q},\,c) \in (0,\,1),\, c \in [1..K_i]$	A consensus function assessing the likelihood of having $q_i = c$ knowing the rest of $\boldsymbol{q}$

Algorithm 1: Collaborative "EM".

Initialize, $t = 0$ and $\boldsymbol{\Theta}(0)$ with the local step
while *the global entropy* $\mathcal{H}$ *decreases* **do**
  E-Step: $\boldsymbol{S}(t) = \text{Argmax}_{\boldsymbol{S}}\, L_g(\boldsymbol{S}, \boldsymbol{\Theta}(t))$,
  M-Step: $\boldsymbol{\Theta}(t+1) = \text{Argmax}_{\Theta}\, L_g(\boldsymbol{S}(t), \boldsymbol{\Theta})$,
  $t = t + 1$
end
Return $\boldsymbol{S}(t)$

3.3. Objective function

The fundamental question in horizontal collaborative setting is to find the right functional to optimize so that we can properly answer the problem of having several algorithms working together by exchanging their information with a goal of mutual improvement. To do so, we have the following constraints: We want a functional similar to Eq. (3) based on the partitions instead of distributions prototypes, where we attempt to bias each local solution S_t^i so that S_{t+1}^i takes into account the information from the other partitions without using any prototypes. The problem therefore consists in finding the right local and collaborative terms.

Defining the local term is relatively easy and can be done using any kind of likelihood function for probabilistic algorithms, and ad-hoc normalized quality criterion for other types of algorithms. The literature is also full of potential divergence and consensus functions between partitions for the collaborative term that measure the divergence or consensus between two partitions (NMI, entropies, Rand Index, etc.). However, if we add the extra-constraint that the partitions are mostly non-binary and that Eq. (3) should be optimized in a reasonable amount of time, we face the following problem: For vector partitions of size N, most of these operators have a complexity in $O(N^2)$. Therefore, the final cost of updating all partitions for the J algorithms looking on average for $\bar{K}$ clusters would be equivalent to call these operators $J \times N \times \bar{K}$ times, hence a final complexity of $O(N^3)$ just to optimize the collaborative term.

Since such complexity obviously does not scale well, in the remainder of this section we explain how we re-designed a likelihood function from scratch using a solid probabilistic model. Then, in Section 3.4, we show how to optimize this new function with a low complexity of $O(N)$. Very much like in Eq. (3), we consider that the functional in the collaborative setting is decoupled into two different terms, the *local term* $L(\boldsymbol{S}, \boldsymbol{\Theta})$ computed from all local log-likelihood or quality indexes, and the *collaborative term* $C(\boldsymbol{S})$ in the form of a global consensus function between the partitions. More precisely the global likelihood function writes:

$$L_g(\boldsymbol{S}, \boldsymbol{\Theta}) = L(\boldsymbol{S}, \boldsymbol{\Theta}) + \lambda \cdot C(\boldsymbol{S}), \tag{4}$$

where $\boldsymbol{X}$ is the observed variable, $\boldsymbol{\Theta}$ the set of parameters and $\boldsymbol{S} = (S^1, \dots, S^J)$ is the set of all partitions.

In the first term L in Eq. (4), just as in Eq. (3), we express the log-likelihood of $\boldsymbol{S}$ based only on the local information and model of each algorithm taken individually and the data x_n. We evaluate then the log-likelihood of the completed sample against the a posteriori distribution of $(Z^i \mid X_n^i, \Theta^i)$.

$$L(\boldsymbol{S}, \boldsymbol{\Theta}) = \sum_{i=1}^{J} \sum_{n=1}^{N} P(Z_n^i \mid X_n^i, \Theta^i) \cdot \log P(X_n^i, Z_n^i \mid \Theta^i). \tag{5}$$

The second term of Eq. (4) is detailed in Eq. (6). It is computed from the likelihood that each element x_n be linked to the right cluster based on the other algorithms' partitions and the choice of cluster for the same data in the local view. The difference between the local likelihood and the likelihood based on the other algorithms gives us the collaborative term. This term $C(\boldsymbol{S})$ therefore is the likelihood of $\boldsymbol{S}$ based on all the solutions.

$$C(\boldsymbol{S}) = \sum_{i=1}^{J} \sum_{n=1}^{N} \left(P(Z_n^i \mid X_n \setminus X_n^i, \boldsymbol{S}) - P(Z_n^i \mid X_n^i, \Theta^i) \right) \cdot \log P(X_n^i, Z_n^i \mid \Theta^i) \tag{6}$$

Then using Eqs. (5) and (6) we obtain following a posteriori probability for the completed sample X_n^i, Z_n^i corresponding to algorithm $\mathcal{A}^i$:

$$\begin{aligned} P(Z_n^i = c \mid X_n^i, \Theta^i, \boldsymbol{S}) &= (1-\lambda) \cdot P(Z_n^i = c \mid X_n^i, \Theta^i) \\ &+ \lambda \cdot P(Z_n^i = c \mid X_n \setminus X_n^i, \boldsymbol{S}) \end{aligned} \tag{7}$$

Note that due to the lack of independence $P(Z^i \mid X_n \setminus X_n^i, \boldsymbol{S})$ is not tractable. Nevertheless, in the next section we show tractable update rules for the responsibilities.

3.4. Update rules

In this section, we will proceed with the practical description of the update rules for the responsibilities $s_{n,c}^i$ so that we can actually compute the partitions that are solutions of the functional from Eq. (7). For fuzzy clustering we then infer that the update rule for the responsibility for all data x_n and all cluster c from iteration t to iteration $t+1$ during the E-step of Algorithm (1) is the following:

$$\begin{aligned} s_{n,c}^i(t+1) = \; & (1-\lambda) \cdot s_{n,c}^i(t) \\ & + \lambda \cdot \sum_{\boldsymbol{q} \in Q \mid q_i = c} P(\boldsymbol{q} \mid X_n \setminus X_n^i, \boldsymbol{\Theta}_t \setminus \Theta^i(t)) \cdot P(Z_n^i = q_i \mid \boldsymbol{q}) \end{aligned} \tag{8}$$

The first term $s^i_{n,c}|_t$ comes from the local partition, and is actually given by the *a posteriori* probability $P(Z^i_n = c|x^i_n, \Theta^i(t))$ for the data x_n by using the Bayes rule.

The second term is a key element in this paper: we have J algorithm running parallel, and each of these algorithm can assign the data x_n to any cluster in $[1..K_i]$. Let $\boldsymbol{q} = \{q_1, \cdots, q_J\}, \forall i \quad q_i \in [1..K_i], \boldsymbol{q} \in Q$ be one combination of cluster chosen by the J algorithms among all possible sets of combinations Q. Based on these notations, the collaborative term assess the likelihood of such combination $\boldsymbol{q}$ for the data x_n based on all algorithms except the local algorithm $\mathcal{A}^i$, hence the notations $X_n \setminus X^i_n$ and $\boldsymbol{\Theta}_t \setminus \Theta^i(t)$. Then the collaborative term asses the probability of having $q_i = c$ knowing the rest of the combination $\boldsymbol{q}$. Since we are considering the case of fuzzy clustering, all possibles combinations in Q must be evaluated, hence the sum.

To sum up, the second term sums all possibles combinations of clusters $q \in Q$ where $q_i = c$, then assess the probability of such combination for the data x_n for the other algorithms. This probability is then multiplied by the probability of $q_i = c$ knowing the other elements of the combination q. We will approach this second probability using a consensus function $g^i(\boldsymbol{q}, c) \approx P(q_i = c|\boldsymbol{q})$. Since Q the set of all possible combination grows exponentially large with the number of algorithms, and because most of the combination probabilities are very close to 0, we make the simplification of only considering the most likely combination $\boldsymbol{q}^*_n = \text{Argmax}_q\, P(\boldsymbol{q}|X_n \setminus X^i_n, \boldsymbol{\Theta}_t \setminus \Theta^i_t)$.

Therefore the update rule (8) becomes:

$$s^i_{n,c}(t+1) = (1-\lambda) \cdot s^i_{n,c}(t) + \lambda \cdot g^i(\boldsymbol{q}^*_n, c) \tag{9}$$

where we remind that λ is a weight parameter between local and external information.

As one can see from Eq. (9), the discretization of our model leads to very simple update rules which require only the local likelihood proposed by each algorithm for the possibles clusters of each data, the partitions produced by all the algorithms, and a good combination function g^i. This combination function, through which the algorithms will collaborate, has the key role of assessing the likelihood of a local decision based on the other algorithms' partitions.

Since the M-Step of our proposed algorithm only used information from the local term of the functional, the update rules are identical to these of the local algorithm in their non-collaborative version. For instance, in the case of a Gaussian mixture model, the *mean, variance-covariance* and *mixing probabilities* of each clusters are computed using the usual rules.

3.4.1. Combination functions

In this Section we give some example of a particular class of "combination functions" that are tractable and can be used in our collaborative framework.

First, we want to begin by explaining the intuitive meaning of g^i as a consensus function: Given a partitioning problem processed in parallel by several algorithms (or a vote process in which several algorithms take part), $g^i(\boldsymbol{q}, c)$ assesses the consensus or degree of compatibility of a cluster c from the algorithm $\mathcal{A}^i$ with the group of clusters $\boldsymbol{q} = \{c_1, \cdots, c_j, \cdots, c_J\}, j \neq i$ from the other algorithms.

Definition 1. A function $g^i : \mathcal{Q} \times [1..K_i] \to [0, 1]$ is a combination function for the algorithm i if it satisfies:

1. $g^i(\boldsymbol{q}, c)$ needs to increase strictly between 0 and 1 when the consensus between the different algorithms grows on the likelihood of having $q_i = c$ for a given combination $\boldsymbol{q}$.
2. $g^i(\boldsymbol{q}, c)$ needs to be normalized so that for any cluster combination $\boldsymbol{q}$ that occurs at least once, we have: $\sum_{c \in [1..K_i]} g^i(\boldsymbol{q}, c) = 1$.
3. When the algorithms have the exact same partitions and $c = argmax_{q_i} s^i_{n,q_i}$, then: $g^i(\boldsymbol{q}^*_n, c) = 1$.

Note that the properties of the combination function are naturally satisfied by any marginal of a probability density function defined on latent space.

To be more precise on the computation and increasing property of g, let i be a fixed algorithm, be c a fixed cluster and $\boldsymbol{q}$ be a fixed cluster combination such that $q_i = c$. The value $g^i(\boldsymbol{q}, c)$ is computed by considering $\boldsymbol{S}$ the set of all partitions, in the following way: we compute the likelihood of $q_i = c$ with respect to all others choices q_j, $j \neq i$ for the cluster c and a given partition $S \in \boldsymbol{S}$. This likelihood is computed directly from the cardinality of the intersections of all involved clusters. We propose thereafter 3 possible combination functions abiding by the axioms exposed before. All have different strengths and weaknesses. They are shown in Eqs. (10)–(12).

$$g^i_\cap(\boldsymbol{q}, c) = \frac{|\bigcap_{j \neq i} q_i \cap q_j|}{|\bigcap_{j \neq i} q_j|}, \quad q_i = c \tag{10}$$

The formula from Eq. (10) assesses consensus between the local algorithm and the other algorithms divided by the consensus between the other algorithms. This combination function is the one that should be picked in absence of the independence hypothesis between the different algorithms. This combination function is normalized, However it is costly to compute due to the K^J possible intersections. It is also worthy to mention that this combination function does not allow to weight the influence of the different algorithms.

$$g^i_+(\boldsymbol{q}, c) = \frac{1}{B} \sum_{j \neq i} \tau_{j,i} \frac{|q_i \cap q_j|}{|q_j|} = \frac{1}{B} \sum_{j \neq i} \tau_{j,i} \cdot \omega^{j,i}_{q_j,q_i}, \quad q_i = c \tag{11}$$

In Eq. (11), making the hypothesis that all algorithms are independent, we compute the mean pairwise consensus between the partitions, and in (12) the geometric mean consensus. In both Equations, the $\tau_{j,i}$ are weights that can be set to different values in order to change the influence of the algorithms on each other, and B is a normalization constant that is needed to respect axiom 2. Both equations are based on the same PCM Matrices $\Omega^{j,i} = (\omega^{j,i}_{q_j,q_i})_{(K_j \times K_i)}$ from the SAMARAH method described in Eq. (2) and which are relatively cheap to compute. Beyond the fact that both combination functions require a normalization, g_* also has the issue that it always returns 0 whenever one of the intersection is null.

$$g^i_*(\boldsymbol{q}, c) = \frac{1}{B} \prod_{j \neq i} \left(\frac{|q_i \cap q_j|}{|q_j|} \right)^{\tau_{j,i}} = \frac{1}{B} \prod_{j \neq i} \left(\omega^{j,i}_{q_j,q_i} \right)^{\tau_{j,i}}, \quad q_i = c \tag{12}$$

Given that all 3 combinations functions have their pros and cons, picking one is context dependent. For instance, $g^i_\cap$ certainly is the most interesting one to have a global consensus combination function, but should be avoided with a large number of collaborators due to its complexity and is unpractical when weighting the collaborators is a requirement. Then g^i_* has a behavior that is very close from $g^i_\cap$ with less computational complexity. Another advantage of g^i_* is that it has a Bayesian interpretation if we assume the hypothesis that all partitions are independent. On the other hand g^i_+ behaves a bit differently but it will not tend as fast towards zero when one or more intersections are null. Furthermore, both g^i_+ and g^i_* scale better with a large number of collaborators. Further discussions on the complexity of these functions are available in section 3.5, and some experimental results are shown in Section 4.1.

Finally, as one can see, the 3 combination functions are in practice based solely on the local clustering partitions and can be used regardless of the type of algorithm and the number of clusters it is searching for. This property is fundamental in the sense that it lifts off the previous limitations of collaborative frameworks allowing only algorithms of the same kind to work together and forcing them to search for the same number of clusters. Using these parti-

tion consensus functions is therefore a key element in making our method more generic than the previous ones.

3.4.2. Algorithmic complexity

We now want to discuss the complexity of our proposed method. To this end, let us consider J collaborators looking on average for K clusters and working in together on a data set of size N. Then, we have:

$$cpx = J \times \left(\overline{cpx(\mathcal{A}(N,K))} + N \times cpx(g)\right) \tag{13}$$

where $\overline{cpx(\mathcal{A}(N,K))}$ is the average complexity of the collaborators and $cpx(g)$ is the complexity of the chosen combination function.

All three combinations functions have a complexity in $O(J \times N)$. However, using g_+ and g_*, the combinations functions' values can be computed only once at the beginning of each iteration and stored in an array of size $J^2 \times K^2$ instead of being computed on the flight for each of the N data. Using this technique, the right complexity term involving the combination function disappears. However, this is not an option with the function $g_\cap$ where at best $J \times K^{J-1}$ values would have to be computed and stored.

Therefore, using g_+ or g_* while storing the values in memory, the best possible complexity is:

$$cpx_{min} = J \times \overline{cpx(\mathcal{A}(N,K))} + O(J \times N) \tag{14}$$

Otherwise, we have:

$$cpx_{max} = J \times \overline{cpx(\mathcal{A}(N,K))} + O(J^2 N^2) \tag{15}$$

To conclude on the complexity of our proposed method: In the less favorable scenario using $g_\cap$ without the independence hypothesis, or using a suboptimal version of g_+ or g_*, the collaboration adds complexity term in $O(J^2N^2)$. This term is therefore only negligible when using algorithms the complexity of which is superior or equal to $O(N^2)$.

However, in the best case scenario using the optimized version of g_+ or g_* with the memory trade off, the collaboration adds a linear complexity term in $O(N)$. Considering that the best clustering algorithms also have a linear complexity, the loss of performance is negligible when compared with using the original clustering algorithms in parallel. Therefore, using fast algorithm, we can get a complexity in $O(N)$ to optimize the functional in Eq. (7).

3.5. Stopping criterion

The stopping criterion used by our algorithm is the probabilistic confusion entropy [29,30] as shown in Eq. (16) bellow:

$$\mathcal{H} = \sum_{i=1}^{J} \sum_{j \neq i}^{J} \frac{-1}{K_i \times \log(K_j)} \sum_{l=1}^{K_i} \sum_{m=1}^{K_j} \omega_{l,m}^{i,j} \log(\omega_{l,m}^{i,j}) \tag{16}$$

This entropy assess the pairwise divergences between the algorithms, and is equal to 0 when all algorithms have identical partitions, and 1 when there is a full disagreement. In short, $\mathcal{H}$ is the system global entropy under the conditions that all algorithms are independent. We chose to use this entropy because it uses the $\omega_{l,m}^{i,j}$ from the probabilistic confusion matrix in Eq. (2) that we already compute for two of our combination functions g. As such, the entropy $\mathcal{H}$ is much less costly to compute than any other divergence or consensus measure in the literature.

The justification that this entropy is a good stopping criterion is the following: from Eq. (6), we know that the collaboration of algorithm $\mathcal{A}^i$ with all the others collaborators can be measured by the difference between the cross entropy of the two distributions $P(Z^i|X \backslash X^i, \boldsymbol{S})$ and $P(Z^i|X^i, \Theta^i)$, and the entropy of the distribution $P(Z^i|X^i, \Theta^i)$. Therefore, the collaborative term is oppositely proportional to the system global entropy $\mathcal{H}$. From there, since we use an EM-like optimization process the form of which is a local term minus a difference of two entropies , we know from the proof of the variational EM [31] that both involved entropies increase strictly, and therefore that their difference decreases. As such, the global entropy $\mathcal{H}$ is a is a valid stopping criterion. Furthermore, this type of entropic criterion is consistent with earlier studies that have shown the importance of diversity and entropy in collaborative clustering [32–34].

3.6. Setting the weights parameters

We now want to discuss the role of the weighting parameter $\tau_{j,i}$. These parameters weight the strength of the collaborative link from an algorithm $\mathcal{A}^j$ to an algorithm $\mathcal{A}^i$, and ultimately they determine the value of the parameter λ_i used as a weight between the local and the collaborative term.

There are several techniques to set up these weights:

- Arbitrarily setting the same value for all weights. While this is not the best method to avoid negative collaboration, it is certainly the lest computationally expensive one and it is widely used in the literature [13,17,20]. It is this method that we used in this paper.
- Using expert knowledge to set them up, for instance using quality and diversity criterion between the solutions [35]. This method can prove useful when expert knowledge is available or specific shapes are expected for the clusters, but it is biased towards certain types of algorithms.
- Searching the weights that optimize the collaborative term when the partitions and parameters are fixed [24]. This method is very effective at reducing the risks of negative collaborations because it tends to favor the most stable solutions. However, it is also known to favor collaborations between already similar partitions, which also tend to reduce the overall performances.

4. Experimental results

Our experimentation will be separated in 4 distinct parts: in the first part we will demonstrate a practical calculation of the 3 combinations functions g using an artificial data set with the goal of showing how the calculus is done in practice and also to demonstrate that all functions have a similar behavior. In the second part, a second experiment is proposed, in which we show the performances of our proposed method in term of collaborative power. In the third part, we show two comparative experiments: First, comparison of our method with other state of this art collaborative and multi-view frameworks. And second, we propose an application of our method for the multi-scale analysis of image data in which we compare it with non-collaborative algorithms. Finally in part 4, we show the average computation times of our methods under various parameters.

Offer settings that the other methods do not.

4.1. Example of empirical calculi with the combination functions

Let us consider an artificial data set X containing 81 observations. We suppose that 3 algorithms are working on a multi-view analysis of this data set, each of them searching for 2 clusters. In Fig. 1, we show the partitions found by each algorithm in a 2-dimension projection that is very convenient to visualize the problem.

The first algorithm (in red on the figure) is searching for two clusters $\{a', a''\}$, the second algorithm (in blue) is searching for the clusters $\{b', b''\}$ and the third (in green) for $\{c', c''\}$. Due to the multi-view nature of this experiment, we can see that they find very dissimilar partitions.

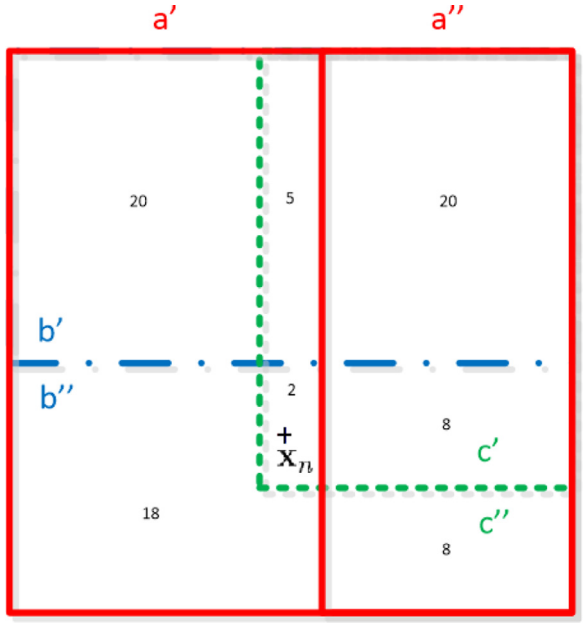

Fig. 1. 2-dimension projection of 3 partitions of 2 clusters each, on a 81 observations data set. The small numbers in the figure highlight the number of data in each intersection of clusters.

Table 2
Example of results for different combination functions.

$g_{\cap}^{red}(\boldsymbol{q}, a') = \frac{|a' \cap b'' \cap c'|}{|b'' \cap c'|} = \frac{2}{10} = 0.2$
$g_{\cap}^{red}(\boldsymbol{q}, a'') = \frac{|a'' \cap b'' \cap c'|}{|b'' \cap c'|} = \frac{8}{10} = 0.8$
$g_{+}^{red}(\boldsymbol{q}, a') = \frac{1}{B}\left(\frac{|a' \cap b'|}{|b'|} + \frac{|a' \cap c'|}{|c'|}\right) = \frac{1}{2}\left(\frac{20}{36} + \frac{7}{35}\right) \approx 0.38$
$g_{+}^{red}(\boldsymbol{q}, a'') = \frac{1}{B}\left(\frac{|a'' \cap b''|}{|b''|} + \frac{|a'' \cap c'|}{|c'|}\right) = \frac{1}{2}\left(\frac{16}{36} + \frac{28}{35}\right) \approx 0.62$
$g_{*}^{red}(\boldsymbol{q}, a') = \frac{1}{B}\left(\frac{|a' \cap b''|}{|b''|} \times \frac{|a' \cap c'|}{|c'|}\right) = \frac{1}{2}\left(\frac{20}{36} \times \frac{7}{35}\right) \approx 0.24$
$g_{*}^{red}(\boldsymbol{q}, a'') = \frac{1}{B}\left(\frac{|a'' \cap b''|}{|b''|} \times \frac{|a'' \cap c'|}{|c'|}\right) = \frac{1}{2}\left(\frac{16}{36} \times \frac{28}{35}\right) \approx 0.76$

Table 3
Example of results for different combination functions.

$g_{\cap}^{blue}(\boldsymbol{q}, b') = \frac{|a' \cap b' \cap c'|}{|a' \cap c'|} = \frac{5}{7} = 0.71$
$g_{\cap}^{blue}(\boldsymbol{q}, b'') = \frac{|a'' \cap b'' \cap c'|}{|a' \cap c'|} = \frac{2}{7} = 0.29$
$g_{+}^{blue}(\boldsymbol{q}, b') = \frac{1}{B}\left(\frac{|a' \cap b'|}{|a'|} + \frac{|b' \cap c'|}{|c'|}\right) = \frac{1}{B}\left(\frac{25}{45} + \frac{25}{35}\right) \approx 0.63$
$g_{+}^{blue}(\boldsymbol{q}, b'') = \frac{1}{B}\left(\frac{|a' \cap b''|}{|a'|} + \frac{|b'' \cap c'|}{|c'|}\right) = \frac{1}{B}\left(\frac{20}{45} + \frac{10}{35}\right) \approx 0.37$
$g_{*}^{blue}(\boldsymbol{q}, b') = \frac{1}{B}\left(\frac{|a' \cap b'|}{|a'|} \times \frac{|b' \cap c'|}{|c'|}\right) = \frac{1}{B}\left(\frac{25}{45} \times \frac{25}{35}\right) \approx 0.76$
$g_{*}^{blue}(\boldsymbol{q}, b'') = \frac{1}{B}\left(\frac{|a' \cap b''|}{|a'|} \times \frac{|b'' \cap c'|}{|c'|}\right) = \frac{1}{B}\left(\frac{20}{45} \times \frac{10}{35}\right) \approx 0.24$

In Fig. 1, we are interested in the data x_n which has been assigned to a', b'' and c' by the 3 algorithms respectively. Let us suppose now, that we use our combination function g to see whether or not the decision of the first algorithm to put x_n in the cluster a' makes consensus with the partition of the two other algorithms which put it in b'' and c'. In Table 2, we show how to practically use the intersections of the clusters to compute $g^i(\boldsymbol{q}, a')$ and $g^i(\boldsymbol{q}, a'')$ with all 3 combination functions that we have introduced earlier (using $\tau = 1$). The same experiment is done in Tables 3 and 4, to check the consensus on b'/b'' and c'/c'' respectively for the same data x_n.

The results are interesting in several ways:

- First we have the confirmation that all 3 functions roughly behave the same way and agree on the same most consensual clusters.
- We can observe the complementary relationships between the different intersections.

Table 4
Example of results for different combination functions.

$g_{\cap}^{green}(\boldsymbol{q}, c') = \frac{|a' \cap b'' \cap c'|}{|a' \cap b''|} = \frac{2}{20} = 0.1$
$g_{\cap}^{green}(\boldsymbol{q}, c'') = \frac{|a' \cap b'' \cap c''|}{|a' \cap b''|} = \frac{18}{20} = 0.9$
$g_{+}^{green}(\boldsymbol{q}, c') = \frac{1}{2}\left(\frac{|a' \cap c'|}{|a'|} + \frac{|b'' \cap c'|}{|b''|}\right) = \frac{1}{2}\left(\frac{7}{45} + \frac{10}{36}\right) \approx 0.22$
$g_{+}^{green}(\boldsymbol{q}, c'') = \frac{1}{2}\left(\frac{|a' \cap c''|}{|a'|} + \frac{|b'' \cap c''|}{|b''|}\right) = \frac{1}{2}\left(\frac{38}{45} + \frac{26}{36}\right) \approx 0.78$
$g_{*}^{green}(\boldsymbol{q}, c') = \frac{1}{2}\left(\frac{|a' \cap c'|}{|a'|} \times \frac{|b'' \cap c'|}{|b''|}\right) = \frac{1}{2}\left(\frac{7}{45} \times \frac{10}{36}\right) \approx 0.07$
$g_{*}^{green}(\boldsymbol{q}, c'') = \frac{1}{2}\left(\frac{|a' \cap c''|}{|a'|} \times \frac{|b'' \cap c''|}{|b''|}\right) = \frac{1}{2}\left(\frac{38}{45} \times \frac{26}{36}\right) \approx 0.93$

- We have an empirical confirmation that g_*^i is a very good approximation of $g_{\cap}^i$, while g_+^i leads to more flexible results.

What we can observe from this experiment is that the combinations functions that we have proposed serve their intended purpose and encourage changing the partitions in a way that will lower the diversity between the algorithms' solutions. According to Tables 2–4, the data x_n should be moved to cluster a'', b' and c'' in order to increase the consensus.

However, one should keep in mind that these changes may only happen if the collaborative term is strong enough compared with the local term which we do not mention in this experiment.

In the light of this first experiment that complete the study on the complexity of our algorithm performed in Section 3.5, we think that the function $g_{\cap}^i$ should be favored when the data set is small enough to do the computations in a reasonable amount of time, and that otherwise g_*^i should be favored over g_+^i because it is the best approximation.

4.2. Multi-view collaborative clustering experiments

4.2.1. Experimental setting

In this experiment, we propose to evaluate our framework for collaborative multi-view clustering task. To this end, our experimental setting is the following: We considered the VHR Strasbourg (9 clusters), the Images (7 clusters), the WDBC (2 clusters) and the Spam Base (2 clusters) data sets in a multi-view setting where their attributes were split between different algorithms.

In the list bellow we explain how we created our views by splitting the data sets depending on their attributes.

- For WDBC: one view with only cell 1 (attributes 1–10), one view with only cell 2 (attributes 11–20), one view with only cell 3 (attributes 21–30), three views combining theses (cells 1 and 2, 1 and 3, 2 and 3). All 6 mentioned combinations using only appearance attributes only (texture, smoothness, compactness, fractal dimension), or only geometric attributes only (radius, perimeter, area, concavity, concave points, symmetry).
- For Image Segmentation: Region based attributes (attributes 1–9), local attributes (attributes 10–19), region-based + red attributes (attributes 1–9, 11 and 14), region-based + green attributes (attributes 1–9, 13 and 16), region-based + blue attributes (attributes 1–9, 12 and 15), raw attributes only (attributes 1–5, 11, 12 and 13), post-processed attributes only (6–10 and 14–19).
- For Spam Base: Any random combination of 19 attributes among the 57 total attributes was considered a view.
- For VHR Strasbourg: Geometric attributes only, radiometric attributes only, comparison with neighboring segments only, color attributes only (redundant with radiometric and comparison attributes), saturation and texture attributes only (also redundant with radiometric and comparison attributes).

We invite you to see appendix A for details on the data sets.

Given this setting, each view was first processed individually by a clustering algorithm (local step), and then they were all pro-

Table 5
Multi-view collaboration improvement results on internal indexes.

Data Set	Simulations	Silhouette Index		DB-Index	
		Average Improvement	Min/Max	Average Improvement	Min/Max
WDBC	100×10	$\mu = 0.122$ $\sigma = 0.057$	+0.241 −0.09	$\mu = -0.013$ $\sigma = 0.011$	+0.042 −0.036
Imgseg	100×7	$\mu = 0.091$ $\sigma = 0.020$	+0.133 −0.017	$\mu = 0.102$ $\sigma = 0.071$	+0.251 −0.101
Spam Base	100×5	$\mu = 0.037$ $\sigma = 0.026$	+0.081 −0.122	$\mu = 0.165$ $\sigma = 0.129$	+0.440 −0.106
Battalia3	100×3	$\mu = 0.025$ $\sigma = 0.004$	+0.092 −0.147	$\mu = 1.375$ $\sigma = 0.327$	+1.793 −1.184
MV2	100×4	$\mu = 0.04$ $\sigma = 0.008$	+0.083 +0.01	$\mu = 0.45$ $\sigma = 0.08$	+1.320 +0.02
VHR Strasbourg	35×5	$\mu = 0.027$ $\sigma = 0.005$	+0.037 −0.049	$\mu = 0.377$ $\sigma = 0.047$	+0.475 −0.094

cessed using our proposed collaborative method. To assess the efficiency of our method, we measured the results of 2 internal index and 1 external index before and after the collaborative step so that we could see whether or not the collaboration was beneficial. The indexes used are the Davies–Bouldin index [36] and the Silhouette index [37] for the internal indexes and the Adjusted Rand Index [38,39] for the external index.

We justify the use of two internal indexes because they do not assess the same things: The Silhouette Index assesses whether or not each data is on average closer to the data from its own cluster than from the data of the other clusters, while the Davies-Bouldin index is a more direct measure of the compactness of the clusters around their centroids and whether or not they are well separated.

This experiment was conducted with all collaborators having the same collaboration weights ($\lambda = 1 - \frac{1}{J}$).

The algorithms used in the collaboration process were a mix of Fuzzy C-Means algorithms, EM algorithms for the Gaussian Mixture Model, plus the GTM algorithm [40] for Spam Base, and the SR-ICM algorithm [41] for the VHR Strasbourg data set. These algorithms were chosen for several reasons:

- They have a random initialization which makes them non-deterministic and therefore interesting both from a collaboration point of view, and also to run a larger number of simulations without using always the same solutions.
- They all have a solid convergence proof and will not hinder the convergence of the collaborative process.
- Even if they don't use the same prototypes and cannot exchange directly on a prototype level, the fact that they are all prototype based makes it possible to use them directly in our collaborative framework without having to adapt them first.
- In the case of the SR-ICM algorithm, it is one of the few available algorithm specialized in pre-segmented high resolution satellite images.

4.2.2. Results

In Table 5, we show the change in the internal indexes before and after collaborations. For readability purposes, the sign of all variations for the Davies-Bouldin index have been inversed so that all positive values mean improvement for both indexes. The results for the change in the Adjusted Rand Index are shown in Table 6. In both tables, we indicate how many simulations were done, and the number collaborators is displayed in the "Simulations" columns.

For all indexes, we indicate the average improvement and its standard deviation, as well as the range of change in the considered indexes in the "Min/Max" column where we show the best improvement and worst deterioration achieved over all simulations.

The first striking result from Table 5 is that the gain for the internal quality indexes (Silhouette and Davies–Bouldin) has a lot of

Table 6
Multi-view collaboration improvement results on the adjusted Rand Index.

Data Set	Simulations	Adjusted Rand Index	
		Average Improvement	Min/Max
WDBC	100×10	$\mu = 2\%$ $\sigma \approx 0$	+3% −2%
Imgseg	100×7	$\mu = -2\%$ $\sigma \approx 0$	+5% −2%
Spam Base	100×5	$\mu = 10\%$ $\sigma = 6\%$	+22% −4%
Battalia3	100×3	$\mu = 6\%$ $\sigma = 1\%$	+9% −2%
MV2	100×4	$\mu = -3\%$ $\sigma \approx 0$	+2% −6%
VHR Strasbourg	35×5	$\mu = -8\%$ $\sigma = 5\%$	+6% −20%

variations. The explanation lies in the fact that while our proposed framework aims at improving all the results, in practice the best collaborators' results are often negatively impacted by weaker algorithms. Nevertheless, we can see that the collaboration results for the Silhouette and Davies–Bouldin indexes remain positive on average, which tends to prove the robustness of our proposed collaborative Framework.

The second point highlighted by this experiment and that is very obvious in Table 6 is that our proposed collaborative framework does not solve the issue of achieving good results on external indexes (the Adjusted Rand Index here) with purely unsupervised clustering algorithms. The weaker performances achieved on the Adjusted rand index can be explained by two factors:

First, without external knowledge, there is no reason for the collaborative process to converge toward the ground truth. The idea of adding external knowledge into our collaborative process may be considered in our future works. Second, in the case of the VHR Strasbourg data set, the ground expert truth contains 15 clusters covering only 90% of the data set, several of them very unlikely to be found by a clustering algorithm. As a consequence, the collaborative process only worsened the situations where the clustering algorithms found only a reduced number of clusters, therefore boosting indexes such as the Davies–Bouldin index -which is very high for this data set- while severely worsening the results on the Adjusted Rand Index.

4.3. Comparison with other methods

4.3.1. Comparison with other collaborative algorithms

In this section, we propose a comparison with other algorithms from the literature: we compare our method using several EM algorithms for the Gaussian mixture model collaborating together (with g_+ and $\lambda = 0.5$) with the multi-view EM algorithm, the col-

Table 7
Experimental results.

Dataset	Our Model		MV-EM		GTM_{collab}		SOM_{collab}	
	Rand	DB	Rand	DB	Rand	DB	Rand	DB
Wdbc (2 clusters)	95.50	0.85	92.30	0.97	96.57	0.9	**97.08**	**0.84**
SpamBase (2 clusters)	**86.77**	0.94	74.69	1.27	83.79	0.92	84.27	**0.87**
Battalia3 (6 clusters)	**80.00**	**2.43**	77.37	2.83	78.04	2.68	78.75	2.51
MV2 (4 clusters)	**94.32**	**1.34**	93.72	**1.34**	89.61	1.61	90.21	1.44
VHR Strasbourg (9 clusters)	**74.56**	**2.89**	73.37	3.21	68.97	4.15	70.14	3.78

laborative SOM algorithm [42] and the collaborative GTM algorithm [24].

All methods are used in a setting similar to the previous paragraph: the data sets are split in several views and each collaborative model is applied to all the views. Then, in Table 7, we show the averable results achieved after collaboration for the Rand Index (Rand) and the Davies–Bouldin Index (DB). We remind that the Rand index is better when it is close to 1, and that the Davies–Bouldin index is not normalized and better when smaller.

As one can see, when comparing our proposed method with the Multi-view EM we can see that our method achieves better results. This is interesting because the only difference between our method and theirs is that the MV-EM is based on prototypes and our method is based on partitions. This proves the efficiency of our method. Regarding the other two methods, we can see that we achieve comparative results: prototype based algorithms do better on the WDBC and SpamBase dataset, and we do better with the other datasets.

However, please note that comparing collaborative algorithms is very difficult and that these experiments may not be very significant to determine which method is more effective: for instance both the collaborative SOM and GTM algorithms only allow pairwise communication during the collaboration process, while in our method and in the multi-view EM all algorithms communicate at the same time. Another difference is that unlike our method, the 3 others have objective functions using prototypes instead of partitions which makes communication easier between algorithms but also restricts the collaboration between similar algorithms looking for the same number of clusters, hence why we decided to use only EM algorithms for our methods in order to have settings as similar as possible. Furthermore, the collaborative SOM and GTM algorithms computes topographic maps and not directly clusters. The partition can only be found by using the K-Means or EM algorithm on the final map, thus affecting the performances of both methods. Finally, when comparing a collaborative EM algorithm to a collaborative GTM algorithm, one can wonder if it is really the collaboration process that is evaluated, or the efficiency of the EM algorithm versus the GTM algorithm. For these reasons, the results of this section have to be taken with caution.

4.3.2. Multi-scale collaborative clustering experiments

4.3.2.1. Experimental setting. In this section, we propose an experiment in which we use our proposed collaborative framework for hierarchical clustering purposes. In very high resolution satellite images, depending on the scale there may be different types of elements of interest: At the first level, we can usually distinguish three main types of objects, namely water areas, vegetation areas and urban areas. At a second level we can separate different types of urban blocs, different types of vegetation areas, and start to distinguish elements such as roads. When zooming even more, very high resolutions images enable detecting small urban elements such as individual houses, cars, trees, or swimming pools.

As one can see, there is an obvious hierarchical relationship between the different objects of interests that can be detected when searching for different numbers of clusters. However, the huge size of these data sets usually makes them ineligible for hierarchical clustering algorithms because of their high computational complexity. We therefore propose an experiment in which we use our collaborative Framework on several instances of the previously mentioned SR-ICM algorithm searching for 3, 6 and 9 clusters with access to all attributes. In this experiment, we use the g_* combination function and $\lambda = \frac{1}{2}$.

Table 8
Experimental results: Hierarchical collaborative clustering.

Algorithm	Davies–Bouldin Index	Rand Index
EM 3	2.36928	0.67454
SR-ICM 3	2.32855	**0.67606**
Co SR-ICM 3	**2.32674**	0.67435
EM 6	2.88014	0.75867
SR-ICM 6	2.67816	0.76935
Co SR-ICM 6	**2.49726**	**0.77068**
EM 9	2.62786	0.78225
SR-ICM 9	2.94065	0.79063
Co SR-ICM 9	**2.58836**	**0.792187**

In our experiment, we compare our results with these of two other algorithms: the *EM* algorithm for the Gaussian Mixture Model [43], and the regular SR-ICM algorithm [41] both looking for 3, 6 and 9 clusters. In Fig. 2, we show the hierarchical clusters that we expected to find. The goal of the experiment is to demonstrate that our proposed collaborative method performs as best or better than local clustering methods working individually at different scales, but also that is these hierarchical structures will be reflected in the PCM matrices, and that the collaborative process will strengthen them.

The results were assessed using the Davies–Bouldin index as a an internal criterion. This index assesses the compactness of the clusters and how well they are separated. It is worth mentioning that the Davies–Bouldin index usually gives better results with less clusters. As for the external index, we used the Rand Index to compare our results with the expert ground truth.

4.3.2.2. Results. The results of this experiments over a dozen simulations for each algorithm are shown in Table 8, where the best result for each number of cluster is highlighted in bold.

As one can see, once again the results are non conclusive with the Rand Index where our method is not significantly better than the other. This was to be expected for the reason that like in the previous experiment, our collaborative framework does not have access to the expert ground truth and therefore cannot be expected to improve external indexes. However, we can see that we perform better than the other methods on the 6 and 9 clusters scale, with a much higher level of significance. The slightly lower performance on the 3 clusters scale can be explained by the fact that our collaborative approach mimics hierarchical clustering both ascending and descending since the collaboration goes both ways. However the descending approach is far more beneficial to get a good hierarchy leading to spherical clusters centered around the mean of

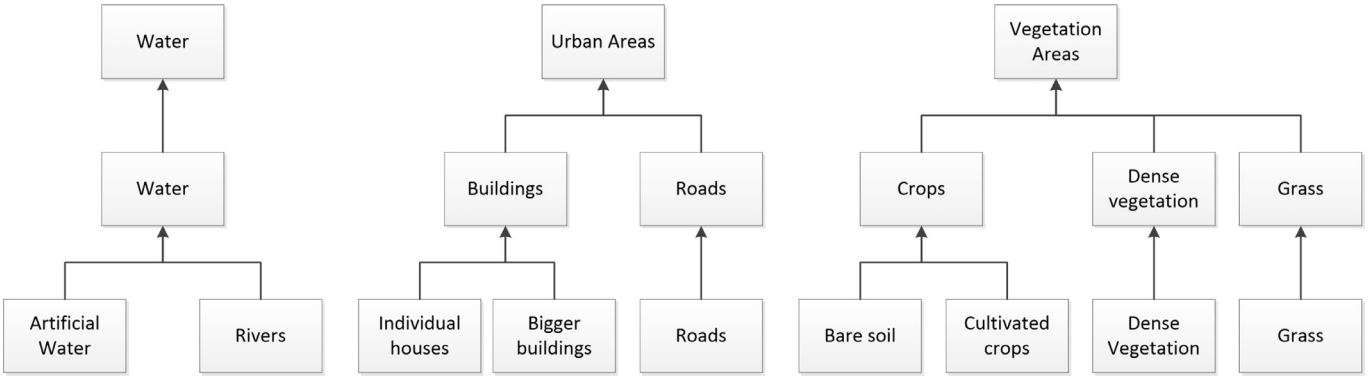

Fig. 2. Expected hierarchical clusters.

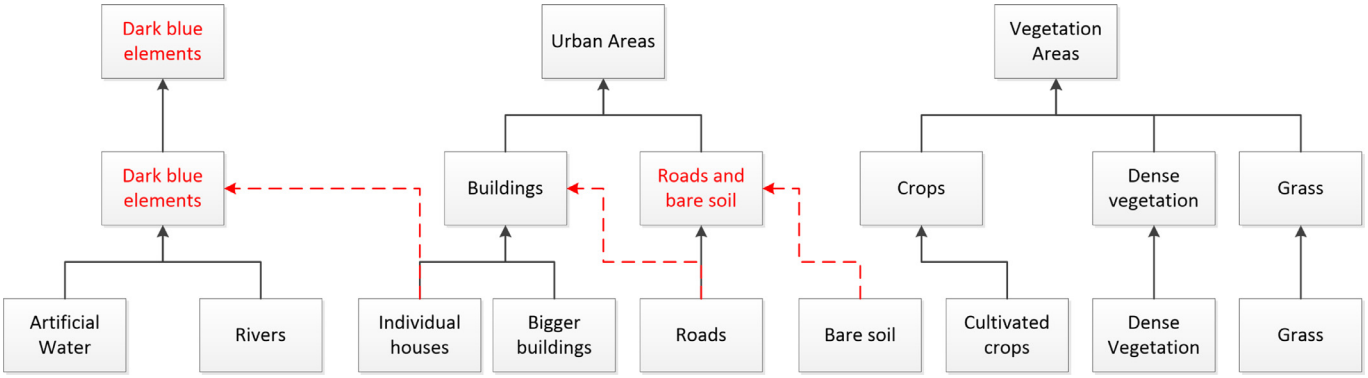

Fig. 3. Found hierarchical clusters.

the parent cluster at the upper scale, and therefore our methods works much better at scales with more clusters.

Finally, if when comparing the results with these of Table 7, an interesting remark is that the multi-scale approach with all the data as we did it in this experiments leads to better results that the multi-view approach of the previous experiment. This was to be expected since having access to all the data plus different scales of clusters leads to more information than just collaborating on partial views of the data.

In Fig. 3, we show the hierarchical structures extracted from the PCM matrices of our method. As one can see, there are some differences with the expected clusters from Fig. 2, that we have highlighted in red. In particular, we note that the hierarchical structure is not perfect with some classes covering each other. More interestingly there seem to be a confusion several blue elements of the image (namely individual houses with blue roofs and water) which may hint that the color attributes remain the dominant ones in the formation of the clusters when not using a multi-view approach.

4.4. Computation times

In Table 9, we show the average computation times achieved by our algorithm with different data sets given different numbers of collaborators and the two types of combination function. We used a C++ implementation of our method, running on a i5-3210M 2.5 GHz processor under a 64 bits version of Microsoft Windows 8. The collaborative framework was not parallelized during these test runs, and the computation time are including both the computations times of local step and the collaborative step.

In all experiments including this one, the collaborative step of our proposed method takes on average 8–10 iterations before reaching a stable global entropy. This number gets slightly lower when there are only 2 or 3 collaborators with very close solutions at the end of the local step, but remains mostly stable when the number of collaborators or the diversity between the initial solutions increases.

Table 9
Computation times.

Data Set	Computation time / number of collaborators				
	2	3	5	7	10
WDBC $g_{\cap}$	7s	15s	45 s	77s	3 min
ImgSeg $g_{\cap}$	2 min	5 min	13 min	27 min	52 min
Spam base $g_{\cap}$	6 min	17 min	42 min	1 h 38	4h
WDBC g_*/g_+	2 s	3 s	6 s	8 s	13 s
ImgSeg g_*/g_+	9 s	16 s	27 s	34 s	46 s
Spam base g_*/g_+	56 s	1 min 25	2 min 23	3 min 18	4 min 27
VHR Strasbourg g_*/g_+	24 min	37 min	1 h 04	1 h 28	2 h

As one can see in Table 9, the g_* combination function is much faster than the exact combination function $g_{\cap}$, and the computation times then increase with the number of clusters and the complexity of the data sets. Please note, that the computations times for $g_{\cap}$ with the VHR Strasbourg data set are not complete due to overly long computation times.

5. Conclusion

In this article, we have proposed a new collaborative framework that enables various algorithms to mutually improve their results. Our main contribution is that our proposed method allows algorithms of different types to work together regardless of the number of clusters they are searching for. The strength of our approach is that it needs neither the subsets, nor the prototypes, or the models used by the different algorithms to be shared during the collaboration step: only the solution vectors produced by all algorithms need to be shared.

Our framework is therefore more generic than previously proposed methods for horizontal collaboration in a sense that it has much less restrictions in terms of which algorithms can collaborate together. The cost of this more generic context is that our method cannot deal with vertical collaboration whereas some early methods could.

The optimization process behind our method is based on the variational EM, and optimizes a collaborative term which is equivalent to an entropy, thus ensuring good convergence properties.

Our framework has been tested on several data sets in a multi-view, a multi-experts and a multi-scale collaborative clustering contexts. Our results have validated the efficiency of our approach in bringing improvements to clustering solutions via collaboration. Furthermore, these experiments have highlighted that our method can find a large number of applications such as multi-view clustering, clustering of distributed data and hierarchical multi-scale clustering.

In our future works, we will focus on improving the overall collaboration process by weighting differently the influence of the different collaborators towards one another depending on quality and diversity measures. By doing so our goal will be to reduce cases of negative collaboration.

Acknowledgement

This work has been supported by the ANR project COCLICO, ANR-12-MONU-0001.

Appendix A. Data sets

A1. VHR Strasbourg data set

The VHR Strasbourg[1] data set [44] contains the description of 187.058 segments extracted from a very high resolution satellite image of the French city of Strasbourg. The original image covers an area of approximately 4×5 km with one pixel being equivalent to a $(10cm)^2$ area. The original image then went through the following process:

- It was corrected for distortion effects and issues due to the satellite angle.
- Then, a first segmentation was done using the software eCognition.
- The first segmentation was corrected to merge small neighboring segments that were too similar. This operation led to the 187.058 final segments.

Each segment is described by numerical 27 attributes, as well as a 28th column containing the IDs of neighbor segments. The first 27 attributes contain different types of features describing the segments: Radiometric features from the original image (brightness, colors, hue, saturation, min/max pixel values, etc.), geometric features (shape, size, coordinates, skewness, orientation, border length, circular mean, etc), comparison features with neighboring segments (contrast, number of brighter objects, number of darker objects, min difference to neighbors, etc.).

As one can see, this data set is good for multi-view clustering by construction due to the different types of available features. But because of the very high resolution of the image, it can also be used for multi-scale clustering. Indeed different scales of interest are available: From only 3 clusters (vegetation, water areas and urban area), to a large number of clusters on urban elements (trees, cars, individual pools, roads, individual houses, etc.). All scales in between can be studied depending on the considered number of clusters.

Finally, we would like to mention that the VHR Strasbourg data set was provided with a partial hybrid ground truth containing 15 classes [41]. The process to build the ground truth was the following:

- Expert geographers determined 15 classes of interest.
- Using on-field observations, Google Maps and city plans, they labeled a high resolution map of the city.
- The expert map was projected on the VHR Strasbourg segmentation so that each segment was given a label using a majority vote based on percentage of covering.
- The 10% of the segments being on the German side of the border are not covered by this ground-truth.

While we are ware that this hybrid ground-truth is not without flaws, the visual results seemed good enough to use it as a reference for our external indexes when conducting experiments using the VHR Strasbourg data set. Furthermore, from the 15 original classes we merged a few that could not possibly be detected by a unsupervised algorithm (e.g. winter crops and summer crops, more than 50 ha vegetation and more than 10 ha vegetation, etc.) and ended up with 9 classes.

A2. Other data sets

Several data sets used in this article are from the UCI repository [45]:

- *Wisconsin Diagnostic Breast Cancer* (WDBC): This data set contains 569 instances having 30 parameters and 2 classes. These 30 parameters contain 10 descriptors for 3 different cells of the same patient. And these descriptors can themselves be split into geometric and other appearance based attributes, therefore making this data set also good for multi-view.
- *Image Segmentation data set* (ImgSeg): The 2310 instances of this data set were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel. Each instance is a 3×3 region represented by 19 attributes and there are 7 classes to be found. The 19 attributes are either color-based (sub-divised into red, green and blue attributes), position based (row, column, pixel count), or other color attributes (contrast, hue, etc.)
- *Spam Base*: The Spam Base data set contains 4601 observations described by 57 attributes and a label column: Spam or not Spam (1 or 0). The different attributes can be split into word frequencies, letter frequencies and capital run sequences attributes.

We also used two artificial data sets:

- The Battalia3 data set[2] (artificial): Battalia3 is an artificial dataset created using the exoplanet random generator from the online game Battalia.fr; This data set describes 2000 randomly generated exoplanets with 27 numerical attributes and their associated class (6 classes). The attributes can be split between system and orbital parameters (7 attributes), planet characteristics (10 attributes) and atmospheric characteristics (10 attributes).
- The "MV2" data set (artificial): A data set created specifically to test this kind of algorithm. It features 2000 randomly generated data, split into 4 views of 6 attributes each, and a total of 4 classes. All attributes were generated either from Gaussian distributions with parameters linked to the matching class, or are random noise, or are linear combinations of other attributes.

[1] Available from Dr. J. Sublime ResearchGate account.

[2] Available from Dr. J. Sublime ResearchGate account.

References

[1] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227, doi:10.1023/A:1022648800760.

[2] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (1992) 241–259.

[3] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239, doi:10.1109/34.667881.

[4] P. Bachman, O. Alsharif, D. Precup, Learning with Pseudo-ensembles, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 3365–3373.

[5] A. Zimek, J. Vreeken, The blind men and the Elephant: on meeting the problem of multiple truths in data from clustering and pattern mining perspectives, Mach.Learn. 98 (1–2) (2015) 121–155, doi:10.1007/s10994-013-5334-y.

[6] A. Kumar, H.D. III, A co-training approach for multi-view spectral clustering., in: L. Getoor, T. Scheffer (Eds.), ICML, Omnipress, 2011, pp. 393–400.

[7] B. Depaire, R. Falcon, K. Vanhoof, G. Wets, PSO driven collaborative clustering: a clustering algorithm for ubiquitous environments, Intell. Data Anal. 15 (2011) 49–68.

[8] X. Cai, F. Nie, H. Huang, Multi-view k-means clustering on big data, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, in: IJCAI '13, AAAI Press, 2013, pp. 2598–2604. http://dl.acm.org/citation.cfm?id=2540128.2540503.

[9] W. Pedrycz, Interpretation of clusters in the framework of shadowed sets, Pattern Recogn. Lett. 26 (15) (2005) 2439–2449, doi:10.1016/j.patrec.2005.05.001.

[10] W. Pedrycz, Knowledge-Based Clustering, John Wiley & Sons, Inc., 2005.

[11] N. Grozavu, Y. Bennani, Topological collaborative clustering, Aust. J. Intell. Inf.Process. Syst. 12 (3) (2010).

[12] J. Sublime, N. Grozavu, Y. Bennani, A. Cornuéjols, Collaborative clustering with heterogeneous algorithms, in: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12–18, 2015, 2015.

[13] W. Pedrycz, Collaborative fuzzy clustering, Pattern Recognit. Lett. 23 (14) (2002) 1675–1686.

[14] S. Zhang, C. Zhang, X. Wu, Knowledge discovery in multiple databases, Advanced Information and Knowledge Processing, Springer, 2004, doi:10.1007/978-0-85729-388-6.

[15] S. Vega-Pons, J. Ruiz-Shulcloper, A survey of clustering ensemble algorithms, IJPRAI 25 (3) (2011) 337–372.

[16] S. Bickel, T. Scheffer, Estimation of mixture models using co-em., in: Proceedings of the ICML Workshop on Learning with Multiple Views, 2005.

[17] S. Bickel, T. Scheffer, Estimation of mixture models using co-em, in: J. Gama, R. Camacho, P. Brazdil, A. Jorge, L. Torgo (Eds.), Machine Learning: ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3–7, 2005, Proceedings, Lecture Notes in Computer Science, 3720, Springer, 2005, pp. 35–46.

[18] W. Pedrycz, Fuzzy clustering with a knowledge-based guidance, Pattern Recogn. Lett. 25 (4) (2004) 469–480.

[19] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[20] G. Cleuziou, M. Exbrayat, L. Martin, J. Sublemontier, Cofkm: a centralized method for multiple-view clustering, in: W. Wang, H. Kargupta, S. Ranka, P.S. Yu, X. Wu (Eds.), ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6–9 December 2009, IEEE Computer Society, 2009, pp. 752–757, doi:10.1109/ICDM.2009.138.

[21] T. Hu, Y. Yu, J. Xiong, S.Y. Sung, Maximum likelihood combination of multiple clusterings, Pattern Recogn. Lett. 27 (13) (2006) 1457–1464, doi:10.1016/j.patrec.2006.02.013.

[22] N. Grozavu, Collaborative Unsupervised Learning and Cluster Characterization, The Paris 13 University, 2009 Ph.D. thesis.

[23] B.Y. Grozavu N., Topological collaborative clustering, in: 17th International Conference on Neural Information Processing, in: LNCS, Springer of ICONIP'10, 2010.

[24] M. Ghassany, N. Grozavu, Y. Bennani, Collaborative clustering using prototype-based techniques, Int. J. Comput. Intell. Appl. 11 (3) (2012).

[25] C. Wemmert, Classification Hybride Distribuée Par Collaboration De Methodes Non Supervisées, The University of Strasbourg, 2000 Ph.D. thesis.

[26] G. Forestier, C. Wemmert, P. Gancarski, Collaborative multi-strategical classification for object-oriented image analysis, in: Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications in conjunction with IbPRIA, 2007, pp. 80–90.

[27] C. Wemmert, P. Gancarski, A multi-view voting method to combine unsupervised classifications, Artif. Intell. Appl. , Malaga, Spain, (2002) 447–452.

[28] S. Dasgupta, M. Littman, D. McAllester, Pac generalization bounds for co-training, in: Proceedings of Neural Information Processing Systems, 2001.

[29] X.-N. Wang, J.-M. Wei, H. Jin, G. Yu, H.-W. Zhang, Probabilistic confusion entropy for evaluating classifiers, Entropy 15 (11) (2013) 4969–4992.

[30] J.-M. Wei, X.-J. Yuan, Q.-H. Hu, S.-Q. Wang, A novel measure for evaluating classifiers., Expert Syst. Appl. 37 (5) (2010) 3799–3809.

[31] R.M. Neal, G.E. Hinton, A view of the EM algorithm that justifies incremental, sparse, and other variants, in: Learning in Graphical Models, 1998, pp. 355–368.

[32] J. Azimi, X. Fern, Adaptive cluster ensemble selection., in: C. Boutilier (Ed.), IJCAI, 2009, pp. 992–997.

[33] N. Grozavu, G. Cabanes, Y. Bennani, Diversity analysis in collaborative clustering, in: 2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6–11, 2014, 2014, pp. 1754–1761.

[34] M. Zarinbal, M.F. Zarandi, I. Turksen, Relative entropy collaborative fuzzy clustering method, Pattern Recognit. 48 (3) (2015) 933–940. http://dx.doi.org/10.1016/j.patcog.2014.09.018.

[35] P. Rastin, G. Cabanes, N. Grozavu, Y. Bennani, Collaborative clustering: how to select the optimal collaborators? in: IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7–10, 2015, IEEE, 2015, pp. 787–794, doi:10.1109/SSCI.2015.117.

[36] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227.

[37] P.J. Rousseeuw, A.M. Leroy, Robust Regression and Outlier Detection, John Wiley & Sons, Inc., New York, NY, USA, 1987.

[38] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1985) 193–218.

[39] W. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. (1971) 846–850.

[40] C.M. Bishop, M. Svensén, C.K.I. Williams, GTM: the generative topographic mapping, Neural Comput. 10 (1) (1998) 215–234.

[41] J. Sublime, A. Troya-Galvis, Y. Bennani, P. Gancarski, A. Cornuéjols, Semantic rich ICM algorithm for VHR satellite image segmentation, in: IAPR International Conference on Machine Vision Applications, Tokyo, 2015.

[42] Y.B.M.L. Nistor Grozavu, From variable weighting to cluster characterization in topographic unsupervised learning, in: in Proc. Proc. of IJCNN09, International Joint Conference on Neural Network, 2009.

[43] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. R. Stat. Soc. Series B 39 (1) (1977) 1–38.

[44] S. Rougier, A. Puissant, Improvements of urban vegetation segmentation and classification using multi-temporal pleiades images, in: 5th International Conference on Geographic Object-Based Image Analysis, 2014, p. 6.

[45] A. Frank, A. Asuncion, UCI machine learning repository, 2010. http://archive.ics.uci.edu/ml.

Jérémie Sublime received a Ph.D. degree in Computer Science from the University Paris-Saclay in 2016. He now is an Associate Professor at the ISEP. He is also an associate researcher at the LIPN - CNRS UMR 7030. His research interests include unsupervised learning, collaborative and multi-view clustering as well as unsupervised neural networks.

Basarab Matei received the Ph.D. degree in applied mathematics from the Paris VI university, France, in 2002. He is an associate professor of machine learning at Paris 13 university. His research interests include wavelets, adaptative representations, irregular sampling, quasicrystals, tillings, compressed sensing and machine learning.

Guénaël Cabanes received a Ph.D. in Computer Science at the University of Paris 13 in 2010. He is an Associate Professor of the University of Paris 13, and a member of the Machine Learning research team at the LIPN-CNRS laboratory. His research interests are in data mining, unsupervised learning and complex structures.

Nistor Grozavu received a Ph.D. in Computer Science at the University of Paris 13 in 2009. He is an Associate Professor of the University of Paris 13, and a member of the Machine Learning research team at the LIPN-CNRS laboratory. His research interests include Unsupervised numerical learning, Data mining, Clustering, Feature selection and weighting and Self-organizing maps.

Youn'es Bennani received the Ph.D. degree in Computer Science from The University of Paris 11, Orsay, in 1992, and the Accreditation to lead research degree from the Paris 13 University in 1998. He is Full Professor of computer science in the Paris 13 University. His research interests are in Machine Learning and Data Science. His areas of expertise are unsupervised learning, transfer learning, cluster analysis, dimensionality reduction, features selection, features construction, data visualisation, and large-scale data mining.

Antoine Cornuéjols received a Ph.D. degree in applied Computer Science from the University Paris 11 in 1989. He is now a Full Professor in Computer Science at AgroParis-Tech (Université Paris-Saclay) where he is the head of the "Modélisation Mathématique, Informatique et Physique" (MMIP) department. His research interests include Machine Learning and Data Mining mostly with applications in biology and genetics.