This is the author's version of an article published in Information Fusion. The final authenticated version is available online at: https://doi.org/10.1016/j.inffus.2017.04.008.

Collaborative Clustering: Why, When, What and How

Antoine Cornuéjols[1], Cédric Wemmert[2,*], Pierre Gançarski[2], Younès Bennani[3]

[1] *UMR MIA-Paris, AgroParisTech, INRA - Université Paris-Saclay, Paris, France*
[2] *UMR ICube - Unistra, CNRS - Université de Strasbourg, Illkirch, France*
[3] *LIPN - UMR CNRS - Université Paris 13 - Sorbonne Paris Cité, Villetaneuse, France*

Abstract

Clustering is one type of unsupervised learning where the goal is to partition the set of objects into groups called clusters. Faced to the difficulty to design a general purpose clustering algorithm and to choose a good, let alone perfect, set of criteria for clustering a data set, one solution is to resort to a variety of clustering procedures based on different techniques, parameters and/or initializations, in order to construct one (or several) final clustering(s). The hope is that by combining several clustering solutions, each one with its own bias and imperfections, one will get a better overall solution.

In the cooperative clustering model, as Ensemble Clustering, a set of clustering algorithms are used in parallel on a given data set: the local results are combined to get an hopefully better overall clustering. In the collaborative framework, the goal is that each local computation, quite possibly applied to distinct data sets, benefit from the work done by the other collaborators.

This paper is dedicated to collaborative clustering. In particular, after a brief overview of clustering and the major issues linked to, it presents main challenges related to organize and control the collaborative process.

Keywords: Collaborative Clustering, Clustering combining, Cooperative clustering

1. Introduction

Unsupervised learning is often defined in contrast with supervised learning. In *supervised learning*, the goal is to make predictions about output value y given an input object or instance $\mathbf{x}$. This is done through a decision procedure $h : \mathcal{X} \longrightarrow \mathcal{Y}$ that is learned from a training set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and some prior knowledge, where each example of $\mathcal{S}$ is composed of an object $\mathbf{x}_i \in \mathcal{X}$ and a corresponding output value $y_j \in \mathcal{Y}$.

By contrast, the objective of *unsupervised learning* is not to make predictions from as yet unknown input values to output values, but to reveal possible hidden structures in the available data set, $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. In a way, this can be compared to signal analysis by which one seeks a decomposition of the signal into underlying basis functions. If these putative structures or regularities may sometimes be extrapolated to make predictions about future events, this is not the primary goal of unsupervised learning. Another crucial distinction with supervised learning is that *there is no absolute way to measure the relevance of the uncovered regularities*, whatever their form [1]. In supervised learning, one can use validation sets or cross-validation to estimate the predictive value of the learned decision function. If the predictive performance is low, then either the data or the learning algorithm is wanting. Unfortunately, there is no equivalent to the predictive performance in unsupervised learning. The algorithms can only find the kind of underlying structures that the user has predefined either implicitly or explicitly in their code. In the best of worlds, the methods also provides some level of significance of the discovered structure. But there is no objective way of measuring the value of the findings, that is whether they correspond to some "true" underlying structure of the data set or if they are just figments of the imagination of the user and the algorithm chosen. Indeed, the significance tests that are often used as referees are themselves, by necessity, biased towards some types of regularities. This is this property that makes unsupervised learning so challenging, both to find a solid theoretical theory about what is a good or best technique, and to apply it with some level of confidence to data in need of interpretability.

Clustering is one type of unsupervised learning where the goal is to partition the set of objects into groups called clusters. These groups can be mutually exclusive or they may overlap, depending on the approach used. Clusters are defined by the fact that the objects within are more similar to each other than to objects from other groups.

*Corresponding author
Email addresses: antoine.cornuejols@agroparistech.fr (Antoine Cornuéjols), cedric.wemmert@unistra.fr (Cédric Wemmert), pierre.gancarski@unistra.fr (Pierre Gançarski), younes.bennani@lipn.univ-paris13.fr (Younès Bennani)

The similarity measure is of course of paramount importance to define the kind of structures or clusters that can be uncovered in the data, and hundreds of distances have been proposed in the literature according to the problem and context at hand.

Two major approaches of clustering exist: *generative* and *discriminative*, both relying more or less directly on a chosen distance. The former supposes that a generative model has been defined, often in the form of a statistical model, and the goal is to find the model parameters maximizing the probability that the data was generated by the model. The latter relies on similarity measures and on optimization criteria to find groups in the data. In either case, before an algorithm can be properly defined, numerous questions have to find an answer.

1.1. The questions raised in Clustering

The exploration of ill-known data sets and the uncovering of hidden regularities are marred with an array of questions and potential pitfalls.

The question arising immediately is: what is clustering? Is there a clear definition and hence, hopefully, some measurable criterion that ought to be optimized?

Intuitively, clustering is the grouping of objects such that similar objects end up in the same group and dissimilar objects are assigned to different groups. Formally, clustering a data set $\mathcal{S}$ of N objects means finding a partition $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ of $\mathcal{S}$ such that:

$$\bigcup_{k=1}^{K} \mathcal{C}_k = \mathcal{S},$$

where the groups $\mathcal{C}_k$ are:

1. As ***homogeneous*** as possible (small intra-group variability)
2. As ***distinct*** as possible (large inter-group variability)

Most clustering techniques output *partitions* (disjoint clusters):

$$\mathcal{C}_k \cap \mathcal{C}_{k'} = \varnothing \quad \text{if} \quad k \neq k'$$

which is not always desirable.

For all its seemingly clear definition, clustering is *an ill-defined problem*. One fundamental issue is that clustering is based on the idea that similar objects should be clustered together while dissimilar objects should be separated in different groups. But, mathematically, similarity is not a transitive relation, while belonging to the same cluster is.

Thus, on Figure 1, which seems a reasonable clustering of the given data points, $\mathbf{x}_1$ appears to be close to $\mathbf{x}_2$, and $\mathbf{x}_2$ to $\mathbf{x}_3$ and so on until $\mathbf{x}_{11}$, and, as a consequence, they should all be put in the same cluster. But, if the shown clustering is correct, it violates the first requirement (all similar elements should end up in the same cluster): $\mathbf{x}_5$ and $\mathbf{x}_6$ should belong to the same cluster; as well the second one (dissimilar elements should be put in distinct clusters): $\mathbf{x}_5$ and $\mathbf{x}_3$ should not belong to the same cluster.

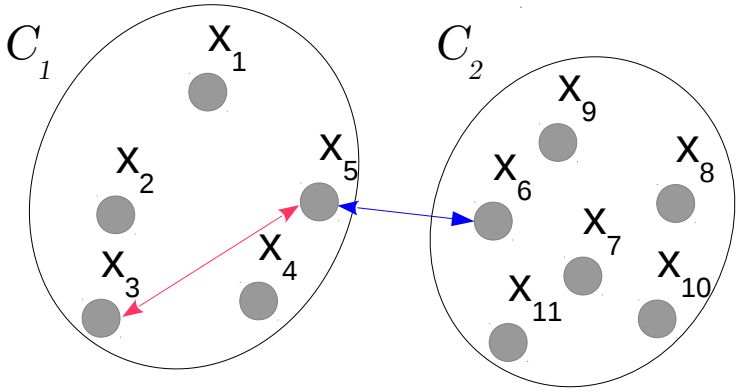


Figure 1: Clustering is an ill-defined problem

There is therefore an ambiguity in the definition of clustering that can only be removed through some additional bias. For instance, the distance used for measuring the inter-group dissimilarity (e.g. single linkage, average linkage, complete linkage, and so on) will favor one type of structure over others. However, this bias impacts the clustering process and not the optimization criterion which remains therefore intrinsically ambiguous.

Another major source of problems is that an ideal clustering would entail the exploration of an impossibly large space of possible answers. Thus, the number of partitions of N objects in K groups is:

$$S_{N,K} = \frac{1}{K!} \sum_{k=0}^{K} (-1)^k \, (K-k)^N \binom{K}{k} \simeq \frac{K^N}{K!} \text{ as } N \to \infty \tag{1}$$

If the number of partitions K is not known beforehand, then the number of all partitions to be examined is given by the Bell number:

$$B_N = \sum_{k=1}^{N} S_{N,k} \tag{2}$$

As an illustration, a computer handling one million partitions per second would take more than 147,000 years to study all partitions of a set of only 25 elements: there are indeed 4,638,590,332,229,999,353 possible partitions of such a set!

One therefore has two perspectives: either finding an optimization criterion such that the optimization problem becomes convex in the search space, or designing a heuristic search algorithm that can search the space of solutions efficiently and, to some extent, escape local minima. No convex optimization criterion is known, and thus one must solve the second alternative. As it happens, most of the resulting optimization problems are NP-hard.

To sum up, clustering, is not only an *ill-defined* problem [2, 3], it is also an *ill-posed problem* that requires some

prior bias in order to be practically solved. Different algorithms may yield dramatically different outputs for the same input sets. Additionally, the entailed computational costs are huge if no proper heuristics is employed.

Consequently, several concrete questions must be answered before a clustering method can be defined and applied.

1.1.1. Formally defining the types of clusters looked for

In clustering, we wish to organize the data in some meaningful way, but "meaningful" depends on the context and on our focus of interest. The same given set of objects can be clustered in various different meaningful ways. For instance, we could be interested in categorizing speakers by the language they speak, or by the topic of discussion, or by gender. Accordingly, one would concentrate on different descriptors in the spoken signal, and use different distances in order to group the speakers.

The *distance* is a critical part in the definition of what types of clusters will be looked for. Actually, in many algorithms, several distances must be decided upon: a distance between instances in the input space, but also a distance between an instance and a cluster, and a distance between clusters. As is well-known by practitioners, any single difference in these choices points may alter considerably the result of a clustering.

Another problem is the choice of the relevant *number of clusters* when the number of "true" underlying categories is not known beforehand. This is related to the model selection problem [4]. Often, what is looked for are clusters that are compact (within inertia) and well separated (extra-inertia). It happens that these two requirements tend to be inversely correlated, when one improves, the other deteriorates. The relative weights put on these aspects control therefore the result in the same manner that the choice of the hypothesis space or of a regularization term controls the result in supervised learning. However, unlike for supervised learning, there is no ground truth in the data that can help choosing the right "model". Hence, the temptation not to have to choose, which is one motivation for ensemble methods in clustering, as described below.

1.1.2. Organizing the search space

Once the type of clusters that is looked for in the data is outlined and similarity measures have been defined accordingly, it remains to set up how the space of possible solutions will be searched. Because it is impossible to evaluate the whole set of solutions, it is necessary to devise algorithms that perform local search. Some, like hierarchical ascendent clustering, iteratively merge the clusters obtained at one step to get larger clusters at the next step until a stopping criterion is met. Other, like k-means, are conspicuously trying to optimize a criterion, such as:

$$G_{k-means}((\mathcal{S}, d), (\mathcal{C}_1, \ldots, \mathcal{C}_K)) = \underset{c_1,\ldots,c_k,\ldots,c_K}{\text{Argmin}} \sum_{k=1}^{K} \sum_{\mathbf{x}\in\mathcal{C}_k} d(\mathbf{x}, c_k) \tag{3}$$

where d stands for the distance used and the c_k is the centroid of the cluster $\mathcal{C}_k$. The number of clusters has to be given a priori.

But because this type of criterion is NP-hard to satisfy, an iterative algorithm, the k-means algorithm, is used instead. Its output is generally highly dependent upon the initialization of the centroids c_i.

Aside the criterion and the search algorithm, the space of solutions plays an important role too. The number of *attributes*, their degree of (in)dependence, the normalization used, all these aspects may strongly affect the result of learning. Furthermore, when dimensionality increases, the volume of the space increases exponentially causing the available data to become extremely sparse. One unexpected and frightful consequence is that, for most data distributions, the relative difference of the distances of the closest and farthest data points of an independently selected point goes to zero as the dimensionality increases. Consequently, any algorithm that bases its analysis on the distribution of distances is doomed to fail at finding structures in the data if there is no truly marked structure. Furthermore, the computational complexity of finding clusters is in $\mathcal{O}(N^D)$ where D is the dimension of the space. Since, in addition, the set of all possible structures is extremely large (see equations 1 and 2), and the optimization criterion generally entails the existence of a complex landscape with many local optima, the algorithms for exploring such spaces must, per force, use heuristics and possibly iterative procedures. In turn, this often requires that several parameters be set and tuned by the user.

1.1.3. The validation

Because clustering relies on many a priori choices (i.e. similarity measures, quality function, heuristics, parameters values, and so on) and since there is no possible post validation with test data where true classes can be compared with the classes discovered by the algorithm, the validation of the output of clustering methods is a thorny question. The user is faced with the interrogation: does the structure produced by the method "really" exist in the data, or is it merely an artifact of the clustering method?

Some principled statistical procedures have been proposed for testing the the significance of a clustering result [5]. They are based on the idea of measuring deviations of some statistical quantities, like the largest nearest neighbor distance, from what would be expected if the data points were homogeneously distributed in the input space. However, the approach has several limits. First, the statistical quantities suggested are not general, they are biased towards the discovery of convex clusters. Second, the convergence rate of the statistical test is so slow ($\mathcal{O}(\log n)^{-1}$)

that it requires enormous quantities of data to give some useful clues. Finally, it requires that the user is able to guess some quantities, like the volume in which lies the data, hard to compute when the input space is not low dimensional.

Similarly, bayesian approaches to model-based clustering (e.g. mixture of Gaussian distributions) have been put forward to yield validity measures such as the Bayes factor [6]. However, these proposals are dependent upon some preconceived models of the world, and they are often based on asymptotic approximations that are hard to satisfy in real applications.

Short of a satisfying theoretical ground, a whole area of methods and indexes have been designed in order to assess the validity of clustering results. However, all are arbitrary in some sense, in that they favor one type of structures over others, and one must be very careful not to over-interpret seemingly optimistic results from a clustering quality measure (see [7] for an overview of assessment methods for clustering and their limits).

Because all clustering methods are based, whether implicitly or explicitly, on a priori assumptions about the data-generating process, or about the interesting structures that ought to be found, the final judge is often the domain expert interested in what the algorithm has discovered in the data. The expert may thus evaluate the output, and suggests modifications of the a priori choices in order to test the robustness of the findings. Indeed the variability of the results when the data sampling, or the parameter values, or the clustering algorithm is changed is often taken as an indication of the reliability of the structure uncovered in the data.

To measure this variability, several *indexes* exist: *External indexes* (used to measure the extent to which cluster labels match externally supplied class labels), of which a prominent one is the Rand index [8]; *Internal indexes* (used to measure the goodness of a clustering structure without respect to external information); *Relative indexes* (used to compare two different clusterings or clusters). Unfortunately, even if the stability of the clustering results is often a good indicator of the fact that the uncovered structure is really present in the data, it can also be misleading under certain circumstances. Indeed the shape of the data manifold can cause a given clustering method to converge strongly towards certain suboptimal solutions.

Overall, one guide for evaluation that is often found useful, and is at the basis of some recent attempt at putting clustering on a sound foot, is the fact that the cluster results are robust to variations either in the data (aka. permutation invariance) and/or in the algorithms, that is variations in their principles, or their parameter values, or the initializations. This forms one justification for the recourse to ensemble methods for clustering, as will be described below.

When taken together, the issues facing a practitioner are numerous, and, correspondingly the number of choices to be made and parameters to be set. And because the results output by the clustering algorithms are generally very dependent upon these choices, they have to be made cautiously. However, it is often difficult to identify the ideal recipe for a given problem.

1.2. Combining Clusterings

Faced with the difficulty of choosing a good, let alone perfect, set of criteria for clustering a data set: distances, optimal number of clusters, measures of compactness, and so on, a temptation is to avoid as much as possible to have to commit to pre-arranged methods or parameters. One solution for this is to resort to a variety of clustering procedures based on different techniques, parameters and/or initializations, in order to construct one (or several) final clustering(s).

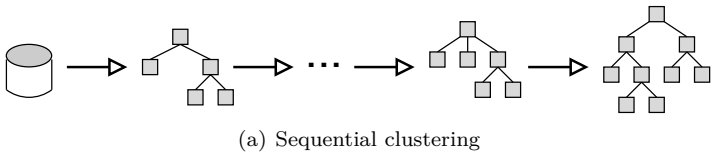
Because, in addition to this motivation, it is increasingly realized that it can be interesting to detect multiple cluster structures that describe alternative aspects in the same data set rather than trying to find "the" best clustering, a recent and very active stream of research has focused on *Multiple clustering* which is devoted to capture multi-faceted information in a data set. Accordingly, the goal is no longer to get a single overall view of the data, but instead of favoring multiple non redundant groupings.

Meanwhile, the success of ensemble methods in supervised learning has lent credit to the power of panels of "experts" that combine their beliefs or decisions in order to reach an overall solution. And while it is far from obvious that the same success can be ensured in the context of unsupervised learning, many works have been inspired by this idea.

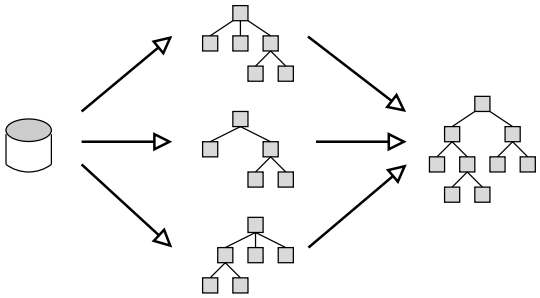
Accordingly, approaches for *combining clusterings* have emerged that bring together clusterings based on different sources and/or different clustering algorithms [9]. The hope is that by combining several clustering solutions, each one with its own bias and imperfections, one will get a better overall solution [10]. The main idea is that fortuitous and not meaningful solutions will cancel each other out, and that the real structure in the data should emerge as the most represented in the outputs of the algorithms because it would be the more robust to variations in the algorithms's settings.

It is useful to characterize multiple clustering based methods by looking at three sets of possibilities:

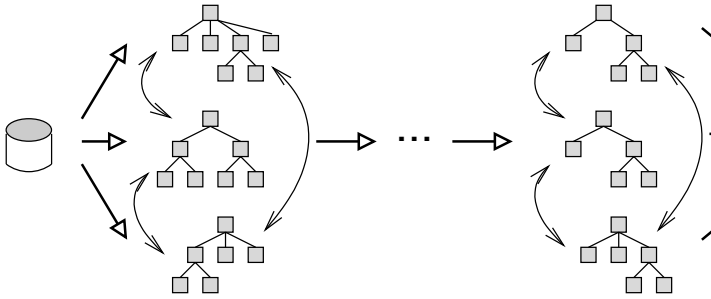
1. Either the clustering algorithms work **sequentially** or they work in **parallel**.
   In sequential approaches (Figure 2-a), like cascading [11], each method in turn may use information provided by the previous clustering system(s) while possibly exploiting new additional data. In both the sequential and parallel regimes, one question is to determine when to end the procedure, which ideally should converge to a fixed point.
2. The clustering algorithms work **in isolation** versus **in interaction**:



(a) Sequential clustering



(b) Cooperative clustering



(c) Collaborative clustering

Figure 2: Different types of combinations of clusterings

- In *cooperative clustering* (see Section 1.2.1), 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 *collaborative clustering* (see Section 1.2.2), 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.

3. Each algorithm works on the **whole dataset** or **only on a subpart**. The advent of "big data" (large data sets often described in high-dimensional spaces) renders distributed computing quite alluring, and sometime compulsory. For instance, for reasons of privacy, ownership or storage, it can happen that the data cannot be pooled together. If each algorithms accesses only a partial view (different features), this situation is called *vertical clustering*. If all the classifier work using the same set of features, but only on a subset of the dataset, then one is faced with *horizontal clustering* [12]. Horizontal and vertical clusterings can be combined in one application (see Section 2). For instance, several health centers located in different countries may want to discover structures in the diseases of the citizens of the world in an attempt to identify latent causes, using data on different populations and exploiting multiple partial information sources at possibly varying resolutions.

1.2.1. Cooperative Clustering

In the *cooperative clustering model*, a set of clustering algorithms are used in parallel on a given data set. The diversity of the local results is obtained through the diversity of the local algorithms, their parameter values, and their initializations. Once all local computations are completed, a master algorithm takes control and combines the local results to get an 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 (Figure 2-b).

The cooperative framework is 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 [9]. Bayesian averaging can be considered as a precursor method. Numerous new ones have been developed, from error-correcting output coding to Bagging, and Boosting [13, 14, 15, 16] and their application in various domains have become routine with often good results.

Ensemble methods are easy to implement in supervised learning for two reasons. First, it is straightforward to define a combination of predictive functions h_i to get an aggregated prediction function H. For instance, a linear combination is used in boosting, which, for a classification task gives:

$$H(\mathbf{x}) \;=\; \texttt{Majority}\Big\{\sum_{i=1}^{P} w_i \; h_i(\mathbf{x})\Big\}$$

where w_i is the weight of classifier h_i.

Second, it is simple to measure both the performance of individual prediction functions h_i 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. For instance, it is not practical to assign each example $\mathbf{x}_j$ to some "majority cluster", i.e. the cluster where the majority of the clustering algorithms would have put $\mathbf{x}_j$. The first obstacle indeed being that,

usually, there is no direct correspondence between the clusters found by the local algorithms. Searching for a correspondence between each cluster C_k^p of a local clustering C^p and each cluster C_l^q of another local clustering C^q, and doing this for all pairs of clusterings (C^p, C^q) taken in the set $\mathcal{C}$ of local clusterings is a daunting task with no simple and practical solution. In response, various heuristic proposals have been made for these problems, among them, the following:

- [17] introduced a new voting method to perform a combination of the results even when there is no strict bijection between the clusters but only a "good" correspondence.
- [18] presented three voting iterative methods: Iterative Voting Consensus (IVC), Iterative Probabilistic Voting Consensus (IPVC) and Iterative Pairwise Consensus (IPC). These algorithms use a feature map built from the set of base clusterings and apply an EM-like consensus clustering.
- [19] proposed a cumulative voting method to come to a consensus from partitions with a variable number of clusters. They described several cumulative vote weighting schemes, with the corresponding algorithms, to compute an empirical probability distribution summarizing the partitions.
- [20] tackled the problem of correspondence by using the cooccurrences of pairs of patterns in the same cluster as votes for their association. In this framework, the data partitions are mapped into a $N \times N$ co-association matrix of patterns which represents a new similarity measure between patterns. The final clusters are obtained by applying a MST-based clustering algorithm on this matrix.
- [21] presented a cooperative clustering model mainly based on four components (1) Co-occurred sub-clusters, (2) An histogram representation of the pair-wise similarities within sub-clusters, (3) The cooperative contingency graph, and (4) The coherent merging between the set of histograms. This kind of Cooperative Clustering model is based on finding the intersection between the multiple clusterings in terms of a set of sub-clusters. These sub-clusters are represented by similarity histograms. The model applies a homogeneous merging procedure on the cooperative contingency graph to attain the same number of clusters by carefully monitoring the pairwise similarities between objects in the sub-clusters [22].

Overall, research in ensemble clustering has blossomed over the recent years [23, 24, 25, 26, 27, 28, 29, 16].

Ensemble (and cooperative) clustering follow the lines sketched in Algorithm 1.

Algorithm 1: Algorithm for *ensemble clustering*

Data: The set $\mathcal{S}$ of data points
P algorithms: $\{A_{i\ (1\leq i\leq P)}\}$
Result: A clustering G of the whole data set
Run in parallel the P algorithms A_i each on the set $\mathcal{S}$;
Compute all the P clusterings $\mathcal{C}_i$;
Compute the consensus solution G from $\{\mathcal{C}_{i\ (1\leq i\leq P)}\}$;

Algorithm 2: Algorithm for *collaborative clustering*

Data: P subsets of $\mathcal{S}$ not necessarily disjoint: $\{\mathcal{S}_{i\ (1\leq i\leq P)}\}$
P algorithms: $\{A_{i\ (1\leq i\leq P)}\}$
Result: A clustering G of the whole data set
Run in parallel the P algorithms A_i each on its own data set $\mathcal{S}_i$;
Compute all the P clusterings $\mathcal{C}_i$;
repeat
  Compute the consensus solution G from $\{\mathcal{C}_{i\ (1\leq i\leq P)}\}$ or local consensus solutions ;
  Exchange information between the/some algorithms ;
  Compute new clusterings $\mathcal{C}_i$;
until *stabilization of the global clustering or of the local solutions*;

1.2.2. Collaborative Clustering

By contrast to the cooperative clustering model, the *collaborative model* does not seek an overall hopefully better clustering of a given data set $\mathcal{S}$ 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 instead of a two step process, as 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 agents, one question is how to measure the evolution at each node, and when to stop each local process.

Collaborative clustering is realized along the lines of Algorithm 2.

The sequel of the paper is dedicated to collaborative clustering. Section 2 gives the flavor of some issues raised in collaborative learning through simple examples. In Section 3, the questions of why and when collaborative clustering should be expected to work are examined. Then, Section 4 is devoted to the questions of how to organize and control a collaborative process. Section 5 presents previous works related to collaborative learning that have bearings on the issue of collaborative clustering and reports some applications.

2. Simple illustrations of Collaborative Clustering

There are two kinds of information that clustering algorithms use and update during their computations: information about the *membership of each data point* (e.g. $\mathbf{x}_{23} \in \mathcal{C}_2$ where $\mathcal{C}_2$ is the label given by one algorithm to a cluster), and information about *internal parameters*, like the current number of clusters envisioned, the coordinates of prototypes, and so on. Exchanges of information can take place at these two levels.

2.1. Examples of collaboration

In order to illustrate the issues raised by Collaborative Clustering, it is useful to contemplate simple examples. We will consider different scenarios in turn:

1. The algorithms have access to the *same dataset*: same objects and same attributes.
2. The algorithms have access to the *same dataset*, but they only see *partial views* (a.k.a. vertical clustering scenario).
3. The algorithms have access to *different objects* supposedly drawn from the same distribution (virtual dataset) measured with the *same set of attributes* (horizontal clustering).
4. The algorithms have access to *different objects* supposedly drawn from the same distribution (virtual dataset) measured with *different sets of attributes*.

To simplify the discussion, we will suppose that all the algorithms are of the k-means variety, but possibly with different values of k and/or different definitions of distances (e.g. ℓ_1 distance, or ℓ_2, or ...). They may also start from different initial states. Thus, they can obtain different results even on the very same data set. These algorithms can communicate:

- the number of clusters k they are contemplating
- the proportion of objects affected to each cluster
- the identifiers of the objects in each cluster
- the coordinates of the prototypes μ_i that define each cluster $\mathcal{C}_i$.

However, they do not change their own settings: k and the distance used.

Now, for each scenario above, we will examine what communication could be set up among the algorithms in order to carry out a collaborative clustering.

Scenario 1: Same data

This scenario is similar to the one of ensemble clustering where a consensus solution that escapes the limits of each biased solution is hoped for. The difference with ensemble clustering may lie in the protocol of exchanges between the learning algorithms. In ensemble clustering, each algorithm computes its own local solution, and then a master algorithm computes a consensus solution. In collaborative clustering, there is no master algorithm and the communications between algorithms can alter the local computations for a solution. Furthermore, one can be happy with different outputs from the local algorithms since alternative clusterings for the same data may well be what is looked for. Collaborative clustering is thus a means used to help local algorithms to escape local minima and discover better solutions.

When the algorithms can share the identifiers of the objects, it is an easy matter to compare cluster memberships. For instance, in Figure 3, we have 8 data points and two clustering algorithms A and B. The cluster memberships can be compared via the matrix:

Algorithm	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$
A	$\mathcal{C}_a$	$\mathcal{C}_a$	$\mathcal{C}_a$	$\mathcal{C}_a$	$\mathcal{C}_a$	$\mathcal{C}'_a$	$\mathcal{C}'_a$	$\mathcal{C}'_a$
B	$\mathcal{C}_b$	$\mathcal{C}_b$	$\mathcal{C}_b$	$\mathcal{C}'_b$	$\mathcal{C}'_b$	$\mathcal{C}''_b$	$\mathcal{C}''_b$	$\mathcal{C}''_b$

In this case, it is apparent that the cluster $\mathcal{C}_b$ found by algorithm B is compatible with cluster $\mathcal{C}_a$ found by algorithm A since $\mathcal{C}_b \subseteq \mathcal{C}_a$. Similarly, $\mathcal{C}'_b \subseteq \mathcal{C}'_a$. However, there is a conflict regarding the cluster $\mathcal{C}''_b$ since it has an intersection with both $\mathcal{C}_a$ and $\mathcal{C}'_a$. Aware of this conflict, each algorithm could take steps to eliminate it. For instance, algorithm A could start its next loop by putting the offending point $\mathbf{x}_8$ in cluster $\mathcal{C}_a$ before updating the prototype μ_a, therefore biasing its own solution toward a solution compatible with the current one of algorithm B. Simultaneously, algorithm B could put $\mathbf{x}_8$ in cluster $\mathcal{C}'_b$. Of course, if this is done simultaneously by both algorithms, the conflict will persist.

It is apparent that this naïve collaborative approach needs much work out before it can be implemented and extended to more than 2 algorithms. First, the communication bandwidth tends to increase as $\mathcal{O}(P^2)$ where P is the number of participating algorithms. Second, the computation of the "offending" data points is more complicated than the simple example above shows. Finally, it is not obvious to define collaboration so that the whole process tends to discover stable local solutions. Indeed, a change in the solution of algorithm A_i can trigger changes in solutions by other algorithms which themselves can modify the solution of A_i. These kinds of loops are not easy to

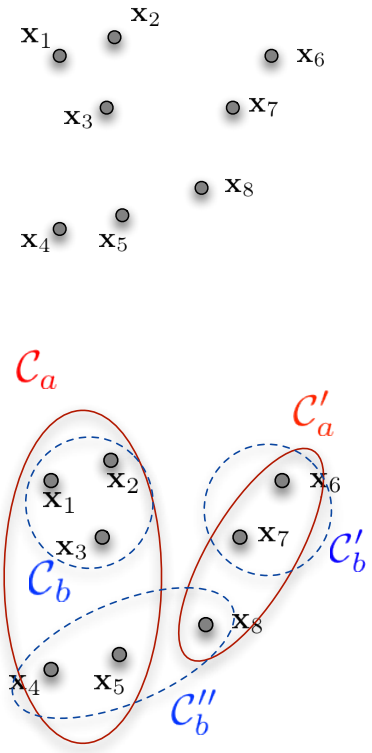


Figure 3: (Left) the data set shared by the clustering algorithms, here in $\mathbb{R}^2$. (Right) The clustering found by algorithm A: $\mathcal{C}_a$ and $\mathcal{C}'_a$, and by algorithm B: $\mathcal{C}_b$, $\mathcal{C}'_b$ and $\mathcal{C}''_b$.

control if an energy function of the whole system cannot be identified and if the overall collaboration scheme is not proved to lead to a decrease of this energy function.

The communication between the algorithms can also involve other characteristics than the memberships of the objects. For instance, the algorithms could also exchange the coordinates of their prototypes together with the weights of these prototypes given by the proportion of the objects they represent.

Coming back to the data set of Figure 4, algorithm A could inform algorithm B that it has two prototypes with coordinates $\mu_a = [\mu_a^{(1)}, \mu_a^{(2)}]^\top$ and $\mu_{a'} = [\mu_{a'}^{(1)}, \mu_{a'}^{(2)}]^\top$ and weights 5/8 and 3/8. Algorithm B in turn would communicate $\mu_b = [\mu_b^{(1)}, \mu_b^{(2)}]^\top$, $\mu_{b'} = [\mu_{b'}^{(1)}, \mu_{b'}^{(2)}]^\top$, and $\mu_{b''} = [\mu_{b''}^{(1)}, \mu_{b''}^{(2)}]^\top$ with respective weights 3/8, 2/8 and 3/8.

In this case, this would push algorithm A toward the absorption of the data point $\mathbf{x}_5$ in the cluster $\mathcal{C}_a$, ending the conflict with the solution of algorithm B.

Again, this gives only an outline of what is involved when exchanging information about the models themselves, and not the data points. For instance, if an algorithm takes into account the prototypes computed by other algorithms, what should be the weights of these prototypes?

Scenario 2: Same objects, different attributes

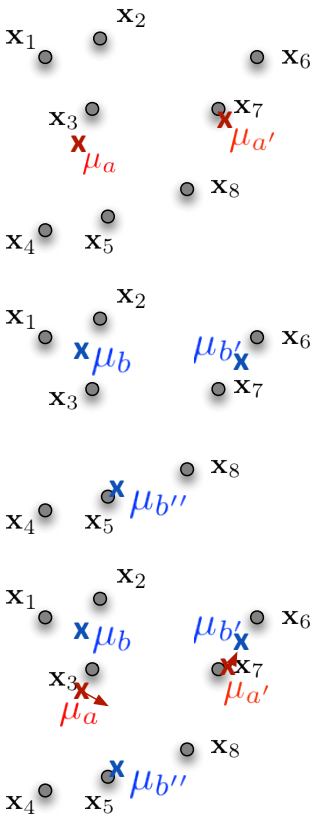
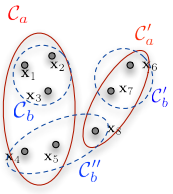


Figure 4: (Left) The prototypes computed by algorithm A at time step t. (Center) The prototypes computed by algorithm B. (Right) The updates of the prototypes of algorithm A when taking into account the prototypes communicated by algorithm B.



	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$
$\mathbf{x}_1$	-	1	1	½	½	0	0	0
$\mathbf{x}_2$	1	-	1	½	½	0	0	0
$\mathbf{x}_3$	1	1	-	½	½	0	0	0
$\mathbf{x}_4$	½	½	½	-	1	0	0	½
$\mathbf{x}_5$	½	½	½	1	-	0	0	½
$\mathbf{x}_6$	0	0	0	0	0	-	1	½
$\mathbf{x}_7$	0	0	0	0	0	1	-	½
$\mathbf{x}_8$	0	0	0	½	½	½	½	-

Figure 5: (Left) The data set and the clusterings by algorithms A and B. (Right) The corresponding consensus matrix. The shaded parts denotes the objects for which there is an agreement between the clusterings.

When the algorithms can share the identifiers of the objects but the attributes they measure are different, they can no longer use information about the centroids computed by other algorithms. However, they can still compare the way they group objects.

One way of doing this is to use *consensus matrices* [30]. A consensus matrix for a set of algorithms is a $N \times N$ matrix where each element $M(i,j)$ is the frequency that the objects i and j are put together in the same cluster by the algorithms. Ideally, if the algorithms all agreed on the clustering, then we would have $M(i,j) \in \{0,1\}, \forall i,j \in \{1,N\}$, and it would be easy to recover the clusters. Any departure from this ideal case is a sign of disagreement between the clusterings made by the algorithms.

For the data set depicted in Figure 3, the consensus matrix is given in Figure 5. Again, it is apparent that the one offending object is $\mathbf{x}_8$, so that this is where the two collaborating algorithms should focus in order to improve their agreement.

Scenario 3: Different objects, same attributes

When different objects are considered by the various algorithms, a collaboration has meaning only if it is assumed that the subsets of data are drawn from the same distribution.

A general case is when there are partial intersections between the subsets of data that the algorithms consider. Then, consensus matrices can still be computed, using a normalization of the entries $M(i,j)$ taking into account the number of times the two objects of the pair have been considered together by the algorithms. This can be used as a basis for the detection of disagreements triggering further adaptations.

Another type of collaboration involves the exchange of the coordinates of the centroids found by the collaborating algorithms, which can be done since the algorithms share the same input space. As was indicated in the scenario 1, the exchange of centroids may allow for modifications that tend to reduce the disagreement between the solutions found locally.

Scenario 4: Different objects, different attributes

Interestingly, even if the attributes considered by the various algorithms are not the same, the exchange of the centroids's coordinates can still be a channel of exchange as long as there is still some non empty intersection between the subsets of attributes. Of course, the smaller the intersection, the higher the risk that the remaining common attributes are in fact not relevant to the true underlying regularities, and the higher the risk of exchanging noisy signals.

The most difficult case is when both the subsets of objects and the subsets of attributes are disjoint. Then the only bits of information that can be of interest to the various algorithms is the relative proportions of the clusters found at each location. If for instance, algorithm A has found the proportions $\{0.25, 0.75\}$ for two clusters and algorithm B has found $\{0.10, 0.13, 0.77\}$ for three, then it is not absurd to suppose that the clusters found by the two algorithms indeed can be put in close match. However, that kind of simple situation is just too simplistic to be realistic. In most cases, a much more involved analysis should be carried out, with likely weak results so far as the collaboration is concerned.

2.2. Illustration: two algorithms collaborating, one well-informed and the other not

In collaborative clustering, it is hoped that by combining several algorithms, each with its own data source and characteristics, one gets better results than by using each algorithm independently. But, is this hope granted?

Taking a very simple setting, what happens if two clustering algorithms are made to collaborate, both applied to the same objects, but using different attributes (vertical clustering)? And particularly, what happens if the set of attributes used by one algorithm, A_{good} is relevant to the underlying existing structure of the data set, while the other subset, used by a A_{bad}, is totally irrelevant, containing only a noisy signal?

In the following, the algorithm F-VBGTM [31] was used on a split *Waveform* data set. This data set was chosen because of its structure: it contains 5,000 observations described by 21 relevant variables and 19 noisy variables. The data set was split into two subsets, the first one described using the relevant variables (dimension $5,000 \times 21$) and the second one using the useless variables (dimension $5,000 \times 19$).

The clustering results by F-VBGTM on the two subsets of the Waveform data set without collaboration are shown in Figure 6. It is apparent that algorithm A_{good}, using the relevant variables, was able to capture the underlying structure of the data set, while algorithm A_{bad} produced essentially a random solution.

Here an oriented collaboration scheme was employed, in which the results of algorithm A_i was sent to algorithm A_j $(j \neq i)$ before its own processing of the data set. As could be expected, when algorithm A_{bad} influences algorithm A_{good}, the structure discovered is seriously impaired (left on Figure 7). On the contrary, algorithm A_{bad} benefits a lot from information sent by A_{good} as shown on the

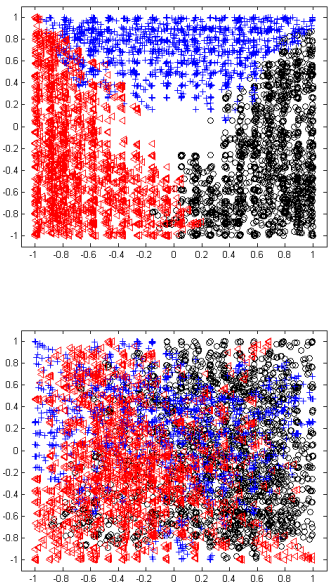


Figure 6: Visualization of the two subsets of the Waveform data set using posterior mean projection, with labels obtained using F-VBGTM before collaboration. We can see good clustering results on the first subset (left). The results of clustering on the second subset of noisy variables are bad (right).

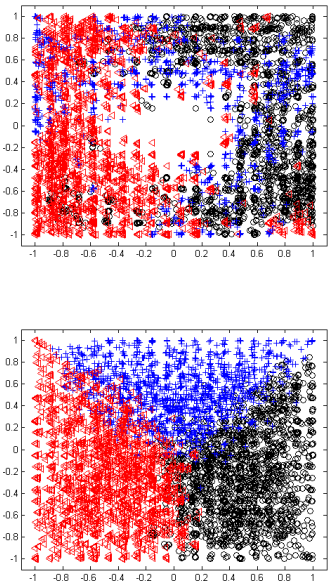


Figure 7: Effect of the Collaborative Clustering. When A_{bad} sends results to algorithm A_{good}, the uncovered underlying structure is severely impaired (left figure). By contrast, clustering is much improved when A_{good} sends information prior to the data processing of A_{bad} (right figure).

right of Figure 7. These results are only qualitative, but they show the potential for harmful as well as beneficial collaboration between clustering algorithms.

3. Why and when collaboration should be beneficial?

In cooperative clustering, the goal is to find a structure in a data set using several algorithms that have access to the whole data set. Therefore the notion of *consensus* among the individual solutions is central in order to control the cooperative process. And while the measurement of consensus can depend on the specificities of the application domain, it is still a relatively easy matter to define such measures.

The situation is quite different in collaborative clustering where the goal is to identify structures in each of the local data sets considered by the various algorithms. There, it is more problematic to ensure that collaboration can bring improvement locally, and it is also much more difficult to control the process since the notion of global consensus is no longer operative. Intuitively, the collaboration is valuable if the final local clusterings have higher quality than if there had been no exchange of information between the local processes.

The potential benefits of collaboration can be due to three causes:

1. The fact that *more data is better*. One can hope therefore that other algorithms bring more information about the data generating process than is locally available.
2. The fact that *perturbations can help* an algorithm *to escape local optima*.
3. The fact that *it might be profitable to modify the bias* of a given algorithm by feeding it with external information coming from algorithms that may have different biases.

Note that identifying these three possible reasons for benefiting from collaborative clustering does not translate trivially into an overall control procedure that would lead to better performance.

First, determining if one of these three situations is valid in one's current context is not obvious. Second, even if it is established that one, or more, of these three ingredients is/are present in some collaborative clustering scheme, none of these three potential causes is guaranteed to bring improvements. Indeed, a local algorithm that is working on its own data set can only benefit from information coming from another algorithm if the data set of the latter shares enough regularities with its own data set. Likewise, perturbations can help the exploration process of an algorithm, but it can also hamper it if they draw it to poor regions of the search space. Finally, and similarly, there is no a priori reason to believe that external information that in effect modifies the optimization criterion of the local algorithm, modifies it in the right direction.

Therefore, extreme attention must be taken in order to ensure that the collaborative process can improve the performance of each local algorithm, and the controlling strategy must be carefully crafted. The lack of a natural notion of overall consensus makes it all the more complicated to achieve.

In the following, we draw attention to each of the three possible causes in turn. And for each of these, we discuss why the outcome should be a benefit, and how could a control procedure become aware of this improvement.

3.1. More data is better

More (training) data entails in general that a better solution is reached by an inductive procedure, be it supervised or unsupervised. This is because considering a larger data sample reduces the variance, and thus the approximation error while leaving the bias, or approximation error, untouched. In the case of collaborative clustering, more data means more unlabeled data.

Let us suppose that the data generating process associated with the local algorithm is characterized by the distribution $\mathbf{P}_{\mathcal{X}}$ (e.g. the distribution that characterize the patient in a given hospital), then the incoming unlabeled data (or its model as sent by the collaborative algorithm(s)) can be useful insofar as its (unknown) marginal distribution with respect to the description space $\mathcal{X}$ of the local algorithm is equal, or very close, to $\mathbf{P}_{\mathcal{X}}$. For instance, this is what would happen if the incoming data from another hospital describes patients that have the same distribution of symptoms than the patients at the local hospital, at least along the dimensions measured at the local hospital.

If this property is warranted, then the local algorithm should benefit from the incoming data, in that it should return a better structure over the local data set. It is expected that the clusters over the local data should be of better quality, at least with respect to the bias enforced by the local algorithm.

One problem is to ensure that the incoming data indeed obeys the same distribution. Measures of divergence of probability distributions such as Kullback-Leibler divergence, relative entropy, information gain and so on can be of use in this respect [32].

3.2. Incoming information can help one participating algorithm to escape local optima

Inductive algorithms evaluate the level of structure of the data set using some inductive criterion. In the case of supervised learning, this is realized generally through a regularized empirical risk measure:

$$\hat{R}(h) \;=\; \frac{1}{m}\sum_{i=1}^{m}\ell(h(\mathbf{x}_i), y_i) \;+\; \mathrm{reg}(h)$$

where there are m labeled data points $(\mathbf{x}_i, y_i)$, ℓ is a loss function that penalizes incorrect predictions $h(\mathbf{x}_i)$, and $\mathrm{reg}(h)$ is regularization function that usually penalizes non smooth functions h.

In the case of unsupervised learning, inductive criteria are often related to some measure of the compactness and distinctness of the sub structures that one is interested in discovering in the data. For instance, in the case of the k-means algorithm the criterion takes the form of Equation 3 of section 1.1.2:

$$G_{k-means}((\mathcal{S}, d), (\mathcal{C}_1, \ldots, \mathcal{C}_K)) \;=\; \underset{c_1,\ldots,c_k,\ldots,c_K}{\mathrm{Argmin}} \sum_{k=1}^{K}\sum_{\mathbf{x}\in\mathcal{C}_k} d(\mathbf{x}, c_k)$$

where d stands for the distance used and the c_k is the centroid of the cluster $\mathcal{C}_k$.

Thus, in principle, an unsupervised learning algorithm explores the parameter space in order to minimize the posited ideal inductive criterion, and may fall into local minima. Any source of perturbation could therefore help the algorithm to escape these traps, and information coming from other collaborative algorithms may play the perturbative role.

Ideally, it would be easy to detect if the information communicated by another algorithm is useful or not, that is if it allows the local algorithm to escape suboptimal minima. It would suffice to track the value of the criterion.

However, a large part of the unsupervised algorithms are heuristic in nature, and they do not directly optimize an inductive criterion. Rather, as is the case for the k-means algorithm, they follow a myopic and iterative process with no explicit optimizing criterion. As a consequence, it might be difficult to assess whether the incoming information is useful or not. And in the absence of such insurance, it is quite possible that the incoming information pushes the exploration towards poor regions of the space of solutions. The difficulty stems from the difference between ideal inductive criteria and the true, heuristic, optimization taking place in the learning algorithm. There is need here for more research investigating this issue, but some works are interesting in this respect [33, 34, 35, 36, 37].

3.3. Incoming information modifies the bias of one participating algorithm

Suppose that the algorithm A is biased towards the discovery of convex subgroups in the data while algorithm B looks for threadlike structures. When algorithm A sends information to algorithm B, in the form, for instance, of the data points with labels corresponding to the discovered structures, then algorithm B will group together points that would not be grouped by algorithm A. If algorithm A takes the groups of algorithm B as suggestions that should be considered to some extent in order to form groups, then this effectively modifies the inductive bias of algorithm A. The shapes found will tend to be not as convex as they would in the absence of the information coming from algorithm B.

But why would that be an interesting modification of the initial bias rather than a damaging one? And how could one assess the merit, if any, of such a modification?

Again, we are drawn back to the issue of evaluating the appropriateness of a clustering method, a question for which there does not exist a definite answer.

In addition, the fact that each local algorithm is influenced by the results of other algorithms and affects them in return, causes the overall process to be recursive with no obvious terminal fixed point. The analysis of such algorithms is thus delicate to carry out.

One paradigmatic illustration of collaborative approaches is the *"blackboard" architecture* developed in the Hearsay II system during the 1970's [38, 39, 40], itself influenced by the Pandemonium multi-agent system of Selfridge [41]. In these systems, a set of modules, each dedicated to a subtask and a domain knowledge, e.g. phonemes identification, word recognition, syntax, phrase, ..., analyze separately the incoming voice signal, make hypotheses about the utterances and post these partial and uncertain hypotheses on a "blackboard" so that other modules can use them as input and can raise or lower their confidence. A result is output when a coherent solution is found with enough confidence level.

4. Collaborative Clustering: the What and How issues

Given that collaborative clustering is defined by the fact that a set of "base methods" or "experts" exchange information while they perform their local task, information that may influence their running, a set of questions arise in order to implement such collaborative methods. Among them, and quite prominent are:

1. What information should (optimally) be exchanged between the algorithms?
2. How to measure the diversity of the methods (agreement)?
3. How to measure the performance of each method?
4. How to measure the quality of the common goal if there is one?
5. How to control the collaborative process? This, in turn, entails questions about the estimation of degree of advancement by the algorithms and about the measurement of diversity of the local results and/or local algorithms.

4.1. Communication: what should be transferred between experts?

The base methods can exchange information of different types:

1. About the candidate structures hypothesized in the data sets.
2. About the memberships of the instances in the data set.

Information on the structure.. The exchange of information can bear on defining characteristics of the structure that are looked for by the "experts". For instance, the contributing algorithms could communicate about the attributes they have found useful, or about the distance(s) they use, or the number of clusters they contemplate. This will be especially helpful in horizontal clustering where the local methods do not operate on the same data points, or even in the same input spaces.

One advantage of this communication channel is that it preserves the secrecy of the objects. It is also parsimonious in that it usually involves only a few values.

Information on the data points.. The other main option is to exchange information about the hypothesized membership of the data points. Often, this is done through the communication of the subgroups of objects found so far. Of course, this requires that there exists an agreement on the identity of the objects. One difficulty is that each algorithm uses its own labeling of the subgroups and, therefore, a translation has to be made by each local algorithm in order to compare its own result with the ones that are communicated.

This type of exchange is more onerous than the one involving structural information since it is proportional to the number of instances in the data set. Furthermore, it also generally requires more computations of each of the local method.

Aside from the type of information exchanged between experts, one question regards the directionality and the intensity of the communication. For instance, suppose that an expert appears very good according to a measure of performance, and much more so than another one. Should they exchange information in a symmetric way? The question is at least worth considering, and is not settled. This is also tied to the way performance is measured (see 4.3).

4.2. Diversity: what methods should be selected?

In most existing ensemble clustering methods, the whole set of available base methods is used in order to obtain a final result. This stems from the hypothesis that all methods are relevant to the task, at least to some (positive) extent, and should therefore be useful. This, however, is questionable.

It is obvious that if there is no variety in the ensemble of methods, nothing can be gained through collaboration. Diversity is thus a requirement. But to what extent?

If the experts must at least "speak the same language", e.g. use the same input space or the same set of data points, do they have also to agree to a minimal level in order to contribute constructively to each other task? This is no simple question, and if there have been some works aimed at exploring this issue, their findings are not quite conclusive since many factors are at play to explain the results. It is however tempting to hypothesize that in order

to have the most fruitful collaboration, an intermediate level of diversity between the experts should be struck.

In [42], the authors suggest a method in order to measure the diversity of local experts in an unsupervised task by looking at how much more they agree on the specific data set as compared to a randomized one. They then use that measure in order to weight the contributions of the experts.

We go back to this issue when discussing specific lines of works (see Section 5).

4.3. Level of performance of each method. How to measure it?

Measurement of performance in clustering is one of the hardest question. Or, rather, this is a question with no perfect, or best, answer. Intrinsically, clustering is an exploratory technique which aims at uncovering unsuspected structures in the data. It is therefore questionable that a definite optimization criterion can be defined a priori. It will always be biased towards some kind of regularities. If the underlying structure of the data does not obey this prescription, clustering will miss it.

There is therefore no objective way of measuring the absolute quality of a partitioning of data points. All existing classical evaluation criteria implicitly favor one type of structure above others. If the measure used is "aligned" with the type of structure that is looked for, then this a lucky conjunction and it is legitimate to use this measure as a performance gauge. Otherwise, the measure will be misleading.

Let us suppose then that the chosen criterion is adequate, and is shared by all participating algorithms. In that case, using an ensemble approach can be a good idea. Indeed, because clustering data points into subgroups involves a gigantic search space and a hard to optimize function, all algorithms realize inexact approximations of the ideal best partitioning. Mutualizing their strengths while trying to factor out their weaknesses is therefore a tempting approach.

Suppose on the other hand that there is no ground to prefer one evaluation criterion over other ones. Should one then hope that by using several algorithms, each seeking to optimize a different performance measure, one would get a better clustering result? This is a recipe to define a new performance criterion, but one difficult to characterize. And, again, there is no reason to hope that this criterion should be better adapted to the discovery of the "right" type of structures. This however can help the algorithms to escape local traps by submitting them to "alien" hypotheses on the structures.

4.4. If there is a common goal: how to measure a "consensus"?

In clustering aggregation, the goal is to reach, if possible, a better clustering of a data set using P clusterings. The idea is generally to produce a single clustering that agrees as much as possible with the P clusterings. The notion of *agreement*, or, conversely, of *disagreement*, must therefore be formalized in order to give rise to a measurement.

Formally, the collaborative framework modifies the criterion that each local algorithm A_i seeks to optimize in order to discover an underlying structure (clustering) $\mathcal{C}_i$ in the local data set $\mathcal{S}_i$. It becomes:

$$G_i(\mathcal{S}_i) \;=\; \underset{\mathcal{C}_i}{\text{Argmin}}\Big\{ f(\mathcal{S}_i) + g\big(\text{agreement}(A_i, \{A_j \;\; {\scriptstyle (1\leq j\leq P,\, j\neq i)}\})\big)\Big\}$$

where the term $g\big(\text{agreement}(A_i, \{A_j \;\; {\scriptstyle (1\leq j\leq P,\, j\neq i)}\})\big)$ provides a measure of the agreement between what has been found by the local algorithm A_i and all other collaborating algorithms. It is important to note that the *agreement* function generally does not take into account the original descriptions of the data points, but only the partitions to which they are allocated by the different clustering algorithms. Therefore, agreement must be defined as a kind of distance or similarity between assignments or between high level descriptions of the clusterings.

Suppose several clustering algorithms wish to compare their results on a common data set, how can this be done? This depends on the learning task, wether it is *consensus clustering* where the goal is to reach a common clustering for the same data set, *vertical clustering* where the description space is the same, but not the examples, and one still looks for a common shared characterization, or *collaborative clustering* where only local clusterings are looked for, but still assuming that information from other local clustering can help each local computation.

The following gives a flavor of the solutions proposed for each of these three settings.

Consensus clustering. In consensus clustering, the goal is to find one clustering for one data set based on several input clusterings. The idea, in general, is to search for a clustering G that is most similar, on average, to all the input clusterings $\mathcal{C}_i (1 \leq i \leq P)$.

One method for measuring similarity between clusterings is to rely on the *mutual information* between the assignments of the data points to the clusters in the target clustering G and the various input clusterings $\mathcal{C}_i$. Thus, for instance [23]) propose what they call the mutual information (that is not in fact the proper mutual information measure which is not symmetrical [32]):

$$I(G, \mathcal{C}_i) \;=\; \sum_{k=1}^{K(G)} \sum_{l=1}^{K(\mathcal{C}_i)} p(G^{(k)}, \mathcal{C}_i^{(l)}) \log\left(\frac{p(G^{(k)}, \mathcal{C}_i^{(l)})}{p(G^{(k)})\, p(\mathcal{C}_i^{(l)})}\right)$$

The goal is to find the optimal reference clustering G with respect to this measure, which results in a difficult combinatorial optimization problem which has been shown to be NP-complete. Heuristics exist that lead to approximate solutions, but guidance is necessary as to which

heuristic to use depending on the characteristics of the data sets and the number of clusters that are looked for.

An alternative strategy relies on an *iterative procedure* where input clusterings are computed at successive time steps, compared thanks to some consensus measure, and iteratively modified until convergence to a unique clustering is obtained, or some stopping criterion is satisfied. This obeys the schema of Figure 2, and is a form of collaborative clustering where all experts work on the same data set.

One solution to define the "similarity" between clusterings is to rely on a consensus measure that compares the partitions in which the data points are cast by each input clustering. A perfect consensus is obtained when the partitions induced by the various clusterings match perfectly.

The archetypal approach for measuring consensus is the so-called *consensus matrix*. This is a $N \times N$ matrix of which each element $M_{i,j}$ stores, for each pair of examples i and j, the proportion of clusterings in which the two examples have been clustered together. Thus, for each participating algorithm A_ℓ, we have what is called a *connectivity matrix*:

$$M^{(\ell)}(i,j) \;=\; \begin{cases} 1 & \text{if examples } i \text{ and } j \text{ belong to the same cluster} \\ 0 & \text{otherwise} \end{cases}$$

The *consensus matrix* can then be defined as:

$$M(i,j) \;=\; \frac{1}{P}\sum_{\ell=1}^{P} M^{(\ell)}(i,j)$$

where P is the number of algorithms.

Each entry of the consensus matrix is a real number between 0 and 1, a perfect consensus matrix being one with only entries that are 0 or 1.

Other techniques to define similarity between input clusterings have been proposed, noticeably methods based on graph structures (e.g., [43]).

Vertical clustering. If one looks for a common structure of several *different data sets*, as is done in "vertical clustering", it is no longer possible to make direct comparisons at the level of the examples since they are different. Only descriptions of the clusters found by each expert can be exchanged, and a consensus measure must be defined at this level. Very few works look at this framework. [44] is a noteworthy exception. Here, the problem is to estimate Gaussian mixtures at different nodes of a sensor network. The authors show how to adapt an Expectation-Maximization procedure to tackle it. They formalize what is assumed a priori to exist as commonalities between supposedly independent nodes (i.e. a common underlying signal measured with different offsets at each node and the same noise level) and what should be communicated between nodes (i.e., the estimated values of the mixtures parameters).

Collaborative clustering. In collaborative clustering, one is interested in the local clusterings attained by each local algorithm on the local data sets. In this setting, the notion of consensus becomes even more slippery than for vertical clustering. It has to be defined to reflect the kind of information that one would deem interesting to share between the local algorithms. It could be that the expected number of clusters is very similar, or that the hierarchical structure of the clustering is the same, or any other such expectation that can be translated into constraints that local clustering algorithms can send to each other and, in return, take into account.

One instance of a consensus function with an associated distributed clustering algorithm can be found in [45] in the context of discovering clusters of pixels in satellite images.

4.5. How to control the collaboration?

As soon as a collection of "agents" is interacting in pursuit of some task(s), several control strategies are possible. This is also true in the case of collaborative clustering. The set of possibilities can be organized along the following, strongly dependent, dimensions:

- *Synchronous* versus *asynchronous* operations. The experts can be left functioning at their own pace and exchanging information as leisurely or as rapidly as they see fit. This asynchronous strategy is best when each expert has its own goal and exchanges information only insofar that it may help in the pursuit of its local goal. When all experts are involved in a single overall task, synchronicity is usually required since the final result depends (equally) on all local achievements. The choice is obviously heavily dependent upon the type of process: iterative or one-shot.

- *Iterative* versus *one-shot* process. In one-shot process, all experts compute their local solution, and when this is done, and only then, a master algorithm combines them and outputs the final solution. Experts may well vary in the time they take to produce a solution, the master algorithm must wait until the slowest one has contributed before starting the combination process. Most techniques proposed for consensus clustering use this type of control strategy because there is only one clustering pass before the combination process takes place.
  When, by contrast, the computations performed by each expert can take into account partial solutions communicated by other experts, and is therefore iterative, synchronicity becomes an issue. Either, the process is organized with a master clock: each expert must produce and communicate its temporary solution before another clustering phase starts, or the agents communicate freely, asynchronously, at their own initiative. This later type of organization, or lack thereof, can be encountered in collaborative clustering, where each expert pursues its own local

clustering task. However, most proposals use a synchronous control strategy.

- *Local* versus *global* control. Local control goes generally hand in hand with an asynchronous control strategy, while global control is linked to the need for a master clock, and often also with the computation of a final combined overall solution.

One central concern with collaborative approaches is the *termination property*. While it is required that a clustering algorithm stops at somme point, even though the solution reached might not correspond to a global optimum of the, sometimes hidden, optimization criterion, it might be difficult to ensure that all algorithms stop when collaborating with other clustering algorithms in a iterative way. Indeed, it is easy to imagine cycles where one agent A_i produces some temporary solution $\mathcal{C}_i(t)$ at step t that modifies the solution $\mathcal{C}_j(t+1)$ produced by another agent A_j at time $t+1$, which in turn pushes A_i to change its solution at time $t+2$ to $\mathcal{C}_i(t+2)$, only to influence A_j to revert to its earlier solution $\mathcal{C}_ijt)$ at time $t+3$, and so on.

In the same way that convergence is an issue in multi-agent systems, it is a central question in iterative collaborative clustering. One general method is to define an "energy function" that is guaranteed to decrease with each iteration. This is, for instance, the approach taken in [45].

5. Related work

Collaborative learning denotes distributed algorithms that influence each other during their own computations by exchanging information. We already underlined the pioneering works of Oliver Selfridge on the Pandemonium in 1958 [46, 41] and the highly sophisticated Hearsay II system developed for speech understanding in the 1970s [38]. The co-learning framework developed for supervised classification from partial and disjoint views [47] has also a special place in the history of collaborative learning since it was based on a formal analysis leading to the definition of the algorithm.

Regarding unsupervised learning, algorithms for co-clustering or bi-clustering, initialized by [48], can be considered as forerunners of what was not yet called collaborative clustering. Co-clustering is indeed a limit case where the algorithm can be seen as an organized interplay between two agents exchanging information about the same data set, each agent seeing only one dimension (e.g. objects or attributes values) and trying to find clusters along this dimension taking into account what is found by the other agent. More recent work on co-clustering, which is not in the scope of this paper, can be found in [49, 50, 51].

However, this was not before the very end of the 1990s that collaborative approaches emerged as a concept in its own right worthy of interest (e.g. [52]). Many methods have been investigated from various perspectives: collaborative frameworks based on fuzzy clustering or topological maps, higher-level collaborative schemes relying on existing clustering algorithms or collaborative approaches dedicated to distributed datasets.

5.1. Implementations

5.1.1. Fuzzy collaborative clustering

A fuzzy collaborative clustering architecture is introduced by [53], in which several subsets of patterns can be processed together to find a common structure to all of them. In this system, different subsets of the initial data are processed independently. Then, each partition matrix is modified according to the other matrices found: each result produced on a subset is modified according to results found on the other subsets. Extensive experiments of the method are also proposed in [54] along with algorithmic details. An application of this collaborative fuzzy clustering method to semantic web content analysis has been proposed in [55]. The authors discuss a collaborative proximity-based fuzzy clustering and show how this type of clustering is used to discover a structure of web information in the spaces of semantics and data.

Another fuzzy collaborative framework is proposed by [56], where rough sets are used in a collaborative paradigm in which several subsets of patterns are processed together to find a common structure. A clustering algorithm is developed by integrating the advantages of both fuzzy sets and rough sets. A quantitative analysis of the experimental results is also provided for synthetic and real-world data. To tackle the problem of distributed data, [57] proposed a framework to cluster distributed classifiers. They show that clustering distributed classifiers as a preprocessing step for classifier combination enhances the achieved performance of the ensemble.

5.1.2. Topological and topographic maps

In a recent work [58], the authors present a formalism based on topological collaborative clustering using prototype-based clustering techniques. Topological maps representing different sites collaborate without having to access the original data, thus preserving their privacy. Two different approaches of collaborative clustering are given: horizontal and vertical collaboration. The strength of collaboration (exchange of confidence levels) between each pair of datasets is determined by a parameter, called coefficient of collaboration, to be estimated iteratively during the collaboration phase using a gradient-based optimization. The method comprises two steps. In the first one, the SOM (Self-Organizing Map) algorithm [59] is applied to each dataset independently. In the second one, the collaboration phase takes place to enrich the information contained in the different topological maps produced during the first step.

In [31], another collaborative clustering approach based on topographic maps is proposed. The idea is to combine

the Variational Bayesian Generative Topographic Mapping (VBGTM) and Fuzzy c-means (FCM). VBGTM was introduced as a variational approximation of Generative Topographic Mapping (GTM) to control data overfitting, but when the number of latent points is large, similar units need to be clustered to facilitate quantitative analysis of the map and the data. FCM is used to determine the prototypes as well as the clusters and the corresponding membership functions of the input data, based on the latent variables obtained from VBGTM.

5.1.3. Collaboration of existing clustering methods

Besides the development of any sort of specific collaborative methods, general frameworks describing how clustering methods can work together have been proposed.

For example, [60] presented a new architecture for collaboration, in order to help a set of clustering algorithms to reach an agreement on the partitioning of a common dataset. In this setting, the collaboration aims at making the methods agree on the partitioning through a refinement of their results.

Another collaborative framework is given in [61], which goal is to solve multi-view and alternative clustering problems. Ensemble clustering and semi-supervised clustering principles are used to control different clustering algorithms by sharing a common model. The aim of the method is to reach a consensus or alternatively to improve local clustering solutions.

A similar approach has been proposed by [21]. The authors defined a new cooperative clustering model which involves cooperation among multiple clustering techniques to increase the homogeneity of objects within the clusters. The model is capable of handling datasets with different properties by developing two data structures: a histogram representation of the pair-wise similarities and a cooperative contingency graph. The two data structures are designed to find the matching sub-clusters between different clusterings and to obtain the final set of clusters through a coherent merging process.

More recently, [45] proposed a new collaborative framework that works with most clustering algorithms. The collaboration scheme used is horizontal collaboration (see Section 2). All algorithms work either on subsets representing the same data in different feature spaces, or on the exact same data searching for a different number of clusters, or a mix of both.

5.1.4. Distributed data

Due to the recent growth of data automatically gathered, there is a growing need of efficient methods dealing with distributed data.

[62] defined a collaborative clustering approach that focuses on distributed or ubiquitous knowledge discovery, when only the access to the local dataset is available. The authors present two different versions of the method, one where the data are the same but described by different features, and one where the features are the same, but the objects of each local dataset are different. The approach consists in two step. First a collaborative fuzzy clustering step and then a particle swarm optimization to optimize the collaboration.

More recently, [63] proposed a new efficient way to deal with large distributed datasets. The method is based on a collaborative divide-and-conquer algorithm using k-means as base clustering algorithm. The collaboration consists in the exchange of seeds between the clustering methods to accelerate the convergence in each partition.

5.2. Dimensions of Collaborative Methods for Clustering

To sum up, several dimensions can be useful to characterize the tasks and methods of collaborative clustering:

1. single objective vs. multiobjective clustering;
2. global clustering vs. local clusterings;
3. single domain clustering vs. multi-domain clustering;
4. clustering of single scale data vs. multiscale data;
5. single strategy vs. multi-strategy approaches;
6. exchange of information on the membership of the samples to cluster vs. exchange of information on the clustering models.

Table 1 proposes a classification of the main methods presented in this paper according to these characteristics.

5.2.1. Single objective vs. multiobjective clustering

The notion of *objective* relates to the criterion or criteria to optimize.

It may be convenient to define a single expression as a combination (or compromise) between several targets to maximize. For example, it is common in clustering to seek to maximize both clusters compactness (intra-cluster similarity) and difference (inter-cluster dissimilarity). This can be expressed by a criterion: $\texttt{compacity}(C)+\lambda\texttt{.dissimilarity}(C)$, where C is a clustering and λ a parameter to control the relative importance of the two targets. In this case, we talk about *single objective clustering*.

By contrast, the goal of *multiobjective clustering* is to find clusters in a data set by applying several clustering algorithms corresponding to different objective functions (or criteria). A final clustering corresponding to a trade-off between the base criteria can eventually be derived using a Pareto front.

5.2.2. Global clustering vs. local clusterings

The goal of the collaboration can either be to produce a unique partition of the whole dataset: *global clustering*, or it may be to find partitions of subsets of the data, whether they are described by the same attributes or not: *local clustering*.

In the case of a global clustering, the final partition results from a consensus between the solutions produced by the different methods. This consensus function can rely directly on the partitions of the data set produced by the methods, or may take into account the (parametric) models produced by these methods.

5.2.3. Single domain clustering vs. multi-domain clustering

In the *single-domain* case, all objects belong to the same super-class of objects (for example, furniture or living beings). In the *multi-domain* case, the objects categorized by the various local methods may belong to different superclasses (for instance, it could be interesting to see in which way and to what extent insects in a colony and citizens in an economic system share criteria or model parameters).

5.2.4. Clustering of single scale data vs. multiscale data

In some other cases, the data correspond to the same items, but are described at different scales. This is especially the case for remote sensing images for example, but it can also involve objects described at different levels of abstraction.

In all these cases, it may be beneficial to have clustering tools operating at different levels of description to collaborate in order to produce a more informed view of the data.

5.2.5. Single strategy vs. multi-strategy approaches

In a *single strategy* approach, all collaborating methods are based on the same algorithm, but working on different data or having different parameter values. Exchange of information between these methods is easy because they share the same settings.

It is harder to benefit from the collaboration of different methods, like, for example, a k-means algorithm and a hierarchical clustering algorithm, because of the lack of direct correspondence between solutions. The same kind of problem arises in multi-objective clustering.

5.2.6. Exchange of information on the membership of the samples to cluster vs. exchange of information on the clustering models

One way to address the issue of exchanging information between algorithms that differ in their functioning is to communicate at the level of the membership of each data point to the clusters in each local solution. This does not require that the algorithms share any information about their internal state. However, exchanging information about these internal states can enrich communication and speed up the collaborative process. For instance, algorithms based on mixture of probabilities distributions could find useful to exchange information about the shapes of these distributions.

5.3. Applications

Most of the work presented in this section have been compared in the literature on the classical UCI datasets. One reason for this choice was the high execution cost of many of these algorithms, making them hard to execute on large datasets. However, some were used on real data to solve concrete problems.

	single vs. multiobjective	global vs. local	multi-domain	multiscale data	multi-strategy	exchange membership/models
[53]	single	global				membership
[57]	single	local	✓		✓	models
[58]	single	local	✓			models
[31]	multi	local	✓			models
[60]	single	global			✓	membership
[61]	multi	local	✓	✓	✓	models
[21]	single	global				membership
[45]	single	global			✓	membership
[62]	single	local	✓		✓	models
[63]	single	global	✓			membership

Table 1: Classification of the presented collaborative methods, according to the six main characteristics described in Section 5.2.

For example, [64] developed a 3-D motion segmentation method, based on collaborative clustering. The segmentation is computed from two perspective views. A multi-view extension of the Sparse Subspace Clustering [65] algorithm is proposed to combine the information across multiple image frames.

In [66] a study was published on the topic of marketing research. The authors used a multi-objective particle swarm optimization (MOPSO) within a collaborative fuzzy clustering framework. The main contributions of this work consist, first, in giving a method to compute the collaboration matrix between the different data repositories, and second, in proposing a using fitness functions at both levels (data and information levels), that allow one to build a consensus between all the data sites.

In [67], the authors presented an application of their collaborative clustering approach to remote sensing image classification. This multi-strategic method integrates different kinds of clustering algorithms that collaborate to produce a unique consensual result. The paper highlights how clustering methods can collaborate and presents results in the paradigm of object-oriented classification of a very high resolution remotely sensed images of an urban area.

6. Conclusion and Perspectives

Despite the increasing number of methods and tools dedicated to unsupervised collaborative classification, this paradigm is still surprisingly infrequently used. However, the emergence of the Big Data area, with all its demand for distributed but collaborative computations should radically change this situation. Moreover, while, until recently, "ensemble methods" were mostly aimed at supervised learning, a shift of attention towards unsupervised learning is taking place. This is due in part to the pressing need for methods that allow one to explore large data

sets without well-defined preconceptions, and without pre-existing categories that can be used as labels in training instances.

In this paper, we have underlined the differences between approaches that aim at combining clusterings: multi-view clustering, cooperative or consensus clustering, and collaborative clustering. In the latter, the local algorithms usually process different data sets, either because the instances or the descriptors, or both, differ. In addition, and this is essential, these algorithms seek to find structures in the data that may differ from the structures found by other algorithms. However, they are ready to use information coming from the other algorithms in order to discover better structures than if they were operating in isolation. Another key feature of the collaborative clustering framework is that the local algorithms may exchange information among themselves iteratively, and not in prelude of a final consensus making phase.

Still, for all the surge of interest on collaborative clustering, there remain many venues to explore and numerous questions to settle. In this paper, we have tried to expound a systematic set of characteristics that can be used in order to describe algorithms, and above all to devise new schemes. But above all, we have underlined a set of issues and questions that any method must face: the type of information that is exchanged, the protocol controlling the communications, the way that differences are resolved at each node, the stopping criteria, to name the foremost ones.

Even though there now exists a panoply of methods and of successful experiments, we are still lacking a theoretical understanding about the conditions for positive collaboration in clustering tasks. All situations cannot be conducive to fruitful collaboration. But in which ways the local tasks should be related for a fruitful collaboration to be possible? To what extent the local clustering algorithm should share their a priori on the interesting data structures? What information should they exchange? And among which set of collaborators? Is there a way to detect negative collaborations? Finally, in a distributed and collaborative framework, how to define a stopping criterion? All these questions are still largely open.

This survey paper has tried to organize and convey the main issues facing the development of collaborative clustering methods. This is an exciting and promising field that arrives on time for solving knowledge discovery problems in the big data area. We believe that the time is ripe for largely extending the use of these methods. All the while, this is also a domain rich with questions that go beyond the current statistical learning theory, and that should stimulate very interesting new research efforts.

References

[1] Jain, A.K.. Data clustering: 50 years beyond k-means. Pattern Recogn Lett 2010;31(8):651–666.

[2] Shalev-Shwartz, S., Ben-David, S.. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press; 2014.

[3] Ackerman, M., Ben-David, S., Loker, D.. Towards property-based classification of clustering paradigms. In: Advances in Neural Information Processing Systems. 2010, p. 10–18.

[4] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., Tibshirani, R.. The elements of statistical learning. Springer; 2009.

[5] Bock, H.H.. On some significance tests in cluster analysis. Journal of classification 1985;2(1):77–108.

[6] Kass, R.E., Raftery, A.E.. Bayes factors. Journal of the american statistical association 1995;90(430):773–795.

[7] Handl, J., Knowles, J., Kell, D.B.. Computational cluster validation in post-genomic data analysis. Bioinformatics 2005;21(15):3201–3212.

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

[9] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.. On Combining Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998;20(3):226–239.

[10] Topchy, A., Jain, A.K., Punch, W.. Combining multiple weak clusterings. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE; 2003, p. 331–338.

[11] Alpaydin, E., Kaynak, C.. Cascading classifiers. Kybernetika 1998;34(4):369–374.

[12] Pedrycz, W.. Knowledge-Based Clustering: From Data to Information Granules . Wiley-Interscience; 2007. ISBN 9780471469667.

[13] Dietterich, T.G.. Ensemble Methods in Machine Learning. In: Multiple Classifier Systems, LBCS-1857. Springer; 2000, p. 1–15.

[14] Kuncheva, L.I.. Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience; 2004.

[15] Schapire, R.E., Freund, Y.. Boosting: Foundations and Algorithms. MIT Press; 2012.

[16] Zhi-Hua, Z.. Ensemble Methods: Foundations and Algorithms. CRC Press; 2012.

[17] Wemmert, C., Gançarski, P.. A Multi-View Voting Method to Combine Unsupervised Classifications. In: Artificial Intelligence and Applications. Malaga, Spain; 2002, p. 447–452.

[18] Nguyen, N., Caruana, R.. Consensus Clusterings. In: International Conference on Data Mining. IEEE Computer Society; 2007, p. 607–612.

[19] Ayad, H., Kamel, M.S.. Cumulative Voting Consensus Method for Partitions with Variable Number of Clusters. IEEE Transactions on Pattern Analysis and Machine Intelligence 2008;30(1):160–173.

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

[21] Kashef, R., Kamel, M.S.. Cooperative clustering. Pattern Recognition 2010;43(6):2315–2329.

[22] Davis, D., Varghese, B.G.. Survey on cooperative clustering models. International Journal of Engineering Research & Technology 2013;2(12):1810–1815.

[23] Strehl, A., Ghosh, J.. Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions. Journal on Machine Learning Research 2002;3:583–617.

[24] Topchy, A.P., Jain, A.K., Punch, W.F.. Combining Multiple Weak Clusterings. In: International Conference on Data Mining. IEEE Computer Society; 2003, p. 331–338.

[25] Zhou, Z.H., Tang, W.. Clusterer ensemble. Knowledge-Based Systems 2006;19(1):77–83.

[26] Kuncheva, L.I., Hadjitodorov, S.T., Todorova, L.P.. Experimental Comparison of Cluster Ensemble Methods. In: Information Fusion, 2006 9th International Conference on. 2006, p. 1–7.

[27] Gionis, A., Mannila, H., Tsaparas, P.. Clustering aggregation. ACM Transactions on Knowledge Discovery from Data (TKDD)

2007;1(1):4.
[28] Domeniconi, C., Al-Razgan, M.. Weighted cluster ensembles: Methods and analysis. ACM Transactions on Knowledge Discovery from Data (TKDD) 2009;2(4):17.
[29] Vega-Pons, S., Ruiz-Shulcloper, J.. A survey of clustering ensemble algorithms. International Journal of Pattern Recognition and Artificial Intelligence 2011;25(03):337–372.
[30] Monti, S., Tamayo, P., Mesirov, J., Golub, T.. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Machine learning 2003;52(1-2):91–118.
[31] Ghassany, M., Grozavu, N., Bennani, Y.. Collaborative generative topographic mapping. In: Neural Information Processing. Springer; 2012, p. 591–598.
[32] MacKay, D.J.. Information theory, inference and learning algorithms. Cambridge University Press; 2003.
[33] Horn, J., Nafpliotis, N., Goldberg, D.E.. A niched pareto genetic algorithm for multiobjective optimization. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. Ieee; 1994, p. 82–87.
[34] Neal, R.M., Hinton, G.E.. A view of the em algorithm that justifies incremental, sparse, and other variants. In: Learning in graphical models. Springer; 1998, p. 355–368.
[35] Nock, R., Nielsen, F.. An abstract weighting framework for clustering algorithms. In: Proceedings of the 2004 SIAM International Conference on Data Mining. SIAM; 2004, p. 200–209.
[36] Spitkovsky, V.I., Alshawi, H., Jurafsky, D.. Lateen em: Unsupervised training with multiple objectives, applied to dependency grammar induction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics; 2011, p. 1269–1280.
[37] Gimpel, K., Smith, N.A.. Concavity and initialization for unsupervised dependency parsing. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics; 2012, p. 577–581.
[38] Barr, A., Feigenbaum, E., Roads, C.. The Handbook of Artificial Intelligence, Volume 1. 1982.
[39] Pieraccini, R.. The voice in the machine: building computers that understand speech. MIT Press; 2012.
[40] Minsky, M.. The Society of Mind. Simon & Schuster; 1985.
[41] Nilsson, N.J.. The quest for artificial intelligence. Cambridge University Press; 2010.
[42] Cornuéjols, A., Martin, C.. Unsupervised one class identification by selecting and combining ranking functions. In: Proceedings of the 2015 Conférence sur l'Apprentissage Automatique. 2015,.
[43] Lancichinetti, A., Fortunato, S.. Consensus clustering in complex networks. Scientific reports 2012;2.
[44] Fagnani, F., Fosson, S.M., Ravazzi, C.. Consensus-like algorithms for estimation of gaussian mixtures over large scale networks. Mathematical Models and Methods in Applied Sciences 2014;24(02):381–404.
[45] Sublime, J., Grozavu, N., Bennani, Y., Cornuéjols, A.. Collaborative clustering with heterogeneous algorithms. In: Neural Networks (IJCNN), 2015 International Joint Conference on. IEEE; 2015, p. 1–8.
[46] Selfridge, O.G.. Pandemonium: a paradigm for learning in mechanisation of thought processes. HMSO; 1958.
[47] Blum, A., Mitchell, T.. Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory. ACM; 1998, p. 92–100.
[48] Hartigan, J.A.. Direct clustering of a data matrix. Journal of the american statistical association 1972;67(337):123–129.
[49] Cleuziou, G., Exbrayat, M., Martin, L., Sublemontier, J.H.. Cofkm: A centralized method for multiple-view clustering. In: Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on. IEEE; 2009, p. 752–757.
[50] Yan, Y., Chen, L., Tjhi, W.C.. Fuzzy semi-supervised co-clustering for text documents. Fuzzy Sets and Systems 2013;215:74–89.
[51] Jiang, Y., Chung, F.L., Wang, S., Deng, Z., Wang, J., Qian, P.. Collaborative fuzzy clustering from multiple weighted views. IEEE transactions on cybernetics 2015;45(4):688–701.
[52] Johnson, E.L., Kargupta, H.. Collective, hierarchical clustering from distributed, heterogeneous data. In: Large-Scale Parallel Data Mining. Springer; 2000, p. 221–244.
[53] Pedrycz, W.. Collaborative fuzzy clustering. Pattern Recognition Letters 2002;23(14):1675–1686.
[54] Pedrycz, W., Rai, P.. A Multifaceted Perspective at Data Analysis: A Study in Collaborative Intelligent Agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 2008;38(4):1062–1072.
[55] Loia, V., Pedrycz, W., Senatore, S.. Semantic Web Content Analysis: A Study in Proximity-Based Collaborative Clustering. Fuzzy Systems, IEEE Transactions on 2007;15(6):1294–1312.
[56] Mitra, S., Banka, H., Pedrycz, W.. Rough-fuzzy collaborative clustering. IEEE Transactions on Systems, Man, and Cybernetics 2006;36:795–805.
[57] Tsoumakas, G., Angelis, L., Vlahavas, I.. Clustering classifiers for knowledge discovery from physically distributed databases. Data & Knowledge Engineering 2004;49(3):223–242.
[58] Grozavu, N., Bennani, Y.. Topological collaborative clustering. Australian Journal of Intelligent Information Processing Systems 2010;12(2).
[59] Kohonen, T.. The self-organizing map. Proceedings of the IEEE 1990;78(9):1464–1480.
[60] Forestier, G., Gançarski, P., Wemmert, C.. Collaborative clustering with background knowledge. Data & Knowledge Engineering 2010;69(2):211–228.
[61] Sublemontier, J.H.. Unsupervised collaborative boosting of clustering: a unifying framework for multi-view clustering, multiple consensus clusterings and alternative clustering. In: International Joint Conference on Neural Networks (IJCNN 2013). Dallas, United States; 2013,.
[62] Depaire, B., Falcón, R., Vanhoof, K., Wets, G.. PSO driven collaborative clustering: A clustering algorithm for ubiquitous environments. Intelligent Data Analysis 2011;15(1):49–68.
[63] Cui, H., Ruan, G., Xue, J., Xie, R., Wang, L., Feng, X.. A Collaborative Divide-and-conquer K-means Clustering Algorithm for Processing Large Data. In: Proceedings of the 11th ACM Conference on Computing Frontiers. 2014, p. 20:1–20:10.
[64] Li, Z., Guo, J., Cheong, L.F., Zhou, S.. Perspective motion segmentation via collaborative clustering. In: Proceedings of the IEEE International Conference on Computer Vision. 2013, p. 1369–1376.
[65] Elhamifar, E., Vidal, R.. Sparse subspace clustering: Algorithm, theory, and applications. Pattern Analysis and Machine Intelligence, IEEE Transactions on 2013;35(11):2765–2781.
[66] Falcón, R., Depaire, B., Vanhoof, K., Abraham, A.. Towards a suitable reconciliation of the findings in collaborative fuzzy clustering. In: Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on; vol. 3. IEEE; 2008, p. 652–657.
[67] Forestier, G., Wemmert, C., Gancarski, P.. Collaborative multi-strategical clustering for object-oriented image analysis. In: Studies in Computational Intelligence; vol. 126. Springer; 2008, p. 71–88.