k -CEVCLUS: Constrained evidential clustering of large dissimilarity data

Feng Li, Shoumei Li, Thierry Denoeux

k-CEVCLUS: Constrained Evidential Clustering of Large Dissimilarity Data[☆]

Feng Li[a], Shoumei Li[a], Thierry Denœux[a,b,*]

[a] *Beijing University of Technology, College of Applied Sciences, Beijing, China.*
[b] *Sorbonne Universités, Université de Technologie de Compiègne, CNRS, Heudiasyc (UMR 7253), France.*

Abstract

In evidential clustering, cluster-membership uncertainty is represented by Dempster-Shafer mass functions. The EVCLUS algorithm is an evidential clustering procedure for dissimilarity data, based on the assumption that similar objects should be assigned mass functions with low degree of conflict. CEVCLUS is a version of EVCLUS allowing one to use prior information on cluster membership, in the form of pairwise must-link and cannot-link constraints. The original CEVCLUS algorithm was shown to have very good performances, but it was quite slow and limited to small datasets. In this paper, we introduce a much faster and efficient version of CEVCLUS, called k-CEVCLUS, which is both several orders of magnitude faster than EVCLUS and has storage and computational complexity linear in the number of objects, making it applicable to large datasets (around 10^4 objects). We also propose a new constraint expansion strategy, yielding drastic improvements in clustering results when only a few constraints are given.

Keywords: Evidence theory, Dempster-Shafer theory, belief functions, relational data, credal partition, constrained clustering, instance-level constraints

1. Introduction

Cluster analysis, also called data segmentation, is one of the basic tasks in data mining and machine learning. The goal of cluster analysis is to segment a collection of objects into clusters in such a way that similar objects belong to the same cluster, while dissimilar ones are assigned to different clusters. Typically, two data types are considered: *attribute* and *dissimilarity* data. Dissimilarity data, also known as *relational* data or *proximity* data, are composed of distances,

[☆]This research was supported by grant No.11571024 from NSFC, and by the Overseas Talent program from the Beijing Government.

[*]Corresponding author
Email address: tdenoeux@utc.fr (Thierry Denœux)

or dissimilarities between objects. Attribute data can always be transformed into dissimilarity data by using a suitable metric. In this paper, we focus mostly on dissimilarity data.

Several approaches to clustering have been developed over the years. In hard clustering, each object is assigned with full certainty to one and only one cluster; the c-means algorithm is the reference method in this category. In contrast, "soft" clustering algorithms [22] are based on different ways of representing cluster-membership uncertainty. These include fuzzy [3], possibilistic [14] and rough [17] clustering. *Evidential clustering* [9, 20, 8, 18] is a recent approach to soft clustering, in which uncertainty is represented by Dempster-Shafer mass functions [25]. The resulting clustering structure is called *credal partition*. Thanks to the generality of Dempster-Shafer theory, evidential clustering can be shown to extend all other soft clustering paradigms [7]. Some of the recent advances in evidential clustering are briefly summarized here. The Evidential c-Means (ECM) algorithm [20] is an extension of the hard and fuzzy cMeans, in which prototypes are defined not only for clusters, but also for sets of clusters. A cost function is minimized in turn with respect to the prototypes, and with respect to the credal partition. A version of ECM for dissimilarity data, called RECM, was proposed in [21]. In [18], another variant of the ECM algorithm (called CCM) was proposed, based on an alternative definition of the distance between a vector and the prototype of a meta-cluster. This modification produces more sensible results in situations where the prototype of a meta-cluster is close to that of singleton cluster. In [27], Zhou et al. introduced yet another variant of ECM, called Median Evidential c-means (MECM), which is an evidential counterpart to the median c-means and median fuzzy c-means algorithms. An advantage of this approach is that it does not require the dissimilarities between objects to verify the axioms of distances. In [8], the author proposed another evidential clustering method, called Ek-NNclus, which is based on evidential k-nearest neighbor rule [5]. Evidential clustering has been successfully applied in various fields, including machine prognosis [24], medical image processing [19, 15, 16] and analysis of social networks [27].

The notion of credal partition was first introduced in [9], together with the first evidential clustering algorithm, called EVCLUS. The EVCLUS algorithm is similar in spirit to multidimensional scaling procedures [4]. It attempts to build a credal partition such that the plausibility of two objects belonging to the same cluster is higher when the two objects are more similar. This result is achieved by minimizing a stress, or cost function using a gradient-based optimization procedure. A constrained version of EVCLUS allowing for the utilization of prior knowledge about the joint cluster membership of object pairs was later proposed in [1] under the name CEVCLUS. In the CEVCLUS method, pairwise constraints are formalized in the belief function framework and translated as a penalty term added to the stress function of EVCLUS.

Both EVCLUS and CEVCLUS were shown to outperform state-of-the-art clustering procedures [9, 1]. However, their high space and time complexity restricted their application to small datasets with only a few hundred objects. Recently, a new version of EVCLUS, called k-EVCLUS, has been proposed [10].

k-EVCLUS is based on an iterative row-wise quadratic programming (IRQP) algorithm, which makes it much faster than EVCLUS. It also uses only a random sample of the dissimilarities, which reduces the time and space complexity from quadratic to linear, making it suitable to cluster large datasets.

In this paper, we carry out similar improvements to the CEVCLUS algorithm. We show that the cost function composed of a stress term and a penalty term encoding pairwise constraints can also be minimized using the IRQP algorithm, which is several orders of magnitude faster than the gradient-based procedure used in [1]. Together with dissimilarity sampling, this modification makes the new version of CEVCLUS (called k-CEVCLUS) applicable to large datasets composed of tens of thousands of objects with pairwise constraints. We also introduce a new constraint expansion strategy, which brings considerable improvements in clustering results when only a few constraints are provided. Altogether, the contributions reported in this paper considerably extend the applicability of constrained evidential clustering to real-world datasets of realistic size.

The rest of this paper is organized as follows. Basic notions on belief functions and credal partitions, as well as the k-EVCLUS and CEVCLUS algorithms are first recalled in Section 2. The new k-CEVCLUS algorithm and the constraint expansion procedure are then described in Section 3, and experimental results are reported in Section 4. Finally, Section 5 concludes the paper.

2. Background

The purpose of this section is to provide the reader with background information so as to make the paper self-contained. Basic notions of Dempster-Shafer theory are first recalled in Section 2.1, and the concept of credal partition is introduced in Section 2.2. The k-EVCLUS and CEVCLUS algorithms are then presented in Sections 2.3 and 2.4, respectively.

2.1. Mass Functions

Let $\Omega = \{\omega_1, \ldots, \omega_c\}$ be a finite set. A *mass function* on Ω is a mapping from the power set 2^Ω to $[0, 1]$, satisfying the condition

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{1}$$

Each subset A of Ω such that $m(A) > 0$ is called a *focal set*. In Dempster-Shafer theory, a mass function encodes a piece of evidence about some question of interest, for which the true answer is assumed to be an element of Ω. For any nonempty focal set A, $m(A)$ is a measure of the belief that is committed exactly to A [25]. The mass $m(\emptyset)$ assigned to the empty set has a special interpretation: it is a measure of the belief that the true answer might not belong to Ω. As we will see, this quantity is very useful in clustering to identify outliers. A mass function is said to be

- *Bayesian* if all its focal sets are singletons;
- *Logical* if it has only one focal set;
- *Certain* if it is both logical and Bayesian;
- *Consonant* if its focal sets are nested.

Given a mass function m, the corresponding *belief* and *plausibility* functions are defined, respectively, as

$$Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B)$$

and

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B),$$

for all $A \subseteq \Omega$. The quantity $Bel(A)$ represents the degree of total support in A, while $Pl(A)$ can be interpreted as the degree to which the evidence is consistent with A.

The *degree of conflict* [25] between these two mass functions m_1 and m_2 defined on the same frame Ω is

$$\kappa = \sum_{A \cap B = \emptyset} m_1(A) m_2(B). \tag{2}$$

If m_1 and m_2 are mass functions representing evidence about two distinct questions with the same set of possible answers Ω, then the plausibility that the two questions have the same answer is equal to $1 - \kappa$ [9].

2.2. Credal Partition

Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be a set of n objets. We assume that each object belongs to at most one of c clusters. The set of clusters is denoted by $\Omega = \{\omega_1, \ldots, \omega_c\}$. In evidential clustering, the uncertainty about the cluster membership of each object o_i is represented by a mass function m_i on Ω. The n-tuple $\mathcal{M} = (m_1, \ldots, m_n)$ is called a *credal partition*. The notion of credal partition is very general and it encompasses most other types of soft clustering structures [7]. In particular,

- If all mass functions m_i are certain, then we have a hard partition, where object o_i is assigned to cluster ω_k if $m_i(\{\omega_k\}) = 1$.
- If all mass functions m_i are Bayesian, then the evidential partition is equivalent to a fuzzy partition; the degree of membership of object o_i to cluster ω_k is then $u_{ik} = m_i(\{\omega_k\})$, for $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, c\}$.
- If all mass functions m_i are logical with a single focal set $A_i \subseteq \Omega$, then we get a rough partition. The lower and upper approximations of cluster k can be defined, respectively, as $\underline{\omega}_k = \{o_i \in \mathcal{O} | A_i = \{\omega_k\}\}$ and $\overline{\omega}_k = \{o_i \in \mathcal{O} | \omega_k \in A_i\}$.

- If each m_i is consonant, then it is equivalent to a possibility distribution, and it can be uniquely represented by the plausibility of the singletons $pl_{ik} = Pl_i(\{\omega_k\})$ for $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, c\}$. Each number pl_{ik} is the plausibility that object i belongs to cluster k; these numbers form a possibilistic partition of the n objects.

Because a credal partition is more general than other types of hard or soft partitions, it can be converted into any other type [7]. For instance, we obtain a fuzzy partition by defining the degree of membership u_{ik} of object o_i to cluster ω_k as

$$u_{ik} = \frac{pl_{ik}}{\sum_{\ell=1}^{c} pl_{i\ell}}. \tag{3}$$

This fuzzy partition can then be converted to a hard partition by assigning each object to the cluster with the highest membership degree.

2.3. k-EVCLUS Algorithm

The k-EVCLUS algorithm [10, 6] is a faster and more efficient version of the EVCLUS algorithm introduced in [9]. Let $\boldsymbol{D} = (d_{ij})$ be an $n \times n$ matrix of dissimilarities between object. The basic idea of EVCLUS and k-EVCLUS is to construct a credal partition $\mathcal{M} = (m_1, \ldots, m_n)$ in such a way that the degrees of conflict κ_{ij} between any two mass functions m_i and m_j match the dissimilarities d_{ij}. Therefore, similar objects should be assigned mass functions with low conflict (i.e., a high plausibility of belonging to the same cluster), whereas dissimilar objects should have highly conflicting mass functions (corresponding to a low plausibility of belonging to the same cluster). This principle can be implemented by minimizing a stress function such as

$$J(\mathcal{M}) = \eta \sum_{i<j} (\kappa_{ij} - \delta_{ij})^2, \tag{4}$$

where $\eta = \left(\sum_{i<j} \delta_{ij}^2\right)^{-1}$ and the δ_{ij} are transformed dissimilarities defined by $\delta_{ij} = \varphi(d_{ij})$, where φ is an increasing function from $[0, +\infty)$ to $[0, 1]$, such as

$$\varphi(d) = 1 - \exp(-\gamma d^2). \tag{5}$$

In (5), γ is parameter that can be fixed as follows [10]. For $\alpha \in (0, 1)$, let $d_0 = \varphi^{-1}(1 - \alpha)$ be the dissimilarity value such that two objects whose dissimilarity exceeds d_0 have a plausibility at least equal to $1 - \alpha$. For φ defined by (5), we have

$$\gamma = -\log \alpha / d_0^2. \tag{6}$$

We recommend fixing $\alpha = 0.05$ and leaving d_0 as the only parameter to be adjusted. In practice, d_0 can be set to some quantile of distances d_{ij} (See Section 4.2). In [10], the results of k-EVCLUS have been shown to be quite robust to the choice of d_0. However, a smaller value of d_0 should generally be selected when the number of clusters is larger. Practical guidelines for tuning d_0 will be presented in Section 4.2.

The EVCLUS algorithm [9] minimizes a stress function similar to (4) using a gradient-based algorithm. In contrast, k-EVCLUS uses the faster IRQP algorithm [26], which consists in minimizing $J(\mathcal{M})$ with respect to each mass function m_i at a time. Each iteration of this cyclic coordinate descent strategy amounts to solving a linearly constrained positive least-squares problems, which can be done quite efficiently.

Another important innovation of the k-EVCLUS algorithm is to exploit the redundancies in matrix D by minimizing the sum of the squared error terms $(\kappa_{ij} - \delta_{ij})^2$ for only a subset of the object pairs (i, j). This can be done by choosing some value $k < n$ and randomly selecting, for each objet i, k other objects $j_1(i), \ldots, j_k(i)$. We can then minimize the sum of squared errors $(\kappa_{ij} - \delta_{ij})^2$ over the pairs $(i, j_r(i))$,

$$J_k(\mathcal{M}) = \eta \sum_{i=1}^{n} \sum_{r=1}^{k} (\kappa_{i,j_r(i)} - \delta_{i,j_r(i)})^2. \tag{7}$$

As noted in [10], the calculation of $J_k(\mathcal{M})$ requires only $O(nk)$ operations, against $O(n^2)$ for $J(\mathcal{M})$, which makes it possible to apply k-EVCLUS to very large datasets. For small datasets, we can select $k = n - 1$, in which case (7) is identical to (4).

Another important practical issue in the application of k-EVCLUS is the number of focal sets. This number must be controlled to make the method useable for moderate and large values of c. In [9] and [10], very good results have been reported with focal sets limited to the empty set, the singletons of Ω, and the whole set Ω itself. The number f of focal sets is then equal to $c + 2$, which allows the degrees of conflict κ_{ij} in (4) to be computed in linear time as a function of c, making the method useable to datasets with a large number of clusters. This restriction will be applied in all the experiments reported in Section 4.

2.4. CEVCLUS Algorithm

The constrained evidential clustering (CEVCLUS) algorithm [1] is a variant of EVCLUS that makes it possible to take into account prior knowledge about clusters, in the form of pairwise "must-link" (ML) and "cannot-link" (CL) constraints. A ML constraint is a pair of objects that are know to belong to the same cluster, while a CL constraint is pair of objects that surely belong to different clusters.

Let S_{ij} denote the event that objects i and j belong to the same cluster, and $\overline{S}_{ij}$ the complementary event. Given mass functions m_i and m_j about the cluster-membership of objects i and j, the plausibility of S_{ij} and $\overline{S}_{ij}$ can be computed as follows [2],

$$Pl_{ij}(S_{ij}) = 1 - \kappa_{ij} \tag{8a}$$

$$Pl_{ij}(\overline{S}_{ij}) = 1 - m_i(\emptyset) - m_j(\emptyset) + m_i(\emptyset)m_j(\emptyset) - \sum_{k=1}^{c} m_i(\{\omega_k\})m_j(\{\omega_k\}). \tag{8b}$$

Equation (8a) further explains the rationale for stress functions (4) and (7): when the distance between two objects i and j is large, the plausibility $Pl_{ij}(S_{ij})$ that they belong to the same cluster should be small, and the degree of conflict κ_{ij} should be large. Now, if we know for sure that two objects i and j belong to the same cluster, then we should impose the constraints $Pl_{ij}(S_{ij}) = 1$ and $Pl_{ij}(\overline{S}_{ij}) = 0$. Conversely, if we know that objects i and j actually belong to different clusters, then we should have $Pl_{ij}(\overline{S}_{ij}) = 1$ and $Pl_{ij}(S_{ij}) = 0$. To find a credal partition $\mathcal{M}$ that meets these constraints approximately, the CEVCLUS algorithm minimizes the sum of a stress function such as (4), and a penalization term:

$$J_C(\mathcal{M}) = \mathsf{stress} + \frac{\xi}{2(|\mathsf{ML}| + |\mathsf{CL}|)}(J_{\mathsf{ML}} + J_{\mathsf{CL}}), \tag{9}$$

with

$$J_{\mathsf{ML}} = \sum_{(i,j)\in\mathsf{ML}} Pl_{ij}(\overline{S}_{ij}) + 1 - Pl_{ij}(S_{ij}), \tag{10a}$$

$$J_{\mathsf{CL}} = \sum_{(i,j)\in\mathsf{CL}} Pl_{ij}(S_{ij}) + 1 - Pl_{ij}(\overline{S}_{ij}), \tag{10b}$$

where ξ is a hyperparameter that controls the trade-off between the stress and the constraints, and ML and CL are the sets of ML and CL constraints, respectively. The second term on the right-hand side of (9) equals zero for a credal partition $\mathcal{M}$ that meets the constraints exactly. In practice, it is sufficient to make it small enough so that the constraints are met approximately.

The CEVCLUS algorithm [1] minimizes (9) using an iterative gradient-based optimization procedure. As noted in [1], this algorithm is limited to small datasets. In the next section, we introduce several improvements to CEVCLUS, making it much faster and applicable to very large datasets. We also describe a new constraint expansion procedure allowing us to drastically increase the impact of pairwise constraints.

3. Constrained k-EVCLUS Algorithm

This section introduces the main contributions of this paper. The new k-CEVCLUS algorithm, a faster and more efficient version of CEVCLUS, will first be presented in Section 3.1, and the constraint expansion method will be described in Section 3.2.

3.1. k-CEVCLUS Algorithm

In section, we propose a constrained version of the k-EVCLUS algorithm described in Section 2.3. The resulting algorithm, called k-CEVCLUS, will minimize the following cost function,

$$J_{kC}(\mathcal{M}) = \eta \sum_{i=1}^{n}\sum_{r=1}^{k} \left(\kappa_{i,j_r(i)} - \delta_{i,j_r(i)}\right)^2 + \frac{\xi}{2(|\mathsf{ML}| + |\mathsf{CL}|)}(J_{\mathsf{ML}} + J_{\mathsf{CL}}), \tag{11}$$

where, as before, $j_1(i), \ldots, j_k(i)$ are k integers randomly selected in $1, 2, \ldots, i-1, i+1, \ldots, n$. In Eq. (11), the first term on the right-hand side is identical to the stress function (7) of k-EVCLUS, while the second term is equal to the penalty term (10) of CEVCLUS. We note that the calculation of $J_{Ck}(\mathcal{M})$ requires $O(nk + |\mathsf{ML}| + |\mathsf{CL}|)$ operations, against $O(n^2 + |\mathsf{ML}| + |\mathsf{CL}|)$ for CEVCLUS. As with CEVLUS, parameter ξ should be set carefully. Selecting a very large value for ξ will enforce the constraints exactly, but will typically make the optimization problem more difficult. A good strategy is to increase ξ gradually, starting from $\xi = 0$. This strategy will be further discussed in Section 4.2.

To show that the cost function (11) can be minimized using the IRQP algorithm, we need to express (11) in matrix form. We assume that each mass function m_i has at most f focal sets among the set $\mathcal{F} = \{F_1, \ldots, F_f\}$, where $F_1 = \emptyset$ and the c singletons $\{\omega_k\}$, $k = 1, \ldots, c$ are among the $f - 1$ nonempty subsets $\{F_2, \ldots, F_f\}$. Mass function m_i can then be represented by a vector $\boldsymbol{m}_i = (m_i(F_1), \ldots, m_i(F_f))^T$ of length f, and the credal partition $\mathcal{M} = (m_1, \ldots, m_n)$ can be represented by matrix $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n)^T$ of size $n \times f$.

The degree of conflict (2) between two mass functions m_i and m_j can be written as

$$\kappa_{ij} = \boldsymbol{m}_i^T \boldsymbol{C} \boldsymbol{m}_j,$$

where $\boldsymbol{C}$ is the square matrix of size f with general term

$$C_{kl} = \begin{cases} 1 & if\ F_k \cap F_l = \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The plausibility $Pl_{ij}(S_{ij})$ in (8a) can then be written as

$$Pl_{ij}(S_{ij}) = 1 - \boldsymbol{m}_i^T \boldsymbol{C} \boldsymbol{m}_j = \boldsymbol{m}_i^T (\mathbf{1} \cdot \mathbf{1}^T - \boldsymbol{C}) \boldsymbol{m}_j,$$

where $\mathbf{1} = (1, \ldots, 1)^T$. Before rewriting $Pl_{ij}(\overline{S}_{ij})$, we note that the term $m_i(\emptyset) + m_j(\emptyset) - m_i(\emptyset)m_j(\emptyset)$ in the right-hand side of (8b) can be written as $\boldsymbol{m}_i^T \boldsymbol{B} \boldsymbol{m}_j$, where $\boldsymbol{B}$ is the following square matrix of size f,

$$\boldsymbol{B} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}, \tag{13}$$

and the term $\sum_{k=1}^{c} m_i(\{\omega_k\}) m_j(\{\omega_k\}$ can be written as $\boldsymbol{m}_i^T \boldsymbol{A} \boldsymbol{m}_j$, where $\boldsymbol{A}$ is the square matrix of size f with general term

$$A_{k\ell} = \begin{cases} 1 & k = \ell,\ |F_k| = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

With these notations, we have

$$Pl_{ij}(\overline{S}_{ij}) = 1 - \boldsymbol{m}_i^T \boldsymbol{B} \boldsymbol{m}_j - \boldsymbol{m}_i^T \boldsymbol{A} \boldsymbol{m}_j = \boldsymbol{m}_i^T \boldsymbol{E} \boldsymbol{m}_j,$$

where

$$\boldsymbol{E} = \mathbf{1} \cdot \mathbf{1}^T - \boldsymbol{B} - \boldsymbol{A}. \tag{15}$$

The penalty terms in (11) can finally be written as

$$J_{\mathsf{ML}} = \sum_{(i,j) \in \mathsf{ML}} \boldsymbol{m}_i^T (\boldsymbol{C} + \boldsymbol{E}) \boldsymbol{m}_j,$$
$$J_{\mathsf{CL}} = \sum_{(i,j) \in \mathsf{CL}} \boldsymbol{m}_i^T (\overline{\boldsymbol{C}} + \overline{\boldsymbol{E}}) \boldsymbol{m}_j,$$

where

$$\overline{\boldsymbol{C}} = \mathbf{1} \cdot \mathbf{1}^T - \boldsymbol{C} \text{ and } \overline{\boldsymbol{E}} = \mathbf{1} \cdot \mathbf{1}^T - \boldsymbol{E}. \tag{16}$$

With these notations, the stress function (11) equals to

$$J_{kC}(\boldsymbol{M}) = \eta \sum_{i=1}^{n} \sum_{r=1}^{k} \left(\boldsymbol{m}_i^T \boldsymbol{C} \boldsymbol{m}_{j_r(i)} - \delta_{i,j_r(i)} \right)^2 + \rho \left\{ \sum_{(i,j) \in \mathsf{ML}} \boldsymbol{m}_i^T (\boldsymbol{C} + \boldsymbol{E}) \boldsymbol{m}_j + \sum_{(i,j) \in \mathsf{CL}} \boldsymbol{m}_i^T (\overline{\boldsymbol{C}} + \overline{\boldsymbol{E}}) \boldsymbol{m}_j \right\}, \tag{17}$$

where $\rho = \xi \left[2(|\mathsf{ML}| + |\mathsf{CL}|) \right]^{-1}$. From expression (17), it is clear that $J_{kC}(\boldsymbol{M})$ is a quadratic function of each vector $\boldsymbol{m}_i$. Consequently, minimizing $J_{kC}(\boldsymbol{M})$ with respect to $\boldsymbol{m}_i$ alone while leaving the other vectors $\boldsymbol{m}_j$ constant is a quadratic programming problem. This is the underlying principle of the IRQP algorithm [26, 10]. The cost function to be minimized as each iteration is

$$g(\boldsymbol{m}_i) = \eta \| \boldsymbol{M}_i \boldsymbol{C} \boldsymbol{m}_i - \boldsymbol{\delta}_i) \|^2 + \rho \left\{ \left(\sum_{j \in \mathsf{ML}(i)} \boldsymbol{m}_j^T (\boldsymbol{C} + \boldsymbol{E}) \right) \boldsymbol{m}_i + \left(\sum_{j \in \mathsf{CL}(i)} \boldsymbol{m}_j^T (\overline{\boldsymbol{C}} + \overline{\boldsymbol{E}}) \right) \boldsymbol{m}_i \right\}, \tag{18}$$

where $\boldsymbol{M}_i = (\boldsymbol{m}_{j_1(i)}, \ldots, \boldsymbol{m}_{j_k(i)})^T$ is the matrix of size $k \times f$ whose row r is equal to vector $\boldsymbol{m}_{j_r(i)}$, $\mathsf{ML}(i)$ and $\mathsf{CL}(i)$ are the sets of objects linked to object i by, respectievely, ML and CL constraints, and $\boldsymbol{\delta}_i$ is the vector of $(\delta_{i,j_1(i)}, \ldots, \delta_{i,j_k(i)})^T$ of transformed dissimilarities between object o_i and all sampled objects $o_{j_r(i)}$. Developing the right-hand side of (18) and rearranging the terms, we obtain

$$g(\boldsymbol{m}_i) = \boldsymbol{m}_i^T \Sigma \boldsymbol{m}_i + \boldsymbol{u}^T \boldsymbol{m}_i + c, \tag{19}$$

with

$$\Sigma = \eta \boldsymbol{C}^T \boldsymbol{M}_i^T \boldsymbol{M}_i \boldsymbol{C} \tag{20a}$$

$$\boldsymbol{u} = -2\eta \boldsymbol{\delta}_i^T \boldsymbol{M}_i \boldsymbol{C} + \rho \left(\sum_{j \in \mathsf{ML}(i)} \boldsymbol{m}_j^T \right) (\boldsymbol{C} + \boldsymbol{E}) + \rho \left(\sum_{j \in \mathsf{CL}(i)} \boldsymbol{m}_j^T \right) (\overline{\boldsymbol{C}} + \overline{\boldsymbol{E}}) \tag{20b}$$

$$c = \eta \boldsymbol{\delta}_i^T \boldsymbol{\delta}_i. \tag{20c}$$

Minimizing $g(\boldsymbol{m}_i)$ under the constraints $\boldsymbol{m}_i^T \mathbf{1} = 1$ and $\boldsymbol{m}_i \geq \mathbf{0}$ is a quadratic programming (QP) problem, which can be solved efficiently with any QP solver. As we iteratively update each row of $\boldsymbol{M}$, the overall cost $J_{kC}(\boldsymbol{M})$ decreases and eventually reaches a (local) minimum. As in [10], we compute the following running mean after each cycle of the algorithm,

$$e_0 = 1, \tag{21a}$$

$$e_t = 0.5 e_{t-1} + 0.5 \frac{|J_t - J_{t-1}|}{J_{t-1}}, \tag{21b}$$

where t is the iteration counter and J_t is the value of the cost function at iteration t. The algorithm stops when e_t becomes less than some given threshold ϵ. The whole procedure is summarized in Algorithm 1.

Another important aspect of the procedure is the initialization of the credal partition matrix $\boldsymbol{M}$. As the inclusion of a penalization term makes the minimization of (11) more difficult than that of (7), initializing the credal partition randomly may not yield optimal results. We recommend initializing the random partition with k-EVCLUS, i.e., ignoring the constraints, before running k-CEVCLUS. Also, better results are obtained by running k-CEVCLUS a first time with a small value of ξ such as $\xi = 0.05$, before setting ξ to its final value. The tuning of parameters d_0 and ξ will be addressed in greater detail in Section 4.2.

3.2. Constraint Expansion

As shown in [2] and [1], the performances of clustering algorithms (in terms of proximity to the true partition) get better when the number of constraints increases. In particular, as the number of constraints tends to the total number of objects pairs, the partition found by a clustering algorithm can be expected to tend to the true partition. In some applications, pairwise constraints can be obtained by objective methods; however, they more often need to be elicited by visual inspection of the data, which costs time and money. In particular, for large data sets, pairwise constraints will typically be available only for a tiny proportion of all object pairs. It is thus important to "make the most" out of the small number of constraints we usually have.

For that purpose, we propose a *constraint expansion* strategy based on the following idea. Out of the $n(n-1)/2$ object pairs, some are known to belong

Algorithm 1 k-CEVCLUS algorithm.

Require: $\boldsymbol{D} = (d_{ij})$, ML, CL, $\mathcal{F}$, k, d_0, ξ, ϵ, initial credal partition $\boldsymbol{M}$

$\gamma \leftarrow -\log 0.05/d_0^2$

for all $1 \leq i < j \leq n$ **do**

$\delta_{ij} \leftarrow 1 - \exp(-\gamma d_{ij}^2)$

end for

$\eta \leftarrow \left(\sum_{i<j} \delta_{ij}^2\right)^{-1}$

Compute matrices $\boldsymbol{C}$, $\boldsymbol{B}$, $\boldsymbol{A}$, $\boldsymbol{E}$ using (12)-(16)

if $k < n-1$ **then**

for $i = 1$ **to** n **do**

pick k integers $j_1(i), \ldots, j_k(i)$ randomly in $\{1, \ldots, i-1, i+1, \ldots, n\}$

end for

else

for $i = 1$ **to** n **do**

$(j_1(i), \ldots, j_{n-1}(i)) \leftarrow (1, \ldots, i-1, i+1, \ldots, n)$

end for

end if

$t \leftarrow 0$, $e_0 \leftarrow 1$

Compute J_0 using (17)

while $e_t \geq \epsilon$ **do**

$t \leftarrow t + 1$

$J_t \leftarrow 0$

for $i = 1$ **to** n **do**

Delete row i from current matrix $\boldsymbol{M}$ to get $\boldsymbol{M}_i$

Compute Σ, $\boldsymbol{u}$ and c using (20)

Find $\boldsymbol{m}_i^{(t)}$ by minimizing (19) subject to $\boldsymbol{m}_i^T \mathbf{1} = 1$ and $\boldsymbol{m}_i \geq \mathbf{0}$

Replace row i of $\boldsymbol{M}$ by $(\boldsymbol{m}_i^{(t)})^T$

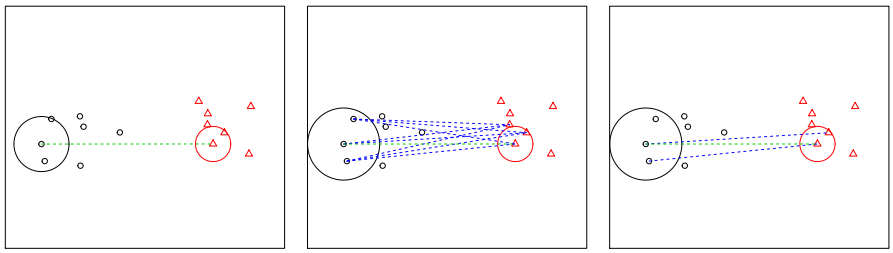
$J_t \leftarrow J_t + g(\boldsymbol{m}_i^{(t)})$

end for

$e_t \leftarrow 0.5 e_{t-1} + 0.5|J_t - J_{t-1}|/J_{t-1}$

end while

return Credal partition $\boldsymbol{M} = (\boldsymbol{m}_1^{(t)}, \ldots, \boldsymbol{m}_n^{(t)})^T$



(a) (b) (c)

Figure 1: (a) A CL constraint $(o_i, o_j) \in \mathsf{CL}$, with the K-neighborhoods $\mathcal{N}_K(o_i)$ and $\mathcal{N}_K(o_j)$ of o_i and o_j, respectively ($K = 2$). (b) The set $\mathcal{P}_K(o_i, o_j)$ of pairs of a neighbor of o_i and a neighbor of o_j. (c) The $K = 2$ new CL constraints.

to the same cluster (those in ML), some are known not to belong to the same cluster (those in CL), and the rest have an unknown status. Determining the ML or CL status of unlabeled object pairs can thus be seen as a binary classification problem. Because there is typically only a small proportion of labeled pairs, we cannot reliably classifying all object pairs. We thus propose to classify only those pairs that are similar to one pair in ML or CL. For each pair (o_i, o_j) in set S, with $\mathsf{S} \in \{\mathsf{ML}, \mathsf{CL}\}$, neighboring pairs that are not already labeled will be found and added to S.

More precisely, the proposed constraint expansion can be described as follows. Let K be a integer such that $K \ll n$. For any object $o \in \mathcal{O}$, let $\mathcal{N}_K(o)$ denote the set composed of o and its K nearest neighbors (NN) in $\mathcal{O}$. For each pair (o_i, o_j) in set S, where $\mathsf{S} \in \{\mathsf{ML}, \mathsf{CL}\}$, let $\mathcal{P}_K(o_i, o_j)$ be the subset of $\mathcal{N}_K(o_i) \times \mathcal{N}_K(o_j)$ composed of pairs (o_r, o_s) such that: $o_r \neq o_s$, $(o_r, o_s) \notin \mathsf{ML}$ and $(o_r, o_s) \notin \mathsf{CL}$. For all pairs (o_r, o_s) in $\mathcal{P}_K(o_i, o_j)$, we compute the dissimilarity with (o_i, o_j), as the sum of the dissimilarity between o_i and o_r, and the dissimilarity between o_j and o_s:

$$\Delta\left[(o_i, o_j), (o_r, o_s)\right] = d_{ir} + d_{js}.$$

We then add the K nearest neighbors of (o_i, o_j) to S. This procedure is illustrated in Figure 1, and described formally in Algorithm 2.

4. Numerical Experiments

In this section, we compare the performances of k-CEVCLUS to those of alternative algorithms for constrained evidential clustering. The experimental settings will first be described in Section 4.1, and the tuning of parameters ξ and d_0 will be specifically addressed in Section 4.2. A comparison with alternative clustering methods will then be presented in Section 4.3, and results with large datasets will be reported in Section 4.4. Finally, the efficiency of the constraint expansion procedure will be demonstrated in Section 4.5.

Algorithm 2 Constraint expansion algorithm.

Require: $\boldsymbol{D} = (d_{ij})$, ML, CL, K
  $\mathsf{ML}' \leftarrow \mathsf{ML}$, $\mathsf{CL}' \leftarrow \mathsf{CL}$
  for all $\mathsf{S} \in \{\mathsf{ML}, \mathsf{CL}\}$ **do**
    for all $(o_i, o_j) \in \mathsf{S}$ **do**
      Find the K NN $\{o_{i_1}, \ldots, o_{i_K}\}$ of o_i in $\mathcal{O}$
      $\mathcal{N}_K(o_i) \leftarrow \{o_i, o_{i_1}, \ldots, o_{i_K}\}$
      Find the K NN $\{o_{j_1}, \ldots, o_{j_K}\}$ of o_j in $\mathcal{O}$
      $\mathcal{N}_K(o_j) \leftarrow \{o_j, o_{j_1}, \ldots, o_{j_K}\}$
      $\mathcal{P}_K(o_i, o_j) \leftarrow [(\mathcal{N}_K(o_i) \setminus \mathcal{N}_K(o_j)) \times (\mathcal{N}_K(o_j) \setminus \mathcal{N}_K(o_i))] \setminus (\mathsf{ML}' \cup \mathsf{CL}')$
      for all $(o_r, o_s) \in \mathcal{P}_K(o_i, o_j)$ **do**
        Compute the dissimilarity $\Delta\left[(o_i, o_j), (o_r, o_s)\right] = d_{ir} + d_{js}$
      end for
      Find the K NN $\{(o_{r_1}, o_{s_1}), \ldots, (o_{r_K}, o_{s_K})\}$ of (o_i, o_j) in $\mathcal{P}_K(o_i, o_j)$
      if $\mathsf{S} == \mathsf{ML}$ **then**
        $\mathsf{ML}' \leftarrow \mathsf{ML}' \cup \{(o_{r_1}, o_{s_1}), \ldots, (o_{r_K}, o_{s_K})\}$
      else
        $\mathsf{CL}' \leftarrow \mathsf{CL}' \cup \{(o_{r_1}, o_{s_1}), \ldots, (o_{r_K}, o_{s_K})\}$
      end if
    end for
  end for
  return ML′, CL′

4.1. Experimental Settings

Datasets. The datasets used in our experiments are summarized in Table 1. The Banana datasets are synthetic data composed of two-dimensional attribute vectors uniformly distributed along intertwined circular segments with standard normal additive noise (Figure 2). This distribution was chosen to generate two clusters separated by a complex boundary, as such clusters are typically difficult to identify without prior information. All other datasets contain real data. The Glass, Iris, Ecoli and Letter were downloaded from the UCI repository[1]. Dissimilarities were computed as Euclidean distances in the attribute space for all datasets except for the Zongker data, which already consist of dissimilarities.

The Letter dataset is composed of a large number of black-and-white rectangular pixel displays of the 26 capital letters in the English alphabet; each object is characterized by 16 dimensional attributes. As in Refs. [2, 1], we only kept three classes corresponding to letters I, J, and L because these letters are hard to recognize. However, the authors of Refs. [2, 1] used only 10% of the data. As k-CEVCLUS can deal with large datasets, we used all the data with $n = 2263$.

The Zongker digit dissimilarity dataset[2], which was also used in Ref. [10], contains similarities between 2000 handwritten digits in 10 classes, based on

[1] Available at `http://archive.ics.uci.edu/ml`.
[2] Available at `http://prtools.org/disdatasets/index.html`.

Table 1: Datasets used in the experiments.

Dataset	Number of objects	Number of clusters	Number of attributes
Banana_n	n	2	2
Glass	214	2	9
Iris	150	3	4
Ecoli	272	3	7
Letter	2263	3	16
Zongker	2000	10	NA

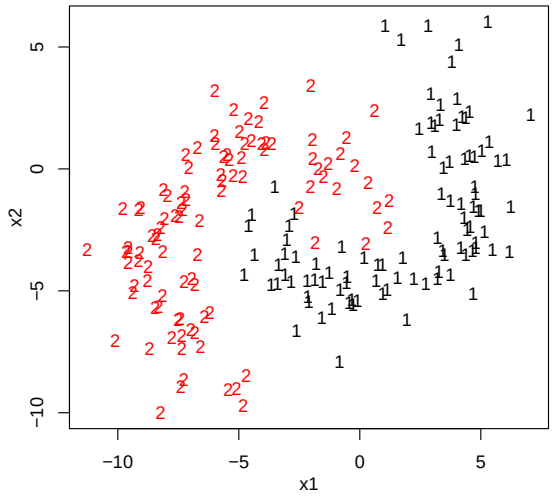


Figure 2: Banana_200 dataset with $n = 200$.

deformable template matching. As the dissimilarity matrix was initially non symmetric, we symmetrized it by the transformation $d_{ij} \leftarrow (d_{ij} + d_{ji})/2$. It should be noted that the dissimilarities are non metric, i.e., they are not Euclidean distances.

Evaluation criteria. For all the datasets considered in this study, some "ground-truth" partition exists. To compare an evidential partition with the true partition, we first converted it to a hard partition using the maximum plausibility rule as explained in Section 2.2, and we computed the Adjusted Rand Index (ARI) between the derived hard partition and the true partition [13]. We recall that the ARI is a chance-adjusted version of the Rand Index (RI), which is a classical measure of similarity between hard partitions [23]; the ARI is designed in such a way that it takes on the value 0 when the RI equals its expected value for random partitions, and it is equal to 1 when the two partitions being compared are identical. The exact definition of the ARI is this. Consider two partitions $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$ of a set with n objects. Let $n_{ij} = |X_i \cap Y_j|$ be the number of objects that belong to both X_i and Y_j, $n_{i\cdot} = |X_i| = \sum_j n_{ij}$ the number of objects in X_i, and $n_{\cdot j} = |Y_i| = \sum_i n_{ij}$ the number of objects in Y_j, for $i = 1, \ldots, r$ and $i = 1, \ldots, r$. Then, we have

$$\mathsf{ARI} = \frac{\displaystyle\sum_{i,j} \binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}\right] \Big/ \binom{n}{2}}{\displaystyle\frac{1}{2}\left[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}\right] - \left[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}\right] \Big/ \binom{n}{2}}. \tag{22}$$

The ARI can only be used when a ground truth partition is available. When there is no such reference partition, as it is the case in real applications, some internal quality index has to be used. In [9], the authors proposed to measure the degree of imprecision of a credal partition by the *average nonspecificity*, defined by

$$N^* = \frac{1}{n \log_2(c)} \sum_{i=1}^{n} \left[\sum_{A \in 2^\Omega \setminus \emptyset} m_i(A) \log_2 |A| + m_i(\emptyset) \log_2(c) \right]. \tag{23}$$

This measure was shown in [9, 10] to be a good internal validity index for credal partitions, allowing us, for instance, to compare credal partitions with different numbers of clusters. Average nonspecificity is comprised between 0 and 1. Smaller values indicate that masses are assigned to non empty focal sets with small cardinality, which is evidence for the adequacy of the credal partition to the data.

4.2. Guidelines for Tuning Parameters ξ and d_0

The k-CEVCLUS procedure (Algorithm 1) depends on the following parameters:

- Parameter d_0 in (5)-(6);
- Parameter ξ in (11);
- The number k of dissimilarities taken into account for each object;
- The set $\mathcal{F}$ of focal sets;
- The threshold ϵ in the stopping criterion (21).

As suggested in [9] and [10], we recommend choosing as focal sets the empty set, the singletons and Ω, i.e., $\mathcal{F} = \{\emptyset, \{\omega_1\}, \ldots, \{\omega_c\}, \Omega\}$. This choice was made for all the simulations reported in this section. For the stopping criterion, we set $\epsilon = 10^{-5}$ in all our simulations. For small datasets, we recommend using $k = n - 1$. The influence of k for large datasets will be studied in Section 4.4. The most important parameters that need some tuning are d_0, used to transform dissimilarities, and ξ, which defines the weight of the constraints in the cost function. In this section, we present some experimental results with four datasets (Banana_200 with $n = 200$, Glass, Iris and Ecoli), from which we derive some guidelines for tuning d_0 and ξ.

Influence of ξ. Table 2 shows the ARI and nonspecificity criteria for nbconst = 100 and nbconst = 200 randomly selected constraints and different values of ξ. As explained in Section 3.1, k-CEVCLUS was first initialized using k-EVCLUS (i.e., with $\xi = 0$) and run a second time with $\xi = 0.05$ before being run with a higher value of ξ. Parameter d_0 was set to the 0.9-quantile of the dissimilarities for the Banana, Glass and Ecoli datasets and to the 0.6-quantile for the Iris data. For each dataset and for each value of ξ, we ran k-CEVCLUS 10 times. The average ARI and nonspecificity values are reported in Table 2. From these results, we can see that for both values of nbconst, the average nonspecificity usually gets smaller with larger values of ξ. In contrast, the ARI is not very sensitive to the choice of ξ, particularly for nbconst = 200. Although the best value of ξ differs for different datasets, the value $\xi = 0.5$ general yields close-to-optimal results considering both ARI and nonspecificity. We thus adopted this value in other experiments, unless otherwise specified.

Influence of d_0. Parameter d_0 determines the size of each class [10]. Typically, d_0 can be set to some quantile q of the dissimilarities D. Table 3 shows results with different values of q, from 0.1 to 1 (by 0.1 increments), with nbconst = 100 and nbconst = 200 randomly chosen constraints. In each experiment, we used $\xi = 0.5$ as mentioned above. For the Ecoli and Glass datasets, we can see that the ARI gets higher when increasing d_0, while for the Iris dataset the best ARI is obtained with $q = 0.6$. In contrast, for the Banana_200 data, the best value of ARI was obtained with the smallest value of d_0. However, if d_0 is too small, the average nonspecificity is high because most objects are classified as outliers. Similarly, when d_0 is large, such as $q = 1$, the nonspecificity is also high, because for the objects near the boundary between two clusters, some mass is assigned to the union of the clusters. But for both values of nbconst, the

Table 2: Average ARI and average nonspecificity as a function of ξ for nbconst = 100 and nbconst = 200, with d_0 = quantile$(D, 0.9)$ for Banana_200, Glass and Ecoli data and d_0 = quantile$(D, 0.6)$ for the Iris data. The best results in terms of ARI are shown in bold.

nbconst	ξ	Banana_200	Glass	Iris	Ecoli
100	0	0.38 (0.17)	0.63 (0.21)	0.75 (0.11)	0.79 (0.16)
	0.05	0.48 (0.16)	0.78 (0.19)	0.86 (0.08)	0.85 (0.13)
	0.1	0.59 (0.14)	0.79 (0.17)	0.87 (0.06)	0.86 (0.10)
	0.2	0.70 (0.12)	0.80 (0.14)	0.89 (0.04)	**0.87** (0.09)
	0.3	0.69 (0.10)	0.80 (0.13)	**0.90** (0.03)	0.86 (0.09)
	0.4	0.70 (0.09)	0.79 (0.11)	0.89 (0.03)	**0.87** (0.09)
	0.5	0.73 (0.08)	0.82 (0.11)	0.89 (0.03)	**0.87** (0.10)
	0.6	0.69 (0.07)	**0.83** (0.11)	**0.90** (0.03)	0.86 (0.09)
	0.7	0.70 (0.07)	0.82 (0.10)	**0.90** (0.03)	0.86 (0.10)
	0.8	0.73 (0.08)	0.77 (0.09)	0.88 (0.03)	0.85 (0.10)
	0.9	**0.75** (0.08)	0.78 (0.10)	0.89 (0.03)	0.85 (0.10)
	1	0.69 (0.08)	0.81 (0.11)	0.88 (0.04)	0.84 (0.09)
200	0	0.38 (0.17)	0.6 (0.2)	0.76 (0.11)	0.79 (0.16)
	0.05	0.51 (0.15)	0.77 (0.19)	0.87 (0.08)	0.87 (0.13)
	0.1	0.61 (0.14)	0.83 (0.18)	0.93 (0.06)	0.89 (0.11)
	0.2	0.77 (0.12)	0.91 (0.14)	**0.97** (0.03)	0.90 (0.08)
	0.3	0.86 (0.10)	0.92 (0.12)	0.95 (0.02)	0.91 (0.06)
	0.4	0.90 (0.09)	0.92 (0.10)	0.96 (0.02)	0.91 (0.05)
	0.5	0.90 (0.07)	0.92 (0.09)	**0.97** (0.01)	0.91 (0.05)
	0.6	0.90 (0.06)	**0.94** (0.08)	**0.97** (0.01)	0.91 (0.05)
	0.7	**0.92** (0.05)	**0.94** (0.07)	**0.97** (0.01)	**0.93** (0.05)
	0.8	**0.92** (0.04)	0.93 (0.06)	0.96 (0.01)	0.92 (0.05)
	0.9	0.91 (0.04)	0.92 (0.06)	**0.97** (0.01)	0.91 (0.05)
	1	**0.92** (0.04)	**0.94** (0.06)	0.96 (0.01)	**0.93** (0.06)

Table 3: Average ARI and average nonspecificity as a function of q with $d_0 = \mathsf{quantile}(D, q)$ for $\mathsf{nbconst} = 100$ and $\mathsf{nbconst} = 200$, with $\xi = 0.5$. The best results in terms of average nonspecificity are shown in bold.

nbconst	q	Banana_200	Glass	Iris	Ecoli
100	0.1	0.94 (0.33)	0.23 (0.38)	0.57 (0.23)	0.27 (0.45)
	0.2	0.93 (0.31)	0.31 (0.36)	0.91 (0.20)	0.46 (0.4)
	0.3	0.92 (0.27)	0.46 (0.31)	0.93 (0.16)	0.71 (0.34)
	0.4	0.86 (0.23)	0.89 (0.28)	0.95 (0.10)	0.8 (0.27)
	0.5	0.84 (0.21)	0.87 (0.23)	0.91 (0.05)	0.83 (0.2)
	0.6	0.84 (0.17)	0.86 (0.16)	0.91 (**0.03**)	0.83 (0.15)
	0.7	0.79 (0.13)	0.82 (**0.11**)	0.83 (0.03)	0.83 (0.11)
	0.8	0.74 (0.09)	0.80 (**0.11**)	0.65 (0.05)	0.85 (**0.09**)
	0.9	0.74 (**0.08**)	0.84 (**0.11**)	0.68 (0.07)	0.86 (**0.09**)
	1	0.69 (0.26)	0.76 (0.27)	0.66 (0.2)	0.84 (0.25)
200	0.1	0.99 (0.21)	0.85 (0.28)	0.93 (0.09)	0.24 (0.22)
	0.2	0.97 (0.19)	0.86 (0.25)	0.97 (0.08)	0.46 (0.20)
	0.3	0.97 (0.17)	0.96 (0.22)	0.98 (0.05)	0.73 (0.18)
	0.4	0.97 (0.15)	0.95 (0.19)	0.99 (0.04)	0.82 (0.14)
	0.5	0.95 (0.13)	0.96 (0.16)	0.98 (0.02)	0.87 (0.12)
	0.6	0.94 (0.12)	0.92 (0.12)	0.96 (**0.01**)	0.88 (0.09)
	0.7	0.94 (0.10)	0.94 (0.10)	0.97 (0.02)	0.89 (0.06)
	0.8	0.91 (**0.07**)	0.94 (**0.09**)	0.92 (0.03)	0.91 (**0.05**)
	0.9	0.90 (**0.07**)	0.91 (**0.09**)	0.88 (0.06)	0.90 (**0.05**)
	1	0.81 (0.24)	0.92 (0.21)	0.87 (0.21)	0.89 (0.20)

average nonspecificity reaches a minimum for some value of q, which is around $q = 0.9$ for the Banana_200, Glass and Ecoli datasets and $q = 0.6$ for the Iris data. These results confirm those reported in [10], in which it was recommended to start with $d_0 = \mathsf{quantile}(D, 0.9)$, but it was also noted that "finding a suitable value of d_0 may sometimes require a trial and error process".

4.3. Performance Comparison

In this section, we compare the performances of the k-CEVCLUS to those of two alternative constrained evidential clustering methods: CEVCLUS with gradient-based cost minimization [1], and the Constrained Evidential c-means (CECM) algorithm [2]. The CECM algorithm is a prototype-based clustering method for attribute data, adapted from the Evidential c-means (ECM) algorithm [20] to take into account pairwise constraints. Note both CEVCLUS and CECM were shown in [1] and [2], respectively, to outperform other relational or attribute constrained clustering methods, such as the constrained Fuzzy C-means [12] and SSCARD [11] algorithms. In the experiments reported in this section, we used $k = n - 1$, so that k-CEVCLUS and CEVCLUS have exactly the same cost function and differ only by the optimization algorithm.

The three methods were compared on the Banana_200, Iris, Glass and Ecoli datasets with different numbers of pairwise constraints nbconst between 0 and 200. All the algorithms were run 20 times for each number of constraints. As suggested in Section 4.2, k-EVCLUS was run twice, with $\xi_0 = 0.05$ and $\xi = 0.5$. Parameter d_0 was fixed to $d_0 = \mathsf{quantile}(D, 0.9)$ for the Banana_200, Glass and Ecoli datasets and to $d_0 = \mathsf{quantile}(D, 0.6)$ for the Iris data. The same parameter values were used for CEVCLUS. For CECM, we used the parameter values as recommended in [2], i.e., $\delta = \max(D)$ and $\xi = 0.5$. All the algorithms were initialized with the corresponding unconstrained method, namely k-EVCLUS for k-CEVCLUS and CEVCLUS and ECM for CECM. Figures 3 and 4 show the median as well as the lower and upper quartiles of ARI, computing time and nonspecificity over the 20 runs.

In most cases, k-CEVCLUS and CEVCLUS yield similar results in terms of ARI, as expected. However, when the number of constraints exceeds 100, k-CEVCLUS outperforms CEVCLUS for the Banana_200 (Figures 3(a)) and Glass (Figure 3(b)) datasets. The CECM method generally performs worse than the other two methods, except on the Banana_200 dataset with a small number of constraints (Figure 3(a)). The results obtained by CECM also have much higher variability than those of the two other methods (Figures 3(b), 4(a) and 4(b)).

As far as computing times are concerned, k-CEVCLUS is much faster than CEVCLUS (see Figures 3(c), 3(d), 4(c), 4(d)), which confirms the superiority of the IRQP algorithm over the gradient-based procedure and is consistent with the results in [10]. The k-CEVCLUS method is also faster than CECM, by a lesser amount.

In terms of average nonspecificity, k-CEVCLUS also outperforms CEVCLUS as it reaches lower values. It also outperforms CECM, except for the Glass data (Figure 3(f)). However, lower nonspecificity values can be obtained by fine-tuning parameters ξ and d_0. For example, if we set $\xi = 1$ for the Glass data, the average nonspecificity is only 0.06, as can be seen from Table 2 with $\mathsf{nbconst} = 200$.

In this section, we have shown that k-CEVCLUS is considerably faster than CEVCLUS, while reaching comparable of better values of ARI and nonspecificity. In the next section, we will demonstrate the performances of k-CEVCLUS on large datasets.

4.4. Results with Large Datasets

In this section, we focus on the performances of the k-CEVCLUS algorithm applied to large datasets with 2000 to 10,000 objects. For such datasets, it is usually not feasible to store the whole dissimilarity matrix. The approach outlined in Section 3.1 is to use only $k \ll n-1$ randomly sampled dissimilarities for each object, which reduces the space and time complexity of the stress function calculation from quadratic to linear. The purpose of this section is to verify that this strategy does not negatively impact the performances of the k-CEVCLUS algorithm. We will not attempt any comparison with CECM, as the complexity of this algorithm limits its use to datasets with only a few hundred objects.

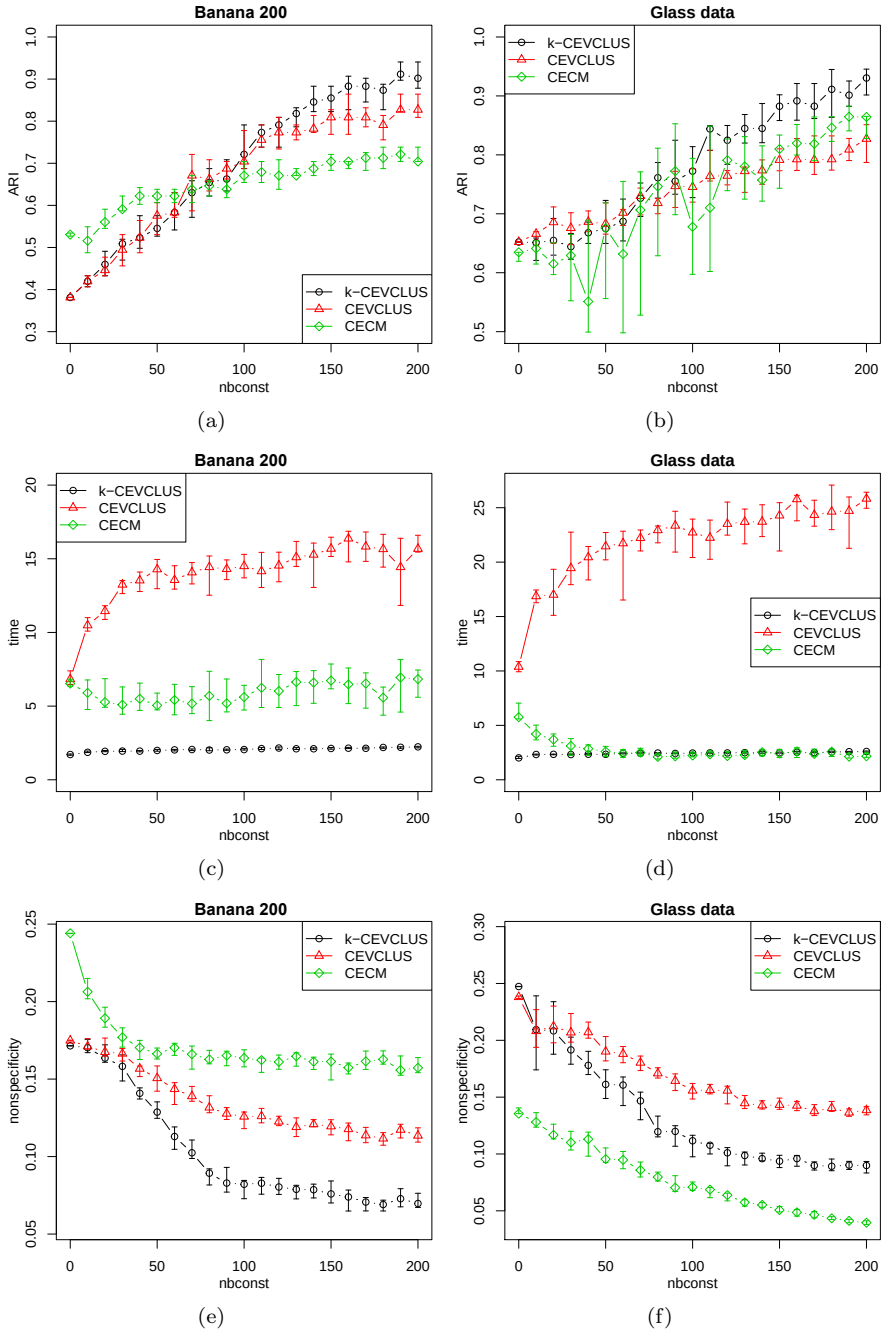


Figure 3: Results obtained with k-CEVCLUS (black), CEVCLUS (red) and CECM (green) for the Banana_200 and Glass datasets, with $\xi = 0.5$ and $d_0 = \mathsf{quantile}(D, 0.9)$.

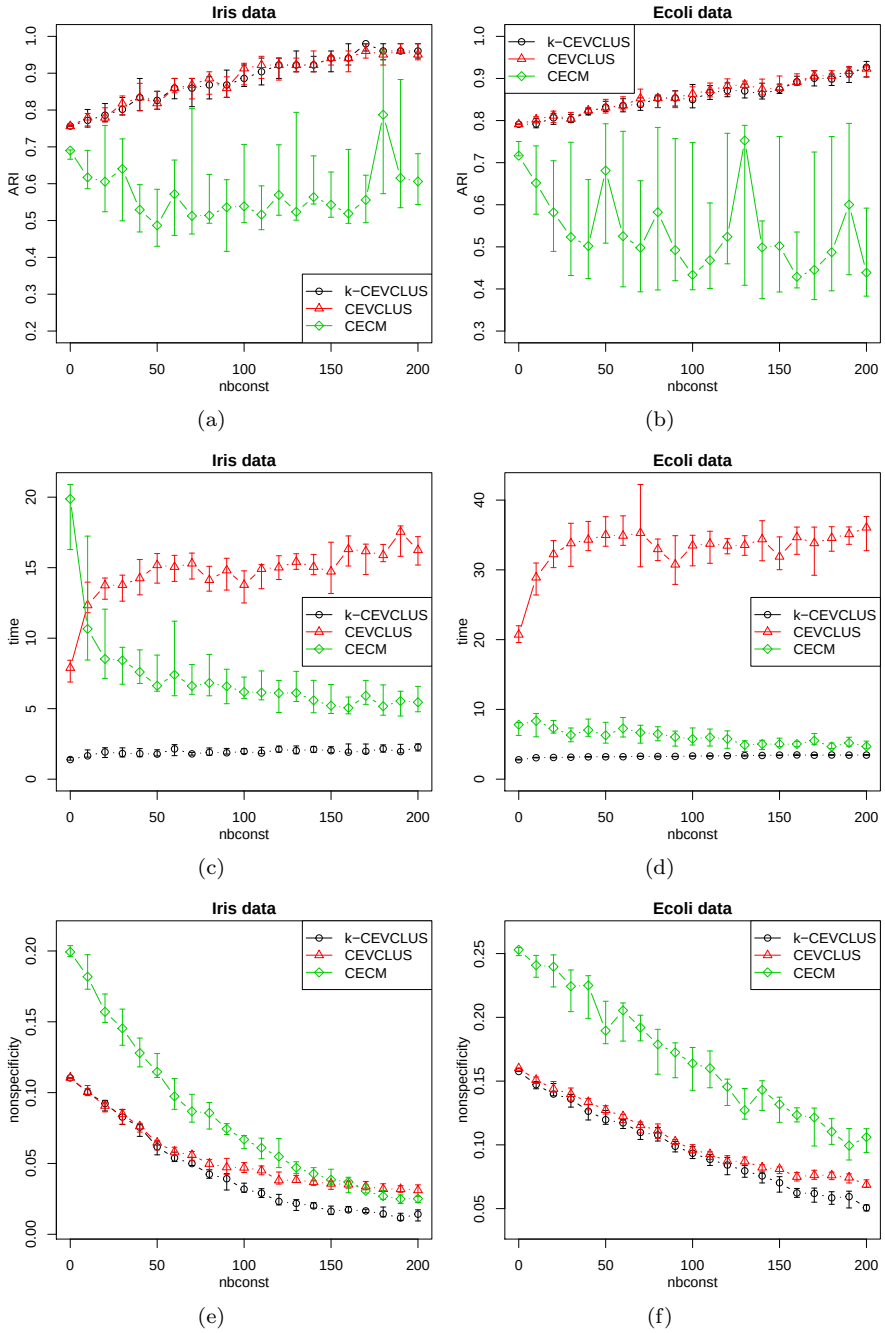


Figure 4: Results obtained with k-CEVCLUS (black), CEVCLUS (red) and CECM (green) for the Iris dataset with $\xi = 0.5$ and $d_0 = \mathsf{quantile}(D, 0.6)$, and for the Ecoli dataset with $\xi = 0.5$ and $d_0 = \mathsf{quantile}(D, 0.9)$.

We applied the k-CEVCLUS algorithm to four datasets: Banana_2000 with 2000 objects, Banana_10,000 with 10,000 objects, Letter and Zongker. Parameters d_0 was set to quantile$(D, 0.9)$ for the Banana datasets and to quantile$(D, 0.8)$ for the Letter and Zongker datasets. Parameter ξ was set to 0.5 for the Banana datasets and to 0.1 and 0.05, respectively, for the Letter and Zongker datasets. In each case, we computed the ARI, running time and nonspecificity as a function of k for a fixed number of constraints nbconst, and as a function of nbconst for fixed k. The results are shown in Figures 5 to 8.

For all three datasets, we can see that setting k to some value between 200 and 500 yields similar solutions in terms of ARI (Figures 5(a), 6(a), 7(a) and 8(a)) and nonspecificity (Figures 5(e), 6(e), 7(e) and 8(e)) as compared to considering the full dissimilarity matrix, while significantly reducing computing time (Figures 5(c), 6(c), 7(c) and 8(c)).

When the number of constraints increases for fixed k, the ARI gets higher (Figures 5(b), 6(b), 7(b) and 8(b)) at the cost of a longer running time (Figures 5(d), 6(d), 7(d) and 8(d)). In constrast, nonspecificity monotonically decreases as a function of nbconst for the Banana datasets (Figures 5(f) and 6(f)), but it exhibits a U-shaped curve for the Letter and Zongker datasets (Figures 7(f) and 8(f)). However, the variation of nonspecificity is smaller than that of ARI.

For the Letter dataset, higher variability of the ARI (Figure 7(b)) and computing time (Figure 7(d)) across repetitions is observed for 1500 to 2000 constraints, due to local minima of the cost function. In general, the optimization problem becomes more difficult with large numbers of objects and constraints, and the algorithm needs to be started several times from different random initial conditions, to avoid being trapped in a local minimum.

Comparing Figures 5(b) and 6(b), we can see that the curves of ARI as a function of nbconst have similar shapes for the Banana datasets with $n = 2000$ and $n = 10,000$ objects. However, with 10,000 objects, we need 10,000 constraints to reach the maximum value of ARI (around 0.88), whereas this value is reached with only 2000 constraints for the smaller dataset. It thus seems that larger datasets require a larger number of constraints, and the number of constraints should be of the same order of magnitude as the number of objects. To test this assumption, we applied k-CEVLUS to Banana datasets with $n = 10^5$ objects, and found that nbconst $= 10^5$ were needed to reach a value of ARI around 0.88. In real applications, it may be impractical to obtain as many constraints, especially if they are elicited by experts. This makes the constraint expansion procedure introduced in Section 3.2 even more useful for large datasets. This procedure will be evaluated in the next section.

4.5. Experiments with Expanded Constraints

In this section, we study experimentally the constraint expansion method introduced in Section 3.2 (Algorithm 2). We recall that this procedure adds K new constraints for each initial constraint, thus increasing the total number of constraints from nbconst to $(K+1) \times$ nbconst without requesting any additional information from the user.

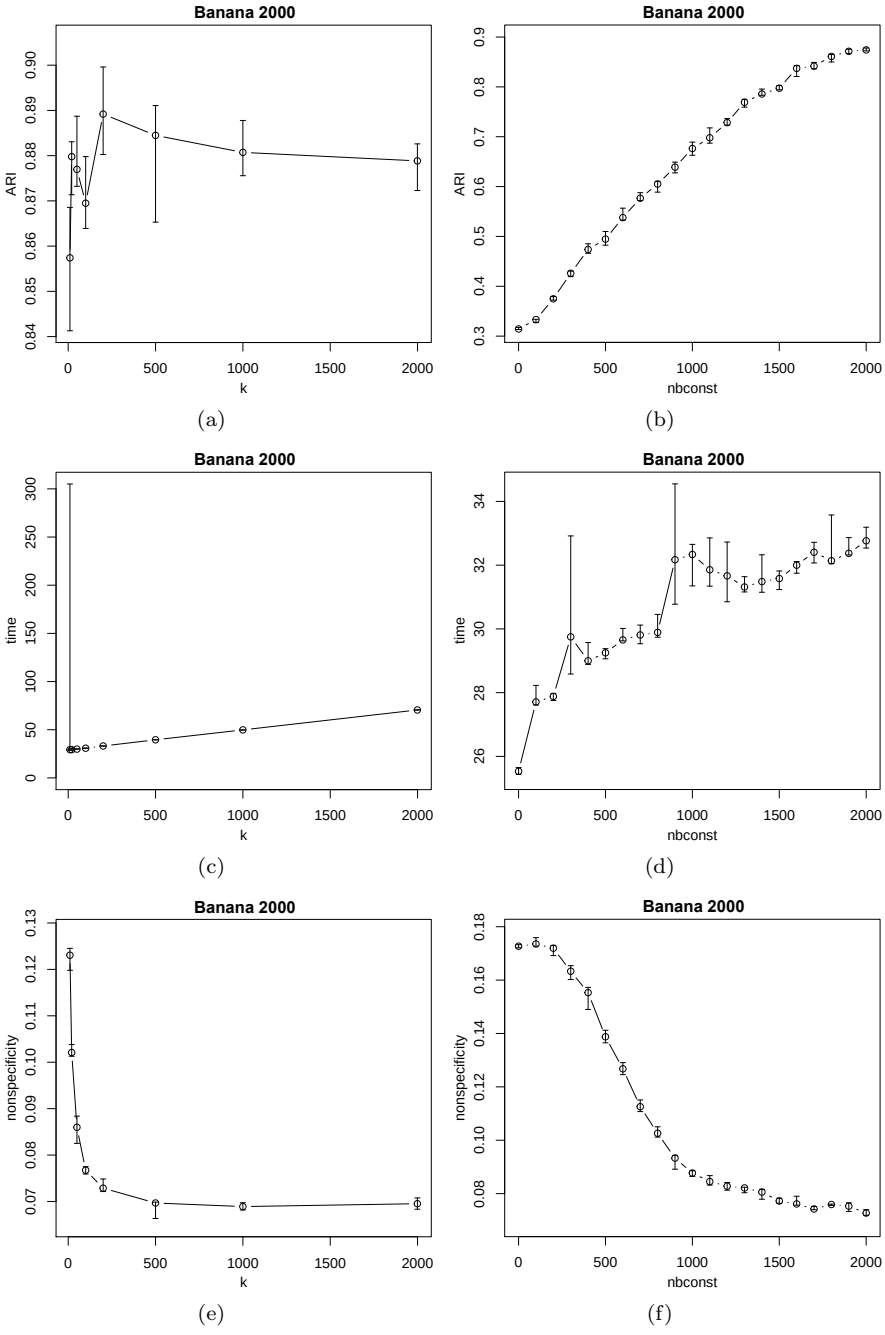


Figure 5: ARI, computing time and average nonspecificity of k-CEVCLUS as a function of k with nbconst $= 2000$ ((a), (c), (e)), and as a function of nbconst with $k = 200$ ((b), (d), (f)) for the Banana_2000 dataset with $d_0 = \mathsf{quantile}(D, 0.9)$ and $\xi = 0.5$. The error bars show the median as well as the lower and upper quartiles over 10 runs of the algorithm.

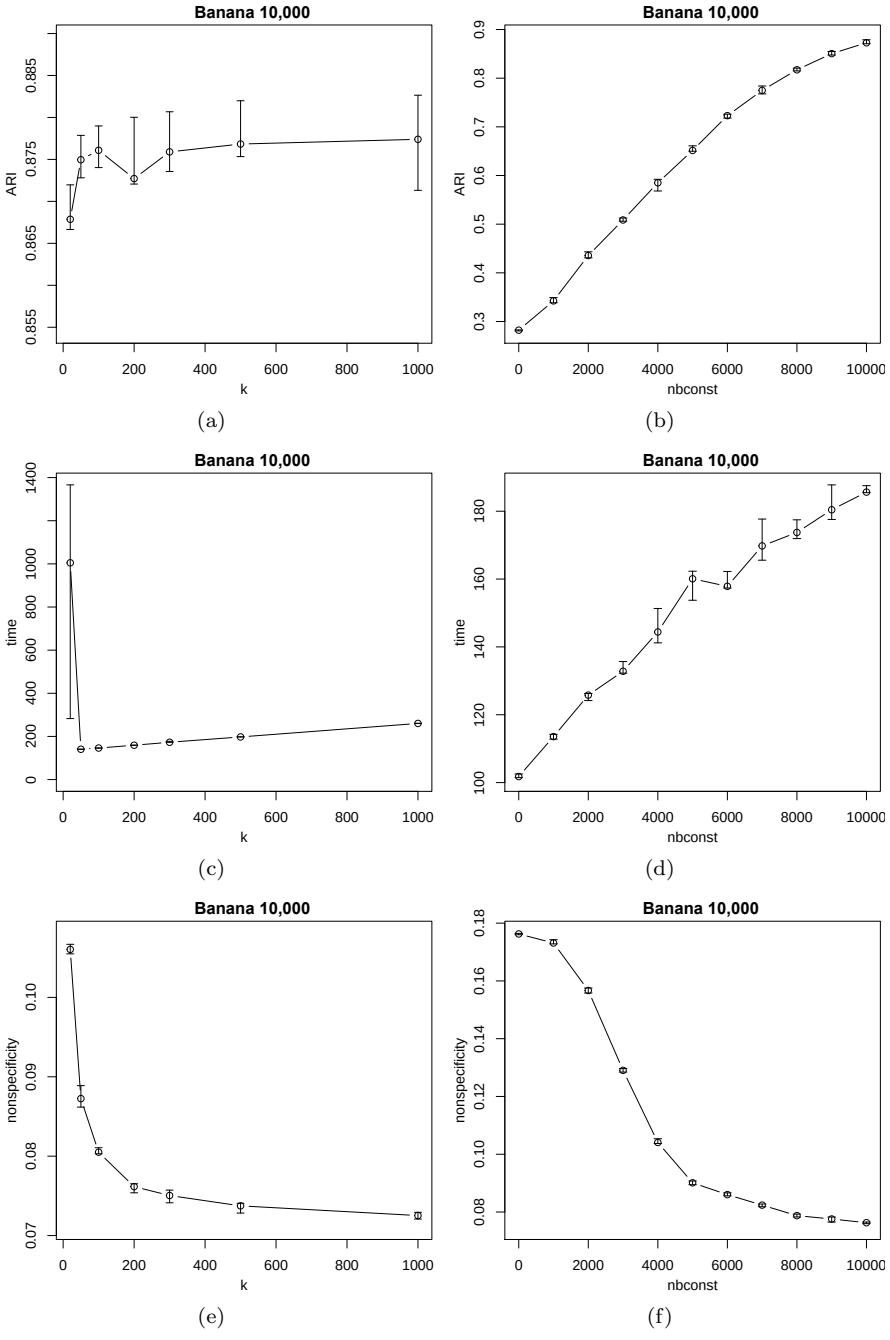


Figure 6: ARI, computing time and average nonspecificity of k-CEVCLUS as a function of k with $\mathsf{nbconst} = 10000$ ((a), (c), (e)), and as a function of $\mathsf{nbconst}$ with $k = 200$ ((b), (d), (f)) for the $\mathsf{Banana\_10{,}000}$ dataset with $n = 10000$, $d_0 = \mathsf{quantile}(D, 0.9)$ and $\xi = 0.5$. The error bars show the median as well as the lower and upper quartiles over 10 runs of the algorithm.

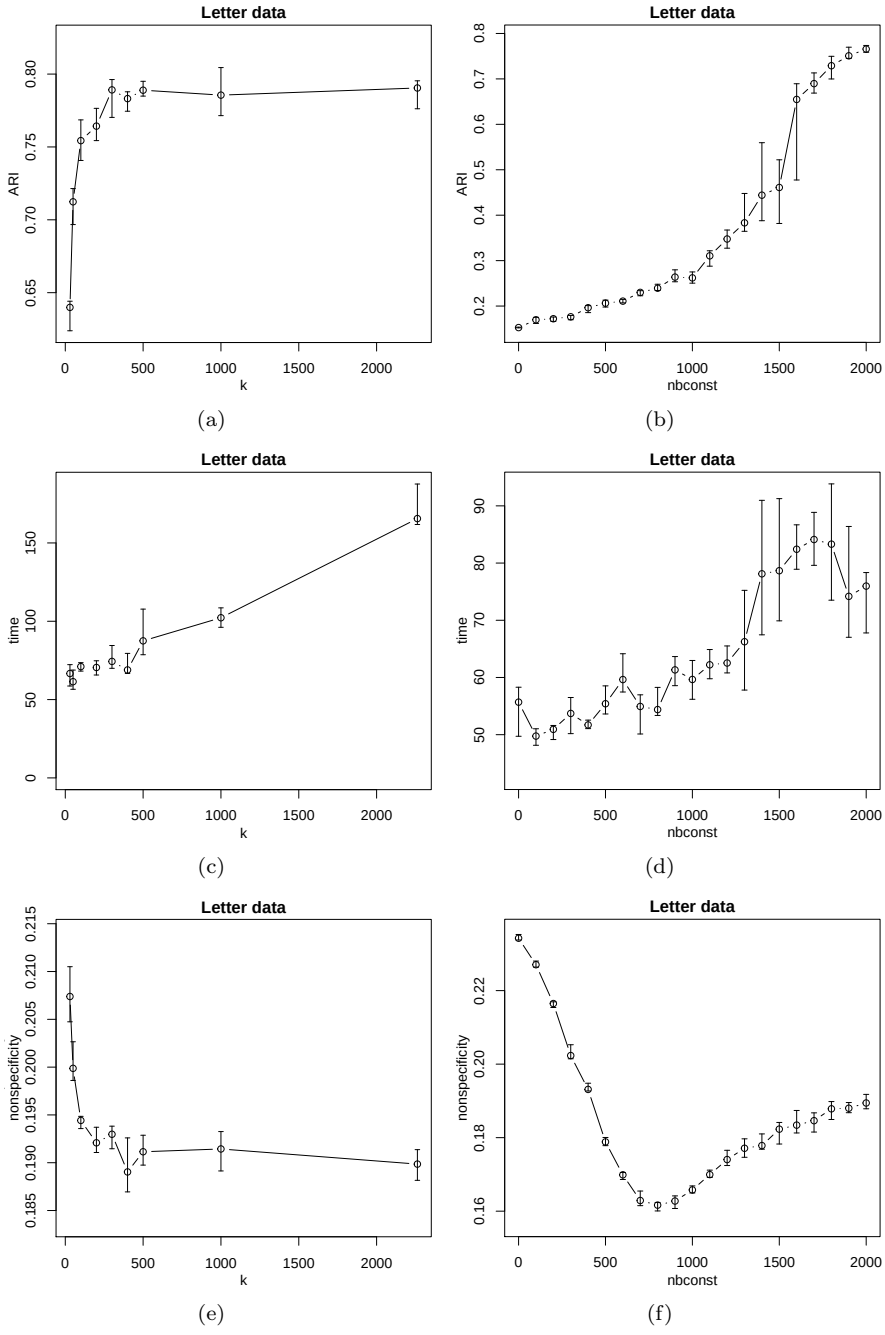


Figure 7: ARI, computing time and average nonspecificity of k-CEVCLUS as a function of k with $\mathsf{nbconst} = 2000$ ((a), (c), (e)), and as a function of $\mathsf{nbconst}$ with $k = 300$ ((b), (d), (f)) for the Letter dataset with $d_0 = \mathsf{quantile}(D, 0.8)$ and $\xi = 0.1$. The error bars show the median as well as the lower and upper quartiles over 10 runs of the algorithm.

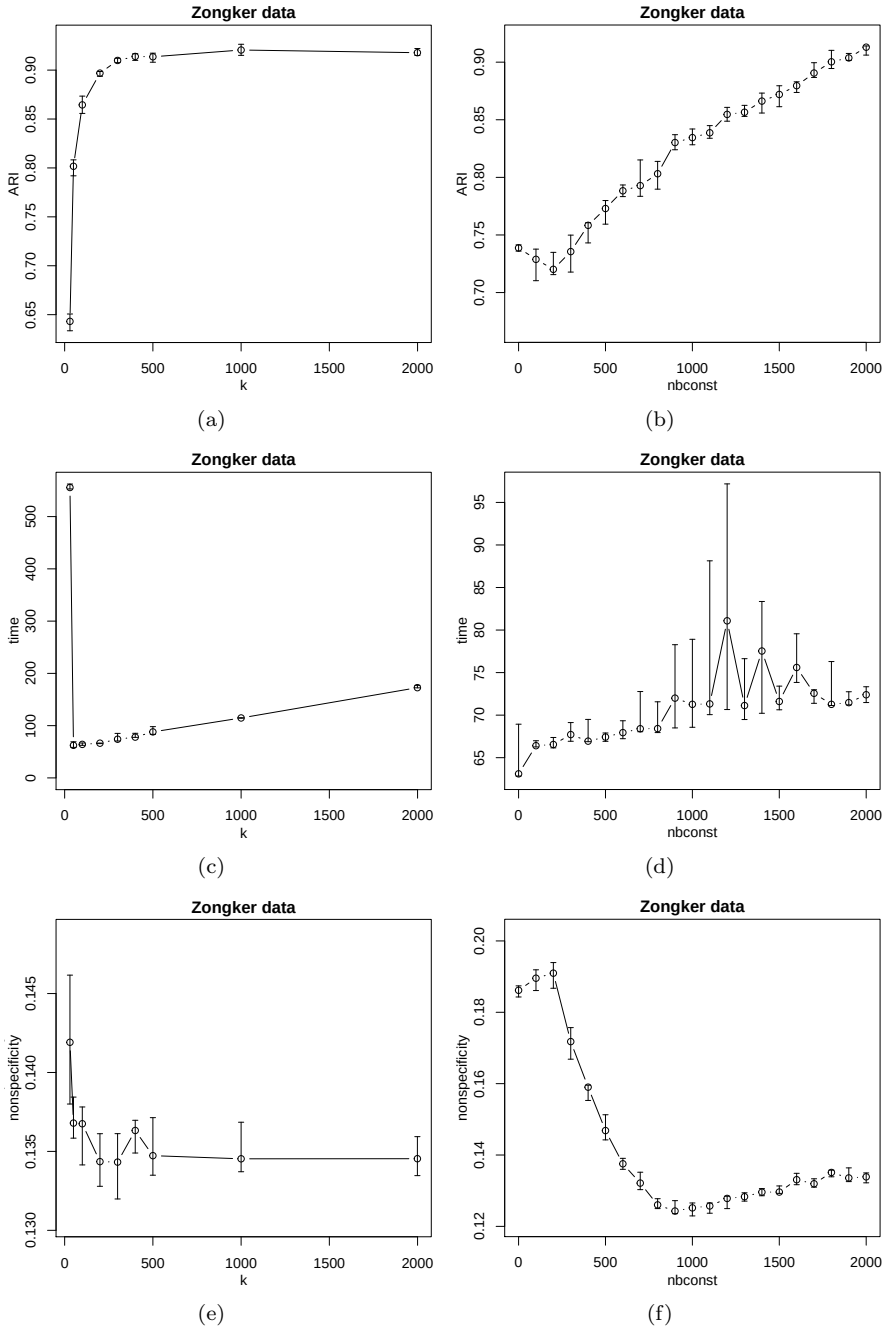


Figure 8: ARI, computing time and average nonspecificity of k-CEVCLUS as a function of k with nbconst $= 2000$ ((a), (c), (e)), and as a function of nbconst with $k = 300$ ((b), (d), (f)) for the Zongker dataset with $d_0 = \mathsf{quantile}(D, 0.8)$ and $\xi = 0.05$. The error bars show the median as well as the lower and upper quartiles over 10 runs of the algorithm.

We considered the Banana_2000, Letter and Zongker datasets as in Section 4.4, with the same parameter settings. Figure 9 shows the ARI as a function of the number nbconst of initial constraints, for $K \in \{0, 1, 3, 5, 10\}$. We can see that, for a given number of initial constraints, better results in terms of ARI can be obtained as K increases. For instance, for the Banana_2000 dataset, when $K = 5$ and only 500 initial constraints are given (resulting in 3000 expanded constraints), the ARI equals that obtained with 1500 initial constraints (Figure 9(a)). For the Letter dataset, the best ARI value, obtained by 2000 initial constraints, can be obtained with $K = 3$ and only nbconst $= 1100$ initial constraints, corresponding 4400 expanded constraints (Figure 9(b)). For the Zongker dataset, the same results can be obtained with 1400 initial constraints or with only 1000 initial constraints and $K = 1$, resulting in 2000 expanded constraints (Figure 9(c)).

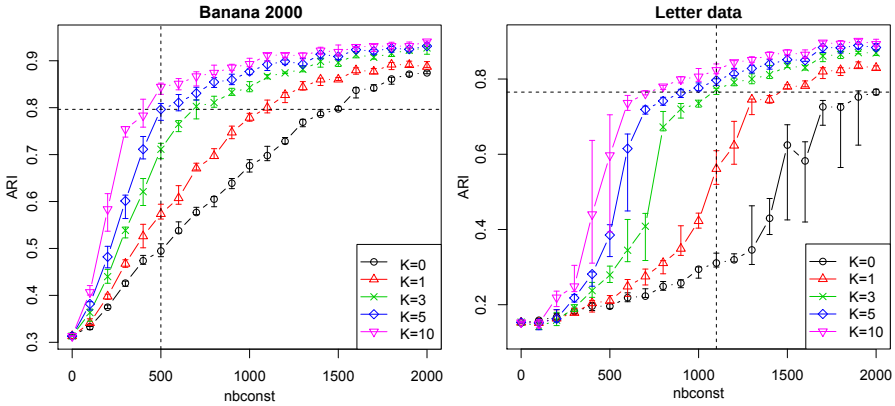
As can be expected, the benefits of expanding the constraints are less remarkable when the initial number of constraints is very large. Also, the difference in ARI between $K = 5$ and $K = 10$ is small, which suggests that increasing the value of K beyond 10 will just increase computing time without significantly improving the results. A large value of K might also have a negative impact on the results, as more pairs might be incorrectly labeled as ML or CL constraints. Overall, the very good results are obtained with moderate numbers of initial constraints and $K = 5$. With this setting, for instance, the ARI can be increased for the Banana_2000 dataset from 0.5 to 0.8 with 500 initial constraints (Figure 9(a)); for the Letter dataset, it can be increased from 0.3 to 0.8 with 1100 constraints 9(b)).

5. Conclusions

Evidential clustering represents a new direction of research in cluster analysis, aiming at a better representation of cluster-membership uncertainty using Dempster-Shafer mass functions. Until recently, the use of evidential clustering algorithms was limited to small datasets, due to their inherent algorithmic complexity. In particular, the EVCLUS algorithm [9], which has been shown to perform very well with non-metric dissimilarity data, could only be applied to datasets with a few hundred objects.

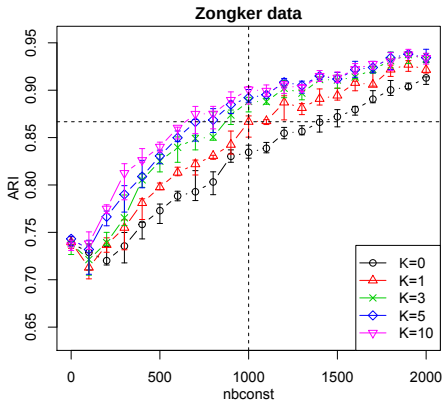
In [10], the authors introduced k-EVCLUS, a variant of EVCLUS applicable to to very large datasets, thanks to (1) a new optimization algorithm and (2) random sampling of the the dissimilarity matrix. In this paper, the same ideas have been applied to CEVCLUS, a constrained version of EVCLUS making it possible to take into account prior knowledge in the form of ML and CL constraints [9]. Contrary to CEVCLUS, the new k-CEVCLUS algorithm can be applied to datasets with several thousands or tens of thousands of objects, which makes it applicable to a wider range of real-world data. Clustering even larger datasets with, e.g., 10^6 objects or more, might require further algorithmic and theoretical developments which are left for further research.

We have also introduced a constraint expansion strategy, which consists in inferring new constraints from initial ones, resulting in drastic improvements of



(a)

(b)



(c)

Figure 9: ARI as a function of the number nbconst of initial constraints, after running the constraint expansion algorithm, for different values of K. The datasets are: Banana_2000 (a), Letter (b) and Zongker (c).

clustering results when a moderate number of constraints are initially provided. We note that the same technique could be used with other constrained clustering methods such as CECM [2]. In the future, it would be interesting to investigate the relation and the synergies between this new constraint expansion method and active learning techniques [2, 12].

References

[1] Antoine, V., Quost, B., Masson, M.-H., Denoeux, T., 2014. CEVCLUS: evidential clustering with instance-level constraints for relational data. Soft Computing 18 (7), 1321–1335.

[2] Antoine, V., Quost, B., Masson, M.-H., Denoeux, T., 2012. CECM: Constrained evidential c-means algorithm. Computational Statistics & Data Analysis 56 (4), 894–914.

[3] Bezdek, J., 1981. Pattern Recognition with fuzzy objective function algorithm. Plenum Press, New-York.

[4] Cox, T. F., Cox, M. A., 1994. Multidimensional scaling. Chapman and Hall, London.

[5] Denœux, T., 1995. A k-nearest neighbor classification rule based on Dempster-Shafer theory. IEEE Trans. on Systems, Man and Cybernetics 25 (05), 804–813.

[6] Denœux, T., 2016. evclust: Evidential Clustering. R package version 1.0.3. URL `https://CRAN.R-project.org/package=evclust`

[7] Denoeux, T., Kanjanatarakul, O., September 2016. Beyond fuzzy, possibilistic and rough: An investigation of belief functions in clustering. In: Soft Methods for Data Science (Proc. of the 8th International Conference on Soft Methods in Probability and Statistics SMPS 2016). Vol. AISC 456 of Advances in Intelligent and Soft Computing. Springer-Verlag, Rome, Italy, pp. 157–164.

[8] Denœux, T., Kanjanatarakul, O., Sriboonchitta, S., 2015. EK-NNclus: a clustering procedure based on the evidential k-nearest neighbor rule. Knowledge-based Systems 88, 57–69.

[9] Denœux, T., Masson, M.-H., 2004. EVCLUS: Evidential clustering of proximity data. IEEE Trans. on Systems, Man and Cybernetics B 34 (1), 95–109.

[10] Denœux, T., Sriboonchitta, S., Kanjanatarakul, O., 2016. Evidential clustering of large dissimilarity data. Knowledge-based Systems 106, 179–195.

[11] Frigui, H., Hwang, C., Rhee, F. C.-H., 2007. Clustering and aggregation of relational data with applications to image database categorization. Pattern Recognition 40 (11), 3053–3068.

[12] Grira, N., Crucianu, M., Boujemaa, N., 2008. Active semi-supervised fuzzy clustering. Pattern Recognition 41 (5), 1834–1844.

[13] Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of Classification 2 (1), 193–ñ218.

[14] Krishnapuram, R., Keller, J., 1993. A possibilistic approach to clustering. IEEE Trans. on Fuzzy Systems 1, 98–111.

[15] Lelandais, B., Ruan, S., Denœux, T., Vera, P., Gardin, I., 2014. Fusion of multi-tracer PET images for dose painting. Medical Image Analysis 18 (7), 1247–1259.

[16] Lian, C., Ruan, S., Denoeux, T., Li, H., Vera, P., 2017. Spatial evidential clustering with adaptive distance metric for tumor segmentation in FDG-PET images. IEEE Transactions on Biomedical Engineering (In Press).

[17] Lingras, P., Peters, G., 2012. Applying rough set concepts to clustering. In: Peters, G., Lingras, P., Ślezak, D., Yao, Y. (Eds.), Rough Sets: Selected Methods and Applications in Management and Engineering. Springer-Verlag, London, UK, pp. 23–37.

[18] Liu, Z.-G., Pan, Q., Dezert, J., Mercier, G., 2015. Credal c-means clustering method based on belief functions. Knowledge-Based Systems 74 (0), 119–132.

[19] Makni, N., Betrouni, N., Colot, O., 2014. Introducing spatial neighbourhood in evidential c-means for segmentation of multi-source images: Application to prostate multi-parametric MRI. Information Fusion 19, 61–72.

[20] Masson, M.-H., Denoeux, T., 2008. ECM: an evidential version of the fuzzy c-means algorithm. Pattern Recognition 41 (4), 1384–1397.

[21] Masson, M.-H., Denœux, T., 2009. RECM: relational evidential c-means algorithm. Pattern Recognition Letters 30, 1015–1026.

[22] Peters, G., Crespo, F., Lingras, P., Weber, R., 2013. Soft clustering: fuzzy and rough approaches and their extensions and derivatives. International Journal of Approximate Reasoning 54 (2), 307–322.

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

[24] Serir, L., Ramasso, E., Zerhouni, N., 2012. Evidential evolving Gustafson-Kessel algorithm for online data streams partitioning using belief function theory. International Journal of Approximate Reasoning 53 (5), 747–768.

[25] Shafer, G., 1976. A mathematical theory of evidence. Princeton University Press, Princeton, N.J.

[26] ter Braak, C. J., Kourmpetis, Y., Kiers, H. A., Bink, M. C., 2009. Approximating a similarity matrix by a latent class model: A reappraisal of additive fuzzy clustering. Computational Statistics & Data Analysis 53 (8), 3183–3193.

[27] Zhou, K., Martin, A., Pan, Q., Liu, Z.-G., 2015. Median evidential c-means algorithm and its application to community detection. Knowledge-Based Systems 74 (0), 69–88.