evclust: An R Package for Evidential Clustering

Thierry Denœux
Université de technologie de Compiègne

Abstract

One of the current trends in clustering is the development of algorithms able not only to discover groups in data, but also to describe group-membership uncertainty. Evidential clustering is a recent approach in this direction based on the Dempster-Shafer theory of belief functions, a mathematical formalism for uncertainty representation. The output of an evidential clustering procedure is a credal partition, defined as a tuple of mass functions representing the uncertain assignment of objects to clusters. The R package **evclust** described in this paper contains a collection of efficient evidential clustering algorithms, as well as functions to display, evaluate and exploit credal partitions. The application of this package is illustrated though the analysis of several datasets with different characteristics.

Keywords: clustering, belief functions, Dempster-Shafer theory, R.

1. Introduction

Since the introduction of the k-means clustering algorithm in the 1960's (Lloyd 1982), cluster analysis has been a very active research area in computational statistics and machine learning (see, e.g. Jain and Dubes (1988); Kaufman and Rousseeuw (1990); Xu and Wunsch (2009)). There is now an abundance of software tools for data clustering: for instance, the Comprehensive R Archive Network (CRAN) task view on "Cluster Analysis & Finite Mixture Models"[1] lists 107 R packages (as of February 23, 2021). Among them, we can mention the packages **RSKC** for robust and sparse k-means clustering (Kondo, Salibian-Barrera, and Zamar 2016), **pdfCluster** for nonparametric clustering based on density estimation (Azzalini and Menardi 2014), **NbClust** for determining the relevant number of clusters in a data set (Charrad, Ghazzali, Boiteau, and Niknafs 2014), etc.

Whereas *hard clustering* algorithms such as the k-means procedure only generate a partition of the dataset, an important research direction has been to develop algorithms that compute a richer representation reflecting *group-membership uncertainty*. For instance, fuzzy clustering algorithms such as the fuzzy k-means (Bezdek 1981) implemented in the R package **fclust** (Ferraro, Giordani, and Serafini 2019), as well as model-based clustering methods based on the EM algorithm (McLachlan and Basford 1988) implemented in **mclust** (Scrucca, Fop, Murphy, and Raftery 2016) compute *fuzzy partitions*. In a fuzzy partition, each object i is assigned a degree of membership u_{ik} in the range $[0, 1]$ to each cluster k, in such as way that, for each object i; the sum $\sum_k u_{ik}$ of membership degrees (which can also be viewed as probabilities) is equal to one. In *possibilistic clustering* (Krishnapuram and Keller 1993; Yang and Wu 2006), this sum constraint is relaxed: each the computed quantity u_{ik} can then be

[1] https://cran.r-project.org/web/views/Cluster.html

interpreted as the *degree of possibility* that object i belongs to cluster k. Degrees of possibility reflect the *typicality* of each observation, atypical patterns (or outliers) having a low degree of possibility of belonging to any of the clusters. Yet another approach is *rough clustering* (Peters, Crespo, Lingras, and Weber 2013; Peters 2014), in which each object is assigned a *set* of possible clusters. For each cluster, we can then define a *lower approximation*, composed of objects that *certainly* belong to that cluster, and an *upper approximation*, composed of objects that *possibly* belong to it. In R, rough clustering algorithms are implemented in the package **SoftClustering** (Peters 2019).

All the above approaches are subsumed by the *evidential clustering* approach (Denœux and Masson 2004; Masson and Denoeux 2008), a relatively new approach based on the Dempster-Shafer (DS) theory of belief functions (Dempster 1967; Shafer 1976; Denœux, Dubois, and Prade 2020). Evidential clustering algorithms quantify clustering uncertainty using *mass functions* assigning masses to *sets* of clusters, called *focal sets*, in such a way that the masses sum to one. The collection of mass functions related to all objects in the dataset is called a *credal* (or *evidential*) *partition*. A credal partition boils down to a fuzzy partition when the focal sets are singletons, and it can be converted into any of the simpler classical representations such as a fuzzy, possibilistic or rough partition for display and interpretation (Denoeux and Kanjanatarakul 2016). Evidential clustering algorithms include the Evidential c-Means (ECM) algorithm (Masson and Denoeux 2008), a prototype-based procedure in the k-means family, and EVCLUS (Denœux and Masson 2004; Denœux, Sriboonchitta, and Kanjanatarakul 2016), an algorithm inspired by multidimensional scaling (Borg and Groenen 1997) and applicable to non-metric proximity data. Until recently, there was no publicly available software tool for evidential clustering. The R package **evclust** described in this paper fills this gap by proposing an implementation of the main evidential clustering available so far, as well as functions for displaying, evaluating and exploiting credal partitions.

The rest of this paper is organized as follows. The needed notions related to DS theory and the main concepts underlying evidential clustering are first introduced in Section 2. The main algorithms implemented in **evclust** are then described in Section 3, and the application of these algorithms to various clustering problems is illustrated in Section 4. Finally, Section 5 summarizes the paper.

2. Evidential Clustering

Evidential clustering is based on the representation of cluster membership uncertainty using the theory of belief functions. We first recall elements of this theory in Section 2.1. We then introduce the notion of credal partition and its representation within package **evclust** in Section 2.2.

2.1. Theory of Belief Functions

We start with the consideration of some question Q, which has one and only answer among a finite set of possibilities $\Omega = \{\omega_1, \ldots, \omega_c\}$ (called the *frame of discernment*). We assume that any evidence about Q can be represented by a *mass function* on Ω, defined as a mapping from the power set 2^Ω to the interval $[0, 1]$, such that $\sum_{A \subseteq \Omega} m(A) = 1$. Each subset A of Ω such that $m(A) > 0$ is called a *focal set* of m. Each mass $m(A)$ represents the degree to which the evidence supports A, without supporting any strict subset of A (Shafer 1976). A

mass function m is said to be *logical* if it has only one focal set, *consonant* of its focal sets are nested (i.e., for any two focal sets A and B, we have either $A \subseteq B$ or $B \subseteq A$), and *Bayesian* if its focal sets are singletons. A mass function that is both logical and Bayesian has only one singleton focal set: it is said to be *certain*. If Ω is the only focal set, mass function m is said to be *vacuous* and it represents total ignorance.

Belief and plausibility functions. Whereas a probability mass function induces a probability measure, a DS mass function induces two dual nonadditive measures: a *belief function*, defined as

$$Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \tag{1}$$

for all $A \subseteq \Omega$ and a *plausibility function* defined as

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B). \tag{2}$$

These two functions are linked by the relation $Pl(A) = Bel(\Omega) - Bel(\overline{A})$, for all $A \subseteq \Omega$. The quantity $Bel(A)$ is a measure of total support for A (taking into account the support given to its subsets), while $Bel(\Omega) - Pl(A) = Bel(\overline{A})$ is a measure of support for the complement $\overline{A}$ of A, so that $Pl(A)$ can be seen as a measure of lack of support for $\overline{A}$. The function $pl : \Omega \to [0, 1]$ that maps each element ω of Ω to its plausibility $pl(\omega) = Pl(\{\omega\})$ is called the *contour function* associated to m. When m is consonant, the whole plausibility function can be recovered from pl as $Pl(A) = \max_{\omega \in \Omega} pl(\omega)$. Function pl is then called a *possibility distribution* (Zadeh 1978).

Conjunctive sum and degree of conflict. Let m_1 and m_2 be two mass functions defined on the same frame Ω. Their *conjunctive sum* (Smets and Kennes 1994) $m_1 \cap m_2$ is the mass function defined as

$$(m_1 \cap m_2)(A) = \sum_{B \cap C = A} m_1(B) m_2(C) \tag{3}$$

for all subset $A \subseteq \Omega$. The quantity $(m_1 \cap m_2)(\emptyset)$ is called *degree of conflict* (Shafer 1976) between m_1 and m_2 and it is denoted as $\kappa(m_1, m_2)$. When m_1 and m_2 represent two independent pieces of evidence pertaining to *the same question*, $\kappa(m_1, m_2)$ can be interpreted as a *measure of conflict* between these two pieces of evidence (Shafer 1976). In contrast, when m_1 and m_2 represent independent pieces of evidence about *two distinct questions* Q_1 and Q_2 with the same frame of discernment Ω, $1 - \kappa(m_1, m_2)$ can be given a different interpretation as the plausibility that the true answers to Q_1 and Q_2 are identical, as shown by Denœux and Masson (2004) (see Section 2.2 below).

Distance measures. The degree of conflict measures the disagreement between two mass functions, but it does not measure their dissimilarity. In particular, the degree of conflict $\kappa(m, m)$ between a mass function and itself is usually strictly positive. Several distance measures for belief functions have been proposed (Jousselme and Maupin 2012). One of the most widely used is Jousselme's distance (Jousselme, Grenier, and Bossé 2001), which is defined as follows. Let $A_1, \ldots, A_N$ be the subsets of Ω arranged in some order, with $N = 2^c$. A mass function m can be represented by an N-vector $\mathbf{m} = (m(A_1), \ldots, m(A_N))^T$. Let $\mathbf{J}$ be

the positive definite and symmetric $N \times N$ matrix whose general term $[\mathbf{J}]_{ij}$ is the Jaccard index between sets A_i and A_j:

$$[\mathbf{J}]_{ij} = \begin{cases} 1 & \text{if } A_i = A_j = \emptyset \\ \frac{|A_i \cap A_j|}{|A_i \cup A_j|} & \text{otherwise.} \end{cases}$$

The Jousselme distance between two mass functions m_1 and m_2 on the same frame Ω is defined as

$$d_J(m_1, m_2) = \left[\frac{1}{2}(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{J}(\mathbf{m}_1 - \mathbf{m}_2)\right]^{1/2}. \tag{4}$$

It verifies $0 \leq d_J(m_1, m_2) \leq 1$ for all m_1 and m_2. Another useful distance measure is the *belief distance* (Denœux 2001), defined as

$$d_B(m_1, m_2) = \frac{1}{2} \sum_{A \subseteq \Omega} \mid Bel_1(A) - Bel_2(A) \mid, \tag{5}$$

where Bel_1 and Bel_2 are, respectively, the belief functions corresponding to m_1 and m_2. We also have $0 \leq d_B(m_1, m_2) \leq 1$ for all m_1 and m_2.

Summarization of a mass function. It is often useful to summarize the information contained in a mass function for communication or display purposes. This can be done in a number of ways. Here, we mention only two approaches implemented in **evclust**. The first approach is to transform a mass function m into a probability distribution. This can be done using the so-called *plausibility transformation* method (Voorbraak 1989), which consists in normalizing the contour function, resulting in the following probability distribution:

$$p_m(\omega) = \frac{pl(\omega)}{\sum_{\omega'=1}^{c} pl(\omega')}, \tag{6}$$

for all $\omega \in \Omega$. Alternatively, the *pignistic transformation* (Smets and Kennes 1994) distributes each normalized mass $m(A)/(1 - m(\emptyset))$ uniformly to the elements of A:

$$betp_m(\omega) = \sum_{\{A \subset \Omega : \omega \in A\}} \frac{m(A)}{(1 - m(\emptyset))|A|}. \tag{7}$$

The second approach is to approximate a mass function m as a set. A simple choice is to select the focal set $A^*(m)$ with the largest mass:

$$A^*(m) = \arg\max_{B \subseteq \Omega} m(B). \tag{8}$$

As an alternative with a more decision-theoretic foundation (Denœux 2019), we may consider the strict *interval dominance* relation $\succ$ on Ω defined as follows: ω dominates ω' (denoted as $\omega \succ \omega'$) iff $Bel(\{\omega\}) > Pl(\{\omega'\})$, i.e., the degree of belief in ω is larger than the degree of plausibility in ω' (meaning that ω is unambiguously more supported than ω'). The set $A^\circ(m)$ of maximal (i.e., non-dominated) elements of this relation is then defined as

$$A^\circ(m) = \{\omega \in \Omega : \forall \omega' \in \omega, Bel(\{\omega'\}) \leq Pl(\{\omega\}\}. \tag{9}$$

Nonspecificity. Several measures have been proposed to quantify the degree of uncertainty of a mass function. A mass function basically represents two kinds of uncertainty: *imprecision* and *conflict* (Klir and Wierman 1999). For instance, the vacuous mass function is maximally imprecise but not conflicting, whereas the uniform Bayesian mass function (such that $m(\{\omega\}) = 1/c$ for all $\omega \in \Omega$) has maximal conflict, but no imprecision. Several uncertainty measures quantifying imprecision, conflict, or both have been proposed. As a measure of imprecision, *nonspecificity* seems particularly well justified (Klir and Wierman 1999). It is defined as follows:

$$N(m) = \sum_{\emptyset \neq A \subseteq \Omega} m(A) \log_2 |A| + m(\emptyset) \log_2 c. \tag{10}$$

Nonspecifity is maximal for the vacuous mass function verifying $m(\Omega) = 1$, and also for the mass function m such that $m(\emptyset) = 1$. The interpretation of the mass assigned to the empty set in evidential clustering will be addressed in Section 2.2. Masson and Denoeux (2008) proposed a method to determine the number of clusters in an evidential partition based on the nonspecificity measure (10), which has been implemented in **evclust** (see Section 2.2 below).

2.2. Credal Partition

Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be a set of n objects. We assume that each object in $\mathcal{O}$ belongs to *at most* one cluster in a set $\Omega = \{\omega_1, \ldots, \omega_c\}$. Using the formalism recalled in Section 2.1, evidence about the cluster membership of each object o_i can be described by a mass function m_i defined on the frame on Ω. The n-tuple $\mathcal{M} = (m_1, \ldots, m_n)$ is called an *credal* (or *evidential*) *partition* of $\mathcal{O}$.

Generality of credal partitions. The notion of evidential partition encompasses most classical clustering structures (Denoeux and Kanjanatarakul 2016). In particular, when all mass functions m_i are certain, $\mathcal{M}$ becomes equivalent to a hard partition; this case corresponds to full certainty about the group of each object. When mass functions are Bayesian, $\mathcal{M}$ boils down to a fuzzy partition; the degree of membership u_{ik} of object i to group k is then $u_{ik} = Bel_i(\{\omega_k\}) = Pl_i(\{\omega_k\}) \in [0, 1]$ and we have $\sum_{k=1}^{c} u_{ik} = 1$. When all mass functions m_i are consonant, they are equivalently represented by their contour functions pl_i, and $pl_{ik} = pl_i(\omega_k)$ can be interpreted as the degree of possibility that object o_i belongs to cluster ω_k, as computed by possibilistic clustering algorithms. Finally, when each mass function m_i is logical with focal set $A_i \subseteq \Omega$, m_i is equivalent to a rough partition: the lower and upper approximations of cluster ω_k are then defined, respectively, as follows:

$$\omega_k^l := \{i \in \mathcal{O} \mid A_i = \{\omega_k\}\} \quad \text{and} \quad \omega_k^u := \{i \in \mathcal{O} \mid \omega_k \in A_i\}. \tag{11}$$

They are interpreted, respectively, as the set of objets *surely* belong to cluster ω_k, and the set of objects *possibly* belong to ω_k. We then have $Bel_i(\{\omega_k\}) = I(i \in \omega_k^l)$ and $Pl_i(\{\omega_k\}) = I(i \in \omega_k^u)$, where $I(\cdot)$ denotes the indicator function.

Summarization of a credal partition. Being more general than classical clustering structures, a credal partition can be summarized into any of them. For instance, we obtain a fuzzy partition by transforming each mass function m_i into a probability distribution using (6) or (7). A hard partition can then be obtained by selecting for each object the cluster with the

highest probability. Alternatively, we may summarize the credal partition into a rough partition by approximating each mass function m_i by a single set A_i using (8) or (9); the lower and upper approximations of each cluster can then be computed using (11).

Representation in evclust. Credal partitions are represented in **evclust** by `S3` objects of class '`credpart`'. Some of the most important attributes of a '`credpart`' object are the following:

`method`: The clustering algorithm used to generate the credal partition;

`F`: The focal sets, encoded as a matrix of binary numbers with c columns and f rows, where f is the number of focal sets; the first row must correspond to the empty set;

`mass`: The mass functions, encoded as a matrix of size $n \times f$;

`pl`: The contour functions, encoded as a matrix of size $n \times c$;

`p`: The probability distributions computed by the plausibility transformation (6), encoded as a matrix of size $n \times c$;

`y.pl`: The n-vector of maximum-plausibility class labels;

`Y`: The maximum-mass sets $A^*(m_i)$, encoded as a matrix of 0's and 1's of size $n \times c$;

`upper.approx`: The upper approximation of the credal partition, computed from the sets $A_i = A^*(m_i)$ using (11), provided as a list of length c. The k-th component of the list is the vector of indices of ω_k^u.

`lower.approx`: The lower approximation of the credal partition, computed from the sets $A_i = A^*(m_i)$ using (11), provided a list of length c. The k-th component of the list is the vector of indices of ω_k^l.

`N`: The average nonspecificity of the credal partition, normalized to range between 0 and 1.

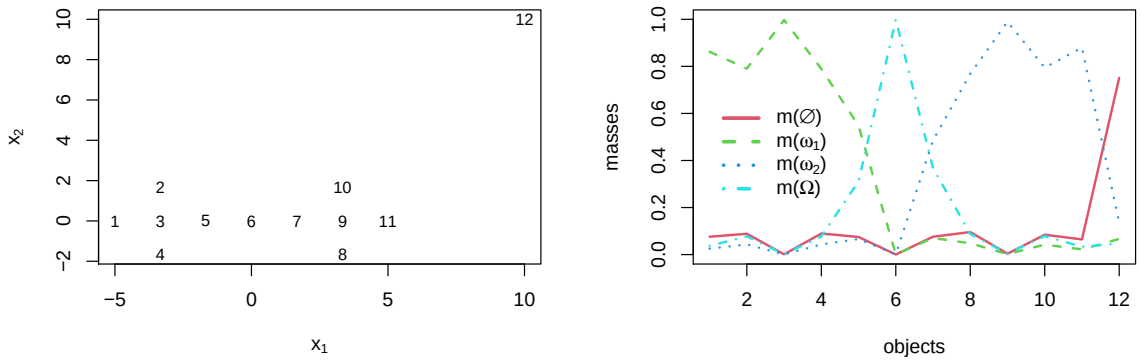
A '`credpart`' object also contains algorithm-specific information such as the prototypes (for prototype-based algorithms such as ECM) and the value of the criterion for algorithms minimizing a cost function. `S3` methods `summary()` and `plot()` display basic information about objects of class '`credpart`'.

Consider, for instance, the `butterfly` data:

```
R> data(butterfly, package="evclust")
R> x<-butterfly
```

This is a toy dataset composed of 12 objects with two attributes (Figure 1a). Objects 1 to 11 can be naturally partitioned in two clusters, and object 12 is an outlier.

Function `ecm()` implements the ECM algorithm (see Section 3.1 below); it returns an object of class '`credpart`':



(a) (b)

Figure 1: `butterfly` dataset (a) and credal partition generated by the ECM algorithm (b).

```
R> library("evclust")
R> set.seed(210121)
R> clus<-ecm(x, c=2, delta=5, disp=FALSE)
R> summary(clus)

------ Credal partition ------
2 classes,12 objects
Generated by ecm
Focal sets:
     [,1] [,2]
[1,]    0    0
[2,]    1    0
[3,]    0    1
[4,]    1    1
Value of the criterion = 35.16
Nonspecificity = 0.29
Prototypes:
          [,1]        [,2]
[1,] -3.521806 0.005540579
[2,]  3.641838 0.069230936
Number of outliers = 1.00
```

The mass functions are encoded in a matrix `clus$mass` of 12 rows and 4 columns, corresponding to the four focal sets in `clus$F`. They are displayed in Figure 1b. We can see that Object 6 has a large mass on $\Omega = \{\omega_1, \omega_2\}$ because it cannot be unambiguously assigned to any of the two classes, while Object 12 has a large mass on the empty set, which signals it as an outlier.

Relational representation. A (hard) partition of a dataset can be represented in two ways: as a list of object subsets, or as the incidence matrix of the corresponding equivalence

relation. Similarly, a credal partition admits a *relational representation* (Denœux, Li, and Sriboonchitta 2018), defined as a tuple $R = (m_{ij})_{1\leq i<j\leq n}$, in which m_{ij} is a so-called *pairwise mass function* on the frame $\Theta_{ij} = \{s_{ij}, \neg s_{ij}\}$, where s_{ij} denotes the proposition "Objects i and j belong to the same cluster", and $\neg s_{ij}$ denotes the negation of s_{ij}. As proved by Denœux *et al.* (2018), paiwise mass function m_{ij} can be obtained from m_i and m_j as

$$m_{ij}(\emptyset) = m_i(\emptyset) + m_j(\emptyset) - m_i(\emptyset)m_j(\emptyset) \tag{12a}$$

$$m_{ij}(\{s_{ij}\}) = \sum_{k=1}^{c} m_i(\{\omega_k\})m_j(\{\omega_k\}) \tag{12b}$$

$$m_{ij}(\{\neg s_{ij}\}) = \sum_{A\cap B=\emptyset} m_i(A)m_j(B) - m_{ij}(\emptyset) \tag{12c}$$

$$m_{ij}(\Theta_{ij}) = \sum_{A\cap B\neq\emptyset} m_i(A)m_j(B) - m_{ij}(\{s_{ij}\}). \tag{12d}$$

The degree of plausibility that objects i and j belong to the same cluster is, thus,

$$pl_{ij}(s_{ij}) = m_{ij}(\{s_{ij}\}) + m_{ij}(\Theta_{ij}) = \sum_{A\cap B\neq\emptyset} m_i(A)m_j(B) = 1 - \kappa_{ij}, \tag{13}$$

where κ_{ij} is the degree of conflict between mass functions m_i and m_j.

In **evclust**, function `pairwise_mass()` takes as input a '`credpart`' object and returns its relational representation as a list of three '`dist`' objects `Me`, `M1` and `M0` corresponding, respectively, to $(m_{ij}(\emptyset))_{i<j}$, $(m_{ij}(\{s_{ij}\}))_{i<j}$ and $(m_{ij}(\{\neg s_{ij}\}))_{i<j}$.

The relational representation is useful, in particular, to compare two credal partitions. Denœux *et al.* (2018) propose two extensions of the Rand index, a classical measure to compare two hard partitions: a *similarity index*, defined as

$$\rho_S(\mathcal{M}, \mathcal{M}') = 1 - \frac{2\sum_{i<j} d(m_{ij}, m'_{ij})}{n(n-1)}, \tag{14}$$

where $\mathcal{M}$ and $\mathcal{M}'$ are two credal partitions and d denotes either the Jousselme distance (4) or the belief distance (5), and a *consistency index*, defined as

$$\rho_C(\mathcal{M}, \mathcal{M}') = 1 - \frac{2\sum_{i<j} \kappa(m_{ij}, m'_{ij})}{n(n-1)}, \tag{15}$$

where κ denotes the degree of conflict. In **evclust**, these two indices are computed by function `credal_RI()`, which has three arguments: two relational representation objects `P1` and `P2`, and `type` with possible values '`c`' for 'consistency', '`j`' for Jousselme's distance, and '`b`' for belief distance.

3. Evidential Clustering Algorithms

Table 1 presents an overview of the evidential clustering algorithms implemented in **evclust**. Some of these algorithms require attribute data, whereas others can handle proximity data, i.e., a matrix of dissimilarities, or distances between objects. Naturally, algorithms in the latter category can also be applied to attribute data after computing a distance matrix.

Function	Name	Inputs
`ecm()`	Evidential c-means	attribute data
`recm()`	Relational evidential c-means	proximity data
`cecm()`	Constrained evidential c-means	attribute data + constraints
`bpec()`	Belief peak evidential clustering	attribute data
`kevclus()`	EVCLUS	proximity data
`kcevclus()`	CEVCLUS	proximity data + constraints
`nnevclus()`	NN-EVCLUS	attribute data (+ constraints + labels)
`bootclus()`	Bootstrap evidential clustering	attribute data
`EkNNclus()`	EK-NNclus	attribute data

Table 1: Overview of clustering algorithms in **evclust**.

Finally, some algorithms can also handle pairwise constraints, or can be trained in supervised learning mode (with class labels for some objects). These algorithms are reviewed below.

3.1. Evidential c-Means and Variants

Introduced by Masson and Denoeux (2008), the evidential c-means (ECM) algorithm is an alternating optimization procedure in the same family as the hard and fuzzy c-means algorithms (Bezdek 1981). As these algorithms, ECM represents each cluster $\omega_k \in \Omega$ by a prototype $\mathbf{g}_k \in \mathbb{R}^p$, where p is the dimension of the data. However, a particularity of ECM is that it also represents each nonempty set of clusters (or *meta-cluster*) $A_j \subseteq \Omega$ by a prototype $\overline{\mathbf{g}}_j$, defined as the barycenter of the prototypes $\mathbf{g}_k$ for $\omega_k \in A_j$:

$$\overline{\mathbf{g}}_j = \frac{1}{|A_j|} \sum_{\omega_k \in A_j} \mathbf{g}_k.$$

Having specified a set $\mathcal{F} = \{A_1, \ldots, A_f\} \subset 2^\Omega$ of f nonempty subsets of Ω, the cost function of ECM is defined as

$$J_{\text{ECM}}(\mathcal{M}, \mathbf{G}) = \sum_{i=1}^{n} \sum_{j=1}^{f} |A_j|^\alpha m_{ij}^\beta d_{ij}^2 + \sum_{i=1}^{n} \delta^2 m_{i\emptyset}^\beta, \tag{16}$$

where $\mathcal{M} = (m_1, \ldots, m_n)$ is the credal partition, $\mathbf{G}$ is the $c \times p$ matrix of prototypes, $d_{ij} = \|\mathbf{x}_i - \overline{\mathbf{g}}_j\|$ is the Euclidean distance between attribute vector $\mathbf{x}_i$ and prototype $\overline{\mathbf{g}}_j$, $m_{ij} = m_i(A_j)$ and $m_{i\emptyset} = m_i(\emptyset)$. Parameters α, β and δ control, respectively, the nonspecificity of the credal partition (larger values of α favor the allocation of mass to smaller focal sets), the "fuzziness" of the partition (higher values of β favor more uniform allocation of masses to focal sets), and the proportion of outliers (smaller values of δ result in more outliers). ECM minimizes cost function $J_{\text{ECM}}(\mathcal{M}, \mathbf{G})$ by alternating two steps:

1. Update of $\mathcal{M}$ for fixed $\mathbf{G}$, as

$$m_{ij} = \frac{|A_j|^{-\alpha/(\beta-1)} d_{ij}^{-2/(\beta-1)}}{\sum_{k=1}^{f} |A_k|^{-\alpha/(\beta-1)} d_{ik}^{-2/(\beta-1)} + \delta^{-2/(\beta-1)}}, \tag{17}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, f$, and $m_{i\emptyset} = 1 - \sum_{j=1}^{f} m_{ij}$ for $i = 1, \ldots, n$.

2. Update of $\mathbf{G}$ for fixed $\mathcal{M}$ by solving a linear system of the form $\mathbf{HG} = \mathbf{B}$, where $\mathbf{B}$ is a matrix of size $c \times p$ and $\mathbf{H}$ a matrix of size $c \times c$.

In **evclust**, ECM is implemented in function `ecm()`:

```
ecm(x, c, g0 = NULL, type = "full", pairs = NULL, Omega = TRUE,
  ntrials = 1, alpha = 1, beta = 2, delta = 10, epsi = 0.001, disp = TRUE)
```

where the arguments without a default value are the input data matrix `x` (of size $n \times p$, where p is the number of attributes) and the number `c` of clusters. Argument `g0` is an optional matrix of initial prototypes. Argument `type` specifies the focal sets: its values are `"full"` for all the subsets of Ω, `"simple"` for the empty set, the singletons and Ω, and `"pairs"` for the empty set, the singletons, Ω and either all pairs, or only the pairs specified by `pairs`. Setting `Omega = FALSE` removes Ω from the list of focal sets. The meaning of the other arguments is either obvious or can be found using `?ecm`. The output is a '`credpart`' object encoding the computed evidential partition.

RECM. The relational evidential c-means (RECM) algorithm is a relational version of ECM, which takes as input a distance matrix instead of a matrix of attribute values (Masson and Denœux 2009). The convergence of the algorithm is guaranteed if the distances are Euclidean, but the algorithm will often converge even if they are not. Function `recm()` has the same arguments as `ecm()`, except that the first argument is a distance matrix `D` instead of an object-attributes matrix `x`. Also, the algorithm can be initialized with a matrix `m0` of masses, instead of the matrix `g0` of initial prototypes.

CECM. The constrained Evidential c-Means (CECM) algorithm (Antoine, Quost, Masson, and Denoeux 2012) is another variant of ECM that takes as inputs not only attribute data, but also pairwise constraints specifying that some pairs of object belong to the same cluster (*must-link* constraint) or belong to different clusters (*cannot-link* constraint). The CECM algorithm minimizes the following penalized cost function:

$$J_{\text{CECM}}(\mathcal{M}, \mathbf{G}) = (1-\xi) J_{\text{ECM}}(\mathcal{M}, \mathbf{G}) + \frac{\xi}{|\mathsf{ML}| + |\mathsf{CL}|} \left[\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathsf{ML}} pl_{ij}(\neg s_{ij}) + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathsf{CL}} pl_{ij}(s_{ij}) \right], \quad (18)$$

where ML and CL are, respectively, the sets of must-link and cannot-link constraints, pl_{ij} is the contour function corresponding to pairwise mass function m_{ij} defined by (12), and ξ is a penalization coefficient. As ECM, CECM uses an alternating optimization scheme, minimizing the cost function with respect to $\mathcal{M}$ and $\mathbf{G}$ in turn. Here, the former optimization problem does not admit a closed-form solution and needs to be solved by a quadratic programming procedure.

A version of CECM uses an adaptive metric to account for elliposidal clusters, in a similar way as the Gustafson-Kessel fuzzy clustering algorithm (Gustafson and Kessel 1979). A symmetric and positive definite matrix $\mathbf{S}_k$ is defined for each cluster ω_k, and the matrix $\overline{\mathbf{S}}_j$ of each meta-cluster $A_j \subseteq \Omega$ is defined as the average of matrices $\mathbf{S}_k$ for $\omega_k \in A_j$. The squared distance d_{ij}

between $\mathbf{x}_i$ and the center $\overline{\mathbf{g}}_j$ of meta-cluster A_j is then defined as $d_{ij}^2 = (\mathbf{x} - \overline{\mathbf{g}}_j)^T \overline{\mathbf{S}}_j (\mathbf{x} - \overline{\mathbf{g}}_j)$. The cost function (18) is then minimized with respect to $\mathcal{M}$, $\mathbf{V}$ and the matrices $\mathbf{S}_k$.

The CECM algorithm is implemented in function `cecm()`, which has the same arguments as `ecm()`, and four additional arguments: `ML`, `CL`, `xi` (with obvious meanings), and `distance` with possible values 0 for Euclidean distance and 1 for adaptive metric.

BPEC. The belief peak evidential clustering (BPEC) method (Su and Denœux 2019) is similar to ECM, except that the cluster centers are selected as the "belief peaks" in a two dimensional "δ-Bel" graph, defined as points of high density (estimated by a K nearest neighbor approach) and located far from points with a higher density. The ECM algorithm is then run, keeping the cluster centers constant. In **evclust**, function `delta_Bel()` takes as input the data matrix `x`, the number `K` or neighbors and a scale parameter `q`. It draws the δ-Bel graph, and the user is invited to manually select the lower right corner of a rectangle containing the belief peaks, which are then returned as output. Function `bpec()`, whose syntax is similar to that of `ecm()` can then be run with the belief peaks as fixed cluster centers.

3.2. EVCLUS and Variants

The EVCLUS algorithm, introduced by Denœux and Masson (2004) and improved by Denœux *et al.* (2016), is another evidential clustering algorithm that takes inspiration from multidimensional scaling (Borg and Groenen 1997). It takes as input a symmetric $n \times n$ dissimilarity matrix $\mathbf{D} = (\delta_{ij})$, where δ_{ij} denotes the dissimilarity between objects o_i and o_j. Dissimilarities may be computed from attribute data, or they may be directly available. They need not satisfy the axioms of a distance such as the triangular inequality.

The fundamental assumption underlying EVCLUS is that the more similar are two objects, the more plausible it is that they belong to the same cluster. From (13), the plausibility $pl_{ij}(s_{ij})$ that two objects o_i and o_j belong to the same cluster is equal to $1 - \kappa_{ij}$, where κ_{ij} is the degree of conflict between m_i and m_j. The credal partition $\mathcal{M}$ should thus be determined in such a way that similar objects have mass functions m_i and m_j with low degree of conflict, whereas highly dissimilar objects are assigned highly conflicting mass functions. This can be achieved by minimizing the following loss function:

$$\mathcal{L}(\mathcal{M}) = \frac{2}{n(n-1)} \sum_{i<j} \left(\kappa_{ij} - \varphi(\delta_{ij})\right)^2, \tag{19}$$

where φ is a fixed nondecreasing mapping from $[0, +\infty)$ to $[0, 1]$, such as $\varphi(\delta) = 1 - \exp(-\eta\delta^2/d_0^2)$. Setting η to $-\log\alpha$, d_0 is the threshold such that two objects o_i and o_j with dissimilarity larger than d_0 have a plausibility at least α of belonging to different clusters. Typically, $\alpha = 0.05$ and d_0 is set to some quantile of the dissimilarities δ_{ij}.

Computing the loss function (19) requires to store the whole dissimilarity matrix, which may not be feasible for large datasets. Denœux *et al.* (2016) showed that it is sufficient to minimize the sum of squared errors for a subset of k object pairs. This is achieved by changing the loss function (19) to

$$\mathcal{L}(\mathcal{M}; J) = \frac{1}{nk} \sum_{i=1}^{n} \sum_{j \in J(i)} \left(\kappa_{ij} - \varphi(\delta_{ij})\right)^2, \tag{20}$$

where $J(i)$ is a randomly selected subset of $\{1,\ldots,n\}\setminus\{i\}$ with cardinality $k \leq n-1$. In **evclust**, EVCLUS is implemented in function `kevclus()`:

```
kevclus(x, k=n-1, D, J, c, type='simple', pairs=NULL, m0=NULL, ntrials=1,
  disp=TRUE, maxit=1000, epsi=1e-5, d0=quantile(D,0.9), tr=FALSE,
  change.order=FALSE, norm=1)
```

where most arguments have obvious meanings. If an object-attribute data matrix `x` of size $n \times p$ is supplied, then the dissimilarities are computed as the Euclidean distances between the rows of `x`. Otherwise, a dissimilarity matrix `D` must be supplied. If `D` is not square and has size $n \times k$, then a matrix `J` of the same size containing the indices to be used for computing the loss function (20) must be provided. Arguments `type` and `pairs` have the same meaning as in `ecm()`. The meaning of the other arguments can be found using `?kevclus`. The output is a '`credpart`' object encoding the computed evidential partition. It contains, in addition to the usual attributes of a '`credpart`' object, a matrix `Kmat` of degrees of conflict (of size $n \times k$) and a matrix `D` of the same size containing the transformed dissimilarities $\varphi(\delta_{ij})$.

CEVCLUS. CEVCLUS is a constrained version of EVCLUS that uses must-link and cannot-link constraints (Antoine, Quost, Masson, and Denoeux 2014; Li, Li, and Denœux 2018). It minimizes the following loss function

$$\mathcal{L}_C(\mathcal{M};J) = \mathcal{L}(\mathcal{M};J) + \frac{\xi}{2(|\mathsf{ML}|+|\mathsf{CL}|)}\left(\sum_{(i,j)\in\mathsf{ML}} pl_{ij}(\neg s_{ij}) + 1 - pl_{ij}(s_{ij})] + \sum_{(i,j)\in\mathsf{CL}} [pl_{ij}(s_{ij}) + 1 - pl_{ij}(\neg s_{ij})]\right) \quad (21)$$

where, as before, ξ is a penalization coefficient, and ML and CL denote, respectively, the sets of must-link and cannot-link constraints. In **evclust**, the CEVCLUS algorithm is implemented in function `kcevclus()`, which has the same arguments as `kevclus()`, and three additional arguments: `ML`, `CL`, and `xi`.

NN-EVCLUS. As opposed to ECM reviewed in Section 3.1, the EVCLUS algorithm does not build a compact representation of clusters as a collection of prototypes, but it learns an evidential partition of the n objects directly. If each mass function is constrained to have f focal sets, the number of free parameters is, thus, $n(f-1)$, i.e., it grows linearly with the number of objects. This characteristic makes EVCLUS impractical for clustering very large datasets. Also, the algorithm learns an evidential partition of a given dataset, but it does not allow us to extrapolate beyond the learning set and make predictions for new objects. The NN-EVCLUS algorithm addresses these issues by learning a mapping from attribute vectors to mass functions using a multilayer feedforward neural network (Denœux 2020b). The network is trained in an unsupervised way by minimizing a loss function similar to (19) or (20) with respect to the network weights. To enhance its robustness to outliers, the learning algorithm can also use the output of a one-class support vector machine such as implemented in the R package **kernlab** (Karatzoglou, Smola, Hornik, and Zeileis 2004); the mass functions computed by the neural network are then transformed so that outliers receive a large mass on

the empty set. The network can also be trained in semi-supervised mode using either must-link and cannot-link constraints (in which case a loss function similar to (21) is minimized), or using class labels for some learning instances. In the latter case, we assume that we have n_s labeled attribute vectors $\{(\mathbf{x}_i, y_i), i \in \mathcal{I}_s\}$, where $\mathcal{I}_s \subseteq \{1, \ldots, n\}$ and $y_i \in \Omega$ is the class label of object i, and the following term is added to the loss function:

$$\frac{\nu}{n_s} \sum_{i \in \mathcal{I}_s} \sum_{l=1}^{c} (pl_{il} - y_{il})^2. \tag{22}$$

where $y_{il} = I(y_i = \omega_l)$, $pl_{il} = pl_i(\omega_l)$, pl_i is the contour function corresponding to m_i, and ν is a penalization coefficient.

The function `nnevclus()` implementing the NN-EVCLUS algorithm has the following syntax:

```
nnevclus(x, k=n-1, D=NULL, J=NULL, c, type='simple', n_H, ntrials=1,
  d0=quantile(D,0.9), fhat=NULL, lambda=0,y=NULL, Is=NULL, nu=0,
  ML=NULL, CL=NULL, xi=0, tr=FALSE, options=c(1, 1000, 1e-4, 10),
  param0=list(U0=NULL, V0=NULL, W0=NULL, beta0=NULL))
```

Function `nnevclus()` shares most arguments with `kevclus()` and `kcevclus()`. Specific arguments include: `n_H`, which specifies the architecture of the neural network: it is either a scalar equal to the number of units in a single hidden layer, or a two-dimensional vector containing the sizes of two hidden layers; `fhat` is an optional vector of one-class support vector machine (SVM) outputs; `lambda` is a weight decay (L_2 regularization) coefficient; `y` is a vector of class labels for some instances, `Is` is a vector of corresponding indices, and `nu` is the penalization coefficient of the supervised part of the loss function; `param0` is a list of initial network parameters. More details can be found using the instruction `?nnevclus`.

Function `nnevclus()` uses a batch learning algorithm, which compute the gradient of the average loss at each iteration. For large datasets, a stochastic gradient descent (SGD) algorithm is more suitable. Function `nnevclus_mb()` implements RMSprop, an accelerated SGD algorithm (Goodfellow, Bengio, and Courville 2016). Function `nnevclus_mb()` is called as follows:

```
nnevclus_mb(x, foncD=function(x) as.matrix(dist(x)), c, type='simple', n_H,
  nbatch=10, alpha0=0.9, fhat=NULL, lambda=0, y=NULL, Is=NULL, nu=0,
  disp=TRUE, options=list(Niter=1000, epsi=0.001, rho=0.9, delta=1e-8,
  Dtmax=100, print=5), param0=list(V0=NULL, W0=NULL, beta0=NULL))
```

where `foncD` is a function that computes distances, `nbatch` is the number of mini-batches used by the SGD algorithm, `alpha0` is the order of the quantile used to compute parameter d_0, and `epsi`, `rho` and `delta` are the usual parameters of RMSprop. Currently, the network architecture is limited to one hidden layer, and semi-supervised learning with pairwise constraints has not been implemented.

3.3. Bootclus

The Bootclus algorithm uses the bootstrap and a model-based clustering approach to construct a credal partition that reflects second-order cluster-membership uncertainty (Denœux 2020a). The method can be summarized as follows:

1. A finite mixture model, typically a Gaussian mixture model (GMM) is postulated;
2. B bootstrap samples are generated;
3. Parameter estimates are computed from each bootstrap sample using the expectation-maximization (EM) algorithm;
4. Let P_{ij} denote the probability that objects i and j belong to the same class. A bootstrap percentile confidence interval $[P_{ij}^l, P_{ij}^u]$ at level $1-\alpha$ on P_{ij} is computed for each object pair (i, j);
5. A normalized credal partition $\mathcal{M} = (m_1, \ldots, m_n)$ (such that $m_i(\emptyset) = 0$ for all i) is obtained as the solution of the following minimization problem:

$$\min_{\mathcal{M}} J(\mathcal{M}) = \sum_{i<j} \left(m_{ij}(\{s_{ij}\}) - P_{ij}^l\right)^2 + \left(m_{ij}(\{\neg s_{ij}\}) - (1 - P_{ij}^u)\right)^2,$$

   where $m_{ij}(\{s_{ij}\})$ and $m_{ij}(\{\neg s_{ij}\})$ are computed using, respectively, Eqs. (12b) and (12c).

The resulting credal partition is such that, for any pair (i, j) of objects, $Bel_{ij}(\{s_{ij}\}) \approx P_{ij}^l$ and $Pl_{ij}(\{s_{ij}\}) \approx P_{ij}^u$, where Bel_{ij} and Pl_{ij} are, respectively, the belief and plausibility functions corresponding to m_{ij}. As a result, the intervals $[Bel_{ij}(\{s_{ij}\}), Pl_{ij}(\{s_{ij}\})]$ are approximate $1-\alpha$ confidence intervals on the pairwise probabilities: the credal partition is said to be *calibrated*.

In **evclus**, the Bootclus algorithm is implemented in function `bootclus()`:

```
bootclus(x, conf=0.90, B=500, param=list(G=NULL), type="pairs", Omega=FALSE)
```

where `x` is the object-attribute matrix of size $n \times p$, `conf` is the confidence degree $1-\alpha$ and `B` is the number of bootstrap samples. Function `bootclus()` calls functions `Mclust()` and `MclustBootstrap()` of package **mclust** for model-based clustering with GMMs (Scrucca *et al.* 2016). The argument `param` contains a list of arguments passed to `Mclust()`. Arguments `type` and `Omega` have the same meanings as the corresponding arguments of function `ecm()` (see Section 3.1).

The output of function `bootclus()` is a list containing the credal partition (a '`credpart`' object), the mixture model estimation results provided as an '`Mclust`' object by function `Mclust()`, as well as the confidence intervals $[P_{ij}^l, P_{ij}^u]$ and the approximating pairwise belief and plausibility degrees.

3.4. EK-NNclus

Many clustering algorithms are based on the so-called "decision-directed" approach, in which an initial classifier is used to label the learning instances; the classifier is then updated, and the process is repeated until no changes occur in the labels. For instance, the c-means algorithm is based on this principle: in that case, the nearest-prototype classifier is used to label the samples, and it is updated by recomputing the prototypes as the cluster centers.

EK-NNclus (Denœux, Kanjanatarakul, and Sriboonchitta 2015) is a decision-directed clustering algorithm in which the base classifier is the evidential K nearest neighbors (E-KNN)

rule (Denœux 1995). Given a labeled training set $\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $y_i \in \Omega$ is the class label of instance i, the E-KNN rule classifies a new instance x using the subset $N_K(x) \subset \mathcal{T}$ of its K nearest neighbors in $\mathcal{T}$. For each neighbor $x_i \in N_K(x)$ a mass function m_i on Ω is computed as

$$\begin{aligned} m_i(\{\omega_k\}) &= \varphi(\|x - x_i\|) I(y_i = \omega_k), \quad k = 1, \ldots, c \\ m_i(\Omega) &= 1 - \varphi(\|x - x_i\|), \end{aligned}$$

where $I(\cdot)$ is the indicator function and $\varphi(\cdot)$ is a decreasing mapping from $[0, +\infty)$ to $[0, 1]$. Typically, $\varphi(d) = \exp(-\gamma d^b)$, where γ and b are positive parameters. The K mass functions m_i corresponding to the K nearest neighbors are then combined using the conjunctive sum operation (3) and renormalized, an operation called *Dempster's rule* (Shafer 1976).

The EK-NNclus algorithm uses the above EK-NN rule for clustering. The algorithm is initialized by assigning each object a random label. If the dataset is not too large, we can initially assume that there are as many clusters as objects and each cluster contains exactly one object. Objects are then considered in random order and classified from their K nearest neighbors using the EK-NN rule. The algorithm stops when the class labels have not changed during the last iteration through the whole training set. By noticing the similarity with Hopfield neural networks (Hopfield 1982), Denœux *et al.* (2015) showed that this algorithm converges in a finite number of iterations. After convergence, we can compute a combined mass function for each object, which gives us a normalized credal partition. As mass functions in this credal partition are obtained by combining K mass functions using Dempster's rule, they are often very specific, with most of the mass concentrated on singletons. However, outlier are characterized by mass functions with a large mass on Ω. We also note that this method does not require the user to specify the number of clusters: starting with an arbitrarily large number of clusters, the algorithm usually converges to a meaningful partition.

In **evclust**, the EK-NNclus algorithm is implemented in function `EkNNclus()`

```
EkNNclus(x, D, K, y0, ntrials=1, q=0.5, b=1, disp=TRUE, tr=FALSE)
```

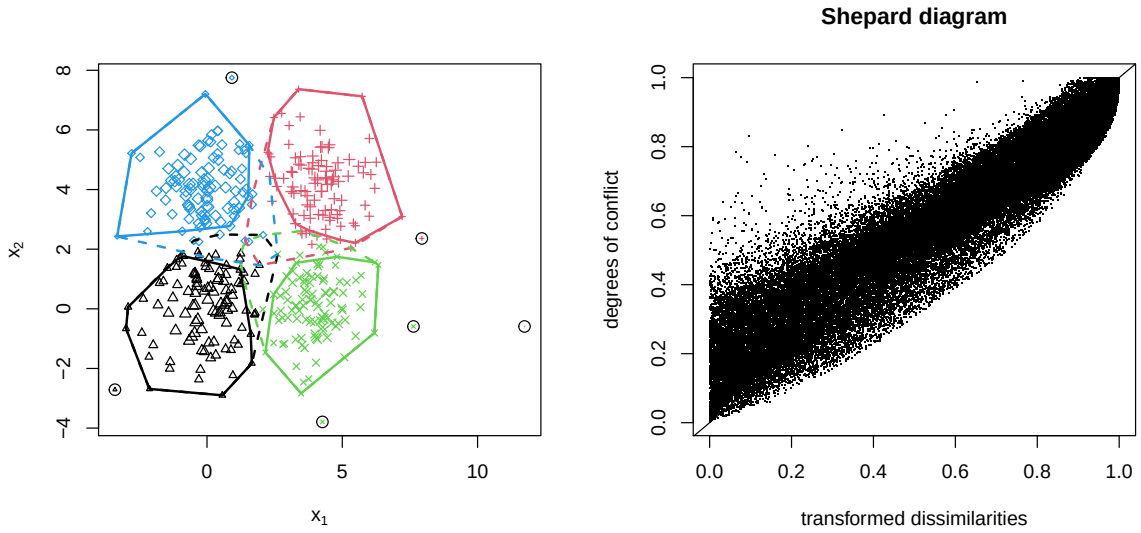
where `x` is the $n \times p$ object-attributes data matrix, `D` is distance matrix (required only if `x` is not provided), `K` is the number of neighbors, `y0` is the n-vector of initial labels, `q` is a number in $(0, 1)$ such that parameter γ is set to the inverse of the q-quantile of distances between any attribute vector and its K nearest neighbors. Arguments `disp` and `tr` control, respectively, the display and storage of intermediate results. The output is a '`credpart`' object encoding the final credal partition.

4. Illustrations

In this section, we demonstrate the use of the main functions in the **evclus** package through the analysis of some datasets.

4.1. `fourclass` dataset

The `fourclass` dataset is a synthetic dataset with two attributes and four classes of 100 points each, generated from a multivariate t distribution with five degrees of freedom:



(a) (b)

Figure 2: Plot of the `fourclass` dataset with the credal partition generated by `kevclus()` (a) and plot of the degrees of conflict κ_{ij} vs. the transformed dissimilarities $\varphi(\delta_{ij})$ (b).

```
R> data(fourclass, package="evclust")
R> x<-fourclass[,1:2]
R> y<-fourclass[,3]
```

EVCLUS. Calling function `kevclus` with the argument `c=4` creates an object `clus.evclus` of class '`credpart`':

```
R> clus.evclus<-kevclus(x, c=4, disp=FALSE)
```

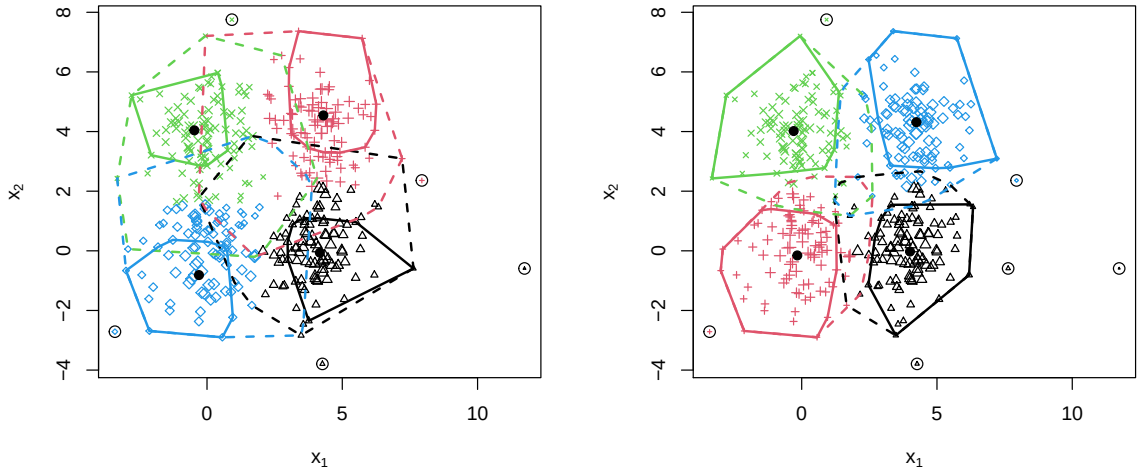
Figure 2 contains the graphs generated by `S3` method `plot()` applied to the credal partition `clus.evclus`:

```
plot(clus.evclus, x, plot_Shepard=TRUE)
```

Figure 2a is a scatterplot of the data together with a view of the credal partition. The solid and broken lines are the convex hulls of, respectively, the lower and upper approximations of the four clusters. The symbol and color of each point indicate the maximum-plausibility cluster, and the size of the symbols is proportional to the plausibility of that cluster. Outliers are indicated by circles. Figure 2b is a "Shepard diagram" showing the degrees of conflict κ_{ij} (vertical axis) vs. the transformed dissimilarities $\varphi(\delta_{ij})$ (horizontal axis). This diagram reflects, in some way, the quality of the credal partition.

ECM The ECM algorithm can be applied to the same data by calling function `ecm()` as:

```
R> clus.ecm<-ecm(x, c=4, type = "pairs", delta=3.5, disp=FALSE, Omega=FALSE)
```



(a) (b)

Figure 3: Plots of the `fourclass` dataset with the credal partitions generated by `ecm()` with $\alpha = 1$ (a) and $\alpha = 3$ (b). The prototypes are displayed as filled circles.

We note that, in this call to function `ecm()`, we have restricted the focal sets to the empty set, singletons, and pairs (thus excluding triplets and Ω). The `delta` argument determines the proportion of outliers. The resulting credal partition is shown in Figure 3a. We notice that the obtained credal partition is more imprecise than that generated by EVCLUS, in the sense that more points are located in the boundary areas of the four clusters (defined as the set difference between the upper and lower approximations). However, we can obtain a more precise credal partition by increasing the value of parameter α in cost function (16) from the default value $\alpha = 1$ to $\alpha = 3$, as

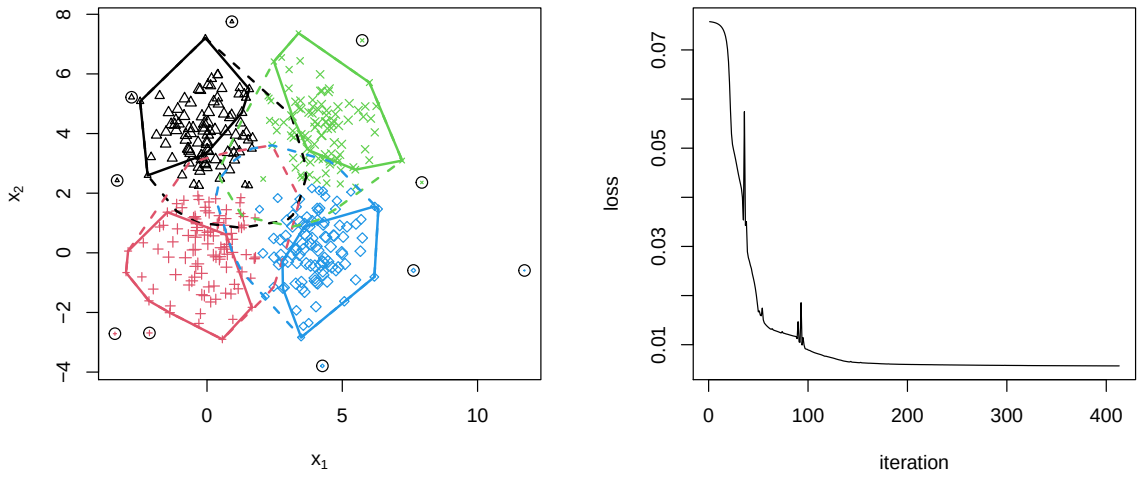
```
R> clus.ecm1<-ecm(x, c=4, type = "pairs", delta=3.5, a=3, disp=FALSE,
+    Omega=FALSE)
```

By comparing Figures 3b and 2a, we can see that the credal partition generated by ECM is then very similar to that computed by EVCLUS.

NN-EVCLUS. Let us now consider the application of NN-EVCLUS to the same data. We may first train a one-class SVM using function `ksvm()` of package **kernlab**:

```
R> library(kernlab)
R> svmfit<-ksvm(~., data=data.frame(x), type="one-svc", kernel="rbfdot",
+    nu=0.2, kpar=list(sigma=0.2))
R> fhat<-predict(svmfit, newdata=x, type="decision")
```

Vector `fhat` containing the one-class SVM output is then provided as input to function `nnevclus()` to train a neural network with one hidden layer of 20 units:



(a) (b)

Figure 4: Plots of the `fourclass` dataset with the credal partitions generated by `nnevclus()` (a) and evolution of the loss as function of the number of iterations (b).

```
R> clus.nn<-nnevclus(x, k=100, c=4, n_H=20, type='pairs', fhat=fhat,
+    options=c(0,1000,1e-4,10), tr=TRUE)
```

By setting the argument `k` to 100, we only use the distances from each input vector to 100 randomly selected input vectors, which speeds up the calculations. The obtained credal partition displayed in Figure 4a is similar to those generated by EVCLUS and ECM. Figure 4b shows the decrease of the loss (20) as a function of the number of iterations.

An advantage of NN-EVCLUS is the possibility to predict the cluster-membership of new data. The `S3` method `predict()` for '`credpart`' objects takes as input a credal partition of class '`credpart`' generated by NN-EVCLUS or ECM as well as a new data matrix, and outputs a credal partition for the new data. The following code computes an array `PL` containing the plausibilities of the four clusters at the nodes of a 50×50 grid in the two-dimensional attribute space:

```
R> nx <- 50
R> xmin <- apply(x,2,min) - 2
R> xmax <- apply(x,2,max) + 2
R> xx <- seq(xmin[1], xmax[1], (xmax[1] - xmin[1]) / (nx-1))
R> yy <- seq(xmin[2], xmax[2], (xmax[2] - xmin[2]) / (nx-1))
R> PL <- array(0, c(nx, nx, 4))
R> for(i in 1:nx){
+    X1 <- matrix(c(rep(xx[i],nx),yy),nx,2)
+    x1 <- data.frame(X1)
+    names(x1) <- c("x1","x2")
+    fhat <- predict(svmfit, newdata=x1, type="decision")
```

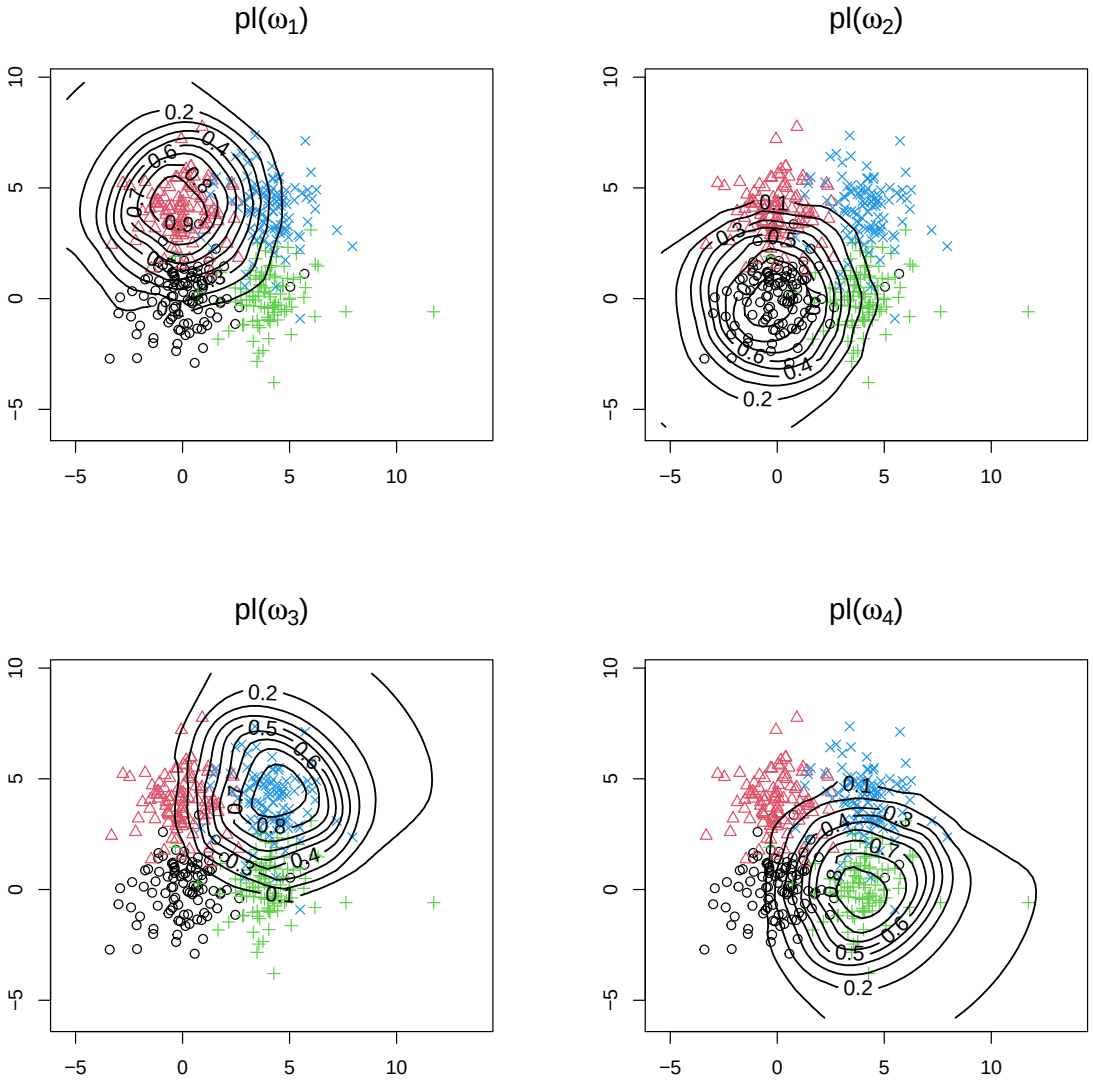


Figure 5: Contour plots of the plausibilities of the four clusters computed from the `fourclass` dataset using the NN-EVCLUS algorithm.

```
+    clus.t <- predict(clus.nn, X1, fhat)
+    PL[i,,] <- clus.t$pl
+  }
```

The corresponding contour plots are shown in Figure 5.

Comparison of credal partitions. The quality of a credal partition can be assessed by its consistency with the true partition (when it is known) and by its nonspecificity. Function `create_hard_credpart()` transforms a vector of class labels encoding a hard partition into a '`credpart`' object, and function `pairwise_mass()` computes the relational representation

of a credal partition (defined in Section 2.2). Function `nonspecificity()` computes the average nonspecifity of pairwise mass functions m_{ij} in the relational representation of a credal partition. The following piece of code computes the consistency degrees and nonspecificities of the credal partitions generated by EVCLUS, ECM and NN-EVCLUS:

```
R> Ptrue <- pairwise_mass(create_hard_credpart(y))
R> P_evclus <- pairwise_mass(clus.evclus)
R> RI_evclus <- credal_RI(Ptrue,P_evclus,type="c")
R> NS_evclus <- nonspecificity(P_evclus)
R> P_ecm <- pairwise_mass(clus.ecm)
R> RI_ecm <- credal_RI(Ptrue,P_ecm,type="c")
R> NS_ecm <- nonspecificity(P_ecm)
R> P_nn <- pairwise_mass(clus.nn)
R> RI_nn <- credal_RI(Ptrue,P_nn,type="c")
R> NS_nn <- nonspecificity(P_nn)
R> print(c(RI_evclus, RI_ecm, RI_nn))

[1] 0.8015894 0.7948521 0.8233088

R> print(c(NS_evclus, NS_ecm, NS_nn))

[1] 0.2690493 0.4504993 0.4241045
```

We can see that the three credal partitions have similar consistency degrees. The credal partition generated by EVCLUS is more specific because, by default, masses are assigned to the empty set, singletons and Ω (and to no strict subset of Ω of cardinality greater than 1).

4.2. `S2` dataset

With c clusters, the number of focal sets of the mass functions in a credal partition can be as high as 2^c, which is not tractable for large c. If we allow masses to be assigned to pairs of clusters, as suggested by Denœux and Masson (2004) and Masson and Denoeux (2008), the number of focal sets becomes proportional to c^2, which is manageable for moderate values of c (say, until 10), but is still impractical when c is very large. It is clear, however, that only the pairs of overlapping clusters will be assigned some mass during the learning process.

To determine which pairs of clusters can potentially become focal sets, a two-step approach was proposed by Denœux *et al.* (2016):

1. In the first step, a credal clustering algorithm is run with focal sets of cardinalities 0, 1 and c. A credal partition $\mathcal{M}_0$ is obtained. The similarity between each pair of clusters (ω_j, ω_ℓ) is measured by

$$S(j,\ell) = \sum_{i=1}^{n} pl_{ij} pl_{i\ell}, \tag{23}$$

   where pl_{ij} and $pl_{i\ell}$ are the normalized plausibilities that object i belongs, respectively, to clusters j and ℓ. We then determine the set $\mathcal{P}_k$ of pairs $\{\omega_j, \omega_\ell\}$ that are mutual k nearest neighbors, according to similarity measure S.

2. In the second step, the credal clustering algorithm is initialized with the previous credal partition $\mathcal{M}_0$, but adding as focal sets the pairs in $\mathcal{P}_k$; it is run again until convergence.

To illustrate this methodology, we consider the `S2` dataset (Fränti and Sieranoja 2018) composed of $n = 5000$ two-dimensional vectors grouped in 15 Gaussian clusters:

```
R> data(s2, package="evclust")
R> n <- nrow(s2)
```

We first use the EK-NNclus algorithm with $K = 200$ neighbors, starting from five different initial random hard partitions in 500 clusters:

```
R> clus.eknnclus <- EkNNclus(s2, K=200, y0=sample(500,n,replace=TRUE),
+    ntrials=5, q=0.9, disp=FALSE)
R> print(clus.eknnclus$N)
```

```
[1] 5.810961e-13
```

The resulting partition is close to a hard partition, with a mean nonspecificity close to zero. It is plotted in Figure 6a. We can see that the algorithm has correctly identified the 15 clusters. This credal partition can be used to initialize the EVCLUS algorithm. We first compute the Euclidean distances between each input vector and `k=100` random input vectors:

```
R> Dist <- createD(s2,k=100)
```

and use these distances as inputs to function `kevclus()`, with focal sets restricted to the empty set, singletons and Ω:

```
R> clus.evclus1 <- kevclus(D=Dist$D, c=15, J=Dist$J, type='simple',
+    d0=quantile(Dist$D,0.25), m0=clus.eknnclus$mass, maxit=100, epsi=1e-4,
+    disp=FALSE)
```

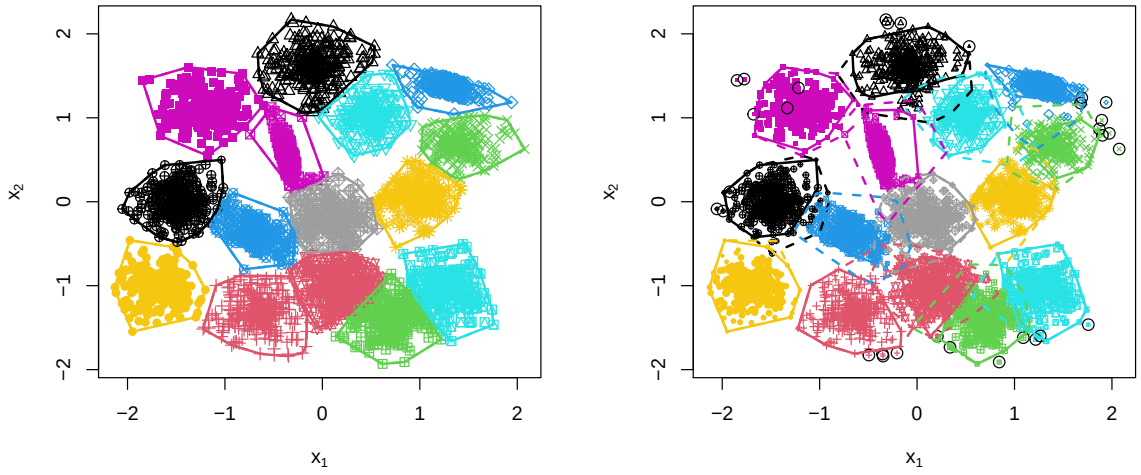
Function `createPairs()` then finds the mutual `k` nearest neighbor pairs of clusters; here, setting `k=2`, we get 11 cluster pairs:

```
R> P <- createPairs(clus.evclus1, k=2)
R> print(t(P$pairs))
```

```
    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
row    3    1    4    3    2   10    8    9   11     6     9
col    4    5    5    7   10   11   12   12   13    14    15
```

Finally, we run the EVCLUS algorithm a second time, starting with the previous credal partition `P$m0` and adding the pairs of clusters `P$pairs` found in the previous step as focal sets:

```
R> clus.evclus2 <- kevclus(D=Dist$D, c=15, J=Dist$J, type='pairs',
+    pairs=P$pairs, d0=quantile(Dist$D,0.25), m0=P$m0, maxit=100, epsi=1e-4,
+    disp=FALSE)
```



(a) (b)

Figure 6: Plots of the `S2` dataset with the credal partitions generated by `EkNNclus()` (a) and `kevclus()` with selected pairs of clusters as focal sets (b).

The final credal partition is displayed in Figure 6b. It is clearly more informative than the initial partition found by EK-NNclus, as overlapping regions between clusters are now clearly identified.

4.3. `Iris` dataset

To illustrate the Bootclus algorithm, we consider the famous `Iris` dataset, which contains the measurements in centimeters of sepal and petal length and width for 50 flowers from each of three species of iris: *Iris setosa*, *versicolor*, and *virginica*:

```
R> data("iris", package="datasets")
R> x<-iris[,1:4]
R> Y<-as.numeric(iris[,5])
R> n<-nrow(x)
```

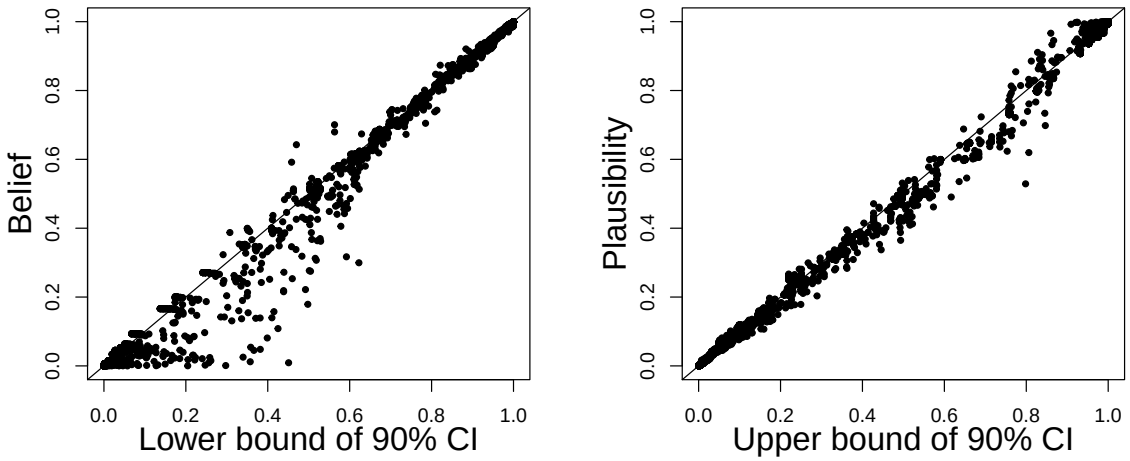
The Bootclus algorithm can be applied to these data by running function `bootclus()`:

```
R> fit <- bootclus(x, param=list(G = 3))
```

where the argument `G = 3` is passed to function `Mclust()` from package **mclust** called internally by `bootclus()`. This function searches for the best GMM out of 14 models with different contraints on the cluster volumes, shapes and orientations. The selected model can be displayed as

```
R> print(fit$clus$modelName)
```

```
[1] "VEV"
```



(a) (b)

Figure 7: Approximation of confidence intervals by belief-plausibility intervals for the Iris data. (a) Lower bound P_{ij}^l of the 90% confidence interval on pairwise probabilties P_{ij} (x-axis) vs. belief degree $Bel_{ij}(\{s_{ij}\})$ (y-axis); (b) Upper bound P_{ij}^u of the 90% confidence interval on P_{ij} (x-axis) vs. plausibility degree $Pl_{ij}(\{s_{ij}\})$ (y-axis).

It is model "VEV" corresponding to ellipsoidal clusters with equal shape. The default confidence level of the bootstrap percentile confidence intervals is $1 - \alpha = 0.9$ (See Section 3.3). By default, the focal sets are the singletons and the pairs; the computed mass functions thus have $f = 6$ focal sets in this case. Figure 7 displays the pairwise belief and plausibility degrees vs. the lower and upper bounds of the 90% bootstrap percentile confidence intervals. We can see that the confidence bounds are quite well approximated by the belief-plausibility intervals. Some belief values are smaller than the lower bounds of the confidence intervals (Figure 7a), which suggests that the coverage probability of the belief-plausibility intervals is actually larger than the 90% specified level.

Figure 8 shows the obtained credal partition encoded in `fit$Clus`. We can see that the *setosa* group, which is well separated from the other two, has a precise representation (for that cluster, the lower and upper approximations are equal). In contrast, the other two groups are overlapping, and assigning some instances to only one group would be highly uncertain. Some instances are thus assigned to the meta-cluster formed by the union of the *versicolor* and *virginica* groups, and belong to the upper approximations of these two clusters.

The following confusion matrix shows that five objects from the *versicolor* group are misclassified into the *virginica* group in the initial partition created by `mclust()`:

```
R> table(iris[,5],fit$clus$classification)

               1  2  3
  setosa      50  0  0
  versicolor   0 45  5
```

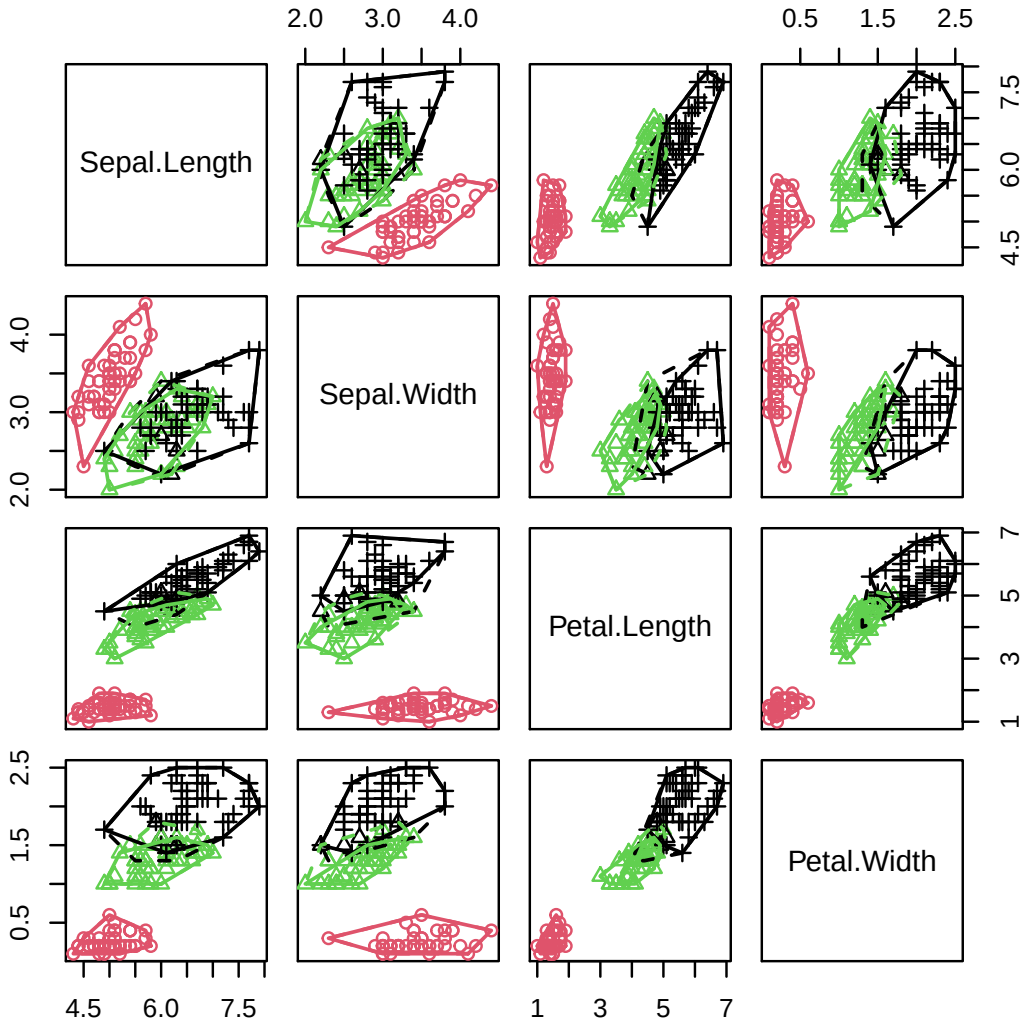


Figure 8: Plots of the `Iris` dataset with the credal partitions generated by `bootclus()`. The true groups are represented by different symbols (o: setosa; triangle: versicolor; +: virginica), and the maximum-plausibility groups are represented by different colors. The solid and broken lines represent, respectively, the convex hulls of the lower and upper approximation of each cluster.

```
virginica    0  0 50
```

In contrast, after summarizing the credal partition into a rough partition by approximating each mass function by a set using (8), nine instances from the *versicolor* class and one instance from the *virginica* class are assigned to the meta-cluster formed by the union of the *versicolor* and *virginica* groups, and ony one instance from the *versicolor* is incorrectly assigned to the *virginica* group:

```
R> rough_partition<-apply(fit$Clus$Y,1,paste,collapse="")
R> table(iris[,5],rough_partition)
```

```
            rough_partition
             001 010 100 101
  setosa       0  50   0   0
  versicolor  40   0   1   9
  virginica    0   0  49   1
```

(In the above table, sets are represented in binary notation, i.e., '100' represents $\{\omega_1\}$, '101' represents $\{\omega_1, \omega_3\}$, etc.).

4.4. Bananas dataset

Discovering clusters with complex shapes is a challenging task for clustering algorithms. For instance, function `bananas()` generates two banana-shaped clusters:

```
R> data<-bananas(300)
R> x<-data$x
R> y<-data$y
```

Clustering algorithms such as ECM or EVCLUS do not perform well on these data, as they are implicitly based on the definition of a cluster as a set of objects similar to each other. To detect the two clusters, we need additional information, such as must-link and cannot link constraints. Given the true class labels, function `create_MLCL()` generates such constraints by randomly picking pairs of objects from the same class and from different classes with equal probabilities. Here, we use it to generate 400 constraints:

```
R> const<-create_MLCL(y,400)
```

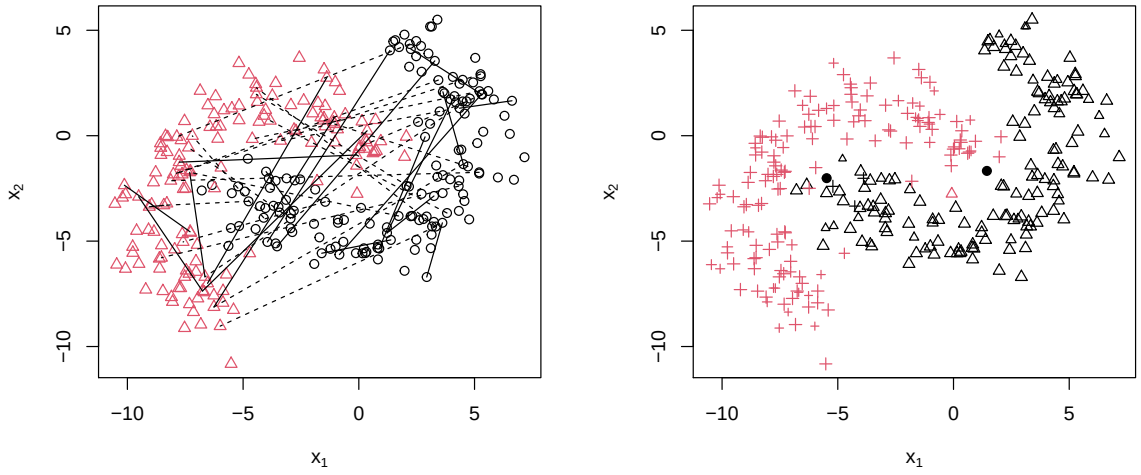
The output `const` is a list with two components: a matrix `ML` of must-link constraints, and a matrix `CL` of cannot-link constraints, both with two columns and as many rows as constraints. Note that these 400 pairwise constraints (some of which represented in Figure 9a) represent less than 1% of the 44850 object pairs.

There are two main complementary approaches to exploit such additional information. The first one is to account for the constraints in the cost function; this is the approach used in the CECM and CEVCLUS algorithms. For instance, we can run the `cecm()` function as

```
R> clus.cecm <- cecm(x, c=2, ML=const$ML, CL=const$CL, ntrials=5, xi=0.9,
+    distance=1, disp=FALSE)
```

where `xi` is the hyperparameter in (18), and the algorithm is run `ntrials=5` times. The resulting credal partition is plotted in Figure 9b. The adjusted Rand index (ARI) between the maximum-plausibility hard partition and the true partition can be computed using function `adjustedRandIndex()` of package **mclust** as

```
R> library("mclust")
R> print(adjustedRandIndex(clus.cecm$y.pl, y))
```



(a) (b)

Figure 9: (a): `Bananas` dataset with 20 must-link constraints (solid lines) and 20 cannot-link constraints (broken lines) out of the 400 constraints generated by function `create_MLCL()` for the `bananas` dataset; (b): credal partition generated by `cecm()` using the 400 pairwise constraints.

```
[1] 0.9472012
```

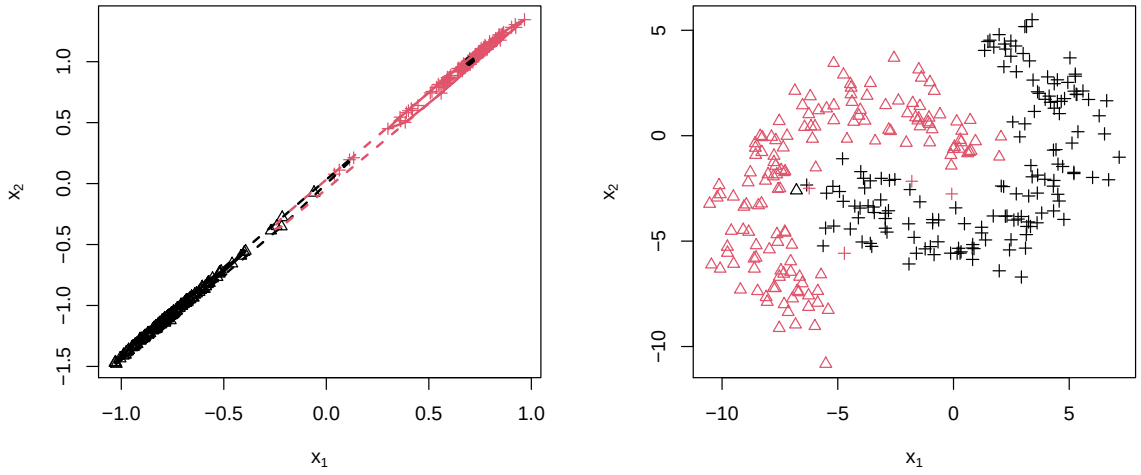
The other approach is to map the data to a new feature space in which pairs of objects known to belong to the same cluster are close to each other, while pairs of objects known to belong to different clusters are far apart. One such method is the kernel pairwise constrained component analysis (KPCCA) algorithm proposed by Mignon and Jurie (2012) and implemented in **evclust** as function `kpca()`; this function takes as input a kernel matrix, which can be generated by function `kernelMatrix()` of package **kernlab**:

```
R> library(kernlab)
R> rbf <- rbfdot(sigma = 0.2)
R> K <- kernelMatrix(rbf, x)
R> res.kpcca <- kpcca(K, d1=2, ML=const$ML, CL=const$CL, epsi=1e-3,
+    disp=FALSE)
```

where `K` is a kernel matrix, `d1` is the number of extracted features, and the algorithm stops when the rate of change of the cost function is less than `epsi`. Function `kpca()` returns a list with three components: the new feature matrix `z` of size $n \times d_1$, the projection matrix `A` of size $d_1 \times n$, and the Euclidean distance matrix `D` in the new feature space.

Having extracted `d_1=2` features, we can run function `ecm()` in the new feature space:

```
R> clus.ecm<-ecm(res.kpcca$z,c=2,disp=FALSE)
```



(a) (b)

Figure 10: Credal partition generated by `ecm()` for the `bananas` dataset in the transformed feature space, with 400 randomly generated pairwise contraints. The partition is shown in the transformed (a) and original (b) feature spaces.

The generated partition is diplayed in the transformed and original feature spaces, respectively, in Figures 10a and 10b. The result is similar to that obtained with the previous method according to the ARI of the maximum-plausibility partition:

```
R> print(adjustedRandIndex(clus.ecm$y.pl, y))
```

```
[1] 0.9342245
```

Combining the two methods further improves result in terms of ARI:

```
R> clus.cecm1 <- cecm(res.kpcca$z, c=2, ML=const$ML, CL=const$CL, ntrials=5,
+    xi=0.9, distance=1, disp=FALSE)
R> print(adjustedRandIndex(clus.cecm1$y.pl, y))
```

```
[1] 0.9866665
```

5. Conclusions

Evidential clustering is a new approach to clustering in which uncertainty about cluster membership is described by Dempster-Shafer mass functions. Evidential clustering algorithms compute credal partitions defined as tuples of mass functions over the set Ω of clusters. For each object i, the mass $m_i(A)$ assigned to nonempty set A of clusters reflects ambiguity in the assignment of the object to clusters in A, while the mass $m_i(\emptyset)$ assigned to the empty

set reflects the possibility that the object might not belong to any of the clusters and allows us to detect outliers. This representation is very general and encompasses other approaches such a fuzzy, possibilistic and rough clustering as special cases.

The **evclust** package described in this paper implements a set of efficient evidential clustering algorithms, as well as functions to display, evaluate and exploit credal partitions. These algorithms are based on different principles and make it possible to address a variety of clustering problems, including difficult ones such as clustering nonmetric dissimilarity data, discovering complex-shaped clusters, and constrained clustering. For lack of space, we did not address the important problem of determining the number of clusters. Whereas the EK-NNclus algorithm does not require to specify that number, other algorithms do. In the case of the Bootclus algorithm relying on model-based clustering, analytical criteria such as AIC can be used. For the other algorithms, the nonspecifity of the credal partition can be used to select a good credal partition (Masson and Denoeux 2008). Another approach is to use generic indices such as implemented in the **NbClust** package (Charrad *et al.* 2014).

Whereas **evclust** is dedicated to clustering, a companion package **evclass** focusses on evidential supervised classification (Denœux 2017). The author hopes that the availability of these packages will contribute to a more widespread dissemination of tools based on belief functions in data analysis and machine learning, and draw the attention of an increasing number of data scientists to this new research direction.

References

Antoine V, Quost B, Masson MH, Denoeux T (2012). "CECM: Constrained evidential c-means algorithm." *Computational Statistics & Data Analysis*, **56**(4), 894–914.

Antoine V, Quost B, Masson MH, Denoeux T (2014). "CEVCLUS: evidential clustering with instance-level constraints for relational data." *Soft Computing*, **18**(7), 1321–1335.

Azzalini A, Menardi G (2014). "Clustering via Nonparametric Density Estimation: The R Package pdfCluster." *Journal of Statistical Software, Articles*, **57**(11), 1–26. ISSN 1548-7660. doi:10.18637/jss.v057.i11. URL https://www.jstatsoft.org/v057/i11.

Bezdek J (1981). *Pattern Recognition with fuzzy objective function algorithm*. Plenum Press, New-York.

Borg I, Groenen P (1997). *Modern multidimensional scaling*. Springer, New-York.

Charrad M, Ghazzali N, Boiteau V, Niknafs A (2014). "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set." *Journal of Statistical Software, Articles*, **61**(6), 1–36. ISSN 1548-7660. doi:10.18637/jss.v061.i06. URL https://www.jstatsoft.org/v061/i06.

Dempster AP (1967). "Upper and lower probabilities induced by a multivalued mapping." *Annals of Mathematical Statistics*, **38**, 325–339.

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.

Denœux T (2001). "Inner and outer approximation of belief structures using a hierarchical clustering approach." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **9**(4), 437–460.

Denœux T (2017). *evclass: Evidential Distance-Based Classification*. R package version 1.1.1, URL https://CRAN.R-project.org/package=evclass.

Denœux T (2019). "Decision-Making with Belief Functions: a Review." *International Journal of Approximate Reasoning*, **109**, 87–110.

Denœux T (2020a). "Calibrated model-based evidential clustering using bootstrapping." *Information Sciences*, **528**, 17–45.

Denœux T (2020b). "NN-EVCLUS: Neural Network-based Evidential Clustering." ArXiv preprint 2009.12795, URL https://arxiv.org/abs/2009.12795.

Denœux T, Dubois D, Prade H (2020). "Representations of Uncertainty in Artificial Intelligence: Beyond Probability and Possibility." In P Marquis, O Papini, H Prade (eds.), *A Guided Tour of Artificial Intelligence Research*, volume 1, chapter 4, pp. 119–150. Springer Verlag.

Denoeux T, Kanjanatarakul O (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)*, volume AISC 456 of *Advances in Intelligent and Soft Computing*, pp. 157–164. Springer-Verlag, Rome, Italy.

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.

Denœux T, Li S, Sriboonchitta S (2018). "Evaluating and Comparing Soft Partitions: an Approach Based on Dempster-Shafer Theory." *IEEE Transactions on Fuzzy Systems*, **26**(3), 1231–1244.

Denœux T, Masson MH (2004). "EVCLUS: Evidential Clustering of Proximity Data." *IEEE Trans. on Systems, Man and Cybernetics B*, **34**(1), 95–109.

Denœux T, Sriboonchitta S, Kanjanatarakul O (2016). "Evidential clustering of large dissimilarity data." *Knowledge-based Systems*, **106**, 179–195.

Ferraro MB, Giordani P, Serafini A (2019). "fclust: An R Package for Fuzzy Clustering." *The R Journal*, **11**(1), 198–210. doi:10.32614/RJ-2019-017. URL https://doi.org/10.32614/RJ-2019-017.

Fränti P, Sieranoja S (2018). "K-means properties on six clustering benchmark datasets." URL http://cs.uef.fi/sipu/datasets/.

Goodfellow I, Bengio Y, Courville A (2016). *Deep Learning*. MIT Press. URL http://www.deeplearningbook.org.

Gustafson EE, Kessel W (1979). "Fuzzy clustering with a fuzzy covariance matrix." In *Proc. IEEE Conf. on Decision and Control*, pp. 761–766. IEEE, Piscataway, NJ.

Hopfield JJ (1982). "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences*, **79**, 2554–2558.

Jain AK, Dubes RC (1988). *Algorithms for clustering data*. Prentice-Hall, Englewood Cliffs, NJ.

Jousselme AL, Grenier D, Bossé E (2001). "A new distance between two bodies of evidence." *Information Fusion*, **2**(2), 91–101.

Jousselme AL, Maupin P (2012). "Distances in evidence theory: Comprehensive survey and generalizations." *International Journal of Approximate Reasoning*, **53**(2), 118–145.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "kernlab – An S4 Package for Kernel Methods in R." *Journal of Statistical Software*, **11**(9), 1–20. URL http://www.jstatsoft.org/v11/i09/.

Kaufman L, Rousseeuw PJ (1990). *Finding groups in data*. Wiley, New-York.

Klir GJ, Wierman MJ (1999). *Uncertainty-Based Information. Elements of Generalized Information Theory*. Springer-Verlag, New-York.

Kondo Y, Salibian-Barrera M, Zamar R (2016). "RSKC: An R Package for a Robust and Sparse K-Means Clustering Algorithm." *Journal of Statistical Software, Articles*, **72**(5), 1–26. ISSN 1548-7660. doi:10.18637/jss.v072.i05. URL https://www.jstatsoft.org/v072/i05.

Krishnapuram R, Keller J (1993). "A Possibilistic Approach to Clustering." *IEEE Trans. on Fuzzy Systems*, **1**, 98–111.

Li F, Li S, Denœux T (2018). "k-CEVCLUS: Constrained evidential clustering of large dissimilarity data." *Knowledge-Based Systems*, **142**, 29–44.

Lloyd S (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, **28**(2), 129–137. doi:10.1109/TIT.1982.1056489.

Masson MH, Denoeux T (2008). "ECM: An evidential version of the fuzzy c-means algorithm." *Pattern Recognition*, **41**(4), 1384–1397.

Masson MH, Denœux T (2009). "RECM: Relational Evidential c-means algorithm." *Pattern Recognition Letters*, **30**, 1015–1026.

McLachlan GJ, Basford KE (1988). *Mixture Models: inference and applications to clustering*. Marcel Dekker, New York.

Mignon A, Jurie F (2012). "PCCA: A new approach for distance learning from sparse pairwise constraints." In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2666–2672.

Peters G (2014). "Rough clustering utilizing the principle of indifference." *Information Sciences*, **277**, 358 – 374.

Peters G (2019). *SoftClustering: Soft Clustering Algorithms*. R package version 1.1902.2, URL https://CRAN.R-project.org/package=SoftClustering.

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.

Scrucca L, Fop M, Murphy TB, Raftery AE (2016). "mclust 5: clustering, classification and density estimation using Gaussian finite mixture models." *The R Journal*, **8**(1), 289–317. URL https://doi.org/10.32614/RJ-2016-021.

Shafer G (1976). *A mathematical theory of evidence*. Princeton University Press, Princeton, N.J.

Smets P, Kennes R (1994). "The Transferable Belief Model." *Artificial Intelligence*, **66**, 191–243.

Su Z, Denœux T (2019). "BPEC: Belief-Peaks Evidential Clustering." *IEEE Transactions on Fuzzy Systems*, **27**(1), 111–123.

Voorbraak F (1989). "A computationally efficient approximation of Dempster-Shafer theory." *Int. J. Man-Machine Studies*, **30**, 525–536.

Xu R, Wunsch D (2009). *Clustering*. Wiley-IEEE Press, Hoboken, New Jersey.

Yang MS, Wu KL (2006). "Unsupervised possibilistic clustering." *Pattern Recognition*, **39**(1), 5–21.

Zadeh LA (1978). "Fuzzy sets as a basis for a theory of possibility." *Fuzzy Sets and Systems*, **1**, 3–28.

Affiliation:

Thierry Denœux
Université de technologie de Compiègne
Compiègne, France *and*
Institut universitaire de France
Paris, France
E-mail: tdenoeux@utc.fr
URL: https://www.hds.utc.fr/~tdenoeux/