NN-EVCLUS: Neural network-based evidential clustering

Thierry Denoeux

▶ **To cite this version:**

Thierry Denoeux. NN-EVCLUS: Neural network-based evidential clustering. Information Sciences, 2021, 572, pp.297-330. 10.1016/j.ins.2021.05.011 . hal-03235749

HAL Id: hal-03235749
https://hal.science/hal-03235749

Submitted on 26 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NN-EVCLUS: Neural Network-based Evidential Clustering

Thierry Denœux[a,b,c]

[a] *Université de technologie de Compiègne, CNRS, Heudiasyc, Compiègne, France*
[b] *Institut universitaire de France, Paris, France*
[c] *Shanghai University, UTSEUS, Shanghai, China*

Abstract

Evidential clustering is an approach to clustering based on the use of Dempster-Shafer mass functions to represent cluster-membership uncertainty. In this paper, we introduce a neural-network based evidential clustering algorithm, called NN-EVCLUS, which learns a mapping from attribute vectors to mass functions, in such a way that more similar inputs are mapped to output mass functions with a lower degree of conflict. The neural network can be paired with a one-class support vector machine to make it robust to outliers and capable of detecting previously unseen clusters when applied to new data. The network is trained to minimize the discrepancy between dissimilarities and degrees of conflict for all or some object pairs. Additional terms can be added to the loss function to account for pairwise constraints or labeled data, which can also be used to adapt the metric. Comparative experiments show the superiority of NN-EVCLUS over state-of-the-art evidential clustering algorithms for a range of unsupervised and constrained clustering tasks involving both attribute and dissimilarity data.

Keywords: Dempster-Shafer theory, evidence theory, belief functions, unsupervised learning, semi-supervised learning, constrained clustering

1. Introduction

One of the most important tasks in machine learning and exploratory data analysis is finding groups of similar objects in a dataset, in such a way that the dissimilarity between groups is maximized. This problem, referred to as *clustering*, has been addressed using a wide range of techniques and from a variety of perspectives (see, e.g. [25, 29, 55]). While the earlier methods such as the hard c-means algorithm do not consider group-membership uncertainty, quantifying this uncertainty has been a major issue in the last 40 years [42, 15]. Among the most widely-used formalisms, we can mention fuzzy sets [3, 16], possibility theory [31, 39], and rough sets [41, 17, 52]. These approaches are, to some extent, subsumed by a relatively new approach, referred to as *evidential clustering*, which is based on the Dempster-Shafer (DS) theory of belief functions [12, 36, 10]. Evidential clustering algorithms quantify clustering uncertainty using DS 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 the n objects is called an *evidential* (or *credal*) *partition*. An evidential partition boils down to a fuzzy partition when the focal sets are

Email address: Thierry.Denoeux@utc.fr (Thierry Denœux)

singletons, and it is equivalent to a rough partition when each mass function has a single focal set [10].

Among the earliest evidential clustering procedures are the evidential c-means (ECM) algorithm [36] and its relational version [37]. The ECM algorithm maximizes an objective function based on distances to prototypes associated not only to individual clusters but also to sets of clusters or"meta-clusters". The prototype representing a meta-cluster is the barycenter of the prototypes representing the clusters it contains. The method was shown to provide meaningful evidential partitions, but it can provide undesirable results when the prototype of a meta-cluster is close to that of an individual cluster. The Belief c-means [34] and Credal c-means (CCM) [35] algorithms are alternative procedures designed to address this problem. The Belief Peak Evidential Clustering (BPEC) method [49] combines ideas from density peak clustering [44, 56, 21] and ECM. The Median Evidential c-means (MECM) [62] is an evidential version of the median c-means for relational data, while the Evidential c-medoid (ECMdd) with either a single prototype per class or multiple weighted prototypes [61] are inspired by the c-medoids algorithm.

In this paper, we revisit an earlier approach to evidential clustering that is not based on prototypes, but that takes inspiration from multidimensional scaling (MDS) [4]. The EVCLUS algorithm [12] constructs an evidential partition in such a way that the degree of conflict between mass functions related to any two objects match the similarity between these two objects. The method has been shown to outperform most other algorithms for handling nonmetric dissimilarity data [12], but it can obviously also be applied to attribute data after a distance matrix has been computed. Whereas the initial algorithm was initially only applicable to small datasets of a few hundred objects, algorithmic improvements introduced in [13] have made it applicable to much larger datasets containing tens of thousands of objects.

Whereas the EVCLUS has good performances in clustering tasks, it has some limitations. First, as it directly constructs mass functions describing the cluster membership of each object as the solution of an optimization problem, it does not allow us to classify new objects, other than by including these objects in the dataset and solving the new optimization problem globally, which may be time-consuming. Also, as EVCLUS does not represent clusters by parametric models such as prototypes, the number of parameters grows linearly with the number of objects, which becomes problematic when the number of objects is very large (say, hundreds of thousands). Finally, a third limitation of EVCLUS is that it does not easily incorporate side information in the clustering process. Semi-supervised versions of EVCLUS using pairwise constraints have been proposed [2, 32], but the performances of these algorithms are limited because constraints imposed on some pairs of objects do not naturally propagate to neighboring objects, which is another consequence of the absence of a parametric model of clusters.

In this paper, we address the above limitations by proposing a new version of EVCLUS, called NN-EVCLUS, in which attributes are mapped to mass functions using a feedforward neural network. Continuing the analogy with MDS, this approach bears some resemblance with the SAMANN model for Sammon's mapping [27] or Webb's radial basis function network implementation of MDS [53]. It is also related to "Siamese" networks for distance learning [5, 58]. In NN-EVCLUS, the network weights are learnt by minimizing the discrepancy between degrees of conflict and dissimilarities over all pairs of objects, as in EVCLUS. However, the number of model parameters, equal to the number of weights in the network, is usually much less than that of EVCLUS. The model can be used to predict the class of new objects, and it can be trained from very large datasets by stochastic gradient descent. It can also easily incorporate side information in the form of labeled

data or pairwise constraints.

The rest of this paper is organized as follows. Background knowledge on DS theory and evidential clustering is first recalled in Section 2. The new model is then described in Section 3 and numerical experiments are reported in Section 4. Finally, conclusions are presented in Section 5.

2. Background

The purpose of this section is to make this paper self-contained. Necessary definitions and results related to DS mass functions will first be recalled in Section 2.1. The notion of evidential partition and its relation with other notions of hard and "soft" partition will then be exposed in Section 2.2. Finally the EVCLUS algorithm will be summarized in Section 2.3.

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 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. In DS theory [46, 9], Ω is the set of possible answers to some question (called the *frame of discernment*), and a mass function m describes a piece of evidence pertaining to that question. Each mass $m(A)$ represents a share of a unit mass of belief allocated to focal set A, and which cannot be allocated to any strict subset of A. 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*.

Just as a probability mass function induces a probability measure, a DS mass function induces two 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 how much subset A is supported by the available evidence, while $Bel(\Omega) - Pl(A) = Bel(\overline{A})$ is a measure of how much the complement $\overline{A}$ is supported, so that $Pl(A)$ can be seen as a measure of lack of support for $\overline{A}$. When m is consonant, the following equality holds for all subsets A and B of Ω:

$$Pl(A \cap B) = \max(Pl(A), Pl(B)).$$

Function Pl is, thus, a possibility measure [60]. 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, it is a possibility distribution.

Given two mass functions m_1 and m_2 on the same frame of discernment Ω, their conjunctive sum [48] is the following mass function:

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

for all $C \subseteq \Omega$. The *degree of conflict* between m_1 and m_2 is then defined as the mass assigned to the empty set,

$$\kappa = (m_1 \cap m_2)(\varnothing) = \sum_{A \cap B = \varnothing} m_1(A) m_2(B). \tag{3}$$

It ranges in the interval [0,1]. When m_1 and m_2 represent two independent pieces of evidence pertaining to *the same question*, κ can be interpreted as a *measure of conflict* between these two pieces of evidence [46]. 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, κ can be given a different interpretation as one minus the plausibility that the true answers to Q_1 and Q_2 are identical [12].

Example 1. *Consider a population of objects partitioned in three classes, and let* $\Omega = \{\omega_1, \omega_2, \omega_3\}$ *denote the set of classes. Assume that a sensor provides information about the class of three objects* o_1*,* o_2 *and* o_3 *as the following three mass functions on* Ω*:*

$$m_1(\{\omega_1\}) = 0.6, \quad m_1(\{\omega_1, \omega_2\}) = 0.3, \quad m_1(\Omega) = 0.1$$

$$m_2(\{\omega_1, \omega_2\}) = 0.5, \quad m_2(\{\omega_3\}) = 0.2, \quad m_2(\Omega) = 0.3$$

$$m_3(\{\omega_1\}) = 0.1, \quad m_3(\{\omega_2\}) = 0.1, \quad m_3(\{\omega_3\}) = 0.8$$

The degree of conflict between m_1 *and* m_2 *is*

$$\kappa_{12} = 0.6 \times 0.2 + 0.3 \times 0.2 = 0.18,$$

while the degree of conflict between m_1 *and* m_3 *is*

$$\kappa_{13} = 0.6 \times 0.1 + 0.6 \times 0.8 + 0.3 \times 0.8 = 0.06 + 0.54 = 0.78,$$

Consequently, the plausibility that objects o_1 *and* o_2 *belong to the same class is* $1 - \kappa_{12} = 0.82$*, whereas this plausibility is only 0.22 for objects* o_1 *and* o_3*.*

2.2. Evidential clustering

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

The notion of evidential partition encompasses several classical clustering structures [10]:

- When all mass functions m_i are certain, then $\mathcal{M}$ is equivalent to a hard partition; this case corresponds to full certainty about the group of each object.

- When mass functions are Bayesian, then $\mathcal{M}$ 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, the corresponding contour functions pl_i are possibility distributions and they define a possibilistic partition.

- When each mass function m_i is logical with focal set $A_i \subseteq \Omega$, $\mathcal{M}$ is equivalent to a rough partition [40]. The lower and upper approximations of cluster ω_k are then defined, respectively, as the set of objects that *surely* belong to group ω_k, and the set of objects that *possibly* belong to group ω_k [36]; they are formally given by

$$\omega_k^l := \{i \in \mathcal{O} \mid A_i = \{\omega_k\}\} \quad \text{and} \quad \omega_k^u := \{i \in \mathcal{O} \mid \omega_k \in A_i\}. \tag{4}$$

  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.

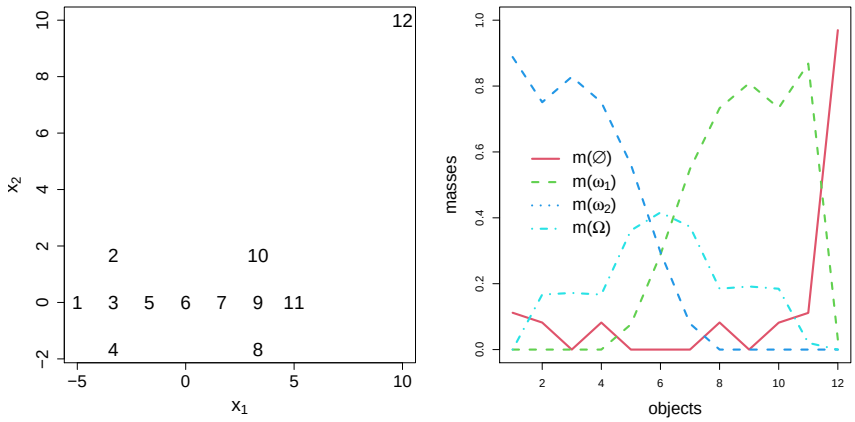
Example 2. *Consider the Butterfly data displayed in Figure 1a, consisting in 12 objects described by two attributes. Table 1 shows an evidential partition of these data obtained by EVCLUS, with $c = 2$ clusters. This evidential partition is represented graphically in Figure 1b. We can see that object 6, which is located between clusters ω_1 and ω_2, has the largest mass assigned to $\Omega = \{\omega_1, \omega_2\}$. In contrast, object 12, which is an outlier, has the largest mass assigned to the empty set. A convenient way to summarize an evidential partition is to approximate each mass function m_i by a logical mass function $\widehat{m}_i$ such that $\widehat{m}_i(A_i) = 1$ with $A_i = \arg\max_A m_i(A)$. We can then compute the lower and approximations of each cluster using (4). Here, we have $\omega_1^l = \{7,8,9,10,11\}$, $\omega_1^u = \{6,7,8,9,10,11\}$, $\omega_2^l = \{1,2,3,4,5\}$ and $\omega_2^u = \{1,2,3,4,5,6\}$. The objects that do not belong to the upper approximation of any cluster are outliers, which is the case for object 12 in this example.*

2.3. EVCLUS algorithm

Evidential clustering aims at generating an optimal evidential partition from attribute or dissimilarity data, based on some optimality criterion. The earliest such procedure is EVCLUS[1], which was introduced in [12] and later improved in [13]. EVCLUS transposes some ideas from MDS [4] to clustering. Let $\boldsymbol{D} = (\delta_{ij})$ be a symmetric $n \times n$ dissimilarity matrix, 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 distances such as the triangular inequality, i.e., we may have $\delta_{ik} > \delta_{ij} + \delta_{k,j}$ for some triple of objects (i,j,k).

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. As recalled in Section 2.1, the plausibility pl_{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

[1] EVCLUS is implemented with other evidential clustering algorithms in the R package `evclust` [8] available at `https://cran.r-project.org`.



(a) (b)

Figure 1: Butterfly dataset (a) and evidential partition with $c = 2$ obtained by ECM (b).

Table 1: Evidential partition of the Butterfly data. The largest mass for each object is printed in bold.

object #	$m(\varnothing)$	$m(\{\omega_1\})$	$m(\{\omega_2\})$	$m(\Omega)$
1	0.11	·	**0.89**	·
2	0.082	·	**0.75**	0.17
3	0.000	·	**0.83**	0.17
4	0.082	·	**0.75**	0.17
5	·	0.077	**0.56**	0.36
6	·	0.29	0.30	**0.42**
7	·	**0.55**	0.079	0.37
8	0.082	**0.73**	·	0.18
9	0.000	**0.81**	·	0.19
10	0.082	**0.73**	·	0.18
11	0.11	**0.87**	·	0.02
12	**0.97**	0.030	·	·

dissimilar objects are assigned highly conflicting mass functions. To derive an evidential partition $\mathcal{M} = (m_1, \ldots, m_n)$ from $\boldsymbol{D}$, we can thus minimize the discrepancy between the pairwise degrees of conflict and the dissimilarities, up to some increasing transformation. This problem bears some resemblance with the one addressed by MDS, which aims to represent objects in some Euclidean space, in such a way that the distances in that space match the observed dissimilarities [4]. Here, we want to find an evidential partition $\mathcal{M}$ that minimizes the following loss function,

$$\mathcal{L}(\mathcal{M}) = \frac{2}{n(n-1)} \sum_{i<j} \left(\kappa_{ij} - \varphi(\delta_{ij})\right)^2, \tag{5}$$

where φ is a fixed nondecreasing mapping from $[0, +\infty)$ to $[0, 1]$, such as

$$\varphi(\delta) = 1 - \exp(-\gamma\delta^2), \tag{6}$$

for some user-defined coefficient γ. In [13], we proposed to set $\gamma = -\log 0.05/\delta_0^2$, where δ_0 is some quantile of the dissimilarities δ_{ij}. This parametrization ensures that $\varphi(\delta) \geq 0.95$ whenever $\delta \geq \delta_0$, i.e., δ_0 is the threshold such that two objects o_i and o_j with dissimilarity larger than δ_0 have a plausibility at least 0.95 of belonging to different clusters. In [13], we recommended to set δ_0 to the 0.9-quantile as the default value, but this parameter sometimes needs to be fine-tuned to get optimal results.

Computing the loss function (5) requires to store the whole dissimilarity matrix, which may not be feasible for large datasets. In [13], it was shown that it is often sufficient to minimize the sum of squared errors for a subset of object pairs. We can thus replace (5) by

$$\mathcal{L}(\mathcal{M}; J) = \frac{1}{np} \sum_{i=1}^{n} \sum_{j \in J(i)} \left(\kappa_{ij} - \varphi(\delta_{ij})\right)^2, \tag{7}$$

where $J(i)$ is a randomly selected subset of $p < n - 1$ indices in $\{1, \ldots, i-1, i+1, \ldots, n\}$. The calculation of $\mathcal{L}(\mathcal{M}; J)$ thus requires to store only np dissimilarities δ_{ij} instead of $n(n-1)/2$ for the calculation of $\mathcal{L}(\mathcal{M})$ in (5). Experiments reported in [13] show that, for a number n of objects between 1000 and 10,000, optimal results are obtained with p in the range 100-500.

In [12], it was originally proposed to minimize a loss function similar to (5) using a gradient-based algorithm. A much more efficient cyclic coordinate descent procedure, called Iterative Row-wise Quadratic Programming (IRQP) [51] was proposed in [13]. This procedure consists in minimizing the loss with respect to one mass function m_i at a time while keeping the other mass functions fixed, which can be shown to be a linearly constrained least-squares problem. The IRQP algorithm together with the loss function (7) allow EVCLUS to cluster datasets containing up to a few tens of thousands of objects.

Example 3. *The* fourclass *dataset*[2] *is composed of 400 two-dimensional vectors generated from four two-dimensional Student distributions. Figure 2 displays the lower and upper approximations of each of the four clusters computed from an evidential partition obtained by EVCLUS, with loss function* (7) *and* $p = 100$*. The focal sets were restricted to subsets of cardinality less than or equal to two, and* Ω*. As shown in Figure 3a, the algorithm converges in a few dozens of iterations. Figure 3b shows the Shepard diagram, which displays the degrees of conflict* κ_{ij} *versus the transformed dissimilarities* $\varphi(d_{ij})$*.*

[2]This dataset is part of the R package `evclust`.

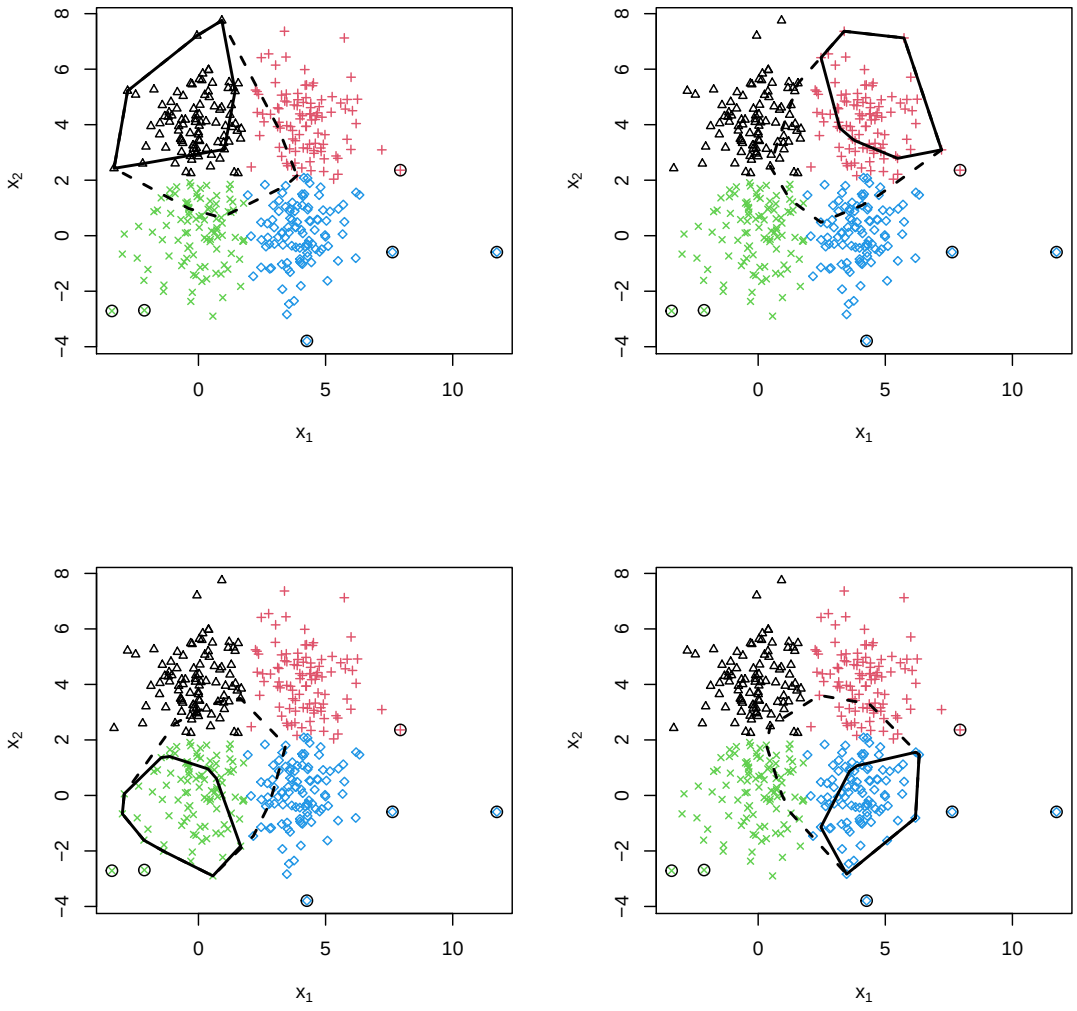
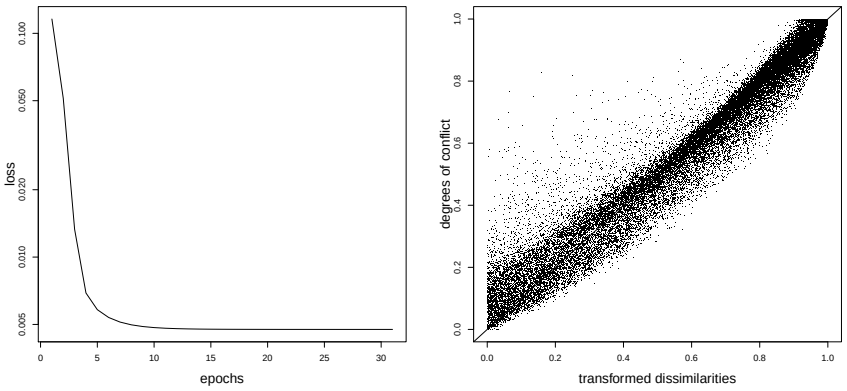


Figure 2: Lower and upper approximations of the four clusters found by EVCLUS for the fourclass dataset. The true classes are displayed with different colors. The identified clusters are plotted with different symbols. The convex hulls of the cluster lower and upper approximations are displayed using solid and interrupted lines, respectively. The outliers are indicated by circles.



(a) (b)

Figure 3: EVCLUS algorithm applied to the fourclass dataset: (a): loss vs. number of epochs (iterations thorugh the whole learning set); (b): plot of the degrees of conflict κ_{ij} vs. the transformed dissimilarities δ^*_{ij}.

3. NN-EVCLUS

As opposed to prototype-based evidential clustering algorithms such as ECM [36] or CCM [35], the EVCLUS algorithm summarized in Section 2.3 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. In this section, we describe a neural network version of EVCLUS that addresses these issues. This new model will also be shown to outperform EVCLUS and ECM in semi-supervised clustering tasks. The model will first be introduced in Section 3.1, and learning algorithms will be described in Section 3.2. Finally, semi-supervised learning will be addressed in Section 3.3.

3.1. Model

Learning data. We assume the learning data to consist in

- A dissimilarity matrix $\boldsymbol{D} = (\delta_{ij})$;
- A collection of n vectors $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, each vector $\boldsymbol{x}_i$ being composed of d attributes describing object o_i.

Most of the time, we get the n attribute vectors first and compute $\boldsymbol{D}$ as, for instance, the matrix of Euclidean distances between vectors $\boldsymbol{x}_i$:

$$\delta_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|.$$

Sometimes, it may be advantageous to compute the dissimilarities using not only the attribute vectors, but also side information such as must-link and cannot-link constraints. This case will

be investigated in Section 3.3. The most delicate situation is that of pure dissimilarity data, i.e. data consisting only in the dissimilarity matrix $\boldsymbol{D}$. Even then, we may still be able to construct a collection of attribute vectors $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ by applying Principal Component Analysis (PCA) to the dissimilarity matrix. Examples of this approach will be presented in Section 4.2.

Basic principle. Any mass function m on Ω with f focal sets $F_1, \ldots, F_f$ can be represented by a mass vector $\boldsymbol{m} \in [0,1]^f$. The basic idea behind our approach is to learn a mapping from attribute vectors to mass vectors, so as to minimize a loss function such as (5) or (7). For this, we can define a parametrized family of mappings

$$\mathcal{G} = \{g(\cdot;\boldsymbol{\theta}) : \mathbb{R}^d \to [0,1]^f \mid \boldsymbol{\theta} \in \Theta\}.$$

For each choice of $\boldsymbol{\theta}$, we then have an evidential partition $\mathcal{M}(\boldsymbol{\theta}) = (\boldsymbol{m}_1(\boldsymbol{\theta}), \ldots, \boldsymbol{m}_n(\boldsymbol{\theta}))$, with $\boldsymbol{m}_i(\boldsymbol{\theta}) = g(\boldsymbol{x}_i;\boldsymbol{\theta})$. Let $\kappa_{ij}(\boldsymbol{\theta})$ be the conflict between $\boldsymbol{m}_i(\boldsymbol{\theta})$ and $\boldsymbol{m}_j(\boldsymbol{\theta})$; it can be written using matrix notations as

$$\kappa_{ij}(\boldsymbol{\theta}) = \boldsymbol{m}_i(\boldsymbol{\theta})^T \boldsymbol{C} \boldsymbol{m}_j(\boldsymbol{\theta}), \tag{8}$$

where $\boldsymbol{C}$ is the symmetric $f \times f$ matrix with general term $C_{qr} = I(F_q \cap F_r = \varnothing)$. We can determine $\boldsymbol{\theta}$ so as to minimize a loss function such as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{2}{n(n-1)} \sum_{i<j} \left(\kappa_{ij}(\boldsymbol{\theta}) - \delta^*_{ij}\right)^2, \tag{9}$$

where $\delta^*_{ij} = \varphi(\delta_{ij})$ denotes the transformed dissimilarities. Let

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

be the solution to this optimization problem. This approach allows us to predict the cluster membership of a new object with attribute vector $\boldsymbol{x}$ by the mass vector $\boldsymbol{m} = g(\boldsymbol{x};\widehat{\boldsymbol{\theta}})$.

Neural network model. A common choice for a parametrized family of function $\mathcal{G}$ is a multi-layer feedforward neural network (NN) model [19]. Without loss of generality, let us consider a network with one hidden layer of n_H neurons with rectified linear units and a softmax output layer. The activation a_h and the output z_h of hidden unit $h \in \{1, \ldots, n_H\}$ in such a network are defined, respectively, as

$$a_h = v_{h0} + \sum_{k=1}^{d} v_{hk} x_k \tag{10a}$$

and

$$z_h = \max(0, a_h), \tag{10b}$$

where v_{hk} is the weight of the connection between input k and hidden unit h and v_{h0} is a bias term. Similarly, the activation of output unit $q \in \{1, \ldots, f\}$ is

$$\mu_q = w_{q0} + \sum_{h=1}^{n_H} w_{qh} z_h, \tag{11a}$$

where w_{qh} is the weight of the connection between hidden unit h and output unit q and w_{q0} is a bias term. The corresponding output of unit q is finally

$$m_q = \frac{\exp(\mu_q)}{\sum_{r=1}^{f} \exp(\mu_r)}. \tag{11b}$$

Outlier detection. NN models such as described above usually have very good performances, but they provide arbitrary predictions for inputs that lie in regions with no training data. A trained NN might thus not be able to detect outliers in a test set of previously unseen input vectors. (We recall that, in EVCLUS, outliers are identified by a large mass assigned to the empty set). To address this issue, we propose to optionally couple a feedforward NN with a one-class support vector machine (SVM) [45]. A one-class SVM is an unsupervised learning method that constructs a "simple" region R of the input space containing a large fraction of the data. This region is described by a function f that returns a value $f(\boldsymbol{x}) > 0$ when $\boldsymbol{x}$ belongs to R, and $f(\boldsymbol{x}) < 0$ when $\boldsymbol{x}$ belongs to the complement of R. Function f has the following expression

$$f(\boldsymbol{x}) = \alpha_0 + \sum_{i \in \mathsf{SV}} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i),$$

where $K(\cdot,\cdot)$ is a kernel function fixed a priori, $\mathsf{SV} \subset \{1,\dots,n\}$ is the set of indices of the support vectors, and the α_i are coefficients. The coefficients α_0 and α_i as well as the support vectors are learnt by minimizing a loss function. Thanks to the kernel trick, one-class SVMs can adapt to arbitrarily complex input vector distributions and make it possible to detect new input vectors generated from a different distribution (a problem often referred to as "novelty detection" [45]).

Let $\boldsymbol{m} = g(\boldsymbol{x})$ be the mass vector computed by the NN for input $\boldsymbol{x}$, and $f(\boldsymbol{x})$ the output of the one-class SVM. We define a transformed mass function $\boldsymbol{m}^*$ as

$$\boldsymbol{m}^* = \gamma \boldsymbol{m} + (1-\gamma)\boldsymbol{m}_\varnothing, \tag{12}$$

where $\boldsymbol{m}_\varnothing$ is the mass vector corresponding to the mass function $m_\varnothing$ such that $m_\varnothing(\varnothing) = 1$, and $\gamma \in [0,1]$ is a coefficient defined as an increasing function of $f(\boldsymbol{x})$ such that $\gamma \to 1$ when $f(\boldsymbol{x}) \to +\infty$ and $\gamma \to 0$ when $f(\boldsymbol{x}) \to -\infty$. For instance, we can define γ as

$$\gamma = \frac{\eta}{1+\eta} \tag{13a}$$

where η is related to an affine function of $f(\boldsymbol{x})$ by the "softplus" mapping [19]

$$\eta = \log\left[1 + \exp(\beta_0 + \beta_1 f(\boldsymbol{x}))\right]. \tag{13b}$$

The complete model is illustrated in Figure 4.

3.2. Learning

The one-class SVM in the above model can be trained separately using standard algorithms described in [45]. Here, we focus on the training of the NN component, which can be done by minimizing a loss function such as (9), which is the average over all objet pairs (i,j) of the error

$$\mathcal{L}_{ij}(\boldsymbol{\theta}) = \left(\kappa_{ij}(\boldsymbol{\theta}) - \delta^*_{ij}\right)^2. \tag{14}$$

We can remark that the error is not computed for every instance as in standard NN training, but for every pair of instances. We can view the learning process as two identical (or "Siamese" [5]) networks operating in parallel, as illustrated in Figure 5. For each $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, one input $\boldsymbol{x}_i$ is presented to the first network and $\boldsymbol{x}_j$ is presented to the second one. The degree of conflict κ_{ij} between output mass functions m^*_i and m^*_j is then computed using (8), and the error is defined as the squared difference between κ_{ij} and the transformed dissimilarity δ^*_{ij}.

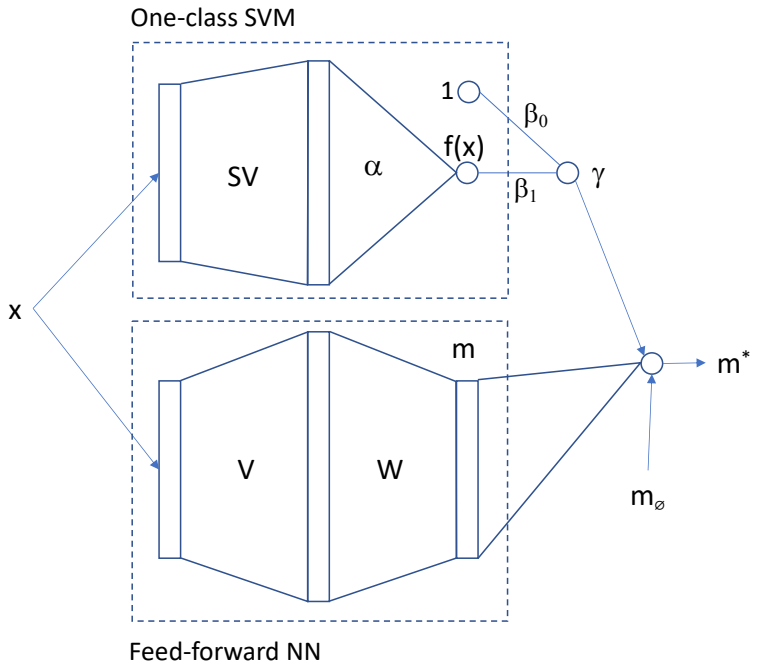


Figure 4: Complete NN-EVCLUS model composed of a multi-layer NN and a one-class SVM. The output for input vector $\boldsymbol{x}$ is a mass vector $\boldsymbol{m}^* = \gamma \boldsymbol{m} + (1-\gamma)\boldsymbol{m}_\varnothing$, where $\boldsymbol{m} = g(\boldsymbol{x})$ is the NN output and γ is a coefficient computed as an increasing function of the linearly transformed one-class SVM output $\beta_0 + \beta_1 f(\boldsymbol{x})$.

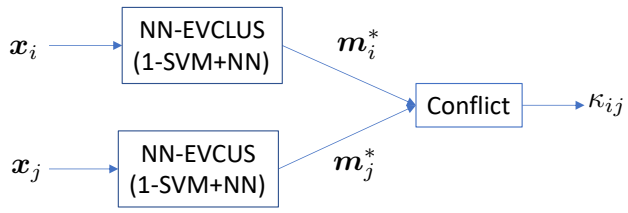


Figure 5: The learning process of EVCLUS seen as two identical "Siamese" networks operating in parallel.

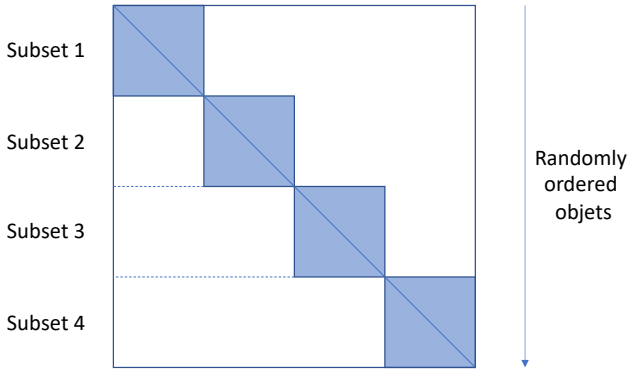


Figure 6: Illustration of the sampling procedure. The objects are randomly ordered and partitioned into s subset ($s = 4$ in the figure). Distances within each subsets are computed. This procedure gives us s mini-batches of approximately $(n/s)[(n/s) - 1]/2$ pairs. (Only half of the distance sub-matrices need to be computed as they are assumed to be symmetric).

The calculation of the error derivatives is detailed in Appendix A. The learning can be done in batch mode for small or medium-size datasets or using mini-batch stochastic gradient descent (SGD) for large datasets. In batch mode, we can directly minimize the average loss (9) over the whole distance matrix, or over a subset of distances as done in EVCLUS (see Eq. (7)). In the latter case, the average loss is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{np} \sum_{i=1}^{n} \sum_{j \in J(i)} \mathcal{L}_{ij}(\boldsymbol{\theta}), \tag{15}$$

where $J(i)$ is a subset of $\{1, \ldots, n\}$ of cardinality p, randomly selected before the learning process. Minimizing (15) instead of (9) allows us to store only np distances instead of $n(n-1)/2$ and to accelerate the calculations. To implement mini-batch SGD, we need to randomly sample q pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and average the gradient of $\mathcal{L}_{ij}$ over these q pairs before each weight update. This sampling can be done in several ways. One approach is to randomly order the objects, partition them in s subsets of approximately equal size n/s, and compute all the pairwise distances within each subset. This gives us s mini-batches of approximately $(n/s)[(n/s) - 1]/2$ pairs (i, j). This sampling procedure is illustrated in Figure 6.

Architecture design and regularization. To implement the NN-EVCLUS method, we need to select the number c of clusters, the focal sets and the NN architecture (number of layers and number of units per layer).

To determine the number of clusters, we can use automatic selection criteria such as the non-specificity measure proposed in [36], or an interactive approach based on visualization techniques, such as the $\delta - Bel$ graph proposed in [49]. The latter approach is less computationally intensive and often more reliable than the former; it will be adopted in this paper.

The choice of the set $\mathcal{F}$ of focal sets depends on the number c of clusters. For small values of c (say, $c \leq 5$), all 2^c focal sets can be considered. For small and medium values of c (typically, $c \leq 10$), we can consider only the subsets of cardinality less than or equal to 2, and the frame Ω. For large c, a common choice is to keep only the empty set, the singletons, and Ω. Alternatively, we can adopt the two-step strategy proposed in [13], in which we first fit the model with singletons (as well as $\varnothing$ and Ω), and then include selected pairs of neighboring clusters in a second step.

As far as the network architecture is concerned, we have found one hidden layer to be sufficient for most datasets. However, it is possible that very complex datasets would require two hidden layers or more. Assuming one hidden layer, the next decision to make is on the number n_H of hidden units. If the emphasis is on clustering a single data set, we need not concern ourselves with generalization and we can choose n_H large enough to reach a small enough discrepancy between degrees of conflict and transformed distances. If we can run EVCLUS, then the loss reached by EVCLUS can be considered as a lower bound of the loss achievable by NN-EVCLUS (as the latter model has fewer free parameters). If we intend to use the model for prediction, then some complexity control technique should be used. A common approach of regularization: for instance, ℓ_2 regularization combined with (15) gives us the following regularized loss function:

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{np}\sum_{i=1}^{n}\sum_{j\in J(i)}\mathcal{L}_{ij}(\boldsymbol{\theta}) + \frac{\lambda}{2}\left(\frac{1}{n_H(d+1)}\sum_{h,k}v_{hk}^2 + \frac{1}{f(n_H+1)}\sum_{q,h}w_{qh}^2\right),$$

where hyperparameter λ can be tuned by cross-validation or using the hold-out method. When using mini-batch SGD, regularization can, alternatively, be obtained by the early stopping technique (interrupting the learning process when the loss computed on a validation set starts increasing).

Complexity. From Eqs. (8) and (14)-(15), the complexity of computing the loss function in batch mode is $O(nkf^2)$, and from Eqs (A.1)-(A.10), the gradient of the loss function with respect to the weights (for a network with one hidden layer) can be computed in $O(nkn_Hf(f+d))$ operations. The complexity is, thus, proportional to the number n of objects and to f^2, which confirms the necessity of limiting the number f of focal sets when the number c of clusters is large. Keeping only the singletons, the empty set and Ω, we have $f = c+2$, and the number of operations for the gradient calculation becomes proportional to c^2.

Example 4. *As an example, we consider again the* fourclass *dataset of Example 3. As this dataset has only two features it is easy to determine the number of clusters by just displaying the data. The four clusters are also very clearly visible in the $\delta - Bel$ graph shown in Figure 7. In this graph, Bel denotes the degree of belief that an attribute vector is a cluster center, and δ is the minimum distance to vectors with a higher value of Bel [49]. Cluster centers are typically located in the upper-right corner of the graph.*

For the one-class SVM part we used the ν-SVM algorithm in R package `kernlab` *[28] with a Gaussian kernel. This method has two hyperparameters: ν, which is an upper bound on the fraction of outliers, and the kernel width σ. We set $\nu = 0.2$ and $\sigma = 0.2$. Contours of the SVM output $f(\boldsymbol{x})$ are shown in Figure 8a.*

For the NN part, we set $n_H = 20$ and $\lambda = 0$. The focal sets were restricted to the subsets of cardinality less than or equal to 2, and the frame Ω. The network was trained in batch mode using a gradient-based procedure quite similar to the method described in [47]. We minimized loss function (15) with $p = 100$. We started the algorithm from five independent random conditions and we kept the best result. The learning curve is shown in Figure 9a with the loss of EVCLUS as a comparison, and the Shepard diagram is displayed in Figure 9b. Comparing Figures 3b and 9b, we can see that the quality of the approximation is similar for EVCLUS and NN-EVCLUS. The former algorithm reaches a loss of 4.75×10^{-3}, while the latter yields 5.64×10^{-3}.

The obtained evidential partition, shown in Figure 10, is similar to that obtained by EVCLUS, which is displayed in Figure 2. However, NN-EVCLUS also allows us to predict the cluster membership of new objects. Figure 11 shows the predicted evidential partition of a data set of 1000

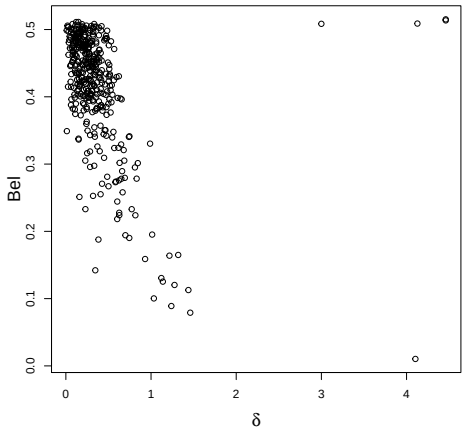
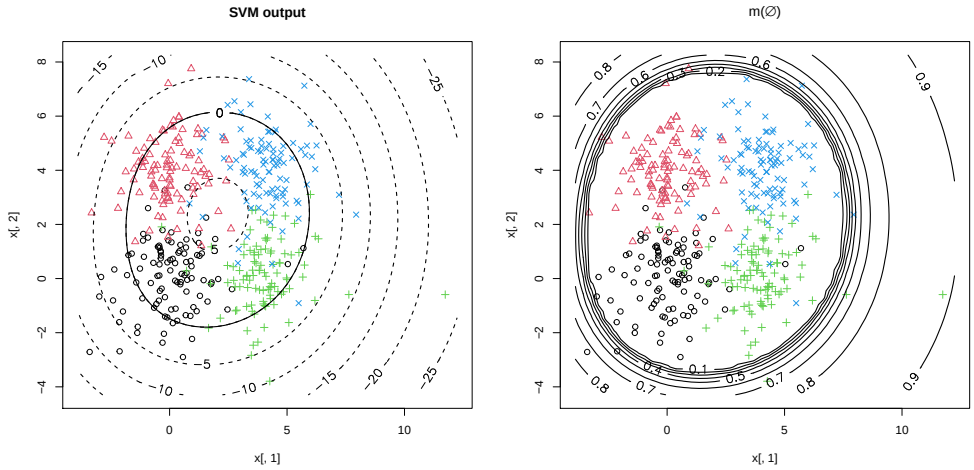
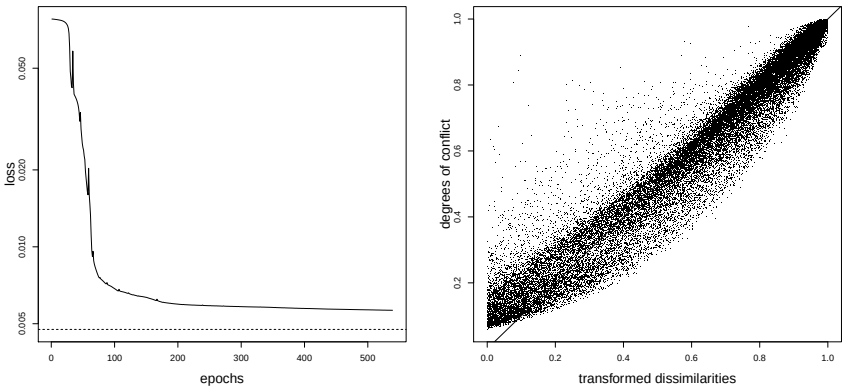


Figure 7: $\delta - Bel$ plot of the fourclass dataset, with $K = 50$ neighbors and $q = 0.9$. The four cluster centers correspond to the four points in the upper-right corner of the graph. (Note that two points are very close and almost indiscernable).



(a) (b)

Figure 8: Output of the one-class SVM (a) and mass on the empty set (b) for the fourclass dataset.



(a) (b)

Figure 9: (a): Learning curve of a NN trained in batch mode on the fourclass dataset (solid line), and value of the loss reached by EVCLUS (horizontal broken line); (b): plot of the degrees of conflict κ_{ij} vs. the transformed dissimilarities δ^*_{ij}.

vectors drawn from the same distribution as fourclass *(a mixture of four multidimensional Student distributions). We can see that the four clusters and the outliers are correctly identified. Figure 12 shows contours of the masses assigned to the singletons (Figures 12a-12d) and some pairs of clusters (Figures 12e-12h) as functions of* $\boldsymbol{x}$*, and Figure 13 displays contour plots of the plausibility of each of the four clusters as functions of* $\boldsymbol{x}$*. A contour plot of the mass on the empty set is shown in Figure 8b.*

Figure 14 shows the influence on the training and test performance of the number n_H *of hidden units (Figure 14a) and of the regularization coefficient* λ *(Figure 14b) with* $n_H = 50$*. The test loss was computed using a dataset represented in Figure 11. As expected, the training error decreases slowly with* n_H *while the generalization reaches a plateau at* $n_H = 45$*. Similarly, the training error increases with* λ *for fixed* n_H*, but the generalization error reaches a minimum for* $\lambda = 10^{-5}$*.*

3.3. Using side information

In many cases, additional knowledge about some objects can guide the learning process. In clustering, such knowledge may take the form of pairwise constraints specifying that some objects belong to the same class (*must-link* constraints), or belong to different classes (*cannot-link* constraints). In evidential clustering, a variant of ECM (called CECM) dealing with such constraints was first introduced in [1]. The constrained version of EVCLUS (called CEVCLUS) was then introduced in [2], and improved in [32].

CEVCLUS minimizes a loss function that is the sum of the squared error loss and a penalization term defined as follows. 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 [1]:

$$Pl_{ij}(S_{ij}) = 1 - \kappa_{ij} \tag{16a}$$

$$Pl_{ij}(\overline{S}_{ij}) = 1 - m_i(\varnothing) - m_j(\varnothing) + m_i(\varnothing)m_j(\varnothing) - \sum_{k=1}^{c} m_i(\{\omega_k\})m_j(\{\omega_k\}). \tag{16b}$$

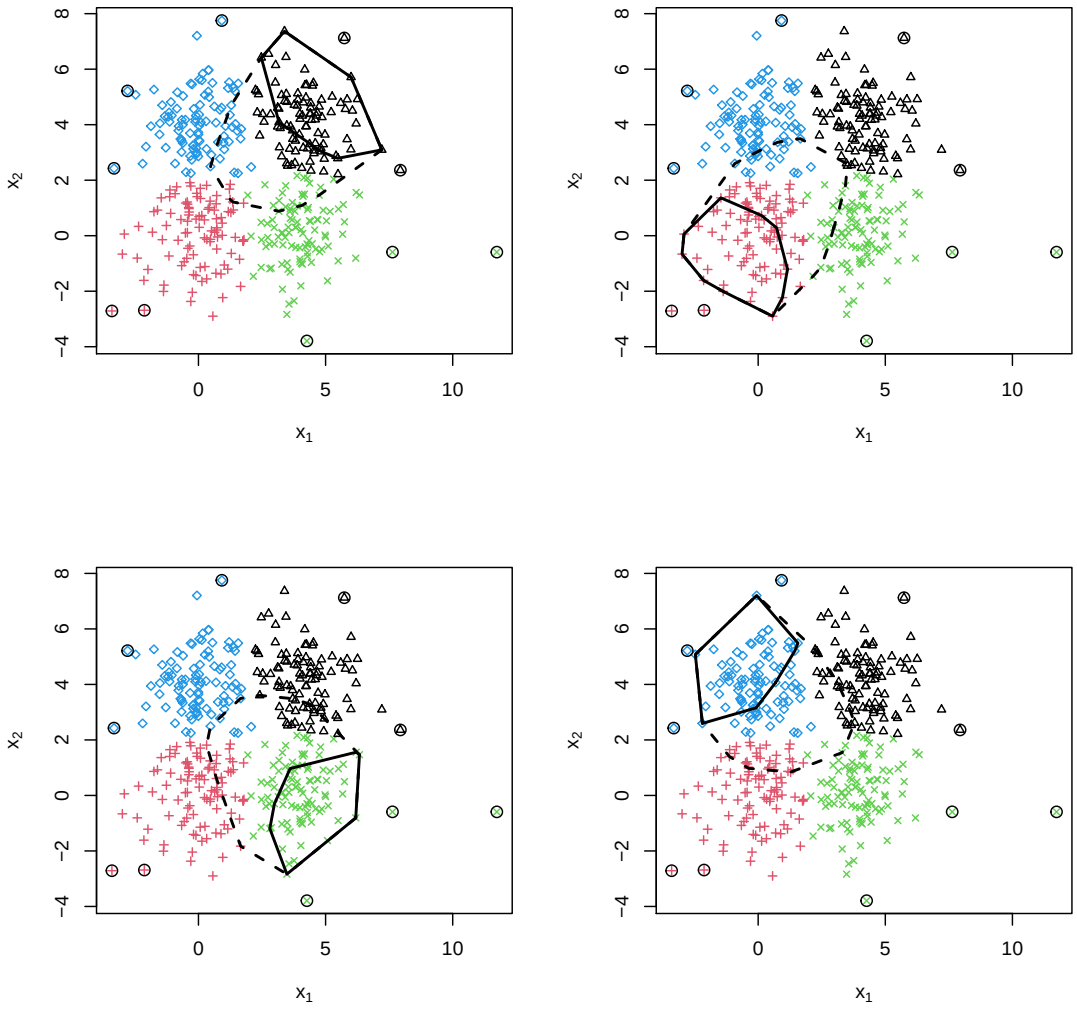


Figure 10: Lower and upper approximations of the four clusters for the fourclass dataset found by NN-EVCLUS. The true classes are displayed with different colors. The identified clusters are plotted with different symbols. The convex hulls of the cluster lower and upper approximations are displayed using solid and interrupted lines, respectively. The outliers are indicated by circles.

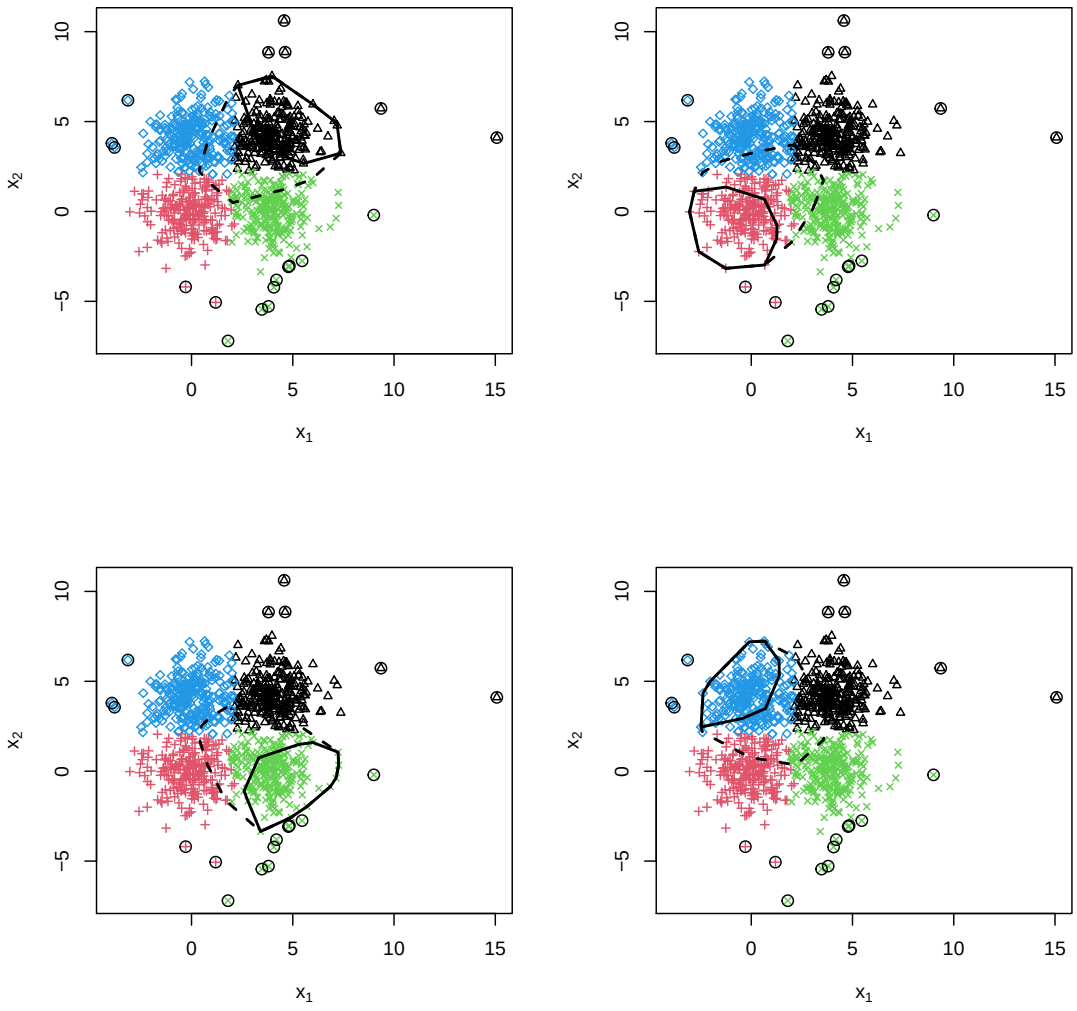


Figure 11: Lower and upper approximations of the four clusters for a test dataset of size 1000 drawn from the same distribution as the fourclass dataset. The true classes are displayed with different colors. The identified clusters are plotted with different symbols. The convex hulls of the cluster lower and upper approximations are displayed using solid and interrupted lines, respectively. The outliers are indicated by circles.

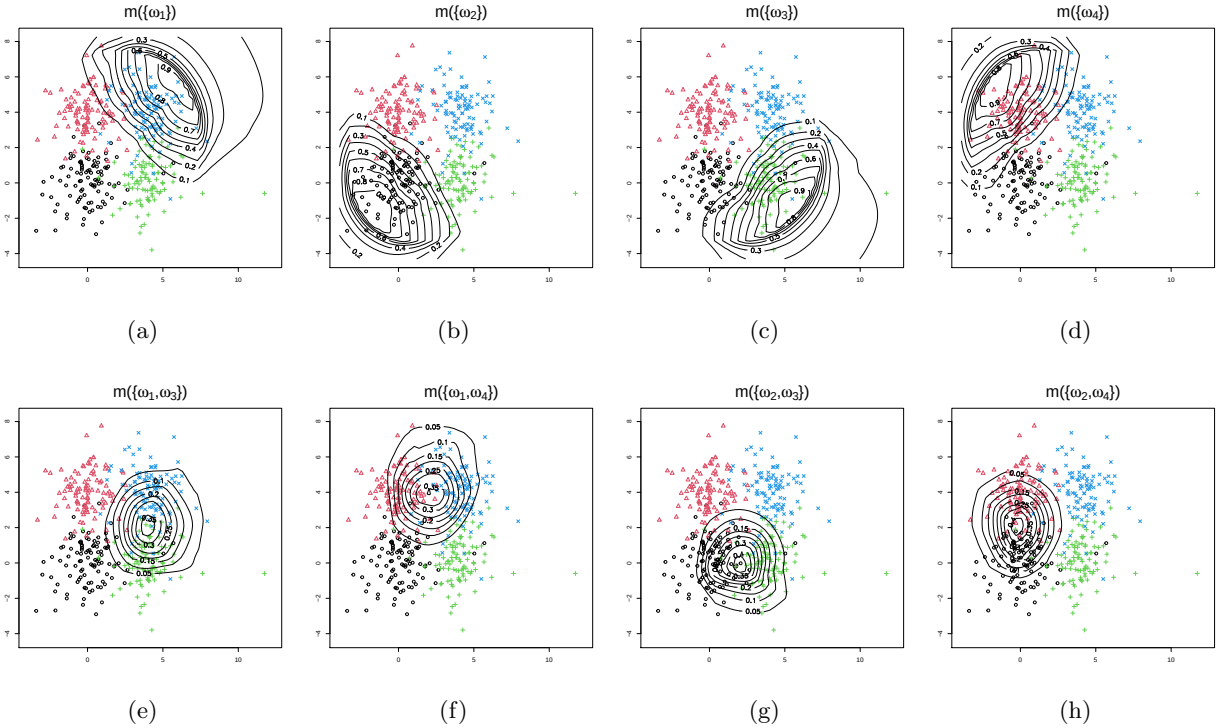
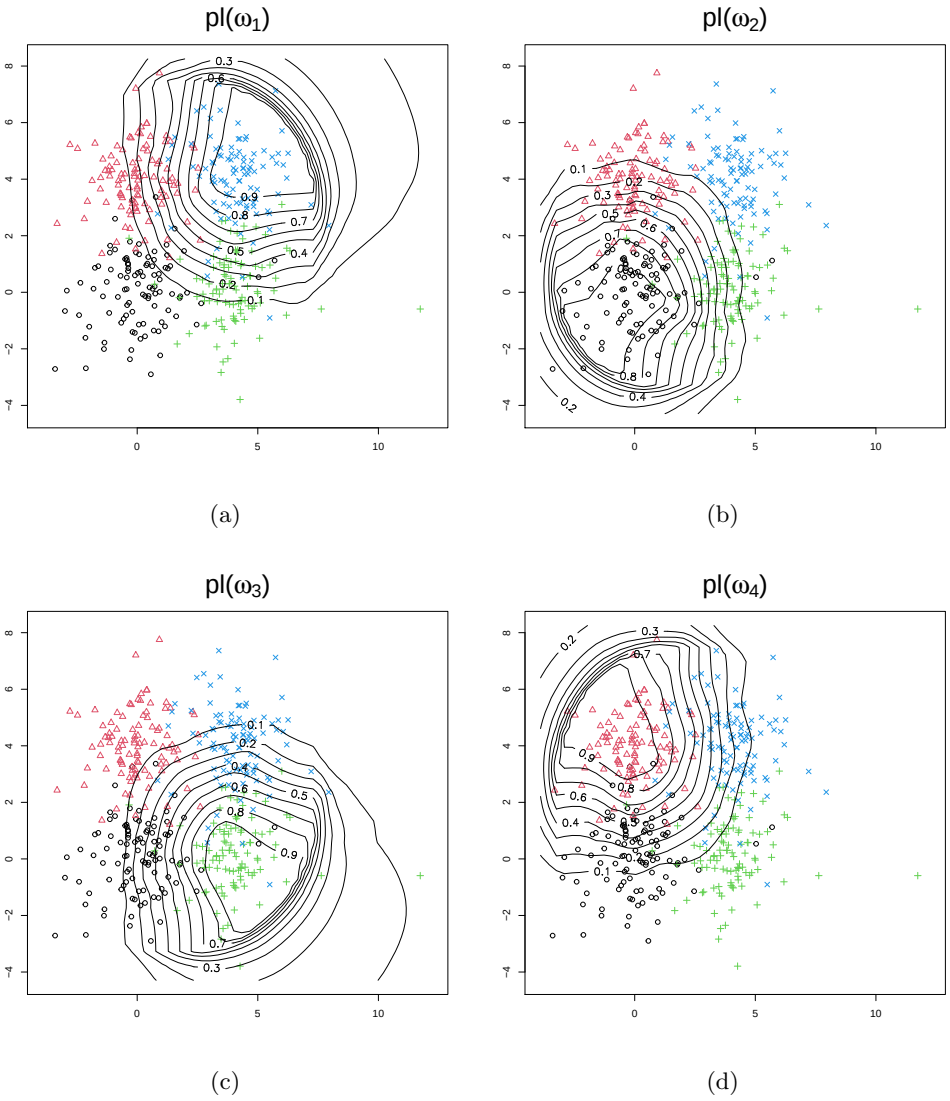
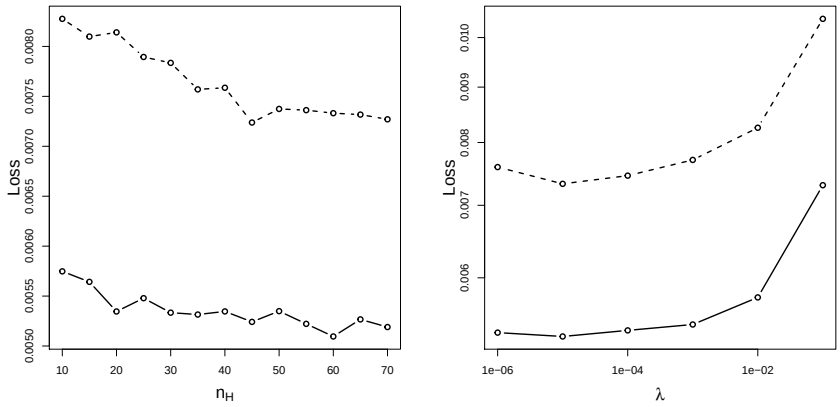


Figure 12: Contour plot of the masses assigned to singletons (a-d) and pairs of contiguous clusters (e-h) by NN-EVCLUS for the fourclass dataset.



(a) (b) (c) (d)

Figure 13: Contour plot of plausibilities of each of the four clusters obtained by NN-EVCLUS for the fourclass dataset.



(a) (b)

Figure 14: Training (solid lines) and test (broken lines) loss of NN-EVCLUS on the fourclass dataset vs. number n_H of hidden units (a) and vs. regularization coefficient λ (b) with 50 hidden units.

We note that the pair $(Pl_{ij}(S_{ij}), Pl_{ij}(\overline{S}_{ij}))$ can take any value in $[0,1]^2$; for instance, if $m_i(\varnothing) = m_j(\varnothing) = 1$, then $Pl_{ij}(S_{ij}) = Pl_{ij}(\overline{S}_{ij}) = 0$, and if $m_i(\Omega) = m_j(\Omega) = 1$, then $Pl_{ij}(S_{ij}) = Pl_{ij}(\overline{S}_{ij}) = 1$. For objects i and j known to belong to the same cluster, $Pl_{ij}(S_{ij})$ should be high and $Pl_{ij}(\overline{S}_{ij})$ should be low, and the converse holds if objects i and j are known to belong to different clusters. The loss function of CEVCLUS is thus defined as

$$\mathcal{L}_c(\mathcal{M}) = \mathcal{L}(\mathcal{M}) + \frac{\xi}{2(|\mathsf{ML}| + |\mathsf{CL}|)}(\mathcal{P}_{\mathsf{ML}} + \mathcal{P}_{\mathsf{CL}}), \tag{17}$$

with

$$\mathcal{P}_{\mathsf{ML}} = \sum_{(i,j)\in\mathsf{ML}} \left(Pl_{ij}(\overline{S}_{ij}) + 1 - Pl_{ij}(S_{ij})\right), \tag{18a}$$

$$\mathcal{P}_{\mathsf{CL}} = \sum_{(i,j)\in\mathsf{CL}} \left(Pl_{ij}(S_{ij}) + 1 - Pl_{ij}(\overline{S}_{ij})\right), \tag{18b}$$

where $\mathcal{L}(\mathcal{M})$ is the squared error loss defined by (5) or (7), ξ is a hyperparameter that controls the trade-off between the stress and the constraints, and ML and CL are the sets of must-link and cannot-link constraints, respectively. The constrained loss (17)-(18) was minimized by gradient descent in the original version of CEVCLUS [2]; a more efficient cyclic coordinate descent algorithm was proposed in [32].

Another form of prior information may come as a subset of labeled objects. This is the point of view of semi-supervised learning [6]. This approach is relevant when classes are already defined, but only a subset of objects can be labeled because of time or cost constraints. Given a subset of labeled data, it is possible to derive must-link and cannot-link contraints, but the converse is false in general: the pairwise constraint formalism is thus more general. Semi-supervised learning is rather seen as an extension of supervised classification, whereas constrained clustering is seen as an extension of fully unsupervised learning. Both types of side information can easily be exploited by NN-EVCLUS, as will be explained below.

Integration of pairwise constraints. Pairwise constraints can be easily integrated in NN-EVCLUS using a penalized loss such as (17). From (8), we have

$$Pl_{ij}(S_{ij}) = 1 - \boldsymbol{m}_i^{*T}\boldsymbol{C}\boldsymbol{m}_j^* = \boldsymbol{m}_1^{*T}(\mathbf{1}\mathbf{1}^T - \boldsymbol{C})\boldsymbol{m}_j^*,$$

where $\mathbf{1}$ is a column vector of length f whose components are all equal to 1. Similarly, we can write

$$Pl_{ij}(\overline{S}_{ij}) = 1 - \boldsymbol{m}_i^{*T}\boldsymbol{E}\boldsymbol{m}_j^* - \boldsymbol{m}_i^{*T}\boldsymbol{S}\boldsymbol{m}_j^* = \boldsymbol{m}_1^{*T}(\mathbf{1}\mathbf{1}^T - \boldsymbol{E} - \boldsymbol{S})\boldsymbol{m}_j^*,$$

where $\boldsymbol{E}$ is a square matrix of size f defined as

$$\boldsymbol{E} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}$$

and $\boldsymbol{S}$ is the square matrix of size f with general term

$$(\boldsymbol{S})_{k\ell} = \begin{cases} 1 & k = \ell,\ |F_k| = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, we can rewrite (18) as

$$\mathcal{P}_{\mathsf{ML}} = \sum_{(i,j)\in\mathsf{ML}} \boldsymbol{m}_i^{*T}\boldsymbol{Q}\boldsymbol{m}_j^* \tag{19a}$$

$$\mathcal{P}_{\mathsf{CL}} = \sum_{(i,j)\in\mathsf{CL}} \left(2 - \boldsymbol{m}_i^{*T}\boldsymbol{Q}\boldsymbol{m}_j^*\right), \tag{19b}$$

with $\boldsymbol{Q} = \mathbf{1}\mathbf{1}^T + \boldsymbol{C} - \boldsymbol{E} - \boldsymbol{S}$. The gradients of $\mathcal{P}_{\mathsf{ML}}$ and $\mathcal{P}_{\mathsf{CL}}$ with respect to the network parameters are given in Appendix B.

Integration of labeled data. Let us assume that we have n_s labeled attribute vectors $\{(\boldsymbol{x}_i, y_i), i \in \mathcal{I}_s\}$, where $\mathcal{I}_s \subseteq \{1, \ldots, n\}$, and $y_i \in \Omega$ is the class label of object i. We can use this information by minimizing a penalized loss of the form

$$\mathcal{L}_S(\boldsymbol{\theta}) = (1 - \nu)\mathcal{L}(\boldsymbol{\theta}) + \nu\mathcal{P}_s, \tag{20}$$

where ν is a coefficient and $\mathcal{P}_s$ is a penalization term defined as

$$\mathcal{P}_s = \frac{1}{n_s}\sum_{i\in\mathcal{I}_s}\sum_{l=1}^{c}(pl_{il}^* - y_{il})^2. \tag{21}$$

In (21), $y_{il} = I(y_i = \omega_l)$, $pl_{il}^* = pl_i^*(\omega_l)$, and pl_i^* is the contour function corresponding to m_i^*. The rationale behind (21) is that, when object i is known to belong to class l, the plausibility of that class should be high, while the plausibility of the other classes should be low. We can notice than, when $n_s = n$, the learning task becomes fully supervised. The gradient of $\mathcal{P}_s$ with respect to the model parameters is given in Appendix C

Metric adaptation. Although it has been shown that CEVCLUS has the ability to use of pairwise constraints to improve clustering results [2, 32], it can fail to do so effectively when the constraints are inconsistent with the distance matrix. As recalled in Section 2.3, EVCLUS is based on the assumption that similar objects are likely to belong to the same cluster and, conversely, dissimilar objects plausibly belong to different clusters. If pairwise constraints are provided, which force similar objects to belong to different clusters, or dissimilar objects to belong to the same cluster, then the two terms in loss function (17) may become strongly inconsistent and CEVCLUS may fail to find a suitable evidential partition.

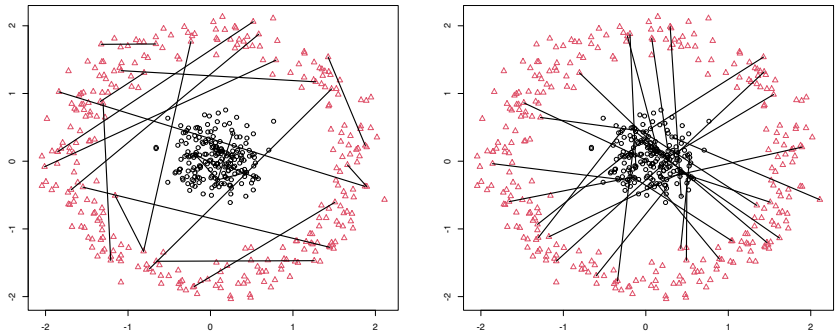
An approach to solve this problem is to use the additional information (pairwise constraints or labeled data) to learn a metric such that objects that are known to belong to different clusters become further apart, while objects in a given cluster are as similar as possible. When labeled data are provided, we can extract discriminant features using classical Fisher Discriminant Analysis (FDA) or a nonlinear version such as Local Fisher Discriminant Analysis (FDA) [50] or Kernel Fisher Discriminant Analysis (KFDA) [57], and compute the distance matrix in the new feature space. When pairwise constraints are available, we can use feature extraction techniques such as Learning with Side Information (LSI) [54], Distance Metric Learning with Eigenvalue Optimization (DML-eig) [59], Pairwise Constrained Component Analysis (PCCA) or its kernelized version KPCCA [38].

Example 5. *The* circles *dataset shown in Figure 15 is composed of 500 two-dimensional vectors distributed in a spherical cluster surrounded by a circular-shaped cluster. The proportions of the two clusters are, respectively, 2/5 and 3/5. NN-EVCLUS cannot find this partition without additional knowledge, because it violates the fundamental assumption that two dissimilar objects are unlikely to belong to the same cluster: maximally distant points on the circle at the extremities of a diameter actually belong to the same cluster. We can, however, use additional information in the form of pairwise constrained or labeled data and modify the distance matrix accordingly.*

To illustrate this approach, we randomly generated 50 object pairs, which gave us 22 must-link constraints and 28 cannot-link constraints as shown, respectively, in Figure 15a and 15b. These constraints represent a tiny fraction of the $500 \times 499/2 = 124750$ *object pairs. We used this information to extract a discriminant feature by KPCCA[38] using a Gaussian kernel with inverse kernel width* $\sigma = 0.3$*. As shown in Figure 16, the data are linearly separable in this new one-dimensional feature space. The Euclidean distance matrix in the original space and in the transformed space are shown, respectively, in Figure 17a and 17b.*

We trained NN-EVCLUS with the original features $\boldsymbol{x}$ *and the Euclidean matrix in the KPCCA feature space, with* $n_H = 30$ *hidden units,* $\lambda = 0.01$*, and the loss function* (17) *with* $\xi = 0.1$*. (The results are not sensitive to* ξ *in this example, and even setting* $\xi = 0$*, i.e., ignoring the constraints gives good results thanks to the very good separation in the transformed feature space). The results are shown as contour plots of the masses assigned to each of the four focal sets in Figure 18, and as contour plots of the plausibilities of the two classes in Figure 19. We can see that a meaningful evidential partition has been found and the two clusters are perfectly separated (as shown by the decision boundary plotted in red in Figure 19).*

Similar results were obtained with 50 randomly selected labeled instances as shown in Figure 20a, using KFDA instead of KPCCA, and penalized loss function (20)*. The distributions of the discriminant feature extracted by KFDA in each of the two classes are shown in Figure 20b, and contour plots of the plausibilities of the two classes computed by NN-EVCLUS trained in semi-supervised mode with the 50 labeled data are displayed in Figure 21. By comparing Figures 19 and*



(a) (b)

Figure 15: The circles dataset with must-link (a) and cannot-link (b) constraints.

21, we can see that the results are almost identical.

4. Comparative experiments

In this section, we compare the performances of NN-EVCLUS with those of alternative evidential clustering algorithms on a sample of publicly available datasets. All the simulations were performed using an implementation of our algorithm in R publicly available in package `evclust` [8]. Fully unsupervised clustering of attribute and dissimilarity data will first be addressed, respectively, in Sections 4.1 and 4.2. Clustering with pairwise constraints will then be studied in Section 4.3.

4.1. Unsupervised clustering of attribute data

Data sets. We considered the 14 publicly available real and artificial datasets summarized in Table 2. These datasets all contain attribute data and have a wide range of characteristics in terms of input dimension and number of clusters. For the Ecoli dataset, we used only the quantitative attributes (2, 3, 6, 7, and 8) and the four most frequent classes: 'im', 'pp', 'imU' and 'cp'; we then merged the classes 'im' and 'imU', resulting in a dataset with 307 objects described by five attributes and partitioned into three clusters. The Letters4p1 is a subset of the "Letter Recognition" dataset from the UCI machine learning repository [14] containing six clusters. The Mice dataset is a part of the data analyzed in [22]; it contains the expression levels of 22 proteins for the trisomic mice: we considered the 26 proteins listed in columns 4 and 5 of Table 3 in [22], and we retained only the 22 of them without missing values. The three classes are: "t-SC-m" (shock-context with memantine), "t-SC-s" (shock-context with saline) and "t-CS" (context-shock with either memantine or saline). The DryBean dataset is a subset of the data analyzed in [30], with 200 randomly selected objects in each class. The Leaves5p1 dataset [18] is a subset of the "One-hundred plant species leaves" in the UCI database [14] containing 320 objects from 20 classes. All datasets contain only numerical attributes; the dissimilarities were computed as Euclidean distances between attribute vectors.

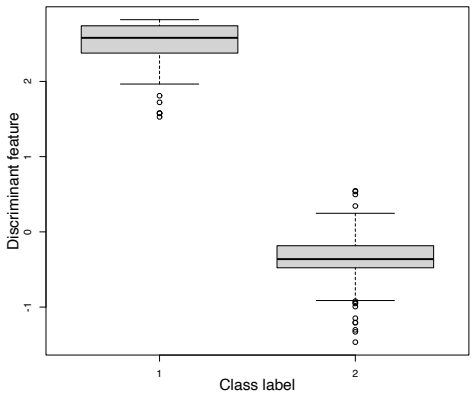
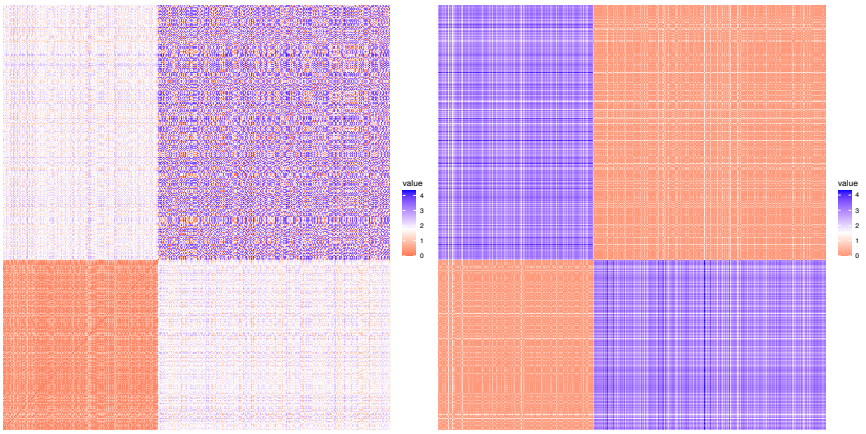
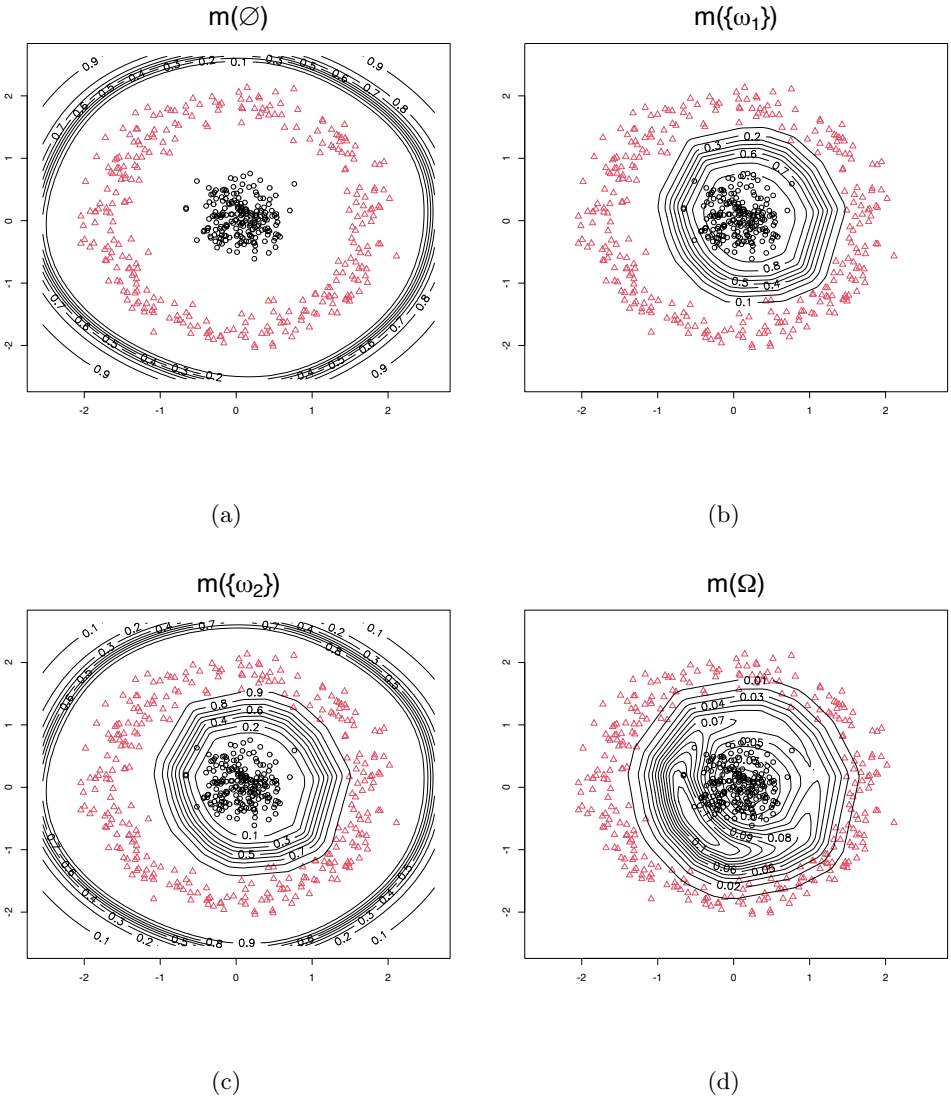


Figure 16: Boxplots of the feature extracted by KPCCA for the circles data with the pairwise constraints shown in Figure 15.



(a) (b)

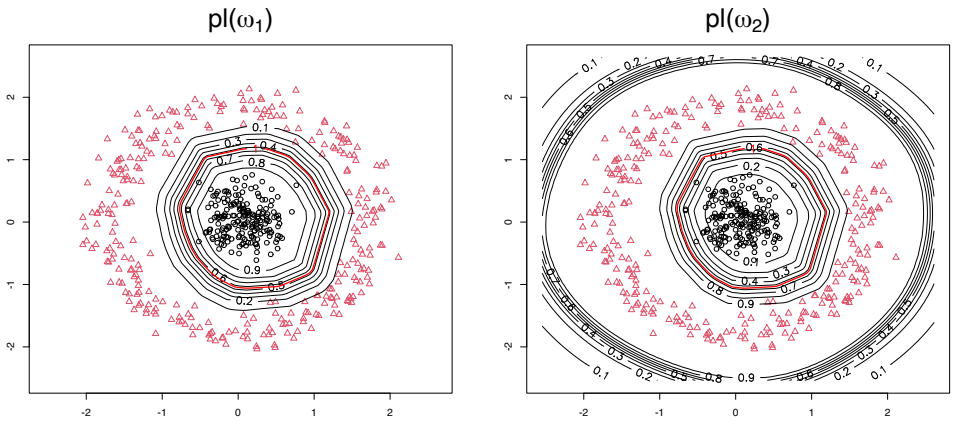
Figure 17: Image representations of the distances matrices of the circles data in the original space (a) and in the transformed space generated by KPCCA (b). (This figure is better viewed in color).



(a) (b)

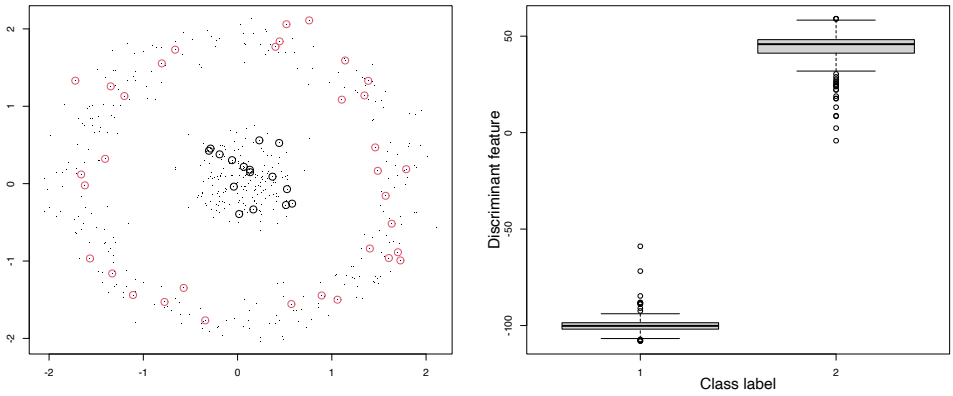
(c) (d)

Figure 18: Contour plots of the masses assigned to each of the four focal set by NN-EVCLUS trained on the circles dataset with the pairwise constraints shown in Figure 15.



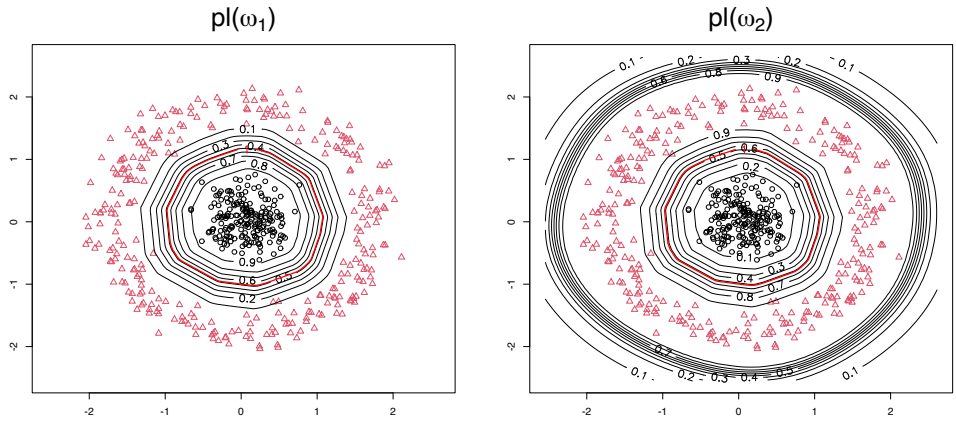
(a) (b)

Figure 19: Contour plot of plausibilities of each of the two clusters obtained by NN-EVCLUS for the circles dataset with the pairwise constraints shown in Figure 15. The red thick line represents the decision boundary between the two clusters defined as the curve with equation $pl(\omega_1) = pl(\omega_2)$.



(a) (b)

Figure 20: The circles dataset with 50 randomly selected labeled instances (a), and boxplot of the discriminant feature extracted by KFDA (b) using the labeled data.



(a) (b)

Figure 21: Contour plot of plausibilities of each of the two clusters obtained by NN-EVCLUS for the circles dataset with the 50 labeled instances shown in Figure 20a. The red thick line represents the decision boundary between the two clusters defined as the curve with equation $pl(\omega_1) = pl(\omega_2)$.

Table 2: Number n of objects, number d of attributes and number c of clusters for each of the 14 attribute datasets used in this study.

Name	n	d	c	Source
Wine	178	13	3	[14]
Iris	150	4	3	[43]
Ecoli	307	5	3	[14]
Heart	270	13	2	[14]
Seeds	210	7	3	[14]
Letters4p1	4634	16	6	[18]
Glass	214	10	6	[14]
Segment	2310	16	7	[14]
S2	5000	2	15	[18]
S4	5000	2	15	[18]
D31	3100	2	31	[18]
Mice	507	22	3	[14][22]
DryBean	1400	16	7	[14][30]
Leaves5p1	320	64	20	[18]

Algorithms. As alternative evidential clustering algorithms, we considered EVCLUS [13], ECM [36], CCM[3] [35], sECMdd, wECMdd[4] [61], BPEC [49] and Bootclus [7]. The number of clusters was assumed to be known. For EVCLUS and NN-EVCLUS, we restricted the focal sets to the empty set, singletons, pairs and Ω, except when the number c of clusters was equal to or larger than 5, in which case the pairs were not included in the focal sets. Parameter δ_0 was set to the 0.9-quantile of distances for $c \leq 4$ and to a smaller value (0.5, 0.2 or 0.1 quantile) for datasets with a larger number of clusters (as a heuristic, δ_0 should be smaller when the number of clusters is larger). For NN-EVCLUS, the number of hidden units was set to 1.5 times the number of focal sets. We used batch learning for small datasets ($n \leq 1000$) and minibatch learning with the RMSprop algorithm [19, page 300] for larger datasets. For ECM, we used as focal sets the empty set, singletons and pairs. When the number of clusters was strictly greater that 3, we used the two-step strategy described in [13]: we first trained the model with the empty set and singletons; we then identified pairs of clusters that are mutual nearest neighbors according to a similarity measure, and we re-trained the model after including these pairs as focal sets. The same strategy was applied with BPEC, for which we used the version of ECM with an adaptive metric [1]. The Bootclus method is based on the bootstrap and Gaussian mixture models (GMMs); we used the default settings of function `bootclus` in package `evclus`: $B = 500$ bootstrap samples and bootstrap percentile confidence intervals at level $1 - \alpha = 0.9$. Function `bootclus` calls function `Mclust` of package `mclust`, which selects the best GMM according to the Bayesian information criterion (BIC).

The $\delta - Bel$ method was used to identify cluster centers. When these centers were clearly discernible in the $\delta - Bel$ graph, they were used with BPEC. We also considered ways of exploiting this information with other methods. For ECM and CCM, the cluster centers were used as initial prototypes (the corresponding methods will be denoted, respectively, as ECM-bp and CCM-bp). For NN-EVCLUS, we treated these centers as labeled data and optimized loss function (20) with $\nu = 0.5$; the corresponding method will be denoted as NN-EVCLUS-bp. For the `Leaves5p1` dataset, as the belief peaks did not identify all clusters, we used the attribute vectors with maximum plausibility in each class (one vector per class) obtained with EVCLUS instead. The CCM algorithm was used with the empty set, the singletons, the pairs and Ω as focal sets; the parameters were set to the default parameters as recommended by the authors [35], i.e., $\gamma = 1$, $\beta = 2$ and $T_c = 2$. As in ECM and BPEC, parameter δ, which controls the number of outliers was set to 5. Similarly, we used the default values recommended in [61] for sECMdd ($\beta = 2$, $\alpha = 2$, $\eta = 1$, $\gamma = 1$) and wECMdd ($\beta = 2$, $\alpha = 2$, $\xi = 5$, $\psi = 2$). All algorithms were run five times and we kept the best solution in terms of loss or objective function.

Performance criteria. Measuring the quality of evidential partitions is a difficult problem. One approach is to convert the evidential partition into a hard partition by assigning each object to the cluster with the highest plausibility, and compare the resulting hard partition to the ground-truth partition using, e.g., the adjusted Rand index (ARI) [24]. This approach provides easily interpretable results and makes it possible to rank evidential clustering algorithms according to the similarity between the evidential partition and the true partition. However, it does not account for the specific characteristics of evidential partitions. In [11], we proposed evidential extensions of the Rand index, and we argued that the quality of an evidential partition could be described

[3] Our R code for CCM was translated from Matlab code provided by Prof. Zhunga Liu.

[4] The R code for sECMdd and wECMdd was provided by Dr. Kuang Zhou.

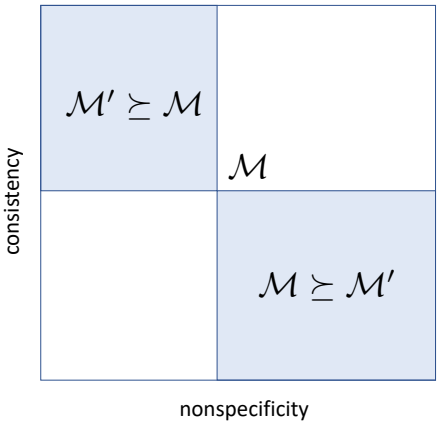


Figure 22: Partial order induced by consistency and nonspecificity: evidential partition $\mathcal{M}'$ dominates $\mathcal{M}$ (denoted as $\mathcal{M}' \succeq \mathcal{M}$) if it has higher consistency and lower nonspecificity, and it is dominated by $\mathcal{M}$ (denoted as $\mathcal{M} \succeq \mathcal{M}'$) if it has lower consistency and higher nonspecificity.

by two numbers: a consistency index (CI) measuring its agreement with the true partition, and a nonspecificity index (NS) measuring its imprecision. We generally aim at high consistency and low nonspecificity, so that the pair of indices (CI,NS) induces a partial order: an evidential partition $\mathcal{M}'$ is better than an evidential partition $\mathcal{M}$ if it has a higher consistency index and a lower nonspecificity (see Figure 22) [11]. Here, we computed the three indices (ARI, CI and NS) for each of the methods applied to each dataset.

Results and discussion. The ARI values obtained by the 11 methods on the 14 datasets are shown in Table 3. For each dataset, the best result is printed in bold, and the values within 5% of the best result are underlined. The $\delta - Bel$ method failed to identify the centers of all clusters for the Letters4p1, Glass, Segment and Leaves5p1 datasets; for this reason, the methods using cluster centers identified by this method (NN-EVCLUS-bp, ECM-bp, CCM-bp and BPEC) are not given for these datasets. Also, our implementations of sECMdd, wECMdd and Bootclus failed to converge in a reasonable amount of time for the S2, S4 and D31 characterized by large numbers of objects and clusters; no results are thus reported for these methods on these datasets.

We can see from Table 3 that NN-EVCLUS yielded either the best results in terms of ARI, or results close to the best ones for all datasets, except Iris and Segment, for which Bootclus gave better results, which can be explained by the presence of nonspherical clusters; even for these two datasets, NN-EVCLUS yielded the second best results in terms of ARI. Using the cluster centers identified by the $\delta - Bel$ method makes it possible to improve the results of NN-EVCLUS most of the time, particular when the number of clusters is large (as in the S2, S4 and D31 datasets). Whereas the methods based on prototypes (ECM, CCM and BPEC) work well on artificial datasets with well-separated clusters, they are outperformed by EVCLUS and NN-EVCLUS on real datasets characterized by highly overlapping clusters. The sECMdd and wECMdd were outperformed by other methods on all datasets. NN-EVCLUS with random initialization performs equally well as, or better than EVCLUS on most datasets, except when the number of clusters is large (as in the S2, S4 and D31 datasets), in which case NN-EVCLUS may fail to find a deep minimum of the loss function; in these cases, using prior information provided by the $\delta - Bel$ method is crucial. When the $\delta - Bel$ fails, using maximum plausibility attribute vectors provided by EVCLUS seems to be a good strategy for training NN-EVCLUS when the number of clusters is large, as shown by the

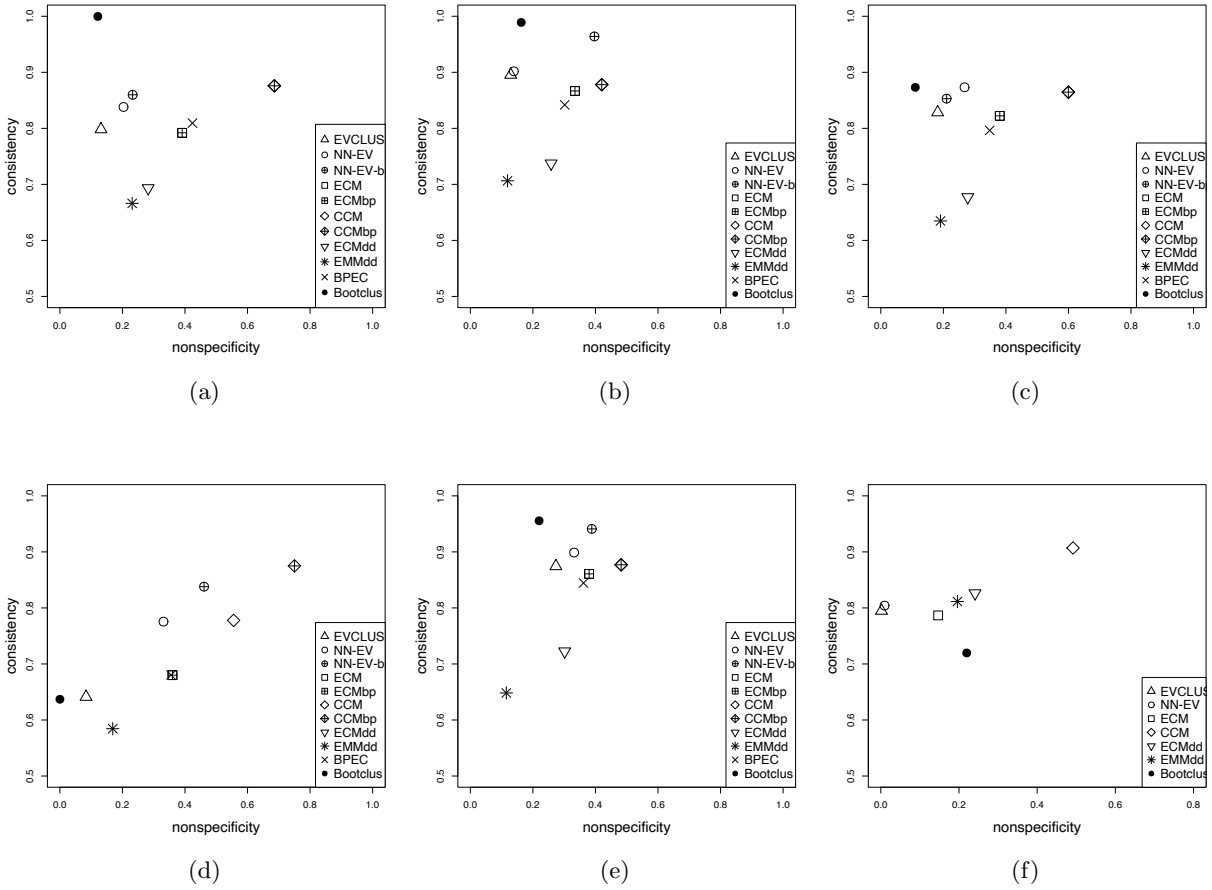


Figure 23: Consistency index (vertical axis) vs. nonspecificity (horizontal axis) for the Wine (a), Iris (b), Ecoli (c), Heart (d), Seeds (e) and Letters4p1 (f) datasets.

good results obtained on the Leaves5p1 dataset.

Figures 23-25 display the consistency indices and nonspecificities of the evidential partitions generated for the 14 datasets. These graphs allow us to visualize the dominance relations between evidential partitions. For instance, in Figure 23d related to the Heart data, we can see that the evidential partitions generated by Bootclus and EVCLUS dominate that generated by EMMdd, and that the evidential partition generated by CCM is dominated by that generated by EVCLUS-bp; the evidential partitions generated by Bootclus, EVCLUS, NN-EVCLUS, NN-EVCLUS-bp, and CCM-bp are not dominated. From Figures 23-25, we can see that the evidential partitions generated by NN-EVCLUS and NN-EVCLUS-bp are generally not dominated by those obtained by any of the others algorithms, except Bootclus on the Wine (Figure 23a), Ecoli (Figure 23c), Seeds (Figure 23e), and Leaves5p1 (Figure 25c) datasets. In contrast, NN-EVCLUS dominates Bootclus on the Letters4p1 data (Figure 23f). These results confirm the very good performance of NN-EVCLUS, which is only outperformed by Bootclus on a minority of datasets.

Computing time. Computing time is an important issue when clustering large datasets. It is not so easy to measure intrinsically because it depends on the implementation of the algorithms. Table 4

Table 3: ARI values for the 11 methods on the 14 attribute datasets. The best value for each dataset is printed in bold, and the values within 5% of the best value are underlined. (NN-EV and NN-EV-bp stand, respectively, for NN-EVCLUS and NN-EVCLUS-bp).

Dataset	EVCLUS	NN-EV	NN-EV-bp	ECM	ECM-bp	CCM	CCM-bp	sECMdd	wECMdd	BPEC	Bootclus
Wine	0.91	0.91	**0.93**	0.85	0.85	0.45	0.45	0.24	0.39	0.80	**0.93**
Iris	0.77	0.77	0.70	0.62	0.62	0.63	0.63	0.54	0.66	0.64	**0.90**
Ecoli	0.80	0.80	**0.82**	0.79	0.79	0.70	0.70	0.11	0.64	0.78	0.60
Heart	0.41	**0.42**	0.41	0.38	0.38	0.37	0.40	-0.0058	0.39	0.22	0.27
Seeds	**0.81**	0.80	0.75	0.73	0.73	0.79	0.79	0.45	0.69	0.72	0.73
Letters4p1	0.50	**0.52**	·	0.050	·	0.089	·	0.075	0.099	·	0.10
Glass	0.35	**0.36**	·	0.31	·	0.21	·	0.20	0.26	·	0.18
Segment	0.51	0.54	·	0.50	·	0.36	·	0.44	0.23	·	**0.61**
S2	0.88	0.81	0.93	0.93	0.93	0.72	**0.94**	·	·	0.93	·
S4	0.63	0.55	0.64	0.63	0.63	0.52	0.63	·	·	**0.65**	·
D31	0.91	0.69	0.94	0.86	**0.95**	0.48	**0.95**	·	·	0.94	·
Mice	0.47	0.69	**0.80**	0.60	0.60	0.66	0.70	-0.02	0.58	0.75	0.77
DryBean	0.62	0.63	**0.75**	0.73	0.66	0.51	0.60	0.47	0.28	0.67	0.67
Leaves5p1	0.60	**0.64**	·	0.42	·	0.25	·	0.20	0.25	·	0.62

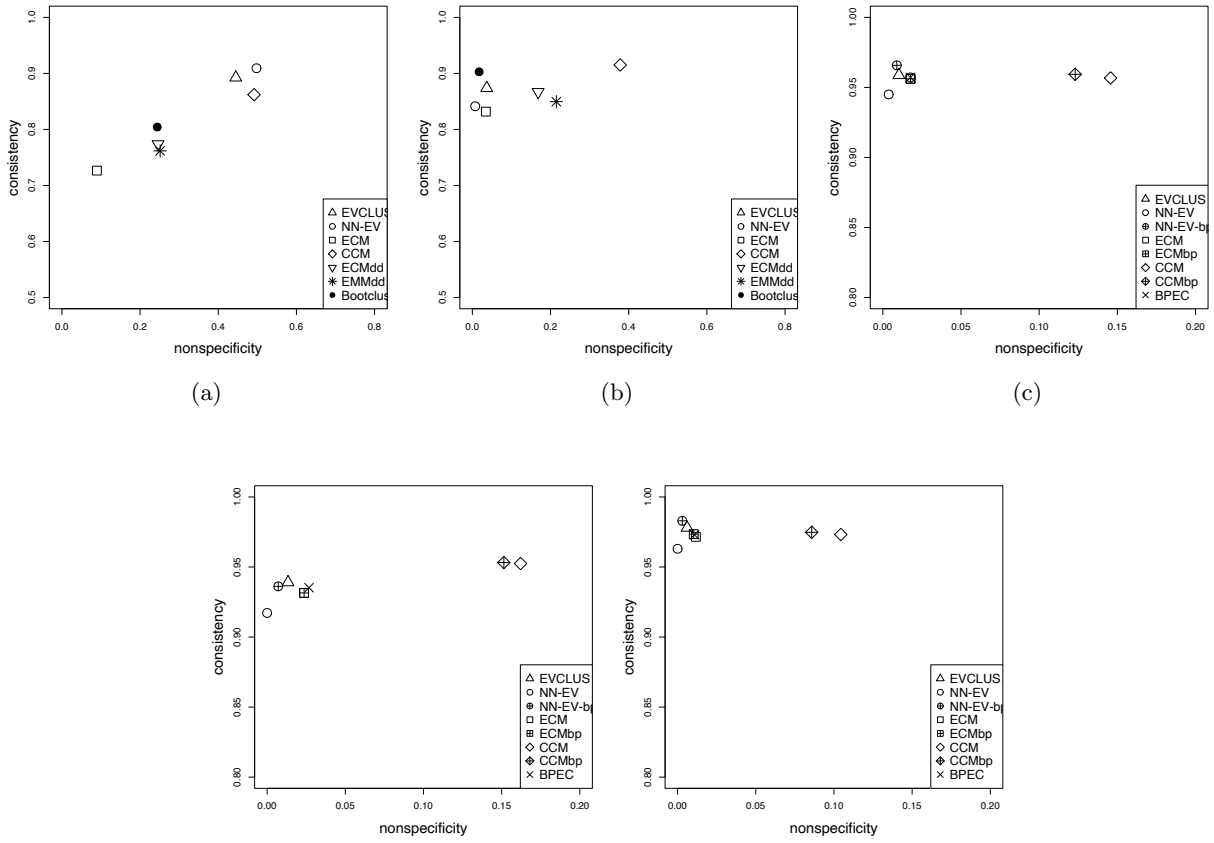


Figure 24: Consistency index (vertical axis) vs. nonspecificity (horizontal axis) for the Glass (a), Segment (b), S2 (c), S4 (d) and D31 (e) datasets.

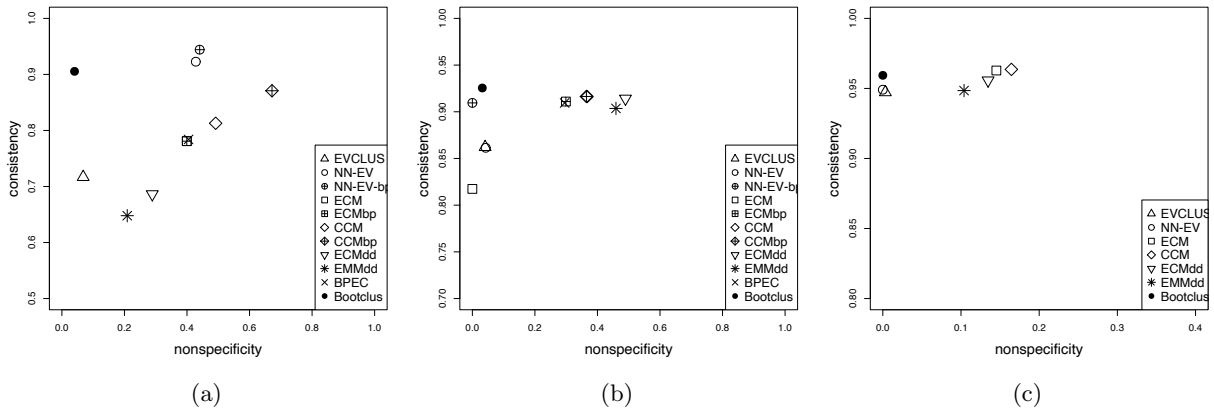


Figure 25: Consistency index (vertical axis) vs. nonspecificity (horizontal axis) for the Mice (a), DryBean (b), and Leaves5p1 (c) datasets.

Table 4: Means and standard deviations (in parentheses) of CPU times (in seconds) over five runs of EVCLUS, NN-EVCLUS, ECM and CCM applied to the S4 and D31 datasets (see implementation details in the text).

	EVCLUS	NN-EVCLUS	ECM	CCM
S4	55.75 (11.2)	430.02 (21.5)	10.51 (3.3)	248.09 (64.7)
D31	94.88 (19.8)	400.60 (82.7)	46.08 (10.6)	1728.82 (430.5)

reports the mean and standard deviations of the CPU times (in seconds) over five runs of EVCLUS, NN-EVCLUS, ECM and CCM applied to two of the largest datasets studied in this section: S4 and D31. The algorithms were coded in R and executed on a 2019 16" MacBook Pro with a 2.4 GHz 8-core Intel i9 processor. EVCLUS was run with $p = 500$ and with the empty set, the singletons and Ω as focal sets. For NN-EVCLUS, we used the same focal sets as with EVCLUS and 30 hidden units; the neural network was randomly initialized and trained using the RMSprop algorithm with $s = 30$ mini-batches and coefficients $\epsilon = 0.001$, $\rho = 0.9$ and $\delta = 10^{-8}$. For ECM, we used the two-step procedure recalled above (the model was first trained with the empty set and singletons, and was then re-trained after including selected pairs of clusters as focal sets). For CCM, we used the empty set, the singletons, the pairs and Ω as focal sets, but the coefficient T_c limiting the cardinality of the focal set was set to 2. We note that we included neither sECMdd nor wECMdd in this comparison as these algorithms require to store the whole dissimilarity matrix and they are extremely slow when applied to dataset of more than 1000 objects. We can see that NN-EVCLUS consumes significantly more time than EVCLUS and, to an even greater extent, ECM. The CCM algorithm was very slow on the D31 dataset because our implementation requires to use all pairs of clusters as focal sets. We can remark that the execution of NN-EVCLUS could be dramatically accelerated by running the code on GPU, which is left for further development.

4.2. Unsupervised clustering of dissimilarity data

Whereas the EVCLUS algorithm was initially introduced for clustering dissimilarity data [12, 13], it may seem that this possibility is lost with NN-EVCLUS, which uses attributes as input, in addition to a distance of dissimilarity matrix. However, it is still possible to cluster dissimilarity data with NN-EVCLUS by using dissimilarities as attributes. More precisely, let $\boldsymbol{D} = (\delta_{ij})$ be the $n \times n$ dissimilarity matrix. Each object i can be described by the n-dimensional vector $\boldsymbol{\delta}_i = (\delta_{i,1}, \ldots, \delta_{i,n})$ of its distances to the n objects (including itself), which corresponds to a row of matrix $\boldsymbol{D}$ and can be regarded as a vector of n attributes. To reduce the dimensionality of the representation, we can apply Principal Component Analysis (PCA) to these vectors and project the data on the subspace spanned by the p first principal components, resulting in the description of each object i by a p-dimensional attribute vector $\boldsymbol{x}_i$. We note that the attributes of any new object can be obtained from its dissimilarities to the n objects in the learning set by multiplying the centered vector of dissimilarities by the projection matrix.

Datasets. To study the application of NN-EVCLUS to nonmetric dissimilarity data, we considered four datasets:

1. The Protein dataset[5] [23, 20, 12] consists of a dissimilarity matrix derived from the structural comparison of 213 protein sequences. Each of these proteins is known to belong to

[5]This dataset is part of the `evclust` R package [8].

one of four classes of globins: hemoglobin-α (HA), hemoglobin-β (HB), myoglobin (M) and heterogeneous globins (G).

2. The ChickenPieces dataset[6] is composed of dissimilarities between 446 binary images represents the silhouettes of five parts of chickens. There are thus $c = 5$ clusters. The dataset is composed of 44 dissimilarity matrices corresponding to different ways of computing the dissimilarities. As in [1], we used matrix `chickenpieces-20-90` in our experiments. Since the data are slightly asymmetric, we computed a new matrix $\boldsymbol{D} = (\delta_{ij})$ by the transformation $\delta_{ij} \leftarrow (\delta_{ij} + \delta_{ji})/2$.
3. The Zongker dataset contains similarities between 2000 handwritten digits in 10 classes, based on deformable template matching. The dissimilarity measure is the result of an iterative optimization of the non-linear deformation of the grid [26]. Again, we made the dissimilarity matrix symmetric by the transformation $\delta_{ij} \leftarrow (\delta_{ij} + \delta_{ji})/2$.
4. The Gestures dataset consists of the dissimilarities computed from a set of gestures in a sign-language study [33]. They were measured by two video cameras observing the positions the two hands in 75 repetitions of creating 20 different signs. There are thus 1500 objects grouped in 20 clusters. The dissimilarities were computed by a dynamic time warping procedure.

Algorithms. As alternative evidential relational clustering algorithms, we considered EVCLUS [13], RECM [37], sECMdd and wECMdd [61]. We used the same focal sets for EVCLUS, NN-EVCLUS and RECM (the empty set, the singletons and Ω). For sECMdd and wECMdd, we used the empty set, the singletons, the pairs and Ω. The other parameter values for each of these algorithms are summarized in Table 5. For RECM, sECMdd and wECMdd, we used the default settings recommended by the authors [37, 61]. The batch version of NN-EVCLUS was run for the Protein dataset, and the mini-batch version (with 10 mini-batches and the RMSprop algorithm) was applied to the three other datasets. Each algorithm was run five times, and the best solution in terms of the loss or objective function was retained.

Results. The performances of the five methods in terms of ARI are shown in Table 6. As we can see, EVCLUS and NN-EVCLUS outperform the other methods for the four datasets, NN-EVCLUS reaching slightly higher values of ARI for the ChickenPieces, Zongker and Gestures datasets. As before, we also display the nonspecificity and consistency indices of the obtained evidential partitions for the four datasets in Figure 26. We can see that sECMdd and wECMdd produce less specific evidential partitions, due to the selection of pairs as focal sets. The evidential partitions produced by EVCLUS and NN-EVCLUS strictly dominate those obtained by ECMdd and wECMdd for the Protein dataset (Figure 26a), and the one produced by RECM for the ChichenPieces dataset (Figure 26b). The evidential partitions produced by NN-EVCLUS are always nondominated.

Computing time. The mean and standard deviations of the CPU times (in seconds) over five runs of EVCLUS, NN-EVCLUS, RECM, sECMdd and wECMdd applied to the Protein, ChickenPieces, Zongker and Gestures datasets are reported in Table 7. Again, the algorithms were coded in R and executed on a 2019 16" MacBook Pro with a 2.4 GHz 8-core Intel i9 processor. These times are only indicative because they obviously depend on implementation. The settings were those described in Table 5. The RECM procedure is the fastest of all four algorithms, but it performs poorly on the fours datasets considered in the eperiment. In contrast, the good performances of

[6] The ChickenPieces, Zongker and Gestures datasets are available at `http://prtools.org/disdatasets`.

Table 5: Parameter values for the five methods applied the three dissimilarity datasets. The notation $\delta_{(\alpha)}$ stands for the α-quantile of dissimilarities.

	EVCLUS	NN-EVCLUS	RECM	sECMdd	wECMdd
Protein	$\delta_0 = \max \delta_{ij}$	$p = 3$, $n_H = 20$, $\delta_0 = \max \delta_{ij}$	$\alpha = 1$, $\beta = 1.5$, $\delta^2 = \max \delta_{ij}$	$\alpha = 2$, $\beta = 2$, $\eta = 1$, $\gamma = 1$, $\delta = \max \delta_{ij}$	$\alpha = 2$, $\beta = 2$, $\xi = 5$, $\psi = 2$, $\delta = \max \delta_{ij}$
ChickenPieces	$\delta_0 = \delta_{(0.2)}$	$p = 5$, $n_H = 30$, $\delta_0 = \delta_{(0.2)}$	$\alpha = 1$, $\beta = 1.5$, $\delta^2 = \delta_{(0.95)}$	$\alpha = 2$, $\beta = 2$, $\eta = 1$, $\gamma = 1$, $\delta = \delta_{(0.95)}$	$\alpha = 2$, $\beta = 2$, $\xi = 5$, $\psi = 2$, $\delta = \delta_{(0.95)}$
Zongker	$\delta_0 = \delta_{(0.3)}$	$p = 20$, $n_H = 50$, $\delta_0 = \delta_{(0.3)}$	$\alpha = 1$, $\beta = 1.5$, $\delta^2 = \delta_{(0.95)}$	$\alpha = 2$, $\beta = 2$, $\eta = 1$, $\gamma = 1$, $\delta = \delta_{(0.95)}$	$\alpha = 2$, $\beta = 2$, $\xi = 5$, $\psi = 2$, $\delta = \delta_{(0.95)}$
Gestures	$\delta_0 = \delta_{(0.2)}$	$p = 20$, $n_H = 50$, $\delta_0 = \delta_{(0.2)}$	$\alpha = 1$, $\beta = 1.5$, $\delta^2 = \delta_{(0.95)}$	$\alpha = 2$, $\beta = 2$, $\eta = 1$, $\gamma = 1$, $\delta = \delta_{(0.95)}$	$\alpha = 2$, $\beta = 2$, $\xi = 5$, $\psi = 2$, $\delta = \delta_{(0.95)}$

Table 6: ARI values for the five methods on the four dissimilarity datasets. The best value for each dataset is printed in bold, and the values within 5% of the best value are underlined.

	EVCLUS	NN-EVCLUS	RECM	sECMdd	wECMdd
Protein	**0.989**	**0.989**	0.863	0.402	0.246
ChickenPieces	<u>0.308</u>	**0.315**	0.251	0.073	0.203
Zongker	<u>0.791</u>	**0.803**	0.217	0.053	0.053
Gestures	<u>0.709</u>	**0.710**	0.096	0.183	0.095

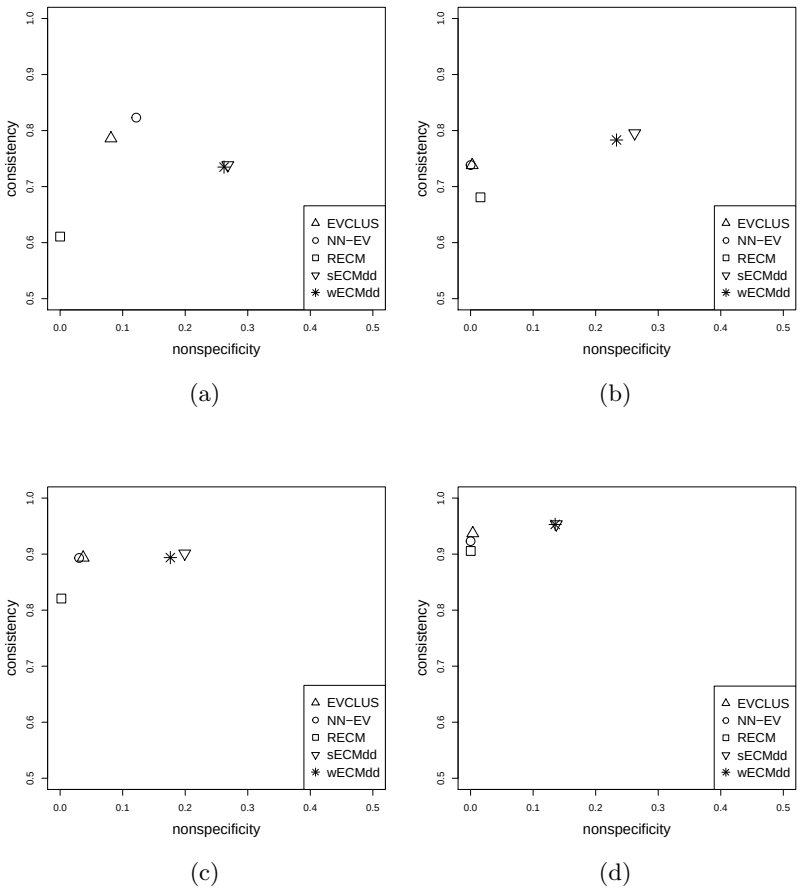


Figure 26: Consistency index (vertical axis) vs. nonspecificity (horizontal axis) for the Protein (a), ChichenPieces (b), Zongker (c) and Gestures (d) datasets.

Table 7: Means and standard deviations (in parentheses) of CPU times (in seconds) over five runs of EVCLUS, NN-EVCLUS, RECM, sECMdd and wECMdd applied to the Protein, ChickenPieces, Zongker and Gestures datasets (see implementation details in the text).

	EVCLUS	NN-EVCLUS	RECM	sECMdd	wECMdd
Protein	2.48 (0.2)	36.26 (3.4)	0.11 (0.02)	0.30 (0.04)	0.37 (0.07)
ChickenPieces	3.16 (0.3)	16.72 (0.1)	0.18 (0.02)	1.30 (0.2)	1.07 (0.05)
Zongker	15.53 (4.9)	251.54 (5.2)	13.16 (0.3)	47.74 (0.8)	25.06 (0.5)
Gestures	63.61 (21.4)	258.51 (4.7)	5.12 (0.2)	134.60 (5.5)	82.95 (1.7)

NN-EVCLUS come at the price of a higher computing time. However, as already emphasized, the NN-EVCLUS lends itself to parallel implementation, which would speed it up by a potentially large factor.

Prediction. Whereas EVCLUS and NN-EVCLUS yield similar results on these datasets, a distinctive advantage of NN-EVCLUS is that it makes it possible to predict the cluster membership of of new objects, without recomputing the evidential partition for the extended dataset. To demonstrate this possibility, we randomly split the Zongker dataset into two subsets of 1000 objects. We computed the attribute vectors by PCA for the first set of objects as explained above (with $p = 20$), and we trained NN-EVCLUS using the dissimilarity matrix for this first set. We then computed the attributes for the other 1000 objects and we computed the evidential partition by propagating the attribute values through the NN. The whole process was repeated 10 times. The average training and test ARI values were, respectively, 0.73 (standard deviation: 0.03) and 0.72 (standard deviation: 0.05). These results show that the relation between attributes and mass functions can be successfully learnt by NN-EVCLUS and generalized to test data, making it possible to predict the cluster membership of new objects.

4.3. Constrained clustering

Finally, we also compared the performance of NN-EVCLUS for exploiting pairwise constraints (as described in Section 3.3) to those of alternative constrained evidential clustering methods, namely: CEVCLUS [32] and CECM [1]. We considered the attribute dataset (Glass) and the dissimilarity dataset (ChickenPieces) with the lowest ARI values in their category (see, respectively, Tables 3 and 6). We also included the Iris dataset in the analysis as it is an example of a dataset with nonsperical clusters, for which pairwise constraints can significantly improve the clustering results.

Each of the three clustering algorithms was used with and without metric adaptation through PCCA [38]. For the Glass and Iris data, we extracted, respectively, five and three features using PCCA and we computed the Euclidean matrix in the feature space. The distance matrix was used by CEVCLUS and the features by CECM; NN-EVCLUS used both. For the ChickenPieces, we first extracted five features from distances using PCA as explained in Section 4.2, and we computed five new features using PCCA. For CEVCLUS and NN-EVCLUS, parameter ξ in (17) was set to 0.5. For CECM, parameter ξ controling the balance between the constraints and the objective function was also set to 0.5. The other parameters were set as in the previous experiments reported in Sections 4.1 and 4.2.

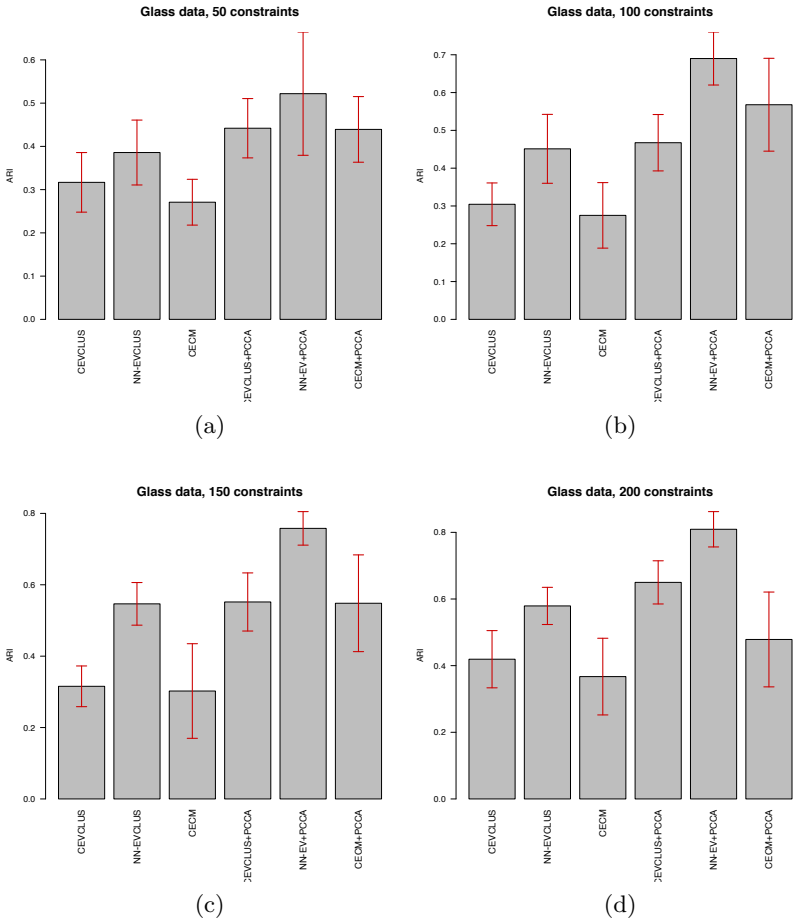


Figure 27: Mean ARI values (over 10 random draws) for the Glass dataset with 50 (a), 100 (b), 150 (c) and 200 (d) constraints. The methods are, from left to right: CEVCLUS, NN-EVCLUS, CECM, and the same methods combined with PCCA.The error bars extend to one standard deviation around the mean.

Pairwise constraints were generated randomly from the set of object pairs. For each number of constraints, we drew 10 different sets. The average ARI values for the three datasets are reported with the standard deviations in Figures 27-29, and the CPU times with 200 constraints are shown in Table 8. We can see that, without PCCA, NN-EVCLUS outperformed both CEVCLUS and CECM for the three datasets. PCCA improved the performances of the three clustering methods. With distances computed in feature space of PCCA, NN-EVCLUS still yielded strictly better results for the Glass and ChickenPieces datasets as shown, respectively, in Figures 27 and 29, and it yielded similar results as CEVCLUS for the Iris dataset (Figure 28). NN-EVCLUS has the highest computing time of the three methods, but it because slightly faster when PCCA is used, as the learning task because simpler. Overall, the combination of NN-EVCLUS and PCCA consistently provided the best results for the three datasets.

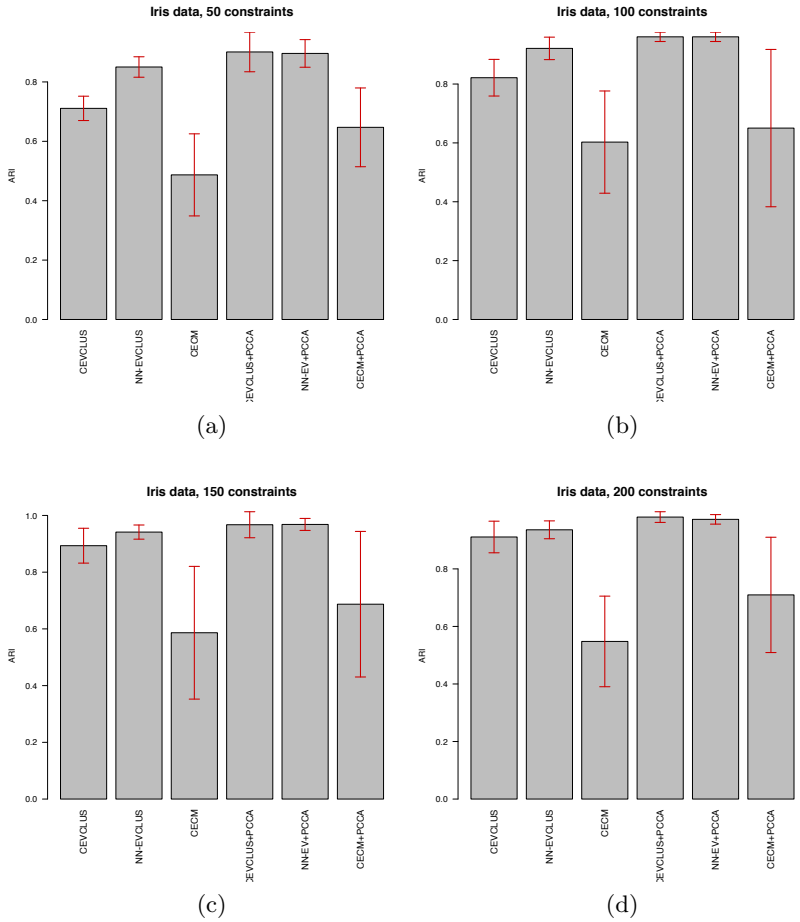


Figure 28: Mean ARI values (over 10 random draws) for the Iris dataset with 50 (a), 100 (b), 150 (c) and 200 (d) constraints. The methods are, from left to right: CEVCLUS, NN-EVCLUS, CECM, and the same methods combined with PCCA. The error bars extend to one standard deviation around the mean.

Table 8: Means and standard deviations (in parentheses) of CPU times (in seconds) over five runs of EVCLUS, NN-EVCLUS, CECM, and the same methods combined with PCCA, applied to the Glass, Iris and ChickenPieces datasets with 200 constraints.

	EVCLUS	NN-EVCLUS	CECM	EVCLUS +PCCA	NN-EVCLUS +PCCA	CECM +PCCA
Glass	18.78 (2.1)	76.22 (12.5)	35.27 (24.1)	31.51 (13.7)	53.74 (15.9)	31.07 (12.4)
Iris	1.55 (0.11)	30.05 (3.17)	4.78 (2.86)	1.21 (0.07)	23.59 (4.58)	2.08 (1.02)
Chicken	6.80 (0.5)	197.74 (23.6)	52.85 (44.0)	3.99 (0.6)	127.18 (21.1)	32.13 (20.5)

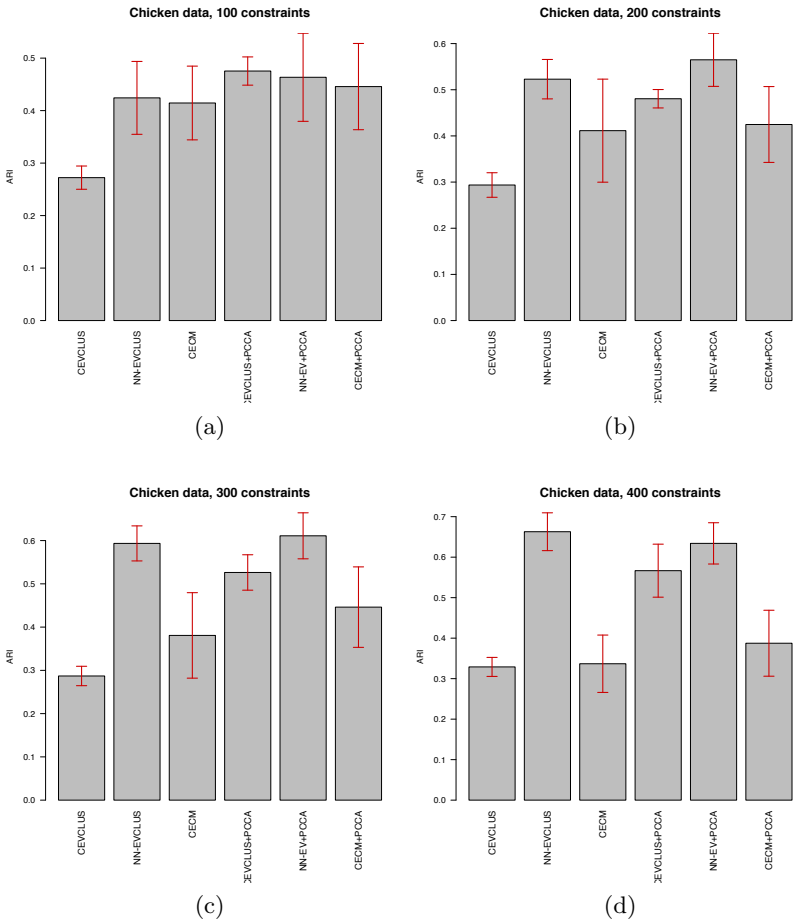


Figure 29: Mean ARI values (over 10 random draws) for the ChickenPieces dataset with 100 (a), 200 (b), 300 (c) and 400 (d) constraints. The methods are, from left to right: CEVCLUS, NN-EVCLUS, CECM, and the same methods combined with PCCA. The error bars extend to one standard deviation around the mean.

5. Conclusions

A new neural network-based evidential clustering algorithm, called NN-EVCLUS, has been introduced. This algorithm learns a mapping from attribute vectors to mass functions on a frame Ω of c clusters, in such a way that more similar inputs are mapped to mass functions with a lower degree of conflict. It, thus, requires two inputs: a set of attribute vectors and a dissimilarity matrix. In the case of attribute data, dissimilarities are typically computed as distances in the attribute space. In the case of proximity data, attributes can be computed by performing PCA on the matrix of dissimilarities. When side information is provided in the form of pairwise constraints or labeled data, feature extraction methods such as PCCA or FDA can be used to learn a metric in such a way that objects that are known to belong to different clusters become further apart, while objects in a given cluster are as similar as possible.

The neural network has a standard multilayer structure but a specific loss function that measures the discrepancy between dissimilarities and degrees of conflict for all or some pairs of objects. Additional error terms can be added to the loss function to account for pairwise constraints or labeled data. The network can be trained in batch mode or using minibatch stochastic gradient descent to handle very large datasets. It can be paired with a one-class SVM to make the method robust to outliers and allow for novelty detection. As opposed to EVCLUS, NN-EVCLUS learns a compact representation of the data in the form of connection weights, which makes it able to generalize beyond the learning set and compute an evidential partition for new data without retraining.

NN-EVCLUS has been compared to alternative evidential clustering algorithms on a range of clustering tasks with both attribute and dissimilarity data. It was shown to outperform other methods for a majority of datasets, by a relatively small margin for EVCLUS and by a larger margin for other algorithms (including ECM, CCM, sECMdd, wECMdd for attribute data, and RECM, sECMdd, wECMdd for dissimilarity data). NN-EVCLUS was only significantly outperformed by Bootclus on a few attribute datasets with elliptical clusters, for which a Gaussian mixture model is a good fit. On constrained clustering tasks, NN-EVCLUS was shown to outperform CECM and CEVCLUS. While metric adaptation using PCCA improved the performances of all methods, the combination of PCCA and EVCLUS yielded the best results overall.

While we used a standard multilayer perception architecture in this work, more complex architectures such as convolutional neural networks could be used with NN-EVCLUS to cluster data with a grid-like topology such as time series, images or videos. Also, training time could be drastically reduced by implementing the learning algorithm on GPUs. These ideas are left for further research.

Acknowledgment

This research was supported by the Labex MS2T, which was funded through the program "Investments for the future" by the National Agency for Research (reference ANR-11-IDEX-0004-02).

References

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

[2] V. Antoine, B. Quost, M.-H. Masson, and T. Denoeux. CEVCLUS: evidential clustering with instance-level constraints for relational data. *Soft Computing*, 18(7):1321–1335, 2014.
[3] J. C. Bezdek, J. Keller, R. Krishnapuram, and N. R. Pal. *Fuzzy models and algorithms for pattern recognition and image processing*. Kluwer Academic Publishers, Boston, 1999.
[4] I. Borg and P. Groenen. *Modern multidimensional scaling*. Springer, New-York, 1997.
[5] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "Siamese" time delay neural network. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 737–744. Morgan Kaufmann, 1994.
[6] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, Ma, 2006.
[7] T. Denœux. Calibrated model-based evidential clustering using bootstrapping. *Information Sciences*, 528:17–45, 2020.
[8] T. Denœux. *evclust: Evidential Clustering*, 2021. R package version 2.0.0. URL: `https://CRAN.R-project.org/package=evclust`.
[9] T. Denœux, D. Dubois, and H. Prade. Representations of uncertainty in artificial intelligence: Beyond probability and possibility. In P. Marquis, O. Papini, and H. Prade, editors, *A Guided Tour of Artificial Intelligence Research*, volume 1, chapter 4, pages 119–150. Springer Verlag, 2020.
[10] T. Denoeux and O. Kanjanatarakul. 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*, pages 157–164, Rome, Italy, September 2016. Springer-Verlag.
[11] T. Denoeux, S. Li, and S. Sriboonchitta. Evaluating and comparing soft partitions: an approach based on Dempster-Shafer theory. *IEEE Transactions on Fuzzy Systems*, 26(3):1231–1244, 2018.
[12] T. Denœux and M.-H. Masson. EVCLUS: Evidential clustering of proximity data. *IEEE Trans. on Systems, Man and Cybernetics B*, 34(1):95–109, 2004.
[13] T. Denœux, S. Sriboonchitta, and O. Kanjanatarakul. Evidential clustering of large dissimilarity data. *Knowledge-based Systems*, 106:179–195, 2016.
[14] D. Dua and C. Graff. UCI machine learning repository, 2017. `http://archive.ics.uci.edu/ml`.
[15] P. D'Urso. Informational paradigm, management of uncertainty and theoretical formalisms in the clustering framework: A review. *Information Sciences*, 400—401:30–62, 2017.
[16] P. D'Urso and R. Massari. Fuzzy clustering of mixed data. *Information Sciences*, 505:513–534, 2019.
[17] A. Ferone and A. Maratea. Integrating rough set principles in the graded possibilistic clustering. *Information Sciences*, 477:148–160, 2019.
[18] P. Fränti and S. Sieranoja. K-means properties on six clustering benchmark datasets, 2018. http://cs.uef.fi/sipu/datasets/.
[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.
[20] T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Advances in Neural Information Processing Systems 11*, pages 438–444, Cambridge, MA, 1999. MIT Press.
[21] Y. He, Y. Wu, H. Qin, J. Z. Huang, and Y. Jin. Improved i-nice clustering algorithm based on density peaks mechanism. *Information Sciences*, 548:177–190, 2021.
[22] C. Higuera, K. J. Gardiner, and K. J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):1–28, 06 2015.
[23] T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.
[24] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–ñ218, 1985.
[25] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Englewood Cliffs, NJ., 1988.
[26] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1391, 1997.
[27] Jianchang Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 1995.
[28] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.
[29] L. Kaufman and P. J. Rousseeuw. *Finding groups in data*. Wiley, New-York, 1990.
[30] M. Koklu and I. A. Ozkan. Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture*, 174:105507, 2020.
[31] R. Krishnapuram and J. Keller. A possibilistic approach to clustering. *IEEE Trans. on Fuzzy Systems*, 1:98–111,

1993.
[32] F. Li, S. Li, and T. Denœux. k-CEVCLUS: Constrained evidential clustering of large dissimilarity data. *Knowledge-Based Systems*, 142:29–44, 2018.
[33] J. Lichtenauer, E. A. Hendriks, and M. J. T. Reinders. Sign language recognition by combining statistical DTW and independent classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:2040–2046, 2008.
[34] Z.-G. Liu, J. Dezert, G. Mercier, and Q. Pan. Belief c-means: An extension of fuzzy c-means algorithm in belief functions framework. *Pattern Recognition Letters*, 33(3):291–300, 2012.
[35] Z.-G. Liu, Q. Pan, J. Dezert, and G. Mercier. Credal c-means clustering method based on belief functions. *Knowledge-Based Systems*, 74(0):119–132, 2015.
[36] M.-H. Masson and T. Denoeux. ECM: an evidential version of the fuzzy c-means algorithm. *Pattern Recognition*, 41(4):1384–1397, 2008.
[37] M.-H. Masson and T. Denœux. RECM: relational evidential c-means algorithm. *Pattern Recognition Letters*, 30:1015–1026, 2009.
[38] A. Mignon and F. Jurie. PCCA: a new approach for distance learning from sparse pairwise constraints. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2666–2672, 2012.
[39] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek. A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems*, 13(4):517–530, 2005.
[40] G. Peters. Some refinements of rough k-means clustering. *Pattern Recognition*, 39(8):1481–1491, 2006.
[41] G. Peters. Rough clustering utilizing the principle of indifference. *Information Sciences*, 277:358 – 374, 2014.
[42] G. Peters, F. Crespo, P. Lingras, and R. Weber. Soft clustering: Fuzzy and rough approaches and their extensions and derivatives. *International Journal of Approximate Reasoning*, 54(2):307–322, 2013.
[43] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.
[44] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344:1492–1496, 2014.
[45] B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.
[46] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, Princeton, N.J., 1976.
[47] F. M. Silva and L. B. Almeida. Speeding up backpropagation. In R. Eckmiller, editor, *Advances neural computers*, pages 151–158, New-York, 1990. Elsevier-North-Holland.
[48] P. Smets. The combination of evidence in the Transferable Belief Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):447–458, 1990.
[49] Z. Su and T. Denœux. BPEC: Belief-peaks evidential clustering. *IEEE Transactions on Fuzzy Systems*, 27(1):111–123, 2019.
[50] M. Sugiyama. Local Fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of 23rd International Conference on Machine Learning*, page 905?912. ACM, 2016.
[51] C. J. ter Braak, Y. Kourmpetis, H. A. Kiers, and M. C. Bink. Approximating a similarity matrix by a latent class model: A reappraisal of additive fuzzy clustering. *Computational Statistics & Data Analysis*, 53(8):3183–3193, 2009.
[52] S. Ubukata, A. Notsu, and K. Honda. Objective function-based rough membership c-means clustering. *Information Sciences*, 548:479–496, 2021.
[53] A. R. Webb. Multidimensional scaling by iterative majorization using radial basis functions. *Pattern Recognition*, 28(5):753–759, 1995.
[54] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 521–528. MIT Press, 2003.
[55] R. Xu and D. Wunsch. *Clustering*. Wiley-IEEE Press, Hoboken, New Jersey, 2009.
[56] X. Xu, S. Ding, Y. Wang, L. Wang, and W. Jia. A fast density peaks clustering algorithm with sparse search. *Information Sciences*, 2020.
[57] J. Yang, Z. Jin, J.-Y. Yang, D. Zhang, and A. F. Frangi. Essence of kernel Fisher discriminant: KPCA plus LDA. *Pattern Recognition*, 37(10):2097–2100, 2004.
[58] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, pages 34–39, 2014.
[59] Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13(1):1–26, 2012.
[60] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.
[61] K. Zhou, A. Martin, Q. Pan, and Z. ga Liu. ECMdd: Evidential c-medoids clustering with multiple prototypes.

Pattern Recognition, 60:239–257, 2016.

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

Appendix A. Gradient of $\mathcal{L}_{ij}$ w.r.t. $\boldsymbol{\theta}$

We consider a neural network with one hidden layer as described in Section 3.1. The vector of parameters is $\boldsymbol{\theta} = (V, W, \beta_0, \beta_1)$. Let us first compute the derivatives of $\mathcal{L}_{ij}$ w.r.t the last-layer weights $W = (w_{qh})$. From (14) and (11), we have

$$\frac{\partial \mathcal{L}_{ij}}{\partial w_{qh}} = \frac{\partial \mathcal{L}_{ij}}{\partial \mu_{iq}} \frac{\partial \mu_{iq}}{\partial w_{qh}} + \frac{\partial \mathcal{L}_{ij}}{\partial \mu_{jq}} \frac{\partial \mu_{jq}}{\partial w_{qh}} = \Delta_{ijq} z_{ih} + \Delta'_{ijq} z_{jh}, \tag{A.1}$$

with

$$\Delta_{ijq} = \frac{\partial \mathcal{L}_{ij}}{\partial \mu_{iq}} = \frac{\partial \mathcal{L}_{ij}}{\partial \kappa_{ij}} \frac{\partial \kappa_{ij}}{\partial \mu_{iq}} = 2(\kappa_{ij} - \delta^*_{ij}) \sum_{r=1}^{f} \frac{\partial \kappa_{ij}}{\partial m^*_{ir}} \frac{\partial m^*_{ir}}{\partial m_{ir}} \frac{\partial m_{ir}}{\partial \mu_{iq}} \tag{A.2}$$

and

$$\Delta'_{ijq} = \frac{\partial \mathcal{L}_{ij}}{\partial \mu_{jq}} = \frac{\partial \mathcal{L}_{ij}}{\partial \kappa_{ij}} \frac{\partial \kappa_{ij}}{\partial \mu_{jq}} = 2(\kappa_{ij} - \delta^*_{ij}) \sum_{r=1}^{f} \frac{\partial \kappa_{ij}}{\partial m^*_{jr}} \frac{\partial m^*_{jr}}{\partial m_{jr}} \frac{\partial m_{jr}}{\partial \mu_{jq}}. \tag{A.3}$$

From (8), we get

$$\frac{\partial \kappa_{ij}}{\partial m^*_{ir}} = (\boldsymbol{C}\boldsymbol{m}^*_j)_r. \tag{A.4}$$

From (12),

$$\frac{\partial m^*_{ir}}{\partial m_{ir}} = \gamma_i,$$

and from (11),

$$\frac{\partial m_{ir}}{\partial \mu_{iq}} = \begin{cases} m_{ir}(1 - m_{ir}) & r = q \\ -m_{ir} m_{iq} & r \neq q. \end{cases} \tag{A.5}$$

Similarly,

$$\frac{\partial \kappa_{ij}}{\partial m^*_{jr}} = (\boldsymbol{C}\boldsymbol{m}^*_i)_r, \quad \frac{\partial m^*_{jr}}{\partial m_{jr}} = \gamma_j, \tag{A.6}$$

and

$$\frac{\partial m_{jr}}{\partial \mu_{iq}} = \begin{cases} m_{jr}(1 - m_{jr}) & r = q \\ -m_{jr} m_{jq} & r \neq q. \end{cases} \tag{A.7}$$

Now, from (10) and (11), the derivatives w.r.t the first-layer weights $V = (w_{hk})$ are

$$\frac{\partial \mathcal{L}_{ij}}{\partial v_{hk}} = \frac{\partial \mathcal{L}_{ij}}{\partial a_{ih}} \frac{\partial a_{ih}}{\partial v_{hk}} + \frac{\partial \mathcal{L}_{ij}}{\partial a_{jh}} \frac{\partial a_{jh}}{\partial v_{hk}} = \Delta_{ijh} x_{ki} + \Delta'_{ijh} x_{kj}, \tag{A.8}$$

with

$$\Delta_{ijh} = \frac{\partial \mathcal{L}_{ij}}{\partial a_{ih}} = \sum_{q=1}^{f} \frac{\partial \mathcal{L}_{ij}}{\partial \mu_{iq}} \frac{\partial \mu_{iq}}{\partial z_{ih}} \frac{\partial z_{ih}}{\partial a_{ih}} = I(z_{ih} > 0) \sum_{q=1}^{f} \Delta_{ijq} w_{qh} \tag{A.9}$$

and

$$\Delta'_{ijh} = \frac{\partial \mathcal{L}_{ij}}{\partial a_{jh}} = \sum_{q=1}^{f} \frac{\partial \mathcal{L}_{ij}}{\partial \mu_{jq}} \frac{\partial \mu_{jq}}{\partial z_{jh}} \frac{\partial z_{jh}}{\partial a_{jh}} = I(z_{ih} > 0) \sum_{q=1}^{f} \Delta'_{ijq} w_{qh}. \tag{A.10}$$

Finally, we now compute the derivatives of $\mathcal{L}_{ij}$ w.r.t. β_0 and β_1. We have

$$\frac{\partial \mathcal{L}_{ij}}{\partial \beta_k} = \frac{\partial \mathcal{L}_{ij}}{\partial \gamma_i} \frac{\partial \gamma_i}{\partial \beta_k} + \frac{\partial \mathcal{L}_{ij}}{\partial \gamma_j} \frac{\partial \gamma_j}{\partial \beta_k} \tag{A.11}$$

for $k \in \{0, 1\}$. From (13), the first term of the sum in the right-hand side of (A.11) can be computed as

$$\frac{\partial \mathcal{L}_{ij}}{\partial \gamma_i} = \frac{\partial \mathcal{L}_{ij}}{\partial \kappa_{ij}} \frac{\partial \kappa_{ij}}{\partial \gamma_i} = 2(\kappa_{ij} - \delta^*_{ij}) \sum_{r=1}^{f} \frac{\partial \kappa_{ij}}{\partial m^*_{ir}} \frac{\partial m^*_{ir}}{\partial \gamma_i},$$

with $\frac{\partial \kappa_{ij}}{\partial m^*_{ir}}$ given by (A.4),

$$\frac{\partial m^*_{ir}}{\partial \gamma_i} = \begin{cases} 1 - m_{ir} & \text{if } r = 1 \\ -m_{ir} & \text{otherwise,} \end{cases} \tag{A.12}$$

and

$$\frac{\partial \gamma_i}{\partial \beta_0} = \frac{\partial \gamma_i}{\partial \eta_i} \frac{\partial \eta_i}{\partial \beta_0} = \frac{1}{(1+\eta_i)^2} \cdot \frac{\exp(\beta_0 + \beta_1 f(\boldsymbol{x}_i))}{1 + \exp(\beta_0 + \beta_1 f(\boldsymbol{x}_i))} \tag{A.13a}$$

$$\frac{\partial \gamma_i}{\partial \beta_1} = \frac{\partial \gamma_i}{\partial \beta_0} f(\boldsymbol{x}_i). \tag{A.13b}$$

The first term of the sum in the right-hand side of (A.11) can be computed in the same way, by replacing i with j.

Appendix B. Gradient of $\mathcal{P}_{\mathsf{ML}}$ and $\mathcal{P}_{\mathsf{CL}}$ w.r.t. $\boldsymbol{\theta}$

We have

$$\mathcal{P}_{\mathsf{ML}} = \sum_{(i,j)\in\mathsf{ML}} \mathcal{P}_{ij} \quad \text{and} \quad \mathcal{P}_{\mathsf{CL}} = \sum_{(i,j)\in\mathsf{CL}} (2 - \mathcal{P}_{ij}),$$

with

$$\mathcal{P}_{ij} = \boldsymbol{m}^{*T}_i \boldsymbol{Q} \boldsymbol{m}^*_j.$$

We thus only need to compute the gradient of $\mathcal{P}_{ij}$. The derivatives w.r.t. the hidden-to-output weights are

$$\frac{\partial \mathcal{P}_{ij}}{\partial w_{qh}} = \frac{\partial \mathcal{P}_{ij}}{\partial \mu_{iq}} \frac{\partial \mu_{iq}}{\partial w_{qh}} + \frac{\partial \mathcal{P}_{ij}}{\partial \mu_{jq}} \frac{\partial \mu_{jq}}{\partial w_{qh}} = \nabla_{ijq} z_{ih} + \nabla'_{ijq} z_{jh}$$

with

$$\nabla_{ijq} = \frac{\partial \mathcal{P}_{ij}}{\partial \mu_{iq}} = \sum_{r=1}^{f} \frac{\partial \mathcal{P}_{ij}}{\partial m^*_{ir}} \underbrace{\frac{\partial m^*_{ir}}{\partial m_{ir}}}_{1-\gamma_i} \underbrace{\frac{\partial m_{ir}}{\partial \mu_{iq}}}_{\text{see (A.5)}}$$

and

$$\frac{\partial \mathcal{P}_{ij}}{\partial m_{ir}} = (\boldsymbol{Q} \boldsymbol{m}^*_j)_r.$$

Similarly,

$$\nabla'_{ijq} = \frac{\partial \mathcal{P}_{ij}}{\partial \mu_{jq}} = \sum_{r=1}^{f} \frac{\partial \mathcal{P}_{ij}}{\partial m^*_{jr}} \underbrace{\frac{\partial m^*_{jr}}{\partial m_{jr}}}_{1-\gamma_j} \underbrace{\frac{\partial m_{jr}}{\partial \mu_{jq}}}_{\text{see (A.7)}}$$

and

$$\frac{\partial \mathcal{P}_{ij}}{\partial m_{jr}} = (\boldsymbol{Q}\boldsymbol{m}^*_i)_r.$$

The derivatives w.r.t. the input-to-hidden weights are,

$$\frac{\partial \mathcal{P}_{ij}}{\partial v_{hk}} = \frac{\partial \mathcal{P}_{ij}}{\partial a_{ih}} \frac{\partial a_{ih}}{\partial v_{hk}} + \frac{\partial \mathcal{P}_{ij}}{\partial a_{jh}} \frac{\partial a_{jh}}{\partial v_{hk}} = \nabla_{ijh} x_{ki} + \nabla'_{ijh} x_{kj}$$

with

$$\nabla_{ijh} = \frac{\partial \mathcal{P}_{ij}}{\partial a_{ih}} = \sum_{q=1}^{f} \frac{\partial \mathcal{P}_{ij}}{\partial \mu_{iq}} \frac{\partial \mu_{iq}}{\partial z_{ih}} \frac{\partial z_{ih}}{\partial a_{ih}} = I(z_{ih} > 0) \sum_{q=1}^{f} \nabla_{ijq} w_{qh}$$

and

$$\nabla'_{ijh} = \frac{\partial \mathcal{P}_{ij}}{\partial a_{jh}} = \sum_{q=1}^{f} \frac{\partial \mathcal{P}_{ij}}{\partial \mu_{jq}} \frac{\partial \mu_{jq}}{\partial z_{jh}} \frac{\partial z_{jh}}{\partial a_{jh}} = I(z_{ih} > 0) \sum_{q=1}^{f} \nabla'_{ijq} w_{qh}.$$

Finally, we have

$$\frac{\partial \mathcal{P}_{ij}}{\partial \beta_k} = \frac{\partial \mathcal{P}_{ij}}{\partial \gamma_i} \frac{\partial \gamma_i}{\partial \beta_k} + \frac{\partial \mathcal{P}_{ij}}{\partial \gamma_j} \frac{\partial \gamma_j}{\partial \beta_k}$$

for $k \in \{0, 1\}$, where the derivatives of γ_i and γ_j are given by (A.13), and

$$\frac{\partial \mathcal{P}_{ij}}{\partial \gamma_i} = \sum_{r=1}^{f} \frac{\partial \mathcal{P}_{ij}}{\partial m^*_{ir}} \frac{\partial m^*_{ir}}{\partial \gamma_i},$$

$$\frac{\partial \mathcal{P}_{ij}}{\partial \gamma_j} = \sum_{r=1}^{f} \frac{\partial \mathcal{P}_{ij}}{\partial m^*_{jr}} \frac{\partial m^*_{jr}}{\partial \gamma_j},$$

with $\frac{\partial m^*_{ir}}{\partial \gamma_i}$ and $\frac{\partial m^*_{jr}}{\partial \gamma_j}$ given by (A.12).

Appendix C. Gradient of $\mathcal{P}_s$ w.r.t. $\boldsymbol{\theta}$

We have

$$\mathcal{P}_s = \frac{1}{n_s} \sum_{i \in \mathcal{I}_s} \mathcal{P}_i$$

with

$$\mathcal{P}_i = \sum_{l=1}^{c} (pl^*_{il} - y_{il})^2.$$

We thus only need to compute the gradient of $\mathcal{P}_i$. The derivatives w.r.t. to the hidden-to-output weights are

$$\frac{\partial \mathcal{P}_i}{\partial w_{qh}} = \frac{\partial \mathcal{P}_i}{\partial \mu_{iq}} \frac{\partial \mu_{iq}}{\partial w_{qh}} = \Delta_{iq} z_{ih},$$

with

$$\Delta_{iq} = \frac{\partial \mathcal{P}_i}{\partial \mu_{iq}} = \sum_{l=1}^{c} \frac{\partial \mathcal{P}_i}{\partial pl^*_{il}} \frac{\partial pl^*_{il}}{\partial \mu_{iq}} = 2(pl^*_{il} - y_{il}) \sum_{l=1}^{c} \frac{\partial pl^*_{il}}{\partial \mu_{iq}}$$

and

$$\frac{\partial pl^*_{il}}{\partial \mu_{iq}} = \sum_{r=1}^{f} \underbrace{\frac{\partial pl^*_{il}}{\partial m^*_{ir}}}_{I(\omega_l \in A_r)} \underbrace{\frac{\partial m^*_{ir}}{\partial m_{ir}}}_{1-\gamma_i} \underbrace{\frac{\partial m_{ir}}{\partial \mu_{iq}}}_{\text{See (A.5)}} .$$

The derivatives w.r.t the input-to-hidden weights are given by

$$\frac{\partial \mathcal{P}_i}{\partial v_{hk}} = \underbrace{\frac{\partial \mathcal{P}_i}{\partial a_{ih}}}_{\Delta_{ih}} \underbrace{\frac{\partial a_{ih}}{\partial v_{hk}}}_{x_{ki}},$$

with

$$\Delta_{ih} = \sum_{q=1}^{f} \frac{\partial \mathcal{P}_i}{\partial \mu_{iq}} \frac{\partial \mu_{iq}}{\partial z_{ih}} \frac{\partial z_{ih}}{\partial a_{ih}} = I(z_{ih} > 0) \sum_{q=1}^{f} \Delta_{iq} w_{qh}.$$

Finally, the derivatives w.r.t β_0 and β_1 can be computed as

$$\frac{\partial \mathcal{P}_i}{\partial \beta_k} = \frac{\partial \mathcal{P}_i}{\partial \gamma_i} \underbrace{\frac{\partial \gamma_i}{\partial \beta_k}}_{\text{See (A.13)}} ,$$

with

$$\frac{\partial \mathcal{P}_i}{\partial \gamma_i} = \sum_{l=1}^{c} \underbrace{\frac{\partial \mathcal{P}_i}{\partial pl^*_{il}}}_{2(pl^*_{il} - y_{il})} \frac{\partial pl^*_{il}}{\partial \gamma_i}$$

and

$$\frac{\partial pl^*_{il}}{\partial \gamma_i} = \sum_{r=1}^{f} \underbrace{\frac{\partial pl^*_{il}}{\partial m^*_{ir}}}_{I(\omega_l \in A_r)} \underbrace{\frac{\partial m^*_{ir}}{\partial \gamma_i}}_{\text{See (A.12)}} .$$