Package ‘evclust’

November 9, 2023

Type Package
Title Evidential Clustering
Version 2.0.3
Date 2023-11-9
Author Thierry Denoeux
Maintainer Thierry Denoeux <tdenoeux@utc.fr>
Description Various clustering algorithms that produce a credal partition, i.e., a set of Dempster-Shafer mass functions representing the membership of objects to clusters. The mass functions quantify the cluster-membership uncertainty of the objects. The algorithms are: Evidential c-Means, Relational Evidential c-Means, Constrained Evidential c-Means, Evidential Clustering, Constrained Evidential Clustering, Evidential K-nearest-neighbor-based Clustering, Bootstrap Model-Based Evidential Clustering, Belief Peak Evidential Clustering, Neural-Network-based Evidential Clustering.
Depends R (>= 3.6.0),
Imports FNN, R.utils, limSolve, Matrix, mclust, quadprog, plyr
License GPL-3
LazyData TRUE
RoxygenNote 7.2.3
Encoding UTF-8
VignetteBuilder utils
Suggests utils, kernlab, MASS
NeedsCompilation no
Repository CRAN
Date/Publication 2023-11-09 14:00:02 UTC

R topics documented:

  bananas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  bootclus . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
**bananas**

**Description**

`bananas` generates a dataset with two classes separated by a nonlinear boundary.

**Usage**

`bananas(n, r = 5, s = 1)`
bootclus

Arguments

n      Number of observations.
r      Radius of the two half circles (default: 5).
s      Standard deviation of noise (default 1).

Details

This function generates a dataset with two complex-shaped classes, useful to test some nonlinear or constrained clustering algorithms.

Value

A list with two attributes:

x      The (n,2) matrix of attributes.
y      The vector of class labels.

Author(s)

Feng Li.

References


See Also

kcevclus

Examples

data<-bananas(1000)
plot(data$x,pch=data$y,col=data$y)

bootclus  Generating a credal partition by bootstraping Gaussian Mixture Models

Description

bootclus generates a credal partition by bootstrapping Gaussian Mixture Models.
Usage

```r
bootclus(
  x,
  conf = 0.9,
  B = 500,
  param = list(G = NULL),
  type = "pairs",
  Omega = FALSE
)
```

Arguments

- `x` attribute matrix or data frame of size (n,p).
- `conf` confidence level (default: 0.90).
- `B` number of bootstrap samples (default=500)
- `param` list of arguments passed to function `Mclust` in addition to 'data'.
- `type` Type of focal sets ("simple": $\emptyset$, singletons and $\Omega$; “full”: all $2^c$ subsets of $\Omega$; "pairs": $\emptyset$, singletons, $\Omega$, and all or selected pairs). Argument passed to `makeF`.
- `Omega` Logical. If TRUE, $\Omega$ is a focal set. Default is FALSE. Argument passed to `makeF`.

Details

This function uses the `mclust` package to generate and bootstrap the mixture models.

Value

A list with the following components:

- `clus` An object of class 'Mclust' returned by `Mclust`.
- `Clus` An object of class 'credpart' providing the output credal partition.
- `CI` An array of dimension (2,n,n) containing the confidence intervals on pairwise probabilities.
- `BelPl` An array of dimension (2,n,n) containing the pairwise Bel-Pl intervals.
- `Time` A matrix of size (3,5) containing the computing time as returned by function `proctime` for (1) the parameter estimation and bootstrap, (2) the computation for the quantiles on pairwise probabilities, and (3) the computation of the credal partition.

References


See Also

cem, recm, cecm, kevclus.
## Examples

```r
## Example with the Faithful geyser data
## Not run:
data("faithful")
X<-faithful
param=list(G=3)
res.faithful<-bootclus(X,conf=0.90,B=100,param=param)
## Plot the results
plot(res.faithful$Clus,X)
## End(Not run)
```

### Description

*bpec* computes a credal partition from a matrix of attribute data using the Belief Peak Evidential Clustering (BPEC) algorithm.

### Usage

```r
bpec(
x, 
g, 
type = "full", 
pairs = NULL, 
Omega = TRUE, 
alpha = 1, 
beta = 2, 
delta = 10, 
epsi = 0.001, 
disp = TRUE, 
distance = 1, 
m0 = NULL
)
```

### Arguments

- **x**: input matrix of size n x d, where n is the number of objects and d the number of attributes.
- **g**: Matrix of size c x d of prototypes (the belief peaks).
- **type**: Type of focal sets ("simple": empty set, singletons and Omega; "full": all \(2^c\) subsets of \(\Omega\); "pairs": \(\emptyset\), singletons, \(\Omega\), and all or selected pairs).
- **pairs**: Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
Omega Logical. If TRUE (default), the whole frame is included (for types 'simple' and 'pairs').
alpha Exponent of the cardinality in the cost function.
beta Exponent of masses in the cost function.
delta Distance to the empty set.
epsi Minimum amount of improvement.
disp If TRUE (default), intermediate results are displayed.
distance Type of distance use: 0=Euclidean, 1=Mahalanobis.
m0 Initial credal partition. Should be a matrix with n rows and a number of columns equal to the number f of focal sets specified by 'type' and 'pairs'.

Details

BPEC is identical to ECM, except that the prototypes are computed from delta-Bel graph using function delta_Bel. The ECM algorithm is then run keeping the prototypes fixed. The distance to the prototypes can be the Euclidean distance or it can be an adaptive Mahalanobis distance as in the CECM algorithm.

Value

The credal partition (an object of class "credpart").

Author(s)

Thierry Denoeux.

References


See Also

ecm, cecm, delta_Bel

Examples

## Clustering of the Four-class dataset
## Not run:
data(fourclass)
x<-fourclass[,1:2]
y<-fourclass[,3]
DB<-delta_Bel(x,100,0.9)
plot(x,pch=".")
points(DB$g0,pch=3,col="red",cex=2)
clus<-bpec(x,DB$g0,type='pairs',delta=3,distance=1)
plot(clus,x,mfrow=c(2,2))

## End(Not run)
butterfly

**Butterfly dataset**

**Description**

A toy dataset used to illustrate fuzzy and evidential clustering algorithms. Also called the 'Diamond' dataset. Adapted from Windham (1985), with one outlier added.

**Usage**

```r
data(butterfly)
```

**Format**

A matrix with 12 rows and 2 column.

**References**


**Examples**

```r
data(butterfly)
plot(butterfly[,1],butterfly[,2],xlab=expression(x[1]),ylab=expression(x[2]))
```

---

**cecm**

**Constrained Evidential c-means algorithm**

**Description**

cecm computes a credal partition from a matrix of attribute data and pairwise constraints using the Constrained Evidential c-means (CECM) algorithm.

**Usage**

```r
ccecm(
  x,
  c,
  type = "full",
  pairs = NULL,
  ntrials = 1,
  ML,
  CL,
```

```r
type = "full",
  pairs = NULL,
  ntrials = 1,
  ML,
  CL,
```
g0 = NULL,
alpha = 1,
delta = 10,
x1 = 0.5,
distance = 0,
eps = 0.001,
disp = TRUE)

Arguments

x  input matrix of size n x d, where n is the number of objects and d the number of attributes.
c  Number of clusters.
type  Type of focal sets ("simple": empty set, singletons and Omega; "full": all 2^c subsets of Ω; "pairs": ∅, singletons, Ω, and all or selected pairs).
pairs  Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
ntrials  Number of runs of the optimization algorithm (set to 1 if g0 is supplied).
ML  Matrix nbML x 2 of must-link constraints. Each row of ML contains the indices of objects that belong to the same class.
CL  Matrix nbCL x 2 of cannot-link constraints. Each row of CL contains the indices of objects that belong to different classes.
g0  Initial prototypes, matrix of size c x d. If not supplied, the prototypes are initialized randomly.
alpha  Exponent of the cardinality in the cost function.
delta  Distance to the empty set.
x1  Tradeoff between the objective function Jecm and the constraints: Jcecm=(1-xi)Jecm + xi Jconst.
distance  Type of distance use: 0=Euclidean, 1=Mahalanobis.
eps  Minimum amount of improvement.
disp  If TRUE (default), intermediate results are displayed.

Details

CECM is a version of ECM allowing the user to specify pairwise constraints to guide the clustering process. Pairwise constraints are of two kinds: must-link constraints are pairs of objects that are known to belong to the same class, and cannot-link constraints are pairs of objects that are known to belong to different classes. CECM can also learn a metric for each cluster, like the Gustafson-Kessel algorithm in fuzzy clustering. At each iteration, the algorithm solves a quadratic programming problem using an interior ellipsoidal trust region and barrier function algorithm with dual solution updating technique in the standard QP form (Ye, 1992).

If initial prototypes g0 are provided, the number of trials is automatically set to 1.

Remark: Due to the use of the Matrix package, messages may be generated by R’s (S4) method dispatch mechanism. They are not error messages, and they can be ignored.
**Value**

The credal partition (an object of class "credpart").

**Author(s)**

Thierry Denoeux (from a MATLAB code written by Violaine Antoine).

**References**


**See Also**

`create_MLCL, makeF, extractMass, ecm, recm`

**Examples**

```r
## Generation of a two-class dataset
## Not run:
n<-30
x<-cbind(0.2*rnorm(n),rnorm(n))
y<-c(rep(1,n/2),rep(2,n/2))
x[(n/2+1):n,1]<-x[(n/2+1):n,1]+1
plot(x[,1],x[,2],asp=1,pch=y,col=y)
## Generation of 10 constraints
const<-create_MLCL(y,nbConst=10)
## Call of cecm
clus<-cecm(x=x,c=2,ML=const$M,CL=const$CL,delta=10)
plot(x[,1],x[,2],asp=1,pch=clus$y.pl,col=y)
## End(Not run)
```

---

**createD**

*Computation of a Euclidean distance matrix*

**Description**

`createD` constructs an n x k matrix of Euclidean distances from an n x p matrix of attribute data. For each object, the distances to k randomly selected objects are computed.

**Usage**

`createD(x, k)`
Arguments

- **x**: n x p data matrix.
- **k**: Number of distances. If missing, an n x n distance matrix is computed.

Value

A list with two elements:

- **D**: n x k distance matrix.
- **J**: n x k matrix of indices. D[i,j] is the Euclidean distance between x[i,] and x[J[i,j],].

See Also

kevclus

Examples

data(fourclass)
x<-as.matrix(fourclass[,1:2])
dist<-createD(x,k=10)
dim(dist$D)
dim(dist$J)

createPairs

Description

createPairs finds pairs of clusters that are mutual k nearest neighbors in a credal partition. The similarity between two clusters k and l is defined as \( \sum_{i=1}^{n} pl_{ik}pl_{il} \), where \( pl_{ik} \) is the plausibility of object i belonging to cluster k.

Usage

createPairs(clus, k = 1)

Arguments

- **clus**: An object of class credpart. It should contain at least two fields: clus$mass (the credal partition) and clus$pl.n (the normalized plausibilities). The focal sets of the credal partition must be the empty set, the singletons, and (optionally) the whole set of clusters.
- **k**: The number of neighbors.
Details

This function allows one to use evidential clustering when the number of clusters is large. A clustering algorithm is first run with a limited number of focal sets (the empty set, the singletons and, optionally, the whole frame). Then, the similarity between clusters is analysed to determine the pairs of neighboring (overlapping) clusters. The clustering algorithm is then run again, adding these pairs to the focal sets (see the example). The focal sets of the passed credal partition must be the empty set (first row), the singletons (next c rows) and, optionally, the whole frame (last row).

Value

A list with the following components:

- `pairs` A matrix with two columns and p rows, containing the p pairs of clusters. This matrix can be passed to `ecm`, `recm`, `cecm` or `kevclus`.
- `m0` A matrix of size (n,c+2+p), encoding the credal partition. The masses assigned to the pairs are null.
- `S` The c x c matrix of similarities between clusters.

References


See Also

`extractMass`, `ecm`, `recm`, `cecm`, `kevclus`.

Examples

```r
## Example with Four-class data
data("fourclass")
x<-fourclass[,1:2]
y<-fourclass[,3]
c=4
## Running k-EVCLUS with singletons
clus<-kevclus(x=x,k=100,c=c,type='simple')
## Plot the results
plot(clus,X=x,mfrow=c(2,2),ytrue=y)
## Creating the pairs of clusters
P<-createPairs(clus,k=2)
## Running k-EVCLUS again, with pairs of clusters
clus1<-kevclus(x=x,k=100,c=c,type='pairs',pairs=P$pairs,m0=P$m0)
## Plot the results
plot(clus1,X=x,mfrow=c(2,2),ytrue=y)
```
create_fuzzy_credpart

Creation of a "credpart" object from a fuzzy or possibilistic partition matrix

Description

create_fuzzy_credpart creates a "credpart" object from a fuzzy or possibilistic partition matrix.

Usage

create_fuzzy_credpart(U)

Arguments

U A fuzzy or possibilistic partition matrix of size n×c, where c is the number of clusters.

Value

An object of class "credpart".

References


See Also

extractMass, create_hard_credpart

Examples

```r
## Not run:
library(fclust)
U<-FKM(fourclass[,1:2],4)$U
clus<-create_fuzzy_credpart(U)
sweep(clus)
## End(Not run)
```
create_hard_credpart  

Creation of a "credpart" object from a vector of class labels

Description
create_hard_credpart creates a "credpart" object from a vector of class labels.

Usage
create_hard_credpart(y)

Arguments
y  A vector of class labels.

Value
An object of class "credpart".

References

See Also
extractMass, create_fuzzy_credpart

Examples
## Not run:
data(fourclass)
y<-kmeans(fourclass[,1:2],4)$cluster
clus<-create_hard_credpart(y)
summary(clus)

## End(Not run)
### create_MLCL

**Random generation of Must-Link and Cannot-Link constraints**

**Description**

`create_MLCL` randomly generates Must-Link (ML) and Cannot-Link (CL) constraints from a vector `y` of class labels.

**Usage**

```
create_MLCL(y, nbConst)
```

**Arguments**

- `y`: Vector of class labels.
- `nbConst`: Number of constraints.

**Value**

A list with two components:

- `ML`: Matrix of ML constraints. Each row corresponds to a constraint.
- `CL`: Matrix of CL constraints. Each row corresponds to a constraint.

**See Also**

`cecm`

**Examples**

```r
y<-sample(3,100,replace=TRUE)
const<-create_MLCL(y,nbConst=10)
const$ML
const$CL
```

### credal_RI

**Credal Rand indices**

**Description**

`credal_RI` computes generalizations of the Rand index to compare credal partitions, as defined in Denoeux et al (2018).

**Usage**

```
credal_RI(P1, P2, type = "c")
```
Arguments

P1  Relational representation of the first credal partition such as generated by function pairwise_mass
P2  Relational representation of the second credal partition such as generated by function pairwise_mass


type  "c" for degree of conflict (default), "j" for Jousselme’s distance and "b" for belief distance.

Details

In Denoeux et al. (2018), two generalizations of the Rand index for comparing credal partitions are defined: one is based on distances between mass function, the other one is based on distances. In the latter case, two distances are proposed: Jousselme’s distance and the L1 distance between belief functions. These three indices can be computed by function credal_RI.

Value

The credal Rand index

References


See Also

nonspecificity, pairwise_mass

Examples

```r
# Butterfly data
data(butterfly)
clus1<-kevclus(butterfly,c=2)
P1<-pairwise_mass(clus1)
clus2<-ecm(butterfly,c=2)
P2<-pairwise_mass(clus2)
RI1<-credal_RI(P1,P2,"c")
RI2<-credal_RI(P1,P2,"j")
RI3<-credal_RI(P1,P2,"b")
print(c(RI1,RI2,RI3))
```
delta_Bel  

Delta-Bel graph for Belief Peak Evidential Clustering (BPEC)

Description

delta_Bel computes the delta-Bel graph used to determine the prototypes in the Belief Peak Evidential Clustering (BPEC) algorithm. The user must manually specify the rectangles containing the prototypes (which are typically in the upper-right corner of the graph if the clusters are well-separated). These prototypes are then used by function bpec to compute a credal partition.

Usage

delta_Bel(x, K, q = 0.9)

Arguments

x  
input matrix of size n x d, where n is the number of objects and d the number of attributes.

K  
Number of neighbors to determine belief values

q  
Parameter of the algorithm, between 0 and 1 (default: 0.9).

Value

A list with three elements:

BelC  The belief values.
delta  The delta values.
g0  A c*d matrix containing the prototypes.
ii  List of indices of the belief peaks.

Author(s)

Thierry Denoeux.

References


See Also

bpec
ecm

Examples

```r
## Not run:
data(fourclass)
x<-fourclass[,1:2]
y<-fourclass[,3]
DB<-delta_Bel(x,100,0.9)
plot(x,pch=".")
points(DB$g0,pch=3,col="red",cex=2)
## End(Not run)
```

desc

Evidential c-means algorithm

des

ecm computes a credal partition from a matrix of attribute data using the Evidential c-means (ECM) algorithm.

Usage

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

Arguments

- **x**: input matrix of size n x d, where n is the number of objects and d the number of attributes.
- **c**: Number of clusters.
- **g0**: Initial prototypes, matrix of size c x d. If not supplied, the prototypes are initialized randomly.
- **type**: Type of focal sets ("simple": empty set, singletons and Omega; "full": all 2^c subsets of \( \Omega \); "pairs": \( \emptyset \), singletons, \( \Omega \), and all or selected pairs).
pairs  Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
Omega  Logical. If TRUE (default), the whole frame is included (for types 'simple' and 'pairs').
ntrials  Number of runs of the optimization algorithm (set to 1 if m0 is supplied).
alpha  Exponent of the cardinality in the cost function.
beta  Exponent of masses in the cost function.
delta  Distance to the empty set.
epsi  Minimum amount of improvement.
init  Initialization: "kmeans" (default) or "rand" (random).
disp  If TRUE (default), intermediate results are displayed.

Details
ECM is an evidential version algorithm of the Hard c-Means (HCM) and Fuzzy c-Means (FCM) algorithms. As in HCM and FCM, each cluster is represented by a prototype. However, in ECM, some sets of clusters are also represented by a prototype, which is defined as the center of mass of the prototypes in each individual cluster. The algorithm iteratively optimizes a cost function, with respect to the prototypes and to the credal partition. By default, each mass function in the credal partition has $2^c$ focal sets, where $c$ is the supplied number of clusters. We can also limit the number of focal sets to subsets of clusters with cardinalities 0, 1 and $c$ (recommended if $c>=10$), or to all or some selected pairs of clusters. If initial prototypes $g_0$ are provided, the number of trials is automatically set to 1.

Value
The credal partition (an object of class "credpart").

Author(s)
Thierry Denoeux (from a MATLAB code written by Marie-Helene Masson).

References

See Also
makeF, extractMass, recm, cecm, plot.credpart

Examples
## Clustering of the Four-class dataset
## Not run:
data(fourclass)
x<-fourclass[,1:2]
y<-fourclass[,3]
EkNNclus

EkNNclus algorithm

Description
EkNNclus computes hard and credal partitions from dissimilarity or attribute data using the EkNNclus algorithm.

Usage
EkNNclus(
  x = NULL,
  D,
  K,
  y0,
  ntrials = 1,
  q = 0.5,
  b = 1,
  disp = TRUE,
  tr = FALSE,
  eps = 1e-06
)

Arguments

- **x**: n x p data matrix (n instances, p attributes).
- **D**: n x n dissimilarity matrix (used only if x is not supplied).
- **K**: Number of neighbors.
- **y0**: Initial partition (vector of length n, with values in (1,2,...)).
- **ntrials**: Number of runs of the algorithm (the best solution is kept).
- **q**: Parameter in (0,1). Gamma is set to the inverse of the q-quantile of distances from the K nearest neighbors (same notation as in the paper).
- **b**: Exponent of distances, $\alpha_{ij} = \phi(d_{ij}^b)$.
- **disp**: If TRUE, intermediate results are displayed.
- **tr**: If TRUE, a trace of the cost function is returned.
- **eps**: Minimal distance between two vectors (distances smaller than eps are replaced by eps).
Details

The number of clusters is not specified. It is influenced by parameters K and q. (It is advised to
start with the default values.) For n not too large (say, until one thousand), y0 can be defined as the
vector (1,2,....,n). For larger values of n, it is advised to start with a random partition of c clusters,
c<n.

Value

The credal partition (an object of class "credpart"). In addition to the usual attributes, the output
credal partition has the following attributes:

trace  Trace of the algorithm (sequence of values of the cost function).

W    The weight matrix.

Author(s)

Thierry Denoeux.

References

T. Denoeux, O. Kanjanata-rakul and S. Sriboonchitta. EK-NNclus: a clustering procedure based on

Examples

## Clustering of the fourclass dataset
## Not run:
data(fourclass)
n<-nrow(fourclass)
N=2
clus<- EkNNclus(fourclass[,1:2],K=60,y0=(1:n),ntrials=N,q=0.9,b=2,disp=TRUE,tr=TRUE)
## Plot of the partition
plot(clus,X=fourclass[,1:2],ytrue=fourclass$y,Outliers=FALSE,plot_approx=FALSE)
## Plot of the cost function vs number of iteration
L<-vector(length=N)
for(i in 1:N) L[i]<-dim(clus$trace[clus$trace[,1]==i,])[1]
imax<-which.max(L)
plot(0:(L[imax]-1),-clus$trace[clus$trace[,1]==imax,3],type="l",lty=imax,
xlab="time steps",ylab="energy")
for(i in (1:N)) if(i != imax) lines(0:(L[i]-1),-clus$trace[clus$trace[,1]==i,3],
type="l",lty=i)
## End(Not run)
Various clustering algorithms that generate a credal partition, i.e., a set of mass functions. Mass functions quantify the cluster-membership uncertainty of the objects. The package consists in the following main functions, implementing different evidential clustering algorithms:

**ecm** Evidential c-means algorithm (Masson and Denoeux, 2008)
**recm** Relational Evidential c-means algorithm (Masson and Denoeux, 2009)
**kevclus** $k$-EVCLUS algorithm (Denoeux and Masson, 2004; Denoeux et al., 2016)
**EkNNclus** ES$k$-NNeclus algorithm (Denoeux et al., 2015)
**cecm** Constrained Evidential c-means algorithm (Antoine et al, 2012)
**kcevclus** Constrained evidential clustering (Antoine et al., 2014; Li et al., 2018)
**bpec** Belief peak evidential clustering (Su and Denoeux, 2019)
**nnevclus** Neural-network based evidential clustering (Denoeux, 2020a)
**nnevclus_mb** Neural-network based evidential clustering, minibatch version (Denoeux, 2020a)
**bootclus** Model-based evidential clustering using bootstrapping (Denoeux, 2020b)

References


**expandlink**  

*Expansion of must-link and cannot-link constraints*

**Description**

`expandlink` returns an expanded set of must-link and cannot-link constraints using the k nearest neighbors of each observation.

**Usage**

`expandlink(link, ind, distan)`

**Arguments**

- `link`: A list with two attributes: a matrix `ML` containing `nbML x 2` must-link constraints and a matrix `CL` containing `nbCL x 2` cannot-link constraints.
- `ind`: An `n*k` matrix containing the k nearest neighbor indices.
- `distan`: An `n*k` matrix containing the k nearest neighbor distances.

**Details**

Using the algorithm described in Li et al (2018), `expandlink` generates new must-link and cannot-link constraints from existing ones, using the k nearest neighbors of each observation. The extended constraint list can be used by constrained clustering algorithms such as `cecm` and `kcevclus`.

**Value**

A list with two attributes:

- `ML`: The new matrix of must-link constraints.
- `CL`: The new matrix of cannot-link constraints.

**Author(s)**

Feng Li and Thierry Denoeux.

**References**


**See Also**

`kcevclus, cecm, create_MLCL, bananas`
Examples

```r
## Not run:
data<-bananas(200)
link<-create_MLCL(data$y,10)
nml<-nrow(link$ML)
plot(data$x,col=data$y)
for(k in 1:nml) lines(data$x[link$ML[k,],1],data$x[link$ML[k,],2],lwd=2,col="red")
ncl<-nrow(link$CL)
for(k in 1:ncl) lines(data$x[link$CL[k,],1],data$x[link$CL[k,],2],lwd=2,col="blue")
library(FNN)
nn<-get.knn(data$x,5)
link1<-expandlink(link,ind=nn$nn.index,distan=nn$nn.dist)
nml<-nrow(link1$ML)
for(k in 1:nml) lines(data$x[link1$ML[k,],1],data$x[link1$ML[k,],2],lwd=1,lty=2,col="red")
ncl<-nrow(link1$CL)
for(k in 1:ncl) lines(data$x[link1$CL[k,],1],data$x[link1$CL[k,],2],lwd=1,lty=2,col="blue")
```

## End(Not run)

---

**extractMass**

* Creates an object of class "credpart"

---

**Description**

extractMass computes different outputs (hard, fuzzy, rough partitions, etc.) from a credal partition and creates an object of class "credpart".

**Usage**

```r
extractMass(
  mass,
  F,
  g = NULL,
  S = NULL,
  method,
  crit = NULL,
  Kmat = NULL,
  trace = NULL,
  D = NULL,
  W = NULL,
  J = NULL,
  param = NULL
)
```
Arguments

mass  A credal partition (a matrix of n rows and f columns, where n is the number of objects and f is the number of focal sets).
F     Matrix (f,c) of focal sets. If the empty set is a focal set, it must correspond to the first row of F.
g     A c x d matrix of prototypes.
S     A list of length f containing the matrices $S_j$ defining the metrics for each cluster and each group of cluster.
method The method used to construct the credal partition (a character string).
crit  The value of the optimized criterion (depends on the method used).
Kmat  The matrix of degrees of conflict. Same size as D (for method kevclus).
trace The trace of criterion values (for methods kevclus and EkNNclus).
D     The normalized dissimilarity matrix (for method kevclus).
W     The weight matrix (for method EkNNclus).
J     The matrix of indices (for method kevclus).
param A method-dependent list of parameters.

Details

This function collects varied information on a credal partition and stores it in an object of class "credpart". The lower and upper approximations of clusters define rough partitions. They can be computed in two ways: either from the set of clusters with maximum mass, or from the set of non dominated clusters. A cluster $\omega_k$ is non dominated if $pl(\omega_k) \geq bel(\omega_l)$ for all l different from k. Once a set of cluster $Y_i$ has been computed for each object, object i belongs to the lower approximation of cluster k if $Y_i = \omega_k$. It belongs to the upper approximation of cluster k if $\omega_k \in Y_i$. See Masson and Denoeux (2008) for more details, and Denoeux and Kanjanatarakul (2016) for the interval dominance rule. The function creates an object of class "credpart". There are three methods for this class: plot.credpart, summary.credpart and predict.credpart.

Value

An object of class "credpart" with the following components:

method  The method used to construct the credal partition (a character string).
F       Matrix of focal sets. The first row always corresponds to the empty set.
conf    Masses assigned to the empty set, vector of length n.
mass    Mass functions, matrix of size (n,f).
mass.n  Normalized mass functions, matrix of size (n,f-1).
g       The prototypes (if defined).
S       The matrices $S_j$ defining the metrics for each cluster and each group of cluster (if defined).
pl      Unnormalized plausibilities of the singletons, matrix of size (n,c).
pl.n    Normalized plausibilities of the singletons, matrix of size (n,c).
p       Probabilities derived from pl by the plausibility transformation, matrix of size (n,c).
extractMass

bel  Unnormalized beliefs of the singletons, matrix of size (n,c).
bel.n Normalized beliefs of the singletons, matrix of size (n,c).
y.pl Maximum plausibility clusters, vector of length n.
y.bel Maximum belief clusters, vector of length n.
betp Unnormalized pignistic probabilities of the singletons, matrix of size (n,c).
betp.n Normalized pignistic probabilities of the singletons, matrix of size (n,c).
Y  Sets of clusters with maximum mass, matrix of size (n,c).
outlier n-vector of 0’s and 1’s, indicating which objects are outliers. An outlier is an object such that the largest mass is assigned to the empty set.
lower.approx Lower approximations of clusters, a list of length c. Each element lower.approx[[i]] is a vector of object indices.
upper.approx Upper approximations of clusters, a list of length c. Each element upper.approx[[i]] is a vector of object indices.
Ynd  Sets of clusters selected by the interval dominance rule, matrix of size (n,c).
lower.approx.nd Lower approximations of clusters using the interval dominance rule, a list of length c. Each element lower.approx.nd[[i]] is a vector of objects.
upper.approx.nd Upper approximations of clusters using the interval dominance rule, a list of length c. Each element upper.approx.nd[[i]] is a vector of objects.
N  Average nonspecificity.
crit The value of the optimized criterion (depends on the method used).
Kmat The matrix of degrees of conflict. Same size as D (for method kevclus).
D  The normalized dissimilarity matrix (for method kevclus).
trace The trace of criterion values (for methods kevclus and EkNNclus).
W  The weight matrix (for method EkNNclus).
J  The matrix of indices (for method kevclus).
param A method-dependent list of parameters.

References


See Also

plot.credpart, summary.credpart
## Examples

```r
## Not run:
## Four-class data
data(fourclass)
x<-fourclass[,1:2]
y<-fourclass[,3]
D<-as.matrix(dist(x))^2
clus<-recm(D,c=4,delta=10,ntrials=1)
summary(clus)
plot(clus,X=x,mfrow=c(1,1),ytrue=y,Outliers=TRUE)
## End(Not run)
```

### fourclass

**Synthetic four-class dataset**

#### Description

A synthetic dataset with two attributes and four classes of 100 points each, generated from a multivariate t distribution with five degrees of freedom and centered, respectively, on [0;0], [0;4], [4;0] and [4;4].

#### Usage

```r
data(fourclass)
```

#### Format

A data frame with three variables: x1, x2 and y (the true class).

#### References


#### Examples

```r
data(fourclass)
plot(fourclass$x1,fourclass$x2,xlab=expression(x[1]),ylab=expression(x[2]),
col=fourclass$y,pch=fourclass$y)
```
harris

Harris gradient-based optimization algorithm

Description

The optimization algorithm implemented in harris is described on Silva & Almeida (1990) and summarized in Denoeux & Masson (2004). The four parameters are:

- **options[1]** Display parameter: 1 (default) displays some results.
- **options[2]** Maximum number of iterations (default: 100).
- **options[3]** Relative error for stopping criterion (default: 1e-4).
- **options[4]** Number of iterations between two displays.

Usage

```r
harris(fun, x, options = c(1, 100, 1e-04, 10), tr = FALSE, ...)
```

Arguments

- **fun** Function to be optimized. The function 'fun' should return a scalar function value 'fun' and a vector 'grad' containing the partial derivatives of fun at x.
- **x** Initial value (a vector).
- **options** Vector of parameters (see details).
- **tr** If TRUE, returns a trace of objective function vs CPU time
- **...** Additional parameters passed to fun

Value

A list with three attributes:

- **par** The minimizer of fun found.
- **value** The value of fun at par.
- **trace** The trace, a list with two attributes: 'time' and 'fct' (if tr==TRUE).

Author(s)

Thierry Denoeux.

References


kcevclus

See Also

pcca

Examples

opt<-harris(function(x) return(list(fun=sum(x^2),grad=2*x)),rnorm(2),tr=TRUE)
print(c(opt$par,opt$value))
plot(opt$trace$fct,type="l")

kcevclus

k-CEVCLUS algorithm

Description

kcevclus computes a credal partition from a dissimilarity matrix and pairwise (must-link and cannot-link) constraints using the k-CEVCLUS algorithm.

Usage

kcevclus(
  x,  # nxp matrix of p attributes observed for n objects (optional).
  k = n - 1,  # Number of distances to compute for each object (default: n-1).
  D,  # nxn or nxk dissimilarity matrix (used only if x is not supplied).
  J,
  c,
  ML,
  CL,
  xi = 0.5,
  type = "simple",
  pairs = NULL,
  m0 = NULL,
  ntrials = 1,
  disp = TRUE,
  maxit = 1000,
  epsi = 1e-05,
  d0 = quantile(D, 0.9),
  tr = FALSE,
  change.order = FALSE,
  norm = 1
)

Arguments

x  # nxp matrix of p attributes observed for n objects (optional).
k  # Number of distances to compute for each object (default: n-1).
D  # nxn or nxk dissimilarity matrix (used only if x is not supplied).
kcevclus

J
n x k matrix of indices. D[i,j] is the distance between objects i and J[i,j]. (Used only if D is supplied and ncol(D)<n; then k is set to ncol(D).)

c
Number of clusters

ML
Matrix nbML x 2 of must-link constraints. Each row of ML contains the indices of objects that belong to the same class.

CL
Matrix nbCL x 2 of cannot-link constraints. Each row of CL contains the indices of objects that belong to different classes.

xi
Penalization coefficient.

type
Type of focal sets ("simple": empty set, singletons and Omega; "full": all 2^c subsets of Ω; "pairs": ∅, singletons, Ω, and all or selected pairs).

pairs
Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".

m0
Initial credal partition. Should be a matrix with n rows and a number of columns equal to the number f of focal sets specified by 'type' and 'pairs'.
ntrials
Number of runs of the optimization algorithm (set to 1 if m0 is supplied and change.order=FALSE).

disp
If TRUE (default), intermediate results are displayed.

maxit
Maximum number of iterations.

epsi
Minimum amount of improvement.

d0
Parameter used for matrix normalization. The normalized distance corresponding to d0 is 0.95.

tr
If TRUE, a trace of the stress function is returned.

change.order
If TRUE, the order of objects is changed at each iteration of the Iterative Row-wise Quadratic Programming (IRQP) algorithm.

norm
Normalization of distances. 1: division by mean(D^2) (default); 2: division par n*p.

Details
k-CEVCLUS is a version of EVCLUS allowing the user to specify pairwise constraints to guide the clustering process. Pairwise constraints are of two kinds: must-link constraints are pairs of objects that are known to belong to the same class, and cannot-link constraints are pairs of objects that are known to belong to different classes. As kevclus, kcevclus uses the Iterative Row-wise Quadratic Programming (IRQP) algorithm (see ter Braak et al., 2009). It also makes it possible to use only a random sample of the dissimilarities, reducing the time and space complexity from quadratic to roughly linear (Denoeux et al., 2016).

Value
The credal partition (an object of class "credpart"). In addition to the usual attributes, the output credal partition has the following attributes:

Kmat The matrix of degrees of conflict. Same size as D.
D The normalized dissimilarity matrix.
trace Trace of the algorithm (Stress function vs iterations).
J The matrix of indices.
Author(s)

Feng Li and Thierry Denoeux.

References


See Also

kevclus, createD, makeF, extractMass, create_MLCL, bananas, nnevclus

Examples

```r
## Not run:
data<-bananas(2000)
D<-as.matrix(dist(data$x))
link<-create_MLCL(data$y,2000)
clus0<-kevclus(D=D,k=200,c=2)
clus1<-kcevclus(D=D,k=200,c=2,ML=link2$ML,CL=link2$CL,Xi=0.1,m0=clus0$mass)
clus2<-kcevclus(D=D,k=200,c=2,ML=link2$ML,CL=link2$CL,Xi=0.5,m0=clus1$mass)
plot(clus2,X=data$x,ytrue=data$y,Outliers=FALSE,Approx=1)
## End(Not run)
```

kevclus

k-EVCLUS algorithm

Description

kevclus computes a credal partition from a dissimilarity matrix using the k-EVCLUS algorithm.

Usage

```r
kevclus(
  x,
  k = n - 1,
  D,
  J,
)```

c,
type = "simple",
pairs = NULL,
m0 = NULL,
ntrials = 1,
disp = TRUE,
maxit = 1000,
epsi = 1e-05,
d0 = quantile(D, 0.9),
tr = FALSE,
change.order = FALSE,
norm = 1
)

Arguments

x nxp matrix of p attributes observed for n objects (optional).
k Number of distances to compute for each object (default: n-1).
D nxn or nxk dissimilarity matrix (used only if x is not supplied).
J nxk matrix of indices. D[i,j] is the distance between objects i and J[i,j]. (Used only if D is supplied and ncol(D)<n; then k is set to ncol(D).)
c Number of clusters
type Type of focal sets ("simple": empty set, singletons and Omega; "full": all 2^c subsets of Ω; "pairs": ∅, singletons, Ω, and all or selected pairs).
pairs Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
m0 Initial credal partition. Should be a matrix with n rows and a number of columns equal to the number f of focal sets specified by 'type' and 'pairs'.
ntrials Number of runs of the optimization algorithm (set to 1 if m0 is supplied and change.order=FALSE).
disp If TRUE (default), intermediate results are displayed.
maxit Maximum number of iterations.
epsi Minimum amount of improvement.
d0 Parameter used for matrix normalization. The normalized distance corresponding to d0 is 0.95.
tr If TRUE, a trace of the stress function is returned.
change.order If TRUE, the order of objects is changed at each iteration of the Iterative Row-wise Quadratic Programming (IRQP) algorithm.
norm Normalization of distances. 1: division by mean(D^2) (default); 2: division par n*p.
Details

This version of the EVCLUS algorithm uses the Iterative Row-wise Quadratic Programming (IRQP) algorithm (see ter Braak et al., 2009). It also makes it possible to use only a random sample of the dissimilarities, reducing the time and space complexity from quadratic to roughly linear (Denoeux et al., 2016). The user must supply: 1) a matrix x or size (n,p) containing the values of p attributes for n objects, or 2) a matrix D of size (n,n) of dissimilarities between n objects, or 3) a matrix D of size (n,k) of dissimilarities between the n objects and k randomly selected objects, AND a matrix J of size (n,k) of indices, such that D[i,j] is the distance between objects i and J[i,j]. In cases 1 and 2, the user may supply the number k of distances to be picked randomly for each object. In case 3, k is set to the number of columns of D.

Value

The credal partition (an object of class "credpart"). In addition to the usual attributes, the output credal partition has the following attributes:

- **Kmat** The matrix of degrees of conflict. Same size as D.
- **D** The normalized dissimilarity matrix.
- **trace** Trace of the algorithm (Stress function vs iterations).
- **J** The matrix of indices.

Author(s)

Thierry Denoeux.

References


See Also

createD, makeF, extractMass

Examples

```r
## Example with a non metric dissimilarity matrix: the Protein dataset
## Not run:
data(protein)
clus <- kevclus(D=protein$D,c=4,type='simple',d0=max(protein$D))
z<- cmdscale(protein$D,k=2) # Computation of 2 attributes by Multidimensional Scaling
plot(clus,X=z,mfrow=c(2,2),ytrue=protein$y,Outliers=FALSE,Approx=1)
## Example with k=30
```
clus <- kevclus(D=protein$D,k=30,c=4,type='simple',d0=max(protein$D))
z<- cmdscale(protein$D,k=2) # Computation of 2 attributes by Multidimensional Scaling
plot(clus,X=z,mfrow=c(2,2),ytrue=protein$y,Outliers=FALSE,Approx=1)
## End(Not run)

knn_dist

K nearest neighbors in a dissimilarity matrix

Description

knn_dist searches for nearest neighbors in a dissimilarity matrix matrix.

Usage

knn_dist(D, K)

Arguments

D Dissimilarity matrix of size (n,n), where n is the number of objects.
K Number of neighbors

Details

This function is called by EkNNclus if argument x is not supplied. It is not optimized and cannot be used for very large D. If an attribute matrix x is supplied and D is the matrix of Euclidean distances, it is preferable to use function get.knn from package FNN.

Value

A list with two components:

- nn.dist An (n,K) matrix for the nearest neighbor dissimilarities.
- nn.index An (n,K) matrix for the nearest neighbor indices.

Author(s)

Thierry Denoeux.

See Also

get.knn, EkNNclus
Examples

```r
data(butterfly)
n <- nrow(butterfly)
D <- as.matrix(dist(butterfly))
knn <- knn_dist(D, K = 2)
knn$nn.dist
knn$nn.index```

kpcca

### Kernel Pairwise Constrained Component Analysis (KPCCA)

**Description**

Using must-link and cannot-link constraints, KPCCA (Mignon & Jury, 2012) learns a projection into a low-dimensional space where the distances between pairs of data points respect the desired constraints, exhibiting good generalization properties in presence of high dimensional data. This is a kernelized version of pcca.

**Usage**

```r
kpcca(K, d1, ML, CL, beta = 1, epsi = 1e-04, etamax = 0.1, disp = TRUE)
```

**Arguments**

- `K`: Gram matrix of size n*n
- `d1`: Number of extracted features.
- `ML`: Matrix nbML x 2 of must-link constraints. Each row of ML contains the indices of objects that belong to the same class.
- `CL`: Matrix nbCL x 2 of cannot-link constraints. Each row of CL contains the indices of objects that belong to different classes.
- `beta`: Sharpness parameter in the loss function (default: 1).
- `epsi`: Minimal rate of change of the cost function (default: 1e-4).
- `etamax`: Maximum step in the line search algorithm (default: 0.1).
- `disp`: If TRUE (default), intermediate results are displayed.

**Value**

A list with three attributes:

- `z`: The n*d1 matrix of extracted features.
- `A`: The projection matrix of size d1*n.
- `D`: The Euclidean distance matrix in the projected space.

**Author(s)**

Thierry Denoeux.
makeF

Creation of a matrix of focal sets

Description

makeF creates a matrix of focal sets

Usage

makeF(c, type = c("simple", "full", "pairs"), pairs = NULL, Omega = TRUE)

Arguments

c
Number of clusters.

type
Type of focal sets ("simple": ∅, singletons and Ω; "full": all 2^c subsets of Ω; "pairs": ∅, singletons, Ω, and all or selected pairs).

pairs
Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".

Omega
Logical. If TRUE (default), Ω is a focal set (for types 'simple' and 'pairs').

Value

A matrix (f,c) of focal sets.
Examples

c <- 4
## Generation of all 16 focal sets
F <- makeF(c, type = 'full')
## Generation of focal sets of cardinality 0, 1 and c
F <- makeF(c, type = 'simple')
## Generation of focal sets of cardinality 0, 1, and 2
F <- makeF(c, type = 'pairs', Omega = FALSE)
## Generation of focal sets of cardinality 0, 1, and c, plus the pairs (1,2) and (1,3)
F <- makeF(c, type = 'pairs', pairs = matrix(c(1, 2, 1, 3), nrow = 2, byrow = TRUE))

Description

nnevclus computes a credal partition from a dissimilarity matrix using the NN-EVCLUS algorithm.

Usage

nnevclus(
  x,
  k = n - 1,
  D = NULL,
  J = NULL,
  c,
  type = "simple",
  n_H,
  n_trials = 1,
  d0 = quantile(D, 0.9),
  fhat = NULL,
  lambda = 0,
  y = NULL,
  Is = NULL,
  nu = 0,
  ML = NULL,
  CL = NULL,
  xi = 0,
  tr = FALSE,
  options = c(1e00, 1e-04, 10),
  param0 = list(U0 = NULL, V0 = NULL, W0 = NULL, beta0 = NULL)
)
Arguments

- **x**: nxp matrix of p attributes observed for n objects.
- **k**: Number of distances to compute for each object (default: n-1).
- **D**: nxn or nxk dissimilarity matrix (optional). If absent, the Euclidean distance is computed.
- **J**: nxk matrix of indices. D[i,j] is the distance between objects i and J[i,j]. (Used only if D is supplied and ncol(D)<n.)
- **c**: Number of clusters
- **type**: Type of focal sets ("simple": empty set, singletons and Omega; "full": all 2^c subsets of Omega; "pairs": [], singletons, Omega, and all or selected pairs).
- **n_H**: Number of hidden units (if one hidden layer), or a two-dimensional vector of numbers of hidden units (if two hidden layers).
- **ntrials**: Number of runs of the optimization algorithm (set to 1 if param0 is supplied).
- **d0**: Parameter used for matrix normalization. The normalized distance corresponding to d0 is 0.95.
- **fhat**: Vector of outputs from a one-class SVM for novelty detection (optional)
- **lambda**: Regularization coefficient (default: 0)
- **y**: Optional vector of class labels for a subset of the training set (for semi-supervised learning).
- **Is**: Vector of indices corresponding to y (for semi-supervised learning).
- **nu**: Coefficient of the supervised error term (default: 0).
- **ML**: Optional nbML*2 matrix of must-link constraints (for constrained clustering). Each row of ML contains the indices of objects that belong to the same class.
- **CL**: Optional nbCL*2 matrix of cannot-link constraints (for constrained clustering). Each row of CL contains the indices of objects that belong to different classes.
- **xi**: Coefficient of the constrained clustering loss (default: 0).
- **tr**: If TRUE, a trace of the stress function is returned.
- **options**: Parameters of the optimization algorithm (see harris).
- **param0**: Optional list of initial network parameters (see details).

Details

This is a neural network version of kevclus. The neural net has one or two layers of ReLU units and a softmax output layer (see Denoeux, 2020). The weight matrices are denoted by U, V and W for a two-hidden-layer network, or V and W for a one-hidden-layer network. The inputs are a feature vector x, an optional distance matrix D, and an optional vector of one-class SVM outputs fhat, which is used for novelty detection. Part of the output belief mass is transferred to the empty set based on beta[1]+beta[2]*fhat, where beta is an additional parameter vector. The network can be trained in fully unsupervised mode, in semi-supervised mode (with class labels for a subset of the learning instances), or with pairwise constraints. The output is a credal partition (a "credpart" object), with a specific field containing the network parameters (U, V, W, beta).
Value

The output credal partition (an object of class "credpart"). In addition to the usual attributes, the output credal partition has the following attributes:

- **Kmat**: The matrix of degrees of conflict. Same size as D.
- **D**: The normalized dissimilarity matrix.
- **trace**: Trace of the algorithm (Stress function vs iterations).
- **J**: The matrix of indices.
- **param**: The network parameters as a list with components U, V, W and beta.

Author(s)

Thierry Denoeux.

References


See Also

nnevclus_mb, predict.credpart, kevclus, kcevclus, harris

Examples

```r
## Not run:
## Example with one hidden layer and no novelty detection
data(fourclass)
x<-scale(fourclass[,1:2])
y<-fourclass[,3]
clus<-nnevclus(x,c=4,n_H=c(5,5),type="Var pairs") # One hidden layer
plot(clus,x,mfrow=c(2,2))

## Example with two hidden layers and novelty detection
library(kernlab)
data(fourclass)
x<-scale(fourclass[,1:2])
y<-fourclass[,3]
x<-data.frame(x)
svmfit<-ksvm(~.,data=x,type="one-svc",kernel="rbfdot",nu=0.2,kpar=list(sigma=0.2))
fhat<-predict(svmfit,newdata=x,type="decision")
clus<-nnevclus(x,k=200,c=4,n_H=c(5,5),type="pairs",fhat=fhat)
plot(clus,x,mfrow=c(2,2))

## Example with semi-supervised learning
data<-bananas(400)
x<-scale(data$x)
y<-data$y
Is<-sample(400, 50) # Indices of labeled instances
plot(x,col=y,pch=y)
```
```r
points(x[Is,],pch=16)
svmfit<-ksvm(~.,data=x,type="one-svc",kernel="rbfdot",nu=0.2,kpar=list(sigma=0.2))
fhat<-predict(svmfit,newdata=x,type="decision")
clus<-nnevclus(x,k=100,c=2,n_H=10,type="full",fhat=fhat,Is=Is,y=y[Is],nu=0.5)
plot(clus,x)

## Example with pairwise constraints
data<-'bananas(400)
x<-scale(data$x)
y<-data$y
const<-create_MLCL(y,500)
clus<-nnevclus(x,k=100,c=2,n_H=10,type="full",fhat=fhat,ML=const$ML,CL=const$CL,rho=0.5)
plot(clus,x)

## Example with pairwise constraints and PCCA
data(iris)
x<-scale(as.matrix(iris[,1:4]))
y<-as.integer(iris[,5])
const<-create_MLCL(y,100)
res.pcca<-pcca(x,3,const$ML,const$CL,beta=1)
plot(res.pcca$z,pch=y,col=y)
clus<-nnevclus(x=x,D=res.pcca$D,c=3,n_H=10,type='full',ML=const$ML,CL=const$CL,rho=0.5)
plot(clus,x[,3:4])

## End(Not run)
```

---

**nnevclus_mb**

**NN-EVCLUS algorithm (minibatch version)**

**Description**

`nnevclus_mb` computes a credal partition from a dissimilarity matrix using the NN-EVCLUS algorithm. Training is done using mini-batch gradient descent with the RMSprop optimization algorithm.

**Usage**

```r
nnevclus_mb(
  x,
  foncD = function(x) as.matrix(dist(x)),
  c,
  type = "simple",
  n_H,
  nbatch = 10,
  alpha0 = 0.9,
  fhat = NULL,
  lambda = 0,
```

\(y = \text{NULL},\)
\(\text{Is} = \text{NULL},\)
\(\text{nu} = 0,\)
\(\text{disp} = \text{TRUE},\)
\(\text{options} = \text{list(}\text{Niter} = 1000, \text{epsi} = 0.001, \text{rho} = 0.9, \text{delta} = 1e-08, \text{Dtmax} = 100, \text{print} = 5),\)
\(\text{param0} = \text{list(}V0 = \text{NULL}, W0 = \text{NULL}, \text{beta0} = \text{NULL})\)

**Arguments**

- **x**
  - nxp matrix of p attributes observed for n objects.
- **foncD**
  - A function to compute distances.
- **c**
  - Number of clusters
- **type**
  - Type of focal sets ("simple": empty set, singletons and Omega; "full": all \(2^c\) subsets of \(\Omega\); "pairs": \(\emptyset\), singletons, \(\Omega\), and all or selected pairs).
- **n_H**
  - Size or the hidden layer.
- **nbatch**
  - Number of mini-batches.
- **alpha0**
  - Order of the quantile to normalize distances. Parameter \(d_0\) is set to the \(alpha0\)-quantile of distances. Default: 0.9.
- **fhat**
  - Vector of outputs from a one-class SVM for novelty detection (optional)
- **lambda**
  - Regularization coefficient (default: 0)
- **y**
  - Optional vector of class labels for a subset of the training set (for semi-supervised learning).
- **Is**
  - Vector of indices corresponding to y (for semi-supervised learning).
- **nu**
  - Coefficient of the supervised error term (default: 0).
- **disp**
  - If TRUE, intermediate results are displayed.
- **options**
  - Parameters of the optimization algorithm (Niter: maximum number of iterations; epsi, rho, delta: parameters of RMSprop; Dtmax: the algorithm stops when the loss has not decreased in the last Dtmax iterations; print: number of iterations between two displays).
- **param0**
  - Optional list of initial network parameters (see details).

**Details**

This is a neural network version of kevclus. The neural net has one layer of ReLU units and a softmax output layer (see Denoeux, 2020). The network is trained in mini-batch mode using the RMSprop algorithm. The inputs are a feature vector x, an optional distance matrix D, and an optional vector of one-class SVM outputs fhat, which is used for novelty detection. Part of the output belief mass is transferred to the empty set based on beta[1]+beta[2]*fhat, where beta is an additional parameter vector. The network can be trained in fully unsupervised mode or in semi-supervised mode (with class labels for a subset of the learning instances). The output is a credal partition (a "credpart" object), with a specific field containing the network parameters (U, V, W, beta).
Value

The output credal partition (an object of class "credpart"). In addition to the usual attributes, the output credal partition has the following attributes:

Kmat The matrix of degrees of conflict. Same size as D.
trace Trace of the algorithm (values of the loss function).
param The network parameters as a list with components V, W and beta.

Author(s)

Thierry Denoeux.

References


See Also

nnevclus, predict.credpart, kevclus, kcevclus, harris

Examples

```r
## Not run:
## Unsupervised learning
data(fourclass)
x<-scale(fourclass[,1:2])
y<-fourclass[,3]
svmfit<-ksvm(~.,data=x,type="one-svc",kernel="rbfdot",nu=0.2,kpar=list(sigma=0.2))
fhat<-predict(svmfit,newdata=x,type="decision")
clus<-nnevclus_mb(x,foncD=function(x) as.matrix(dist(x)),c=4,type=pairs, n_H=10,nbatch=10,alpha0=0.9,fhat=fhat)
plot(clus,x)
## semi-supervised learning
Is<-sample(400,100)
clus<-nnevclus_mb(x,foncD=function(x) as.matrix(dist(x)),c=4,type=pairs, n_H=10,nbatch=10,alpha0=0.9,fhat=fhat,lambda=0, y=y[Is],Is=Is,nu=0.5)
plot(clus,x)
## End(Not run)
```
nonspecificity  

**Nonspecificity of the relational representation of a credal partition**

**Description**

nonspecificity the average nonspecificity of a credal partition, as defined in Denoeux et al (2018).

**Usage**

nonspecificity(P)

**Arguments**

P  
The relation representation of a credal partition as generated by pairwise_mass.

**Value**

The mean nonspecificity (i.e, the average nonspecificity of pairwise mass functions in P).

**References**


**See Also**

credal_RI,pairwise_mass

**Examples**

```r
## Butterfly data
data(butterfly)
clus<-kevclus(butterfly,c=2)
P<-pairwise_mass(clus)
print(nonspecificity(P))
```
normalize.credpart

Normalization of a credal partition

Description

normalize.credpart normalizes a credal partition (a "credpart" object).

Usage

normalize.credpart(clus, method = "d")

Arguments

clus An object of class "credpart", encoding a credal partition.
method Normalization method ("d" for Dempster or "y" for Yager).

Details

The function implements two normalization methods: Dempster's normalization (the mass of each focal set is divided by one minus the mass on the empty set), and yager's normalization (the mass of the empty set is transfered to the whole frame).

Value

The normalized credal partition (a "credpart" object).

References


See Also

extractMass, plot.credpart, summary.credpart.

Examples

data(butterfly)
clus<-kevclus(butterfly,c=2)
print(clus$mass)
clus1<-normalize.credpart(clus,"d") # Dempster normalization
print(clus1$mass)
clus2<-normalize.credpart(clus,"y") # Yager normalization
print(clus2$mass)
pairwise_mass

Computes the relational representation

Description

pairwise_mass computes the relational representation of a credal partition, as defined in Denoeux et al (2018).

Usage

pairwise_mass(clus)

Arguments

clus A credal partition (a matrix of n rows and f columns, where n is the number of objects and f is the number of focal sets).

Details

Given a credal partition, we can compute, for each pair of objects, a "pairwise mass function" on a frame $\Theta = \{s, \neg s\}$, where $s$ means that the two objects belong to the same cluster, and $\neg s$ is the negation of $s$. Function pairwise_mass compute these pairwise mass functions for all object pairs. The result is return as a list with "dist" objects containing the masses of each of the two elements of $\Theta$, and the masses on the empty set.

Value

A list with three "dist" objects:

- **M0** The masses assigned to the assumption that each pair of object (i,j) do not belong to the same class.
- **M1** The masses assigned to the assumption that each pair of object (i,j) belongs to the same class.
- **Me** The masses assigned to the empty set, for each pair of object (i,j).

References


See Also

credal_RI, nonspecificity
Examples

```r
## Butterfly data
data(butterfly)
clus<-kevclus(butterfly,c=2)
P<-pairwise_mass(clus)
```

---

**Pairwise Constrained Component Analysis (PCCA)**

Description

Using must-link and cannot-link constraints, PCCA (Mignon & Jury, 2012) learns a projection into a low-dimensional space where the distances between pairs of data points respect the desired constraints, exhibiting good generalization properties in presence of high dimensional data.

Usage

```r
pcca(x, d1, ML, CL, options = c(1, 1000, 1e-05, 10), beta = 1)
```

Arguments

- **x**: Data matrix of size n*d
- **d1**: Number of extracted features.
- **ML**: Matrix nbML x 2 of must-link constraints. Each row of ML contains the indices of objects that belong to the same class.
- **CL**: Matrix nbCL x 2 of cannot-link constraints. Each row of CL contains the indices of objects that belong to different classes.
- **options**: Parameters of the optimization algorithm (see `harris`).
- **beta**: Sharpness parameter in the loss function (default: 1).

Value

A list with three attributes:

- **z**: The n*d1 matrix of extracted features.
- **L**: The projection matrix of size d1*d.
- **D**: The Euclidean distance matrix in the projected space

Author(s)

Thierry Denoeux.
plot.credpart

Plotting a credal partition

Description

Generates plots of a credal partition.

Usage

```r
## S3 method for class 'credpart'
plot(
  x,
  X = NULL,
  ...,
  mfrow = c(1, 1),
  ytrue = NULL,
  Outliers = TRUE,
  Approx = 1,
  cex = 1,
  cexvar = "pl",
  cex_outliers = 1.3,
  cex_protos = 1,
  lwd = 2,
  ask = FALSE,
  plot_Shepard = FALSE,
  plot_approx = TRUE,
)```

References


See Also

kpcca, harris, create_MLCL

Examples

```r
## Not run:
data(iris)
x<-as.matrix(iris[,1:4])
y<-as.integer(iris[,5])
const<-create_MLCL(y,50)
res.pcca<-pcca(x,1,const$ML,const$CL)
plot(res.pcca$z,col=y,pch=y)
## End(Not run)
```
```r
plot.credpart = TRUE,
xlab = expression(x[1]),
ylab = expression(x[2])
)
```

### Arguments

- **x** An object of class "credpart", encoding a credal partition.
- **X** A data matrix. If it has more than two columns (attributes), only the first two columns are used.
- **...** Other arguments to be passed to the plot function.
- **mfrow** A 2-vector defining the number of rows and columns of the plot. If mfrow=c(1,1), only one figure is drawn. Otherwise, mfrow[1] x mfrow[2] should not be less than x, the number of clusters.
- **ytrue** The vector of true class labels. If supplied, a different color is used for each true cluster. Otherwise, the maximum-plausibility clusters are used instead.
- **Outliers** If TRUE, the outliers are plotted, and they are not included in the lower and upper approximations of the clusters.
- **Approx** If Approx==1 (default), the lower and upper cluster approximations are computed using the interval dominance rule. Otherwise, the maximum mass rule is used.
- **cex** Maximum size of data points.
- **cexvar** Parameter determining if the size of the data points is proportional to the plausibilities ('pl', the default), the plausibilities of the normalized credal partition ('pl.n'), the degrees of belief ('bel'), the degrees of belief of the normalized credal partition ('bel.n'), or if it is constant ('cst', default).
- **cex_outliers** Size of data points for outliers.
- **cex_protos** Size of data points for prototypes (if applicable).
- **lwd** Line width for drawing the lower and upper approximations.
- **ask** Logical; if TRUE, the user is asked before each plot.
- **plot_Shepard** Logical; if TRUE and if the credal partition was generated by kevclus the Shepard diagram is plotted.
- **plot_approx** Logical; if TRUE (default) the convex hulls of the lower and upper approximations are plotted.
- **plot_protos** Logical; if TRUE (default) the prototypes are plotted (for methods generating prototypes, like ECM).
- **xlab** Label of horizontal axis.
- **ylab** Label of vertical axis.

### Details

This function plots different views of a credal partition in a two-dimensional attribute space. If mfrow=c(1,1) (the default), the function plot the dataset with a different symbol for each cluster.
predict.credpart

Value

The maximum plausibility hard partition, as well as the lower and upper approximations of each cluster are drawn in the two-dimensional space specified by matrix X. If prototypes are defined (for methods "ecm" and "cecm"), they are also represented on the plot. For methods "kevclus", "kcevclus" or "nnevclus" a second plot with Shepard’s diagram (degrees of conflict vs. transformed dissimilarities) is drawn. If input X is not supplied and the Shepard diagram exists, then only the Shepard diagram is drawn.

References


See Also

eextractMass, summary.credpart, ecm, recm, cecm, kevclus.

Examples

## Example with Four-class data
data("fourclass")
x<-fourclass[,1:2]
y<-fourclass[,3]
c=4
## Running k-EVCLUS with singletons
clus<-kevclus(x=x,k=100,c=c,type='simple')
## Plot the results
plot(clus,X=x,mfrow=c(2,2),ytrue=y)

predict.credpart

Computation of a credal partition for new data

Description

predict.credpart is the predict method for "credpart" objects generated by nnevclus or ecm.

Usage

## S3 method for class 'credpart'
predict(object, newdata, fhat = NULL, ...)

Arguments

object       An object of class "credpart", encoding a credal partition.
newdata     A matrix of size ntest*p containing the new data.
fhat        An optional vector of one-class SVM outputs (for method nn-evclus only)
...         Additional arguments (not used).

Details

This function computes a credal partial of newdata based on learnt information stored in a "credpart" objects created by ecm or nnevclus.

Value

A credal partition of the new data.

References


See Also

ecm, ceem, nnevclus.

Examples

## Not run:
data(fourclass)
train<-sample(400,200)
x<-fourclass[train,1:2]
x.test<-x[-train,1:2]
clus<-ecm(x,c=4,type='pairs',delta=sqrt(10),epsi=1e-3,disp=TRUE)
clus.test<-predict(clus,x.test)
plot(clus.test,x.test,mfrow=c(2,2))

## End(Not run)
Description

This real data set consists of a dissimilarity matrix derived from the structural comparison of 213 protein sequences. Each of these proteins is known to belong to one of four classes of globins: hemoglobin-alpha (HA), hemoglobin-beta (HB), myoglobin (M) and heterogeneous globins (G).

Usage

data(protein)

Format

A list with three elements:

D  The 213x213 dissimilarity matrix.
class  A 213-vector containing the class encoded a a factor with four levels: "G", "HA", "HB", "M".
y  A 213-vector containing the class encoded by an integer between 1 and 4.

References


Examples

data(protein)
z<- cmdscale(protein$D,k=2)  # Multidimensional scaling
plot(z[,1],z[,2],xlab=expression(z[1]),ylab=expression(z[2]),pch=protein$y,col=protein$y)
Description

`recm` computes a credal partition from a dissimilarity matrix using the Relational Evidential c-means (RECM) algorithm.

Usage

```r
recm(D, c, 
  type = "full", 
  pairs = NULL, 
  Omega = TRUE, 
  m0 = NULL, 
  ntrials = 1, 
  alpha = 1, 
  beta = 1.5, 
  delta = quantile(D[upper.tri(D) | lower.tri(D)], 0.95), 
  epsi = 1e-04, 
  maxit = 5000, 
  disp = TRUE
)
```

Arguments

- **D**  
  Dissimilarity matrix of size (n,n), where n is the number of objects. Dissimilarities must be squared Euclidean distances to ensure convergence.

- **c**  
  Number of clusters.

- **type**  
  Type of focal sets ("simple": empty set, singletons and Omega; "full": all 2^c subsets of Ω; "pairs": ∅, singletons, Ω, and all or selected pairs).

- **pairs**  
  Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".

- **Omega**  
  Logical. If TRUE (default), the whole frame is included (for types 'simple' and 'pairs').

- **m0**  
  Initial credal partition. Should be a matrix with n rows and a number of columns equal to the number f of focal sets specified by 'type' and 'pairs'.

- **ntrials**  
  Number of runs of the optimization algorithm (set to 1 if m0 is supplied).

- **alpha**  
  Exponent of the cardinality in the cost function.

- **beta**  
  Exponent of masses in the cost function.

- **delta**  
  Distance to the empty set.

- **epsi**  
  Minimum amount of improvement.

- **maxit**  
  Maximum number of iterations.

- **disp**  
  If TRUE (default), intermediate results are displayed.
Details

RECM is a relational version of the Evidential c-Means (ECM) algorithm. Convergence is guaranteed only if elements of matrix D are squared Euclidean distances. However, the algorithm is quite robust and generally provides sensible results even if the dissimilarities are not metric. By default, each mass function in the credal partition has \(2^c\) focal sets, where \(c\) is the supplied number of clusters. We can also limit the number of focal sets to subsets of clusters with cardinalities 0, 1 and \(c\) (recommended if \(c\geq10\)), or to all or some selected pairs of clusters. If an initial credal partition \(m_0\) is provided, the number of trials is automatically set to 1.

Value

The credal partition (an object of class "credpart").

Author(s)

Thierry Denoeux (from a MATLAB code written by Marie-Helene Masson).

References


See Also

makeF, extractMass, ecm

Examples

```r
## Clustering of the Butterfly dataset
## Not run:
n <- nrow(butterfly)
D <- as.matrix(dist(butterfly))^2
clus <- recm(D, c=2, delta=sqrt(50))
m <- clus$mass
plot(1:n, m[, 1], type="l", ylim=c(0, 1), xlab="objects", ylab="masses")
lines(1:n, m[, 2], lty=2)
lines(1:n, m[, 3], lty=3)
lines(1:n, m[, 4], lty=4)

## Clustering the protein data
data(protein)
clus <- recm(D=protein$D, c=4, type="full", alpha=0.2, beta=1.1, delta=sqrt(20))
z <- cmdscale(protein$D, k=2)
plot(clus, X=z, mfrow=c(2, 2), ytrue=protein$y, Outliers=FALSE, Approx=1)
## End(Not run)
```
s2  

**S2 dataset**

**Description**

This dataset contains 5000 two-dimensional vectors grouped in 15 Gaussian clusters.

**Usage**

```r
data(s2)
```

**Format**

A matrix with 5000 rows and two columns.

**References**


**Examples**

```r
data(s2)
plot(s2[,1],s2[,2],xlab=expression(x[1]),ylab=expression(x[2]))
```

---

**summary.credpart**  

**Summary of a credal partition**

**Description**

summary.credpart is the summary method for "credpart" objects.

**Usage**

```r
## S3 method for class 'credpart'
summary(object, ...)  
```

**Arguments**

- **object**  
  An object of class "credpart", encoding a credal partition.

- **...**  
  Additional arguments (not used).
Details

This function extracts basic information from "credpart" objects, such as created by ecm, recm, cecm, EkNNclus or kevclus.

Value

Prints basic information on the credal partition.

References


See Also

extractMass, plot.credpart, ecm, recm, cecm, EkNNclus, kevclus.

Examples

```r
## Example with Four-class data
data("fourclass")
x<-fourclass[,1:2]
y<-fourclass[,3]
c=4
## Running k-EVCLUS with singletons
clus<-kevclus(x=x,k=100,c=c,type='simple')
summary(clus)
```
Index

* datasets
  butterfly, 7
  fourclass, 26
  protein, 50
  s2, 53

  bananas, 2, 22, 30
  bootclus, 3, 22
  bpec, 5, 16, 22
  butterfly, 7
  cecm, 4, 6, 7, 11, 14, 18, 22, 48, 49, 54
  create_fuzzy_credpart, 12, 13
  create_hard_credpart, 12, 13
  create_MLCL, 9, 14, 22, 30, 35, 46
  createD, 9, 30, 32
  createPairs, 10
  credal_RI, 14, 42, 44

  delta_Bel, 6, 16
  ecm, 4, 6, 9, 11, 17, 22, 48, 49, 52, 54
  EkNNclus, 19, 22, 24, 25, 33, 54
  evclust, 21
  evclust-package (evclust), 21
  expandlink, 22
  extractMass, 9, 11–13, 18, 23, 30, 32, 43, 48, 52, 54

  fourclass, 26

  get.knn, 33

  harris, 27, 37, 38, 41, 45, 46

  kcevclus, 3, 22, 28, 38, 41
  kevclus, 4, 10, 11, 22, 24, 25, 30, 30, 38, 41, 48, 54
  knn_dist, 33
  kpcca, 34, 46

  makeF, 9, 18, 30, 32, 35, 52

  nnevclus, 22, 30, 36, 41, 49
  nnevclus_mb, 22, 38, 39
  nonspecificity, 15, 42, 44
  normalize.credpart, 43
  pairwise_mass, 15, 42, 44
  pcca, 28, 35, 45
  plot.credpart, 18, 24, 25, 43, 46, 54
  predict.credpart, 24, 38, 41, 48
  protein, 50

  recm, 4, 9, 11, 18, 22, 48, 51, 54

  s2, 53

  summary.credpart, 24, 25, 43, 48, 53