Package ‘EMCluster’

January 20, 2025

Version 0.2-17

Date 2024-12-24

Title EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution

Depends R (>= 4.0.0), MASS, Matrix, methods

Enhances RColorBrewer

LazyLoad yes

LazyData yes

Description EM algorithms and several efficient initialization methods for model-based clustering of finite mixture Gaussian distribution with unstructured dispersion in both of unsupervised and semi-supervised learning.

BugReports https://github.com/snoweye/EMCluster/issues

URL https://github.com/snoweye/EMCluster

NeedsCompilation yes

Maintainer Wei-Chen Chen <wccsnow@gmail.com>

Author Wei-Chen Chen [aut, cre],
Ranjan Maitra [aut],
Volodymyr Melnykov [ctb],
Dan Nettleton [ctb],
David Faden [ctb],
Rouben Rostamian [ctb],
R Core team [ctb] (some functions are modified from the R source code)

Repository CRAN

Date/Publication 2025-01-09 05:00:02 UTC

Contents

`EMCluster-package` *EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution*

Description

EMCluster provides EM algorithms and several efficient initialization methods for model-based clustering of finite mixture Gaussian distribution with unstructured dispersion in both of unsupervised and semi-supervised clustering.

Details

The install command is simply as

```
> R CMD INSTALL EMCluster_0.2-0.tar.gz
```

from a command mode or

```
R> install.packages("EMCluster")
```

inside an R session.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
## Not run:
demo(allinit, 'EMCluster', ask = F, echo = F)
demo(allinit_ss, 'EMCluster', ask = F, echo = F)

## End(Not run)
```

`Assign Class` *Assign Class Id*

Description

This function assigns cluster id to each observation in `x` according to the desired model `emobj` or specified parameters `pi`, `Mu`, and `LTSigma`.

Usage

```
assign.class(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
             lab = NULL, return.all = TRUE)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`pi`	the mixing proportion, length K.
`Mu`	the centers of clusters, dimension $K \times p$.
`LTSigma`	the lower triangular matrices of dispersion, dimension $K \times p(p+1)/2$.
`lab`	labeled data for semi-supervised clustering, length n.
`return.all`	if returning with a whole `emobj` object.

Details

This function are based either an input `emobj` or inputs `pi`, `Mu`, and `LTSigma` to assign class id to each observation of `x`.

If `lab` is submitted, then the observation with label id greater 0 will not be assigned new class.

Value

This function returns a list containing mainly two new variables: `nc` (length K numbers of observations in each class) and `class` (length n class id).

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
library(EMCluster, quietly = TRUE)
set.seed(1234)
x2 <- da2$da

ret <- init.EM(x2, nclass = 2)
ret.new <- assign.class(x2, ret, return.all = FALSE)
str(ret.new)
```

`Conversion` *Convert Matrices in Different Format*

Description

These utility functions are to convert matrices in different formats.

Usage

```
LTSigma2variance(x)
variance2LTSigma(x)
LTsigma2var(x1, p = NULL)
var2LTsigma(x1)
class2Gamma(class)
Gamma2class(Gamma)
```

Arguments

`x`	a matrix/array to be converted, the dimension could be $K \times p(p+1)/2$ or $p \times p \times K$.
`x1`	a vector/matrix to be converted, the length and dimension could be $p(p+1)/2$ and $p \times p$.
`p`	dimension of matrix.
`class`	id of clusters for each observation, length n.
`Gamma`	containing posterior probabilities if normalized, otherwise containing component densities weighted by mixing proportion, dimension $n \times K$.

Details

`LTSigma2variance` converts `LTSigma` format to 3D array, and `variance2LTSigma` is the inversion function.

`LTsigma2var` converts `LTsigma` format to a matrix, and `var2LTsigma` is the inversion function. Note that `LTsigma` is one component of `LTSigma`.

`class2Gamma` converts id to a Gamma matrix where with probability 1 for the cluster where the observation belongs to, and `Gamma2class` converts posterior to cluster id where largest posterior is picked for each observation.

Value

A vector/matrix/array is returned.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
x <- da2$LTSigma
class <- da2$class

y <- LTSigma2variance(x)
str(y)
y <- variance2LTSigma(y)
str(y)
sum(x != y)
```

```
Gamma <- class2Gamma(class)
class.new <- Gamma2class(Gamma)
sum(class != class.new)

## End(Not run)
```

`Dataset` *Dataset for demonstrations*

Description

There are four small datasets to test and demonstrate **EMCluster**.

Usage

```
da1
da2
da3
```

Format

`da1`, `da2`, `da3` are in `list`.

Details

`da1` has 500 observations in two dimensions `da1$da$x` and `da1$da$y`, and they are in 10 clusters given in `da1$class`.

`da2` has 2,500 observations in two dimensions, too. The true parameters are given in `da1$pi`, `da1$Mu`, and `da1$LTSigma`. There are 40 clusters given in `da1$class` for this dataset.

`da3` is similar to `da2`, but with lower overlaps between clusters.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

`EM Algorithm` *EM Algorithm for model-based clustering*

Description

These are core functions of **EMCluster** performing EM algorithm for model-based clustering of finite mixture multivariate Gaussian distribution with unstructured dispersion.

Usage

```
emcluster(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
          lab = NULL, EMC = .EMC, assign.class = FALSE)
shortemcluster(x, emobj = NULL, pi = NULL, Mu = NULL,
               LTSigma = NULL, maxiter = 100, eps = 1e-2)
simple.init(x, nclass = 1)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`pi`	the mixing proportion, length K.
`Mu`	the centers of clusters, dimension $K \times p$.
`LTSigma`	the lower triangular matrices of dispersion, $K \times p(p+1)/2$.
`lab`	labeled data for semi-supervised clustering, length n.
`EMC`	the control for the EM iterations.
`assign.class`	if assigning class id.
`maxiter`	maximum number of iterations.
`eps`	convergent tolerance.
`nclass`	the desired number of clusters, K.

Details

The `emcluster` mainly performs EM iterations starting from the given parameters `emobj` without other initializations.

The `shortemcluster` performs short-EM iterations as described in `init.EM`.

Value

The `emcluster` returns an object `emobj` with class `emret` which can be used in post-process or other functions such as `e.step`, `m.step`, `assign.class`, `em.ic`, and `dmixmvn`.

The `shortemcluster` also returns an object `emobj` with class `emret` which is the best of several random initializations.

The `simple.init` utilizes `rand.EM` to obtain a simple initial.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

References

https://www.stat.iastate.edu/people/ranjan-maitra

See Also

`init.EM`, `e.step`, `m.step`, `.EMControl`.

Examples

```
library(EMCluster, quietly = TRUE)
set.seed(1234)
x1 <- da1$da

emobj <- simple.init(x1, nclass = 10)
emobj <- shortemcluster(x1, emobj)
summary(emobj)

ret <- emcluster(x1, emobj, assign.class = TRUE)
summary(ret)
```

EM Control *EM Control Generator and Controller*

Description

The `.EMControl` generates an EM control (`.EMC`) controlling the options and conditions of EM algorithms, i.e. this function generate a default template. One can either modify `.EMC` or employ this function to control EM algorithms. By default, `.EMC`, `.EMC.Rnd`, and `.EC.Rndp` are three native controllers as the **EMCluster** is loaded.

Usage

```
.EMControl(alpha = 0.99, short.iter = 200, short.eps = 1e-2,
    fixed.iter = 1, n.candidate = 3,
    em.iter = 1000, em.eps = 1e-6, exhaust.iter = 5)
.EMC
.EMC.Rnd
.EMC.Rndp
```

Arguments

`alpha`	only used in `emgroup` for "SVD" initialization.
`short.iter`	number of short-EM steps, default = 200.
`short.eps`	tolerance of short-EM steps, default = 1e-2.
`fixed.iter`	fixed iterations of EM for "RndEM" initialization, default = 1.
`n.candidate`	reserved for other initialization methods (unimplemented).
`em.iter`	maximum number of long-EM steps, default = 1000.
`em.eps`	tolerance of long-EM steps, default = 1e-6.
`exhaust.iter`	number of iterations for "exhaustEM" initialization, default = 5.

Details

`exhaust.iter` and `fixed.iter` are used to control the iterations of initialization procedures.

`short.iter` and `short.eps` are used to control the short-EM iterations.

`em.iter` and `em.eps` are used to control the long-EM iterations.

Moeover, `short.eps` and `em.eps` are for checking convergence of the iterations.

Value

This function returns a list as `.EMC` by default.

The `.EMC.Rnd` is equal to `.EMControl(short.eps = Inf)` and usually used by the `rand.EM` method.

The `.EMC.Rndp` is equal to `.EMControl(fixed.iter = 5)` where each random initials run 5 EM iterations in the `rand.EM` method.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
library(EMCluster, quietly = TRUE)

.EMC <- .EMControl()
.EMC.Rnd <- .EMControl(short.eps = Inf)
.EMC.Rndp <- .EMControl(fixed.iter = 5)
```

`Information Criteria` *Information Criteria for Model-Based Clustering*

Description

These functions are tools for compute information criteria for the fitted models.

Usage

```
em.ic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
      llhdval = NULL)
em.aic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.bic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.clc(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.icl(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.icl.bic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`pi`	the mixing proportion, length K.
`Mu`	the centers of clusters, dimension $K \times p$.
`LTSigma`	the lower triangular matrices of dispersion, $K \times p(p+1)/2$.
`llhdval`	the total log likelihood value of `x` given `emobj`.

Details

The `em.ic` calls all other functions to compute AIC (`em.aic`), BIC (`em.bic`), CLC (`em.clc`), ICL (`em.icl`), and ICL.BIC (`em.icl.bic`). All are useful information criteria for model selections, mainly choosing number of cluster.

Value

`em.ic` returns a list containing all other information criteria for given the data `x` and the desired model `emobj`.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`.

Examples

```
library(EMCluster, quietly = TRUE)
x2 <- da2$da

emobj <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
em.ic(x2, emobj = emobj)
```

`Initialization and EM` *Initialization and EM Algorithm*

Description

These functions perform initializations (including em.EM and RndEM) followed by the EM iterations for model-based clustering of finite mixture multivariate Gaussian distribution with unstructured dispersion in both of unsupervised and semi-supervised clusterings.

Usage

```
init.EM(x, nclass = 1, lab = NULL, EMC = .EMC,
        stable.solution = TRUE, min.n = NULL, min.n.iter = 10,
        method = c("em.EM", "Rnd.EM"))
em.EM(x, nclass = 1, lab = NULL, EMC = .EMC,
      stable.solution = TRUE, min.n = NULL, min.n.iter = 10)
rand.EM(x, nclass = 1, lab = NULL, EMC = .EMC.Rnd,
        stable.solution = TRUE, min.n = NULL, min.n.iter = 10)
exhaust.EM(x, nclass = 1, lab = NULL,
           EMC = .EMControl(short.iter = 1, short.eps = Inf),
           method = c("em.EM", "Rnd.EM"),
           stable.solution = TRUE, min.n = NULL, min.n.iter = 10);
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`nclass`	the desired number of clusters, K.
`lab`	labeled data for semi-supervised clustering, length n.
`EMC`	the control for the EM iterations.
`stable.solution`	if returning a stable solution.
`min.n`	restriction for a stable solution, the minimum number of observations for every final clusters.
`min.n.iter`	restriction for a stable solution, the minimum number of iterations for trying a stable solution.
`method`	an initialization method.

Details

The `init.EM` calls either `em.EM` if `method="em.EM"` or `rand.EM` if `method="Rnd.EM"`.

The `em.EM` has two steps: short-EM has loose convergent tolerance controlled by `.EMC$short.eps` and try several random initializations controlled by `.EMC$short.iter`, while long-EM starts from the best short-EM result (in terms of log likelihood) and run to convergence with a tight tolerance controlled by `.EMC$em.eps`.

The `rand.EM` also has two steps: first randomly pick several random initializations controlled by `.EMC$short.iter`, and second starts from the best of the random result (in terms of log likelihood) and run to convergence.

The `lab` is only for the semi-supervised clustering, and it contains pre-labeled indices between 1 and K for labeled observations. Observations with index 0 is non-labeled and has to be clustered by the EM algorithm. Indices will be assigned by the results of the EM algorithm. See `demo(allinit_ss,'EMCluster')` for details.

The `exhaust.EM` also calls the `init.EM` with different `EMC` and perform `exhaust.iter` times of EM algorithm with different initials. The best result is returned.

Value

These functions return an object `emobj` with class `emret` which can be used in post-process or other functions such as `e.step`, `m.step`, `assign.class`, `em.ic`, and `dmixmvn`.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`emcluster`, `.EMControl`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)
x <- da1$da

ret.em <- init.EM(x, nclass = 10, method = "em.EM")
ret.Rnd <- init.EM(x, nclass = 10, method = "Rnd.EM", EMC = .EMC.Rnd)

emobj <- simple.init(x, nclass = 10)
ret.init <- emcluster(x, emobj, assign.class = TRUE)

par(mfrow = c(2, 2))
plotem(ret.em, x)
plotem(ret.Rnd, x)
```

```
plotem(ret.init, x)

## End(Not run)
```

`Jaccard Index` *Jaccard Index*

Description

This function returns the Jaccard index for binary ids.

Usage

```
Jaccard.Index(x, y)
```

Arguments

`x`	true binary ids, 0 or 1.
`y`	predicted binary ids, 0 or 1.

Details

All ids, `x` and `y`, should be either 0 (not active) or 1 (active). Any value other than 1 will be converted to 0.

Value

Return the value of Jaccard index.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

Examples

```
library(EMCluster, quietly = TRUE)

x.id <- c(1, 1, 1, 0, 0, 0, 3, 3, 3)
y.id <- c(0, 1, 0, 1, 1, 1, 0, 1, 1)

Jaccard.Index(x.id, y.id)
```

`Likelihood Mixture Tests`

Likelihood Mixture Tests

Description

This function test two mixture Gaussian models with unstructured covariance matrix and different numbers of clusters.

Usage

```
lmt(emobj.0, emobj.a, x, tau = 0.5, n.mc.E.delta = 1000,
    n.mc.E.chi2, verbose = FALSE)
```

Arguments

`emobj.0`	a `emret` object for the null hypothesis.
`emobj.a`	a `emret` object for the alternative hypothesis.
`x`	the data matrix, dimension $n \times p$.
`tau`	proportion of null and alternative hypotheses.
`n.mc.E.delta`	number of Monte Carlo simulations for expectation of delta (difference of logL).
`n.mc.E.chi2`	number of Monte Carlo simulations for expectation of chisquare statistics.
`verbose`	if verbose.

Details

This function calls several subroutines to compute information, likelihood ratio statistics, degrees of freedom, non-centrality of chi-squared distributions …etc. Based on Monte Carlo methods to estimate parameters of likelihood mixture tests, this function return a p-value for testing H0: `emobj.0` v.s. Ha: `emobj.a`.

Value

A list of class `lmt` are returned.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)

x <- as.matrix(iris[, 1:4])
p <- ncol(x)
min.n <- p * (p + 1) / 2
.EMC$short.iter <- 200

ret.2 <- init.EM(x, nclass = 2, min.n = min.n, method = "Rnd.EM")
ret.3 <- init.EM(x, nclass = 3, min.n = min.n, method = "Rnd.EM")
ret.4 <- init.EM(x, nclass = 4, min.n = min.n, method = "Rnd.EM")

(lmt.23 <- lmt(ret.2, ret.3, x))
(lmt.34 <- lmt(ret.3, ret.4, x))
(lmt.24 <- lmt(ret.2, ret.4, x))

## End(Not run)
```

`LMT Functions` *Likelihood Mixture Test (LMT) Functions of EMCluster*

Description

All likelihood mixture test (LMT) functions are for testing and can be utilized by advanced developers with caution.

Currently, these are only for workflows.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

`MVN` *Density of (Mixture) Multivariate Normal Distribution*

Description

These functions are tools for compute density of (mixture) multivariate Gaussian distribution with unstructured dispersion.

Usage

```
dmvn(x, mu, LTsigma, log = FALSE)
dlmvn(x, mu, LTsigma, log = TRUE)
dmixmvn(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL, log = FALSE)
logL(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`mu`	the centers of clusters, length p.
`LTsigma`	the lower triangular matrices of dispersion, length $p(p+1)/2$.
`log`	if logarithm returned.
`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`pi`	the mixing proportion, length K.
`Mu`	the centers of clusters, dimension $K \times p$.
`LTSigma`	the lower triangular matrices of dispersion, $K \times p(p+1)/2$.

Details

The `dmvn` and `dlmvn` compute density and log density of multivariate distribution.

The `dmixmvn` computes density of mixture multivariate distribution and is based either an input `emobj` or inputs `pi`, `Mu`, and `LTSigma` to assign class id to each observation of `x`.

The `logL` returns the value of the observed log likelihood function of the parameters at the current values of the parameters `pi`, `Mu`, and `LTSigma`, with the suplied data matrix `x`.

Value

A density value is returned.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
library(EMCluster, quietly = TRUE)
x2 <- da2$da
x3 <- da3$da

emobj2 <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
emobj3 <- list(pi = da3$pi, Mu = da3$Mu, LTSigma = da3$LTSigma)

logL(x2, emobj = emobj2)
logL(x3, emobj = emobj3)

dmixmvn2 <- dmixmvn(x2, emobj2)
dmixmvn3 <- dmixmvn(x3, emobj3)

dlmvn(da2$da[1,], da2$Mu[1,], da2$LTSigma[1,])
log(dmvn(da2$da[1,], da2$Mu[1,], da2$LTSigma[1,]))
```

`Other Initializations` *Other Initializations*

Description

Two more functions with different initialization method.

Usage

```
starts.via.svd(x, nclass = 1, method = c("em", "kmeans"),
               EMC = .EMC)
emgroup(x, nclass = 1, EMC = .EMC)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`nclass`	the desired number of clusters, K.
`method`	method with the svd initializations.
`EMC`	the control for the EM iterations.

Details

The `starts.via.svd` utilizes SVD to initial parameters, and the `emgroup` runs the EM algorithm starting from the initial.

Value

The `starts.via.svd` returns an object with class `svd`, and the `emgroup` returns and object `emobj` with class `emret`.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `.EMControl`.

Examples

```
library(EMCluster, quietly = TRUE)
set.seed(1234)
x1 <- da1$da

emobj <- emgroup(x1, nclass = 10)
summary(emobj)

ret.0 <- starts.via.svd(x1, nclass = 10, method = "kmeans")
summary(ret.0)
```

`Plot EM Results` *Plot Two Dimensional Data with clusters*

Description

The functions plot two dimensional data for clusters.

Usage

```
plotem(emobj, x, main = NULL, xlab = NULL, ylab = NULL,
        ...)
plot2d(x, emobj = NULL, k = NULL, color.pch = 1,
        append.BN = TRUE, ...)
```

Arguments

`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`x`	the data matrix, dimension $n \times p$.
`main`	title of plot.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`...`	other parameters to the plot.

`k`	index for symbols.
`color.pch`	color and style for symbols.
`append.BN`	if appending bivariate normal ellipsoid.

Details

This a simple x-y lot.

Value

A plot is returned.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
x1 <- da1$da

ret.1 <- starts.via.svd(x1, nclass = 10, method = "em")
summary(ret.1)

plotem(ret.1, x1)

## End(Not run)
```

`Plot Multivariate Data`

Plot Multivariate Data

Description

The function plots multivariate data for clusters as the parallel coordinates plot.

Usage

```
plotmd(x, class = NULL, xlab = "Variables", ylab = "Data", ...)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`class`	class id for all observations.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`...`	other parameters to the plot.

Details

This a simplified parallel coordinate plot.

Value

A plot is returned.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)

x <- as.matrix(iris[, 1:4], ncol = 4)
ret <- em.EM(x, nclass = 5)
plotmd(x, ret$class)

## End(Not run)
```

`Plot Projection and Contour`

Plot Contour

Description

The function plots multivariate data on 2D plane with contour. Typically, the contour is built via projection pursuit or SVD algorithms, such as `project.on.2d()`.

Usage

```
plotppcontour(da, Pi, Mu, S, class, class.true = NULL, n.grid = 128,
    angle = 0, xlab = "", ylab = "", main = "")
```

Arguments

`da`	a projected data matrix, dimension $n \times 2$.
`Pi`	proportion, length K.
`Mu`	the projected centers of cluster, dimension $K \times 2$.
`S`	projected matrices of dispersion, dimension $p \times p \times K$.
`class`	id of classifications, length n.
`class.true`	ture id of classifications if available, length n.
`n.grid`	number of grid points.
`angle`	a rotation angle (0 to 2π).
`xlab`	an option for `plot()` function.
`ylab`	an option for `plot()` function.
`main`	an option for `plot()` function.

Details

This function plots projection output of `project.on.2d()`.

`da`, `Mu`, and `S` are projected by some projection matrices obtained via SVD or projection pursuit algorithms. The projection is made on a 2D plane in the direction in which clusters of data `x` are most distinguishable to visualize.

Value

A 2D projection plot is returned.

Note

Only distinguishable for up to 7 clusters due to the limited color schemes.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`project.on.2d()`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
library(MASS, quietly = TRUE)
set.seed(1234)

### Crabs.
x <- as.matrix(crabs[, 4:8])
ret <- init.EM(x, nclass = 4, min.n = 20)
ret.proj <- project.on.2d(x, ret)

### Plot.
pdf("crabs_ppcontour.pdf", height = 5, width = 5)
plotppcontour(ret.proj$da, ret.proj$Pi, ret.proj$Mu, ret.proj$S,
              ret.proj$class, angle = pi/6, main = "Crabs K = 4")
dev.off()

## End(Not run)
```

`Post I Information Functions`

Post I Information Functions of EMCluster

Description

All post I information functions are for computing relative quantities and can be utilized by advanced developers with caution.

Currently, these are only for workflows.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

`Print and Summary` *Functions for Printing or Summarizing Objects According to Classes*

Description

Several classes are declared in **EMCluster**, and these are functions to print and summary objects.

Usage

```
## S3 method for class 'emret'
print(x, digits = max(4, getOption("digits") - 3), ...)
## S3 method for class 'emret'
summary(object, ...)
## S3 method for class 'svd'
summary(object, ...)
```

Arguments

`x`	an object with the class attributes.
`digits`	for printing out numbers.
`object`	an object with the class attributes.
`...`	other possible options.

Details

These are useful functions for summarizing and debugging.

Value

The results will cat or print on the STDOUT by default.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`, `emcluster`, `starts.via.svd`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
x2 <- da2$da

emobj <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
eobj <- e.step(x2, emobj = emobj)
emobj <- m.step(x2, emobj = eobj)
summary(emobj)

ret <- starts.via.svd(x2, nclass = 10, method = "kmeans")
summary(ret)

## End(Not run)
```

`Projection On 2D` *Produce Projection on 2D*

Description

The function projects multivariate data on 2D plane which can be displayed by `plotppcontour()` later.

Usage

```
project.on.2d(x, emobj = NULL, pi = NULL, Mu = NULL,
    LTSigma = NULL, class = NULL, method = c("PP", "SVD"))
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`pi`	the mixing proportion, length K.
`Mu`	the centers of clusters, dimension $K \times p$.
`LTSigma`	the lower triangular matrices of dispersion, $K \times p(p+1)/2$.
`class`	id of classifications, length n.
`method`	either projection pursuit or singular value decomposition.

Details

This function produces projection outputs of `x` and `emobj`.

Value

A projection is returned which is a list contains

- `da` is a $n \times 2$ projected matrix of `x`.
- `Pi` is the original proportion `emobj$pi` of length K.
- `Mu` is a $K \times 2$ projected matrix of `emboj$Mu`.
- `S` is a $2 \times 2 \times K$ projected array of `emboj$LTSigma`.
- `class` is the original class id `emobj$class`.
- `proj.mat` is the projection matrix of dimension $p \times 2$.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`project.on.2d()`.

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)

### Iris.
x <- as.matrix(iris[, 1:4])
ret <- init.EM(x, nclass = 3, min.n = 30)
ret.proj <- project.on.2d(x, ret)

### Plot.
pdf("iris_ppcontour.pdf", height = 5, width = 5)
plotppcontour(ret.proj$da, ret.proj$Pi, ret.proj$Mu, ret.proj$S,
              ret.proj$class, main = "Iris K = 3")
dev.off()

## End(Not run)
```

`Rand Index` *Rand Index and Adjusted Rand Index*

Description

This function returns the Rand index and the adjusted Rand index for given true class ids and predicted class ids.

Usage

```
RRand(trcl, prcl, lab = NULL)
```

Arguments

`trcl`	true class ids.
`prcl`	predicted class ids.
`lab`	known ids for semi-supervised clustering.

Details

All ids, `trcl` and `prcl`, should be positive integers and started from 1 to K, and the maximums are allowed to be different.

`lab` used in semi-supervised clustering contains the labels which are known before clustering. It should be positive integer and started from 1 for labeled data and 0 for unlabeled data.

Value

Return a `Class` `RRand` contains Rand index and adjusted Rand index.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

https://www.stat.iastate.edu/people/ranjan-maitra

Examples

```
library(EMCluster, quietly = TRUE)

true.id <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
pred.id <- c(2, 1, 2, 1, 1, 1, 2, 1, 1)
label   <- c(0, 0, 0, 0, 1, 0, 2, 0, 0)

RRand(true.id, pred.id)
RRand(true.id, pred.id, lab = label)
```

`Recolor Classification IDs`

Recolor Classification IDs

Description

These functions return new classification IDs.

Usage

```
recolor(id.target, id.class, scatter.class = NULL, scatter.target = NULL)
rematch(tg.id, cl.id)
recode(id)
```

Arguments

`id.target`	target class ids.
`id.class`	original class ids.
`scatter.class`	scatter class ids.
`scatter.target`	scatter target class ids.
`id`	class ids.
`tg.id`	target class ids.
`cl.id`	class ids.

Details

The function `recolor` colors `id.target` in accordance with the most likely candidate in `id.class`. Note that if scatter is present, then the class given by 0 is represented as scatter and it is assumed to be the same for both classifications.

The function `rematch` returns a list as `id.trcl` and `id.prcl`. It is the heart of the recolor function and is usually called from recolor.

The function `recode` reoders classes to eliminate group ids without any members. It is assumed that the group ids are integers.

Value

See Details.

Author(s)

Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

Examples

```
## Not run:
library(EMCluster, quietly = TRUE)

true.id <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
pred.id <- c(2, 1, 2, 1, 1, 1, 2, 1, 1)

recolor(pred.id, true.id)

## End(Not run)
```

`Single Step` *Single E- and M-step*

Description

These functions are single E- and M-step of EM algorithm for model-based clustering of finite mixture multivariate Gaussian distribution with unstructured dispersion.

Usage

```
e.step(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
          norm = TRUE)
m.step(x, emobj = NULL, Gamma = NULL, assign.class = FALSE)
```

Arguments

`x`	the data matrix, dimension $n \times p$.
`emobj`	the desired model which is a list mainly contains `pi`, `Mu`, and `LTSigma`, usually a returned object from `init.EM`.
`pi`	the mixing proportion, length K.
`Mu`	the centers of clusters, dimension $K \times p$.
`LTSigma`	the lower triangular matrices of dispersion, $K \times p(p+1)/2$.
`norm`	if returning normalized `Gamma`.
`Gamma`	containing posterior probabilities if normalized, otherwise containing component densities weighted by mixing proportion, dimension $n \times K$.
`assign.class`	if assigning class id.

Details

These two functions are mainly used in debugging for development and post process after model fitting.

Value

The `e.step` returns a list contains `Gamma`, the posterior probabilities if `norm=TRUE`, otherwise it contains component densities. This is one E-step and `Gamma` is used to update `emobj` in the M-step next.

The `m.step` returns a new `emobj` according to the `Gamma` from the E-step above.

Author(s)

Wei-Chen Chen <`wccsnow@gmail.com`> and Ranjan Maitra.

References

`https://www.stat.iastate.edu/people/ranjan-maitra`

See Also

`init.EM`.

Examples

```
library(EMCluster, quietly = TRUE)
x2 <- da2$da

emobj <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
eobj <- e.step(x2, emobj = emobj)
emobj <- m.step(x2, emobj = eobj)
emobj
```

Index