Package ‘EMCluster’

February 1, 2018

Version 0.2-10
Date 2018-01-27
Title EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution
Depends R (>= 3.0.1), MASS, Matrix
Enhances PPrree, 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.
License Mozilla Public License 2.0

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 2018-02-01 04:15:57 UTC
EMCluster-package

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

Package: EMCluster
Type: Package
License: GPL
LazyLoad: yes

The install command is simply as
Assign Class

> 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

http://maitra.public.iastate.edu/

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)

<table>
<thead>
<tr>
<th>Assign Class</th>
<th>Assign Class Id</th>
</tr>
</thead>
</table>

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 $p_i$, $\mu$, 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

http://maitra.public.iastate.edu/

See Also

init.EM, emcluster.

Examples

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

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 $p \times p$.
- **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

http://maitra.public.iastate.edu/

See Also

init.EM, emcluster.
Examples

```r
## 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)
```

Dataset

Dataset for demonstrations

Description

There are four small datasets to test and demonstrate EMCluster.

Usage

da1
da2
da3
myiris

Format

da1, da2, da3 are in list, and myiris is in matrix.

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.
myiris is selected from the original Iris dataset given by R.

Author(s)

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

References

http://maitra.public.iastate.edu/

---

**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

```r
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, \( K \times (p+1)/2 \).
- **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

http://maitra.public.iastate.edu/

See Also

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

Examples

```r
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)
```

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

 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.
 Moreover, 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

 http://maitra.public.iastate.edu/

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

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)
emicl(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, $K \times (p + 1)/2$.
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.
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

```r
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);
```

Examples

```r
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

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

http://maitra.public.iastate.edu/

See Also

`emcluster`, `EMControl`. 
**Jaccard Index**

**Examples**

```r
## Not run:
library(EMcluster, quietly = TRUE)
set.seed(1234)
x <- da$d
er1.em <- init.EM(x, nclass = 10, method = "em.EM")
er1.rnd <- init.EM(x, nclass = 10, method = "Rnd.EM", EMC = .EMC.Rnd)
emobj <- simple.init(x, nclass = 10)
er1.init <- emcluster(x, emobj, assign.class = TRUE)
par(mfrow = c(2, 2))
plotem(er1.em, x)
plotem(er1.rnd, x)
plotem(er1.init, x)
## End(Not run)
```

---

**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.
Likelihood Mixture Tests

References

http://maitra.public.iastate.edu/

Examples

```r
library(EMCluster, quietly = TRUE)

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

Description

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

Usage

```r
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 chi-squared 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 returns a p-value for testing $H_0$: $emobj.0$ v.s. $Ha$: $emobj.a$. 
LMT Functions

Value

A list of class lmt are returned.

Author(s)

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

References

http://maitra.public.iastate.edu/

See Also

init.EM.

Examples

```r
## 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)
```

---

<table>
<thead>
<tr>
<th>LMT Functions</th>
<th>Likelihood Mixture Test (LMT) Functions of EMCluster</th>
</tr>
</thead>
</table>

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

http://maitra.public.iastate.edu/

**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 supplied data matrix `x`.

**Value**

A density value is returned.

**Author(s)**

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

Two more functions with different initialization method.

Usage

\[
\text{starts.via.svd}(x, \text{nclass} = 1, \text{method} = \text{c("em", "kmeans")}, \text{EMC} = \text{.EMC}) \\
\text{emgroup}(x, \text{nclass} = 1, \text{EMC} = \text{.EMC})
\]

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 an object emobj with class emret.

Author(s)

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

References

http://maitra.public.iastate.edu/

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, ...)

minima

The function finds the minimum of an objective function using the EM algorithm.

Value

The minima function returns a list with the following elements:

- $S$: The final value of the objective function.
- $x$: The final value of the parameters.
- $y$: The final value of the log-likelihood.
- $n$: The number of iterations.
- $std$,: The standard deviation of the parameters.
- $start$: The initial value of the parameters.
- $method$: The name of the method used.
- $call$: The call that generated the object.

Author(s)

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

References

http://maitra.public.iastate.edu/

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)
Arguments

- **emobj**: the desired model which is a list mainly contains $\pi$, $\mu$, and $\Sigma$, 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 is a simple x-y lot.

Value

A plot is returned.

Author(s)

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

References

[http://maitra.public.iastate.edu/](http://maitra.public.iastate.edu/)

See Also

- `init.EM`, `emcluster`.

Examples

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

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

plotem(ret.1, x1)
## End(Not run)
```
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

http://maitra.public.iastate.edu/

See Also

init.EM, emcluster.
Examples

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

x <- myiris
ret <- em.EM(x, nclass = 5)
plotmd(x, ret$class)

## End(Not run)
```

---

### 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

```r
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`: true id of classifications if available, length $n$.
- `n.grid`: number of grid points.
- `angle`: a rotation angle (0 to $2\pi$).
- `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

http://maitra.public.iastate.edu/

See Also

`project.on.2d()`.

Examples

```r
## 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$d, ret.proj$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.
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

```r
## 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

http://maitra.public.iastate.edu/
Projection On 2D

See Also

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

Examples

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

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

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

## End(Not run)
```

Description

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

Usage

```r
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`. 
Projection On 2D

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 of length $K$.
- Mu is a $K \times 2$ projected matrix of $emobj$Mu.
- S is a $2 \times 2 \times K$ projected array of $emobj$LT$Sigma$.
- class is the original class id $emobj$class.
- proj.mat is the projection matrix of dimension $p$.

Author(s)

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

References

http://maitra.public.iastate.edu/

See Also

project.on.2d().

Examples

```R
# 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
http://maitra.public.iastate.edu/

Examples
library(EMCluster, quietly = TRUE)

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

RRand(true.id, pred.id)
RRand(true.id, pred.id, lab = label)
Recolor Classification IDs

Description

These functions return new classification IDs.

Usage

```r
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` reorders 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

[http://maitra.public.iastate.edu/](http://maitra.public.iastate.edu/)
Examples

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

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

```r
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 \( \Lambda \Sigma \), 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

http://maitra.public.iastate.edu/

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**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Functions/Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LMT</strong></td>
<td>LMT Functions, 15</td>
</tr>
<tr>
<td><strong>datasets</strong></td>
<td>Dataset, 6</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>EMCluster-package, 2</td>
</tr>
<tr>
<td><strong>post_I_information</strong></td>
<td>Post I Information Functions, 22</td>
</tr>
<tr>
<td><strong>programming</strong></td>
<td>EM Algorithm, 7</td>
</tr>
<tr>
<td></td>
<td>EM Control, 8</td>
</tr>
<tr>
<td></td>
<td>Initialization and EM, 11</td>
</tr>
<tr>
<td></td>
<td>Likelihood Mixture Tests, 14</td>
</tr>
<tr>
<td></td>
<td>Other Initializations, 17</td>
</tr>
<tr>
<td><strong>summary</strong></td>
<td>Jaccard Index, 13</td>
</tr>
<tr>
<td></td>
<td>Print and Summary, 23</td>
</tr>
<tr>
<td></td>
<td>Rand Index, 26</td>
</tr>
<tr>
<td></td>
<td>Recolor Classification IDs, 27</td>
</tr>
<tr>
<td><strong>tool</strong></td>
<td>Assign Class, 3</td>
</tr>
<tr>
<td></td>
<td>Information Criteria, 10</td>
</tr>
<tr>
<td></td>
<td>Single Step, 28</td>
</tr>
<tr>
<td><strong>utility</strong></td>
<td>Conversion, 4</td>
</tr>
<tr>
<td></td>
<td>MVN, 16</td>
</tr>
<tr>
<td></td>
<td>Plot EM Results, 18</td>
</tr>
<tr>
<td></td>
<td>Plot Multivariate Data, 20</td>
</tr>
<tr>
<td></td>
<td>Plot Projection and Contour, 21</td>
</tr>
<tr>
<td></td>
<td>Projection On 2D, 24</td>
</tr>
<tr>
<td></td>
<td>.EMC (EM Control), 8</td>
</tr>
<tr>
<td></td>
<td>.EMControl, 8, 12, 18</td>
</tr>
<tr>
<td></td>
<td>.EMControl (EM Control), 8</td>
</tr>
<tr>
<td></td>
<td>Assign Class, 3</td>
</tr>
<tr>
<td></td>
<td>assign.class (Assign Class), 3</td>
</tr>
<tr>
<td></td>
<td>bh.fdr (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>class2Gamma (Conversion), 4</td>
</tr>
<tr>
<td></td>
<td>col.ppcontour (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>Conversion, 4</td>
</tr>
<tr>
<td></td>
<td>da1 (Dataset), 6</td>
</tr>
<tr>
<td></td>
<td>da2 (Dataset), 6</td>
</tr>
<tr>
<td></td>
<td>da3 (Dataset), 6</td>
</tr>
<tr>
<td></td>
<td>Dataset, 6</td>
</tr>
<tr>
<td></td>
<td>dlmvn (MVN), 16</td>
</tr>
<tr>
<td></td>
<td>dmixmvn (MVN), 16</td>
</tr>
<tr>
<td></td>
<td>dmvn (MVN), 16</td>
</tr>
<tr>
<td></td>
<td>e.step, 8</td>
</tr>
<tr>
<td></td>
<td>e.step (Single Step), 28</td>
</tr>
<tr>
<td></td>
<td>EM Algorithm, 7</td>
</tr>
<tr>
<td></td>
<td>EM Control, 8</td>
</tr>
<tr>
<td></td>
<td>em.aic (Information Criteria), 10</td>
</tr>
<tr>
<td></td>
<td>em.bic (Information Criteria), 10</td>
</tr>
<tr>
<td></td>
<td>em.clc (Information Criteria), 10</td>
</tr>
<tr>
<td></td>
<td>em.EM (Initialization and EM), 11</td>
</tr>
<tr>
<td></td>
<td>em.ic (Information Criteria), 10</td>
</tr>
<tr>
<td></td>
<td>em.icl (Information Criteria), 10</td>
</tr>
<tr>
<td></td>
<td>EMCluster (EMCluster-package), 2</td>
</tr>
<tr>
<td></td>
<td>emcluster, 3–5, 9, 12, 17, 19, 20, 24</td>
</tr>
<tr>
<td></td>
<td>emcluster (EM Algorithm), 7</td>
</tr>
<tr>
<td></td>
<td>EMCluster-package, 2</td>
</tr>
<tr>
<td></td>
<td>emgroup (Other Initializations), 17</td>
</tr>
<tr>
<td></td>
<td>emobj object (EM Algorithm), 7</td>
</tr>
<tr>
<td></td>
<td>exhaust.EM (Initialization and EM), 11</td>
</tr>
<tr>
<td></td>
<td>fill.ppcontour (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>fill.ppmu (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>fill.pppoints (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>Gamma2class (Conversion), 4</td>
</tr>
<tr>
<td></td>
<td>GenDataSet (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>GenMixDataSet (LMT Functions), 15</td>
</tr>
<tr>
<td></td>
<td>get.cov.logit.PI (Post I Information Functions), 22</td>
</tr>
<tr>
<td></td>
<td>get.cov.logit.z (Post I Information Functions), 22</td>
</tr>
</tbody>
</table>
get.cov.param (Post I Information Functions), 22
get.cov.post.z (Post I Information Functions), 22
g.E.chi2 (LMT Functions), 15
g.E.d (LMT Functions), 15
g.logor.stat (Post I Information Functions), 22
Gmat (LMT Functions), 15
gridOne (LMT Functions), 15
Information Criteria, 10
init.EM, 3–5, 8, 9, 11, 15, 17–20, 24, 29
init.EM (Initialization and EM), 11
Initialization and EM, 11
Iy (LMT Functions), 15
Iy2 (LMT Functions), 15
Jaccard Index, 13
Jaccard.Index (Jaccard Index), 13
Likelihood Mixture Tests, 14
lmt (Likelihood Mixture Tests), 14
LMT Functions, 15
lgl (MVN), 16
LTSigma2var (Conversion), 4
LTSigma2variance (Conversion), 4
m.step, 8
m.step (Single Step), 28
MVN, 16
myiris (Dataset), 6
Other Initializations, 17
partial.f.mu.s (Post I Information Functions), 22
partial.logit.p (Post I Information Functions), 22
partial.logL (LMT Functions), 15
partial.post.z (Post I Information Functions), 22
partial.q (LMT Functions), 15
pchisq.my (LMT Functions), 15
Plot EM Results, 18
Plot Multivariate Data, 20
Plot Projection and Contour, 21
plot2d (Plot EM Results), 18
plotem (Plot EM Results), 18
plotmd (Plot Multivariate Data), 20
plotp (LMT Functions), 15
plotppcontour, 24
plotppcontour (Plot Projection and Contour), 21
plotq (LMT Functions), 15
Post I Information Functions, 22
postPI (LMT Functions), 15
Print and Summary, 23
print.emret (Print and Summary), 23
print.lmt (LMT Functions), 15
project.on.2d, 21, 22, 25
project.on.2d (Projection On 2D), 24
Projection On 2D, 24
q.map.new (LMT Functions), 15
Rand Index, 26
rand.EM (Initialization and EM), 11
recode (Recolor Classification IDs), 27
recolor (Recolor Classification IDs), 27
Recolor Classification IDs, 27
rematch (Recolor Classification IDs), 27
RRand (Rand Index), 26
shortemcluster (EM Algorithm), 7
simple.init (EM Algorithm), 7
Single Step, 28
starts.via.svd, 24
starts.via.svd (Other Initializations), 17
summary.emret (Print and Summary), 23
summary.svd (Print and Summary), 23
var2LTSigma (Conversion), 4
variance2LTSigma (Conversion), 4
w.2 (LMT Functions), 15