Package 'RDRToolbox'

April 24, 2025

Type Package

Title A package for nonlinear dimension reduction with Isomap and LLE.

Version 1.58.0

Date 2019-01-22

Author Christoph Bartenhagen

Maintainer Christoph Bartenhagen <c.bartenhagen@uni-koeln.de>

Description A package for nonlinear dimension reduction using the Isomap and LLE algorithm. It also includes a routine for computing the Davis-Bouldin-Index for cluster validation, a plotting tool and a data generator for microarray gene expression data and for the Swiss Roll dataset.

License GPL (>= 2)

LazyLoad yes

Depends R (>= 2.9.0)

Imports graphics, grDevices, methods, stats, MASS, rgl

Suggests golubEsets

biocViews DimensionReduction, FeatureExtraction, Visualization, Clustering, Microarray

git_url https://git.bioconductor.org/packages/RDRToolbox

git_branch RELEASE_3_21

git_last_commit f6af32b

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-04-23

Contents

DBIndex 2
generateData 3
Isomap 4

LLE 5
plotDR 6
SwissRoll 7

Index **9**

`DBIndex` *Davis-Bouldin-Index*

Description

Computes the Davis-Bouldin-Index for cluster validation purposes.

Usage

```
DBIndex(data, labels)
```

Arguments

`data`	N x D matrix (N samples, D features)
`labels`	a vector of class labels

Details

To compute a clusters' compactness, this version uses the Euclidean distance to determine the mean distances between the samples and the cluster centers. Furthermore, the distance of two clusters is given by the distance of their centers.

Value

'DBIndex' returns the Davis-Bouldin cluster index, a numeric value.

Author(s)

Christoph Bartenhagen

Examples

```
## DB-Index of a 50 dimensional dataset with 20 samples separated into two classes
d = generateData(samples=20, genes=50, diffgenes=10, blocksize=5)
DBIndex (data=d[[1]], labels=d[[2]])
```

`generateData` *Simulator for gene expression data*

Description

A simulator for gene expression data, whose values are normally distributed values with zero mean. The covariances are given by a configurable block-diagonal matrix. By default, half of the samples contain differential gene expression values (see parameter `diffsamples`).

Usage

```
generateData(samples=50, genes=10000, diffgenes=200, blocksize=50, cov1=0.2, cov2=0, diff=0.6, diffsam
```

Arguments

`samples`	number of samples
`genes`	number of gene expression values per sample
`diffgenes`	number of differential genes for class 1
`blocksize`	size of each block in the blockdiagonal correlation matrix
`cov1`	covariance within the blocks in the correlation matrix
`cov2`	covariance between the blocks in the correlation matrix
`diff`	difference between the random gene expression values and the differential gene expression values
`diffsamples`	number of samples containing differential gene expression values compared to the rest (if missing, this parameter is set to half of the total number of samples)

Details

The simulator generates two labeled classes:
label 1: samples with differentially expressed genes.
label -1: samples without differentially expressed genes.

Value

'generateData' returns a list containing:

`data`	a (samples x features)-matrix with the simulated gene expression values
`labels`	a vector with labels (1,-1) for the two classes

Author(s)

Christoph Bartenhagen

Examples

```
## generate a dataset with 20 samples and 1.000 gene expression values
d = generateData(samples=20, genes=1000, diffgenes=100, blocksize=10)
data = d[[1]]
labels = d[[2]]
```

`Isomap` *Isomap*

Description

Computes the Isomap embedding as introduced in 2000 by Tenenbaum, de Silva and Langford.

Usage

```
Isomap(data, dims = 2, k, mod = FALSE, plotResiduals = FALSE, verbose = TRUE)
```

Arguments

`data`	N x D matrix (N samples, D features)
`dims`	vector containing the target space dimension(s)
`k`	number of neighbours
`mod`	use modified Isomap algorithm
`plotResiduals`	show a plot with the residuals between the high and the low dimensional data
`verbose`	show a summary of the embedding procedure at the end

Details

Isomap is a nonlinear dimension reduction technique, that preserves global properties of the data. That means, that geodesic distances between all samples are captured best in the low dimensional embedding.
This R version is based on the Matlab implementation by Tenenbaum and uses Floyd's Algorithm to compute the neighbourhood graph of shortest distances, when calculating the geodesic distances. A modified version of the original Isomap algorithm is included. It respects nearest and farthest neighbours.
To estimate the intrinsic dimension of the data, the function can plot the residuals between the high and the low dimensional data for a given range of dimensions.

Value

It returns a N x dim matrix (N samples, dim features) with the reduced input data (list of several matrices if more than one dimension was specified)

Author(s)

Christoph Bartenhagen

References

Tenenbaum, J. B. and de Silva, V. and Langford, J. C., "A global geometric framework for nonlinear dimensionality reduction.", 2000; Matlab code is available at http://waldron.stanford.edu/~isomap/

Examples

```
## two dimensional Isomap embedding of a 1.000 dimensional dataset using k=5 neighbours
d = generateData(samples=20, genes=1000, diffgenes=100, blocksize=10)
d_low = Isomap(data=d[[1]], dims=2, k=5)
## Isomap residuals for target dimensions 1-10
d_low = Isomap(data=d[[1]], dims=1:10, k=5, plotResiduals=TRUE)

## three dimensional Isomap embedding of a 1.000 dimensional dataset using k=10 (nearest and farthest) neighbours
d = generateData(samples=20, genes=1000, diffgenes=100, blocksize=10)
d_low = Isomap(data=d[[1]], dims=3, mod=TRUE, k=10)
```

LLE *Locally Linear Embedding*

Description

Computes the Locally Linear Embedding as introduced in 2000 by Roweis, Saul and Lawrence.

Usage

```
LLE(data, dim=2, k)
```

Arguments

data	N x D matrix (N samples, D features)
dim	dimension of the target space
k	number of neighbours

Details

Locally Linear Embedding (LLE) preserves local properties of the data by representing each sample in the data by a linear combination of its k nearest neighbours with each neighbour weighted independently. LLE finally chooses the low-dimensional representation that best preserves the weights in the target space.
This R version is based on the Matlab implementation by Sam Roweis.

Value

It returns a N x dim matrix (N samples, dim features) with the reduced input data

Author(s)

Christoph Bartenhagen

References

Roweis, Sam T. and Saul, Lawrence K., "Nonlinear Dimensionality Reduction by Locally Linear Embedding",2000;

Examples

```
## two dimensional LLE embedding of a 1.000 dimensional dataset using k=5 neighbours
d = generateData(samples=20, genes=1000, diffgenes=100, blocksize=10)
d_low = LLE(data=d[[1]], dim=2, k=5)
```

`plotDR` *Plotting tool for two and three dimensional data*

Description

Creates two and three dimensional plots of (labeled) data. It uses the library "rgl" for rotatable 3D scatterplots.

Usage

```
plotDR(data, labels, axesLabels=c("x","y","z"), legend=FALSE, text, col, pch, ...)
```

Arguments

`data`	matrix with values to be plotted (rows correspond to samples, columns to features)
`labels`	vector containing labels of the classes within the data (optional)
`axesLabels`	vector containing labels for the axes of the plot
`legend`	logical value whether to automatically insert a legend into the plot
`text`	vector with (short) labels for each point (optional)
`col`	character vector of colours for each class (optional); see `colors()` to display a list of available colours
`pch`	character or integer value specifying the symbol when plotting points (see `?par` or `?points` for more info)
`...`	other common R plot parameters like for example `las`, `cex` or `font` to further customize the plot (see `?par` for possible arguments); some parameters may only work with two dimensional plots

Details

It colours the data points according to given class labels (max. six classes when using default colours). A legend will be printed in the R console by default (for three dimensional plots, a legend is not supported).

Author(s)

Christoph Bartenhagen

Examples

```
## plot a two dimensional LLE embedding of a 1.000 dimensional dataset
d = generateData(samples=20, genes=1000, diffgenes=100, blocksize=10)
d_low = LLE(data=d[[1]], dim=2, k=5)
plotDR(data=d_low, labels=d[[2]])

## plot a two dimensional LLE embedding of a 1.000 dimensional dataset
## add axis labels, a legend and plot a text for each sample
d = generateData(samples=20, genes=1000, diffgenes=100, blocksize=10)
d_low = LLE(data=d[[1]], dim=2, k=5)
text = letters[1:20]
plotDR(data=d_low, labels=d[[2]], axesLabels=c("first component", "second component"), text=text, legend=TRUE)
## manually add a legend to the plot
plotDR(data=d_low, labels=d[[2]], axesLabels=c("first component", "second component"), text=text)
legend("topright", legend=c("class 1","class 2"), col=c("black", "red"), pch=1)
```

SwissRoll *The Swiss Roll dataset*

Description

Computes and plots the Swiss Roll dataset of a given size and height. It uses the library "rgl" for rotatable 3D scatterplots.

Usage

```
SwissRoll(N = 2000, Height = 30, Plot=FALSE)
```

Arguments

N	number of samples
Height	controls the spreading of the samples in the second dimension
Plot	a boolean specifying whether to plot the Swiss Roll dataset or not

Value

'SwissRoll' returns all N samples as a Nx3-matrix

Author(s)

Christoph Bartenhagen

Examples

```
## compute and plot a Swiss Roll dataset with 1.000 samples
data=SwissRoll(N = 1000, Plot=TRUE)
```

Index

DBIndex, 2

generateData, 3

Isomap, 4

LLE, 5

plotDR, 6

SwissRoll, 7