Package 'dimRed'

May 8, 2019

Title A Framework for Dimensionality Reduction

Version 0.2.3

Description A collection of dimensionality reduction techniques from R packages and a common interface for calling the methods.

Depends R (>= 3.0.0), DRR

Imports magrittr, methods

Suggests NMF, MASS, Matrix, RANN, RSpectra, Rtsne, cccd, coRanking, diffusionMap, energy, fastICA, ggplot2, graphics, igraph, keras, kernlab, knitr, lle, loe, optimx, pcaL1, pcaPP, reticulate, rgl, scales, scatterplot3d, stats, tensorflow, testthat, tidyr, tinytex, umap, vegan

VignetteBuilder knitr

License GPL-3 | file LICENSE

URL `https://github.com/gdkrmr/dimRed`

LazyData true

Encoding UTF-8

Collate 'dimRedMethod-class.R' 'misc.R' 'dimRedData-class.R' 'dimRedResult-class.R' 'autoencoder.R' 'dataSets.R' 'diffmap.R' 'dimRed.R' 'drr.R' 'embed.R' 'fastica.R' 'get_info.R' 'graph_embed.R' 'hlle.R' 'isomap.R' 'kpca.R' 'l1pca.R' 'leim.R' 'lle.R' 'loe.R' 'mds.R' 'mixColorSpaces.R' 'nmds.R' 'nnmf.R' 'pca.R' 'plot.R' 'quality.R' 'rotate.R' 'soe.R' 'tsne.R' 'umap.R'

RoxygenNote 6.1.1

NeedsCompilation yes

Author Guido Kraemer [aut, cre]

Maintainer Guido Kraemer <`gkraemer@bgc-jena.mpg.de`>

Repository CRAN

Date/Publication 2019-05-08 08:10:07 UTC

R topics documented:

`dimRed-package` *The dimRed package*

Description

This package simplifies dimensionality reduction in R by providing a framework of S4 classes and methods. dimRed collects dimensionality reduction methods that are implemented in R and implements others. It gives them a common interface and provides plotting functions for visualization and functions for quality assessment.

Funding provided by the Department for Biogeochemical Integration, Empirical Inference of the Earth System Group, at the Max Plack Institute for Biogeochemistry, Jena.

Author(s)

Maintainer: Guido Kraemer <`gkraemer@bgc-jena.mpg.de`>

References

Lee, J.A., Renard, E., Bernard, G., Dupont, P., Verleysen, M., 2013. Type 1 and 2 mixtures of Kullback-Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. Neurocomputing. 112, 92-107. doi:10.1016/j.neucom.2012.12.036

Lee, J.A., Lee, J.A., Verleysen, M., 2008. Rank-based quality assessment of nonlinear dimensionality reduction. Proceedings of ESANN 2008 49-54.

Chen, L., Buja, A., 2006. Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Layout and Proximity Analysis.

See Also

Useful links:

- `https://github.com/gdkrmr/dimRed`

as.data.frame *Converts to data.frame*

Description

General conversions of objects created by `dimRed` to `data.frame`. See class documentations for details (`dimRedData`, `dimRedResult`). For the documentation of this function in base package, see here: `as.data.frame.default`.

Usage

```
as.data.frame(x, row.names, optional, ...)
```

Arguments

`x`	The object to be converted
`row.names`	unused in `dimRed`
`optional`	unused in `dimRed`
`...`	other arguments.

as.dimRedData *Converts to dimRedData*

Description

Conversion functions to dimRedData.

Usage

```
as.dimRedData(formula, ...)

## S4 method for signature 'formula'
as.dimRedData(formula, data)
```

Arguments

`formula`	The formula, left hand side is assigned to the meta slot right hand side is assigned to the data slot.
`...`	other arguments.
`data`	Will be coerced into a `data.frame` with `as.data.frame`

Methods (by class)

- `formula`: Convert a `data.frame` to a dimRedData object using a formula

See Also

Other dimRedData: `dimRedData-class`

Examples

```
## create a dimRedData object using a formula
as.dimRedData(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
              iris)[1:5]
```

`AUC_lnK_R_NX,dimRedResult-method`
Method AUC_lnK_R_NX

Description

Calculate the Area under the R_NX(ln K), used in Lee et. al. (2015). Note that despite the name, this does not weight the mean by the logarithm, but by 1/K. If explicit weighting by the logarithm is desired use `weight = "log"` or `weight = "log10"`

Usage

```
## S4 method for signature 'dimRedResult'
AUC_lnK_R_NX(object, weight = "inv")
```

Arguments

`object`	of class dimRedResult
`weight`	the weight function used, one of `c("inv", "log", "log10")`

Details

The naming confusion originated from equation 17 in Lee et al (2015) and the name of this method may change in the future to avoid confusion.

References

Lee, J.A., Peluffo-Ordonez, D.H., Verleysen, M., 2015. Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. Neurocomputing 169, 246-261. https://doi.org/10.1016/j.neucom.2014.12.095

See Also

Other Quality scores for dimensionality reduction: `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu`
`reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

`AutoEncoder-class` *AutoEncoder*

Description

An S4 Class implementing an Autoencoder

Details

Autoencoders are neural networks that try to reproduce their input. Consider this method unstable, as the internals may still be changed.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

Autoencoder can take the following parameters:

ndim The number of dimensions for reduction.

n_hidden The number of neurons in the hidden layers, the length specifies the number of layers, the length must be impair, the middle number must be the same as ndim.

activation The activation functions for the layers, one of "tanh", "sigmoid", "relu", "elu", everything else will silently be ignored and there will be no activation function for the layer.

weight_decay the coefficient for weight decay, set to 0 if no weight decay desired.

learning_rate The learning rate for gradient descend

graph Optional: A list of bits and pieces that define the autoencoder in tensorflow, see details.

keras_graph Optional: A list of keras layers that define the encoder and decoder, specifying this, will ignore all other topology related variables, see details.

batchsize If NA, all data will be used for training, else only a random subset of size batchsize will be used

n_steps the number of training steps.

Details

There are several ways to specify an autoencoder, the simplest is to pass the number of neurons per layer in `n_hidden`, this must be a vector of integers of impair length and it must be symmetric and the middle number must be equal to `ndim`, For every layer an activation function can be specified with `activation`.

For regularization weight decay can be specified by setting `weight_decay` > 0.

Currently only a gradient descent optimizer is used, the learning rate can be specified by setting `learning_rate`. The learner can operate on batches if `batchsize` is not `NA`. The number of steps the learner uses is specified using `n_steps`.

Further training a model

If the model did not converge in the first training phase or training with different data is desired, the `dimRedResult` object may be passed as `autoencoder` parameter; In this case all topology related parameters will be ignored.

Using Keras layers

The encoder and decoder part can be specified using a list of **keras** layers. This requires a list with two entries, `encoder` should contain a LIST of keras layers WITHOUT the `layer_input` that will be concatenated in order to form the encoder part. `decoder` should be defined accordingly, the output of `decoder` must have the same number of dimensions as the input data.

Using Tensorflow

The model can be entirely defined in **tensorflow**, it must contain a list with the following entries:

encoder A tensor that defines the encoder.

decoder A tensor that defines the decoder.

network A tensor that defines the reconstruction (encoder + decoder).

loss A tensor that calculates the loss (network + loss function).

in_data A `placeholder` that points to the data input of the network AND the encoder.

in_decoder A `placeholder` that points to the input of the decoder.

session A **tensorflow** `Session` object that holds the values of the tensors.

Implementation

Uses **tensorflow** as a backend, for details an problems relating tensorflow, see `https://tensorflow.rstudio.com`.

See Also

Other dimensionality reduction methods: `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
## Not run:
dat <- loadDataSet("3D S Curve")

## use the S4 Class directly:
autoenc <- AutoEncoder()
emb <- autoenc@fun(dat, autoenc@stdpars)

## simpler, use embed():
emb2 <- embed(dat, "AutoEncoder")

plot(emb, type = "2vars")

samp <- sample(floor(nrow(dat) / 10))
embsamp <- autoenc@fun(dat[samp], autoenc@stdpars)
embother <- embsamp@apply(dat[-samp])
plot(embsamp, type = "2vars")
points(embother@data)

## End(Not run)
```

`cophenetic_correlation,dimRedResult-method` *Method cophenetic_correlation*

Description

Calculate the correlation between the distance matrices in high and low dimensioal space.

Usage

```
## S4 method for signature 'dimRedResult'
cophenetic_correlation(object, d = stats::dist,
  cor_method = "pearson")
```

Arguments

`object`	of class dimRedResult
`d`	the distance function to use.
`cor_method`	The correlation method.

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResult-method`, `reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

`dataSets` *Example Data Sets for dimensionality reduction*

Description

A compilation of standard data sets that are often being used to showcase dimensionality reduction techniques.

Usage

```
loadDataSet(name = dataSetList(), n = 2000, sigma = 0.05)

dataSetList()
```

Arguments

`name`	A character vector that specifies the name of the data set.
`n`	In generated data sets the number of points to be generated, else ignored.
`sigma`	In generated data sets the standard deviation of the noise added, else ignored.

Details

The argument `name` should be one of `dataSetList()`. Partial matching is possible, see `match.arg`. Generated data sets contain the internal coordinates of the manifold in the `meta` slot. Call `dataSetList()` to see what data sets are available.

Value

`loadDataSet` an object of class `dimRedData`. `dataSetList()` return a character string with the implemented data sets

Examples

```
## a list of available data sets:
dataSetList()

## Load a data set:
swissRoll <- loadDataSet("Swiss Roll")
plot(swissRoll, type = "3vars")

## Load Iris data set, partial matching:
loadDataSet("I")
```

`DiffusionMaps-class` *Diffusion Maps*

Description

An S4 Class implementing Diffusion Maps

Details

Diffusion Maps uses a diffusion probability matrix to robustly approximate a manifold.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

Diffusion Maps can take the following parameters:

d a function transforming a matrix row wise into a distance matrix or `dist` object, e.g. `dist`.

ndim The number of dimensions

eps The epsilon parameter that determines the diffusion weight matrix from a distance matrix `d`, $exp(-d^2/eps)$, if set to `"auto"` it will be set to the median distance to the 0.01*n nearest neighbor.

t Time-scale parameter. The recommended value, 0, uses multiscale geometry.

delta Sparsity cut-off for the symmetric graph Laplacian, a higher value results in more sparsity and faster calculation. The predefined value is 10^-5.

Implementation

Wraps around `diffuse`, see there for details. It uses the notation of Richards et al. (2009) which is slightly different from the one in the original paper (Coifman and Lafon, 2006) and there is no α parameter. There is also an out-of-sample extension, see examples.

References

Richards, J.W., Freeman, P.E., Lee, A.B., Schafer, C.M., 2009. Exploiting Low-Dimensional Structure in Astronomical Spectra. ApJ 691, 32. doi:10.1088/0004-637X/691/1/32

Coifman, R.R., Lafon, S., 2006. Diffusion maps. Applied and Computational Harmonic Analysis 21, 5-30. doi:10.1016/j.acha.2006.04.006

See Also

Other dimensionality reduction methods: AutoEncoder-class, DRR-class, DrL-class, FastICA-class, FruchtermanReingold-class, HLLE-class, Isomap-class, KamadaKawai-class, LLE-class, MDS-class, NNMF-class, PCA-class, PCA_L1-class, UMAP-class, dimRedMethod-class, dimRedMethodList, kPCA-class, nMDS-class, tSNE-class

Examples

```
dat <- loadDataSet("3D S Curve", n = 300)

## use the S4 Class directly:
diffmap <- DiffusionMaps()
emb <- diffmap@fun(dat, diffmap@stdpars)

## simpler, use embed():
emb2 <- embed(dat, "DiffusionMaps")

plot(emb, type = "2vars")

samp <- sample(floor(nrow(dat) / 10))
embsamp <- diffmap@fun(dat[samp], diffmap@stdpars)
embother <- embsamp@apply(dat[-samp])
plot(embsamp, type = "2vars")
points(embother@data)
```

dimRedData-class *Class "dimRedData"*

Description

A class to hold data for dimensionality reduction and methods.

Usage

```
## S4 method for signature 'dimRedData'
as.data.frame(x, meta.prefix = "meta.",
  data.prefix = "")

## S4 method for signature 'dimRedData'
getData(object)

## S4 method for signature 'dimRedData'
getMeta(object)

## S4 method for signature 'dimRedData'
nrow(x)
```

```
## S4 method for signature 'dimRedData,ANY,ANY,ANY'
x[i]

## S4 method for signature 'dimRedData'
ndims(object)
```

Arguments

`x`	Of class dimRedData
`meta.prefix`	Prefix for the columns of the meta data names.
`data.prefix`	Prefix for the columns of the variable names.
`object`	Of class dimRedData.
`i`	a valid index for subsetting rows.

Details

The class hast two slots, `data` and `meta`. The `data` slot contains a `numeric matrix` with variables in columns and observations in rows. The `meta` slot may contain a `data.frame` with additional information. Both slots need to have the same number of rows or the `meta` slot needs to contain an empty `data.frame`.

See examples for easy conversion from and to `data.frame`.

For plotting functions see `plot.dimRedData`.

Methods (by generic)

- `as.data.frame`: convert to data.frame
- `getData`: Get the data slot.
- `getMeta`: Get the meta slot.
- `nrow`: Get the number of observations.
- `[`: Subset rows.
- `ndims`: Extract the number of Variables from the data.

Slots

`data` of class `matrix`, holds the data, observations in rows, variables in columns

`meta` of class `data.frame`, holds meta data such as classes, internal manifold coordinates, or simply additional data of the data set. Must have the same number of rows as the `data` slot or be an empty data frame.

See Also

Other dimRedData: `as.dimRedData`

Other dimRedData: `as.dimRedData`

Examples

```
## Load an example data set:
s3d <- loadDataSet("3D S Curve")

## Create using a constructor:

### without meta information:
dimRedData(iris[, 1:4])

### with meta information:
dimRedData(iris[, 1:4], iris[, 5])

### using slot names:
dimRedData(data = iris[, 1:4], meta = iris[, 5])

## Convert to a dimRedData objects:
Iris <- as(iris[, 1:4], "dimRedData")

## Convert to data.frame:
head(as(s3d, "data.frame"))
head(as.data.frame(s3d))
head(as.data.frame(as(iris[, 1:4], "dimRedData")))

## Extract slots:
head(getData(s3d))
head(getMeta(s3d))

## Get the number of observations:
nrow(s3d)

## Subset:
s3d[1:5, ]

## Shuffle data:
s3 <- s3d[nrow(s3d)]

## Get the number of variables:
ndims(s3d)
```

`dimRedMethod-class` *Class "dimRedMethod"*

Description

A virtual class "dimRedMethod" to serve as a template to implement methods for dimensionality reduction.

Details

Implementations of dimensionality reductions should inherit from this class.

The `fun` slot should be a function that takes three arguments

data An object of class `dimRedData`.

pars A list with the standard parameters.

keep.org.data Logical. If the original data should be kept in the output.

and returns an object of class `dimRedResult`.

The `stdpars` slot should take a list that contains standard parameters for the implemented methods.

This way the method can be called by `embed(data, "method-name",...)`, where `...` can be used to to change single parameters.

Slots

`fun` A function that does the embedding.

`stdpars` A list with the default parameters for the `fun` slot.

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

`dimRedMethodList` *dimRedMethodList*

Description

Get the names of all methods for dimensionality reduction.

Usage

```
dimRedMethodList()
```

Details

Returns the name of all classes that inherit from `dimRedMethod-class` to use with `embed`.

Value

a character vector with the names of classes that inherit from `dimRedMethod`.

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dimRedMethodList()
```

`dimRedResult-class` *Class "dimRedResult"*

Description

A class to hold the results of of a dimensionality reduction.

Usage

```
## S4 method for signature 'dimRedResult'
predict(object, xnew)

## S4 method for signature 'dimRedResult'
inverse(object, ynew)

## S4 method for signature 'dimRedResult'
as.data.frame(x, org.data.prefix = "org.",
  meta.prefix = "meta.", data.prefix = "")

## S4 method for signature 'dimRedResult'
getPars(object)

## S4 method for signature 'dimRedResult'
getNDim(object)

## S4 method for signature 'dimRedResult'
print(x)

## S4 method for signature 'dimRedResult'
getOrgData(object)

## S4 method for signature 'dimRedResult'
getDimRedData(object)

## S4 method for signature 'dimRedResult'
ndims(object)
```

```
## S4 method for signature 'dimRedResult'
getOtherData(object)
```

Arguments

`object`	Of class `dimRedResult`
`xnew`	new data, of type `dimRedData`
`ynew`	embedded data, of type `dimRedData`
`x`	Of class `dimRedResult`
`org.data.prefix`	Prefix for the columns of the org.data slot.
`meta.prefix`	Prefix for the columns of `x@data@meta`.
`data.prefix`	Prefix for the columns of `x@data@data`.

Methods (by generic)

- `predict`: apply a trained method to new data, does not work with all methods, will give an error if there is no `apply`. In some cases the apply function may only be an approximation.
- `inverse`: inverse transformation of embedded data, does not work with all methods, will give an error if there is no `inverse`. In some cases the apply function may only be an approximation.
- `as.data.frame`: convert to `data.frame`
- `getPars`: Get the parameters with which the method was called.
- `getNDim`: Get the number of embedding dimensions.
- `print`: Method for printing.
- `getOrgData`: Get the original data and meta.data
- `getDimRedData`: Get the embedded data
- `ndims`: Extract the number of embedding dimensions.
- `getOtherData`: Get other data produced by the method

Slots

`data` Output data of class dimRedData.

`org.data` original data, a matrix.

`apply` a function to apply the method to out-of-sampledata, may not exist.

`inverse` a function to calculate the original coordinates from reduced space, may not exist.

`has.org.data` logical, if the original data is included in the object.

`has.apply` logical, if a forward method is exists.

`has.inverse` logical if an inverse method exists.

`method` saves the method used.

`pars` saves the parameters used.

`other.data` other data produced by the method, e.g. a distance matrix.

Examples

```
## Create object by embedding data
iris.pca <- embed(loadDataSet("Iris"), "PCA")

## Convert the result to a data.frame
head(as(iris.pca, "data.frame"))
head(as.data.frame(iris.pca))

## There are no nameclashes to avoid here:
head(as.data.frame(iris.pca,
                   org.data.prefix = "",
                   meta.prefix     = "",
                   data.prefix     = ""))

## Print it more or less nicely:
print(iris.pca)

## Get the embedded data as a dimRedData object:
getDimRedData(iris.pca)

## Get the original data including meta information:
getOrgData(iris.pca)

## Get the number of variables:
ndims(iris.pca)
```

`distance_correlation,dimRedResult-method`
Method distance_correlation

Description

Calculate the distance correlation between the distance matrices in high and low dimensioal space.

Usage

```
## S4 method for signature 'dimRedResult'
distance_correlation(object)
```

Arguments

`object`	of class dimRedResult

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, plot_R_NX, quality,dimRedResult-method, reconstruction_error,dimRedResult-method, reconstruction_rmse,dimRedResult-method, total_correlation,dimRedResult-method

---

DrL-class *Distributed Recursive Graph Layout*

Description

An S4 Class implementing Distributed recursive Graph Layout.

Details

DrL uses a complex algorithm to avoid local minima in the graph embedding which uses several steps.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

DrL can take the following parameters:

ndim The number of dimensions, defaults to 2. Can only be 2 or 3

knn Reduce the graph to keep only the neares neighbors. Defaults to 100.

d The distance function to determine the weights of the graph edges. Defaults to euclidean distances.

Implementation

Wraps around `layout_with_drl`. The parameters maxiter, epsilon and kkconst are set to the default values and cannot be set, this may change in a future release. The DimRed Package adds an extra sparsity parameter by constructing a knn graph which also may improve visualization quality.

References

Martin, S., Brown, W.M., Wylie, B.N., 2007. Dr.l: Distributed Recursive (graph) Layout (No. dRl; 002182MLTPL00). Sandia National Laboratories.

See Also

Other dimensionality reduction methods: AutoEncoder-class, DRR-class, DiffusionMaps-class, FastICA-class, FruchtermanReingold-class, HLLE-class, Isomap-class, KamadaKawai-class, LLE-class, MDS-class, NNMF-class, PCA-class, PCA_L1-class, UMAP-class, dimRedMethod-class, dimRedMethodList, kPCA-class, nMDS-class, tSNE-class

Examples

```
## Not run:
dat <- loadDataSet("Swiss Roll", n = 300)

## use the S4 Class directly:
drl <- DrL()
emb <- drl@fun(dat, drl@stdpars)

## simpler, use embed():
emb2 <- embed(dat, "DrL")


plot(emb)

## End(Not run)
```

DRR-class *Dimensionality Reduction via Regression*

Description

An S4 Class implementing Dimensionality Reduction via Regression (DRR).

Details

DRR is a non-linear extension of PCA that uses Kernel Ridge regression.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

DRR can take the following parameters:

ndim The number of dimensions

lambda The regularization parameter for the ridge regression.

kernel The kernel to use for KRR, defaults to `"rbfdot"`.

kernel.pars A list with kernel parameters, elements depend on the kernel used, `"rbfdot"` uses `"sigma"`.

pca logical, should an initial pca step be performed, defaults to `TRUE`.

pca.center logical, should the data be centered before the pca step. Defaults to `TRUE`.

pca.scale logical, should the data be scaled before the pca ste. Defaults to `FALSE`.

fastcv logical, should `fastCV` from the CVST package be used instead of normal cross-validation.

fastcv.test If `fastcv = TRUE`, separate test data set for fastcv.

cv.folds if `fastcv = FALSE`, specifies the number of folds for crossvalidation.

fastkrr.nblocks integer, higher values sacrifice numerical accuracy for speed and less memory, see below for details.

verbose logical, should the cross-validation results be printed out.

Implementation

Wraps around `drr`, see there for details. DRR is a non-linear extension of principal components analysis using Kernel Ridge Regression (KRR, details see `constructKRRLearner` and `constructFastKRRLearner`). Non-linear regression is used to explain more variance than PCA. DRR provides an out-of-sample extension and a backward projection.

The most expensive computations are matrix inversions therefore the implementation profits a lot from a multithreaded BLAS library. The best parameters for each KRR are determined by cross-validaton over all parameter combinations of `lambda` and `kernel.pars`, using less parameter values will speed up computation time. Calculation of KRR can be accelerated by increasing `fastkrr.nblocks`, it should be smaller than n^1/3 up to sacrificing some accuracy, for details see `constructFastKRRLearner`. Another way to speed up is to use `pars$fastcv = TRUE` which might provide a more efficient way to search the parameter space but may also miss the global maximum, I have not ran tests on the accuracy of this method.

References

Laparra, V., Malo, J., Camps-Valls, G., 2015. Dimensionality Reduction via Regression in Hyperspectral Imagery. IEEE Journal of Selected Topics in Signal Processing 9, 1026-1036. doi:10.1109/JSTSP.2015.2417833

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
## Not run:
dat <- loadDataSet("variable Noise Helix", n = 200)[sample(200)]

## use the S4 Class directly:
drr <- DRR()
pars <- drr@stdpars
pars$ndim <- 3
emb <- drr@fun(dat, pars)

## simpler, use embed():
emb2 <- embed(dat, "DRR", ndim = 3)


plot(dat, type = "3vars")
plot(emb, type = "3vars")
plot(emb@inverse(emb@data@data[, 1, drop = FALSE]), type = "3vars")

## End(Not run)
```

---

embed *dispatches the different methods for dimensionality reduction*

Description

wraps around all dimensionality reduction functions.

Usage

```
embed(.data, ...)

## S4 method for signature 'formula'
embed(.formula, .data, .method = dimRedMethodList(),
  .mute = character(0), .keep.org.data = TRUE, ...)

## S4 method for signature 'ANY'
embed(.data, .method = dimRedMethodList(),
  .mute = character(0), .keep.org.data = TRUE, ...)

## S4 method for signature 'dimRedData'
embed(.data, .method = dimRedMethodList(),
  .mute = character(0), .keep.org.data = TRUE, ...)
```

Arguments

`.data`	object of class `dimRedData`, will be converted to be of class `dimRedData` if necessary; see examples for details.
`...`	the parameters, internally passed as a list to the dimensionality reduction method as `pars = list(...)`
`.formula`	a formula, see `as.dimRedData`.
`.method`	character vector naming one of the dimensionality reduction techniques.
`.mute`	a character vector containing the elements you want to mute (`c("message", "output")`), defaults to `character(0)`.
`.keep.org.data`	`TRUE`/`FALSE` keep the original data.

Details

Method must be one of `dimRedMethodList()`, partial matching is performed. All parameters start with a dot, to avoid clashes with partial argument matching (see the R manual section 4.3.2), if there should ever occur any clashes in the arguments, call the function with all arguments named, e.g. `embed(.data = dat,.method = "mymethod", .d = "some parameter")`.

Value

an object of class `dimRedResult`

Methods (by class)

- `formula`: embed a data.frame using a formula.
- `ANY`: Embed anything as long as it can be coerced to `dimRedData`.
- `dimRedData`: Embed a dimRedData object

Examples

```
## embed a data.frame using a formula:
as.data.frame(
  embed(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
        iris, "PCA")
)

## embed a data.frame and return a data.frame
as.data.frame(embed(iris[, 1:4], "PCA"))

## embed a matrix and return a data.frame
as.data.frame(embed(as.matrix(iris[, 1:4]), "PCA"))

## Not run:
## embed dimRedData objects
embed_methods <- dimRedMethodList()
quality_methods <- dimRedQualityList()
dataset <- loadDataSet("Iris")
```

```
quality_results <- matrix(NA, length(embed_methods), length(quality_methods),
                          dimnames = list(embed_methods, quality_methods))
embedded_data <- list()

for (e in embed_methods) {
  message("embedding: ", e)
  embedded_data[[e]] <- embed(dataset, e, .mute = c("message", "output"))
  for (q in quality_methods) {
    message("  quality: ", q)
    quality_results[e, q] <- tryCatch(
      quality(embedded_data[[e]], q),
      error = function(e) NA
    )
  }
}

print(quality_results)

## End(Not run)
```

`FastICA-class` *Independent Component Analysis*

Description

An S4 Class implementing the FastICA algorithm for Indepentend Component Analysis.

Details

ICA is used for blind signal separation of different sources. It is a linear Projection.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

FastICA can take the following parameters:

ndim The number of output dimensions. Defaults to `2`

Implementation

Wraps around fastICA. FastICA uses a very fast approximation for negentropy to estimate statistical independences between signals. Because it is a simple rotation/projection, forward and backward functions can be given.

References

Hyvarinen, A., 1999. Fast and robust fixed-point algorithms for independent component analysis. IEEE Transactions on Neural Networks 10, 626-634. https://doi.org/10.1109/72.761722

See Also

Other dimensionality reduction methods: AutoEncoder-class, DRR-class, DiffusionMaps-class, DrL-class, FruchtermanReingold-class, HLLE-class, Isomap-class, KamadaKawai-class, LLE-class, MDS-class, NNMF-class, PCA-class, PCA_L1-class, UMAP-class, dimRedMethod-class, dimRedMethodList, kPCA-class, nMDS-class, tSNE-class

Examples

```
dat <- loadDataSet("3D S Curve")

## use the S4 Class directly:
fastica <- FastICA()
emb <- fastica@fun(dat, pars = list(ndim = 2))

## simpler, use embed():
emb2 <- embed(dat, "FastICA", ndim = 2)


plot(emb@data@data)
```

FruchtermanReingold-class
Fruchterman Reingold Graph Layout

Description

An S4 Class implementing the Fruchterman Reingold Graph Layout algorithm.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

ndim The number of dimensions, defaults to 2. Can only be 2 or 3

knn Reduce the graph to keep only the neares neighbors. Defaults to 100.

d The distance function to determine the weights of the graph edges. Defaults to euclidean distances.

Implementation

Wraps around `layout_with_fr`, see there for details. The Fruchterman Reingold algorithm puts the data into a circle and puts connected points close to each other.

References

Fruchterman, T.M.J., Reingold, E.M., 1991. Graph drawing by force-directed placement. Softw: Pract. Exper. 21, 1129-1164. https://doi.org/10.1002/spe.4380211102

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("Swiss Roll", n = 100)

## use the S4 Class directly:
fruchterman_reingold <- FruchtermanReingold()
pars <- fruchterman_reingold@stdpars
pars$knn <- 5
emb <- fruchterman_reingold@fun(dat, pars)

## simpler, use embed():
emb2 <- embed(dat, "FruchtermanReingold", knn = 5)

plot(emb, type = "2vars")
```

`getData` *Method getData*

Description

Extracts the data slot.

Usage

```
getData(object)
```

Arguments

`object`	The object to be converted.

`getDimRedData` *Method getDimRedData*

Description

Extract dimRedData.

Usage

```
getDimRedData(object, ...)
```

Arguments

`object`	The object to extract data from.
`...`	other arguments.

`getMeta` *Method getMeta*

Description

Extracts the meta slot.

Usage

```
getMeta(object, ...)
```

Arguments

`object`	The object to be converted.
`...`	other arguments.

`getNDim` *Method getNDim*

Description

Extract the number of embedding dimensions.

Usage

```
getNDim(object, ...)
```

Arguments

`object`	The object to get the dimensions from.
`...`	other arguments.

`getOrgData` *Method getOrgData*

Description

Extract the Original data.

Usage

```
getOrgData(object, ...)
```

Arguments

`object`	The object to extract data from.
`...`	other arguments.

`getOtherData` *Method getOtherData*

Description

Extract other data produced by a dimRedMethod

Usage

```
getOtherData(object, ...)
```

Arguments

`object`	The object to extract data from.
`...`	other arguments.

`getPars` *Method getPars*

Description

Extracts the pars slot.

Usage

```
getPars(object, ...)
```

Arguments

`object`	The object to be converted.
`...`	other arguments.

`getRotationMatrix` *getRotationMatrix*

Description

Extract the rotation matrix from `dimRedResult` objects derived from PCA and FastICA

Usage

```
getRotationMatrix(x)
```

Arguments

`x`	of type `dimRedResult`

Details

The data has to be pre-processed the same way as the method does, e.g. centering and/or scaling.

Value

a matrix

Examples

```
dat <- loadDataSet("Iris")

pca <- embed(dat, "PCA")
ica <- embed(dat, "FastICA")

rot_pca <- getRotationMatrix(pca)
rot_ica <- getRotationMatrix(ica)

scale(getData(dat), TRUE, FALSE) %*% rot_pca - getData(getDimRedData(pca))
scale(getData(dat), TRUE, FALSE) %*% rot_ica - getData(getDimRedData(ica))
```

`HLLE-class` *Hessian Locally Linear Embedding*

Description

An S4 Class implementing Hessian Locally Linear Embedding (HLLE)

Details

HLLE uses local hessians to approximate the curvines and is an extension to non-convex subsets in lowdimensional space.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

HLLE can take the following parameters:

knn neighborhood size

ndim number of output dimensions

Implementation

Own implementation, sticks to the algorithm in Donoho and Grimes (2003). Makes use of sparsity to speed up final embedding.

References

Donoho, D.L., Grimes, C., 2003. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. PNAS 100, 5591-5596. doi:10.1073/pnas.1031596100

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("3D S Curve", n = 300)

## directy use the S4 class:
hlle <- HLLE()
emb <- hlle@fun(dat, hlle@stdpars)

## using embed():
emb2 <- embed(dat, "HLLE", knn = 45)

plot(emb, type = "2vars")
plot(emb2, type = "2vars")
```

`installSuggests` *getSuggests*

Description

Install packages wich are suggested by dimRed.

Usage

```
installSuggests()
```

Details

By default dimRed will not install all the dependencies, because there are quite a lot and in case some of them are not available for your platform you will not be able to install dimRed without problems.

To solve this I provide a function which automatically installes all the suggested packages.

Examples

```
## Not run:
installSuggests()

## End(Not run)
```

`Isomap-class` *Isomap embedding*

Description

An S4 Class implementing the Isomap Algorithm

Details

The Isomap algorithm approximates a manifold using geodesic distances on a k nearest neighbor graph. Then classical scaling is performed on the resulting distance matrix.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

Isomap can take the following parameters:

knn The number of nearest neighbors in the graph. Defaults to 50.

ndim The number of embedding dimensions, defaults to 2.

get_geod Should the geodesic distance matrix be kept, if `TRUE`, access it as `getOtherData(x)$geod`

Implementation

The dimRed package uses its own implementation of Isomap which also comes with an out of sample extension (known as landmark Isomap). The default Isomap algorithm scales computationally not very well, the implementation here uses `nn2` for a faster search of the nearest neighbors. If data are too large it may be useful to fit a subsample of the data and use the out-of-sample extension for the other points.

References

Tenenbaum, J.B., Silva, V. de, Langford, J.C., 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290, 2319-2323. https://doi.org/10.1126/science.290.5500.2319

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("3D S Curve", n = 500)

## use the S4 Class directly:
isomap <- Isomap()
emb <- isomap@fun(dat, isomap@stdpars)

## or simpler, use embed():
samp <- sample(nrow(dat), size = 200)
emb2 <- embed(dat[samp], "Isomap", .mute = NULL, knn = 10)
emb3 <- emb2@apply(dat[-samp])

plot(emb2, type = "2vars")
plot(emb3, type = "2vars")
```

`KamadaKawai-class` *Graph Embedding via the Kamada Kawai Algorithm*

Description

An S4 Class implementing the Kamada Kawai Algorithm for graph embedding.

Details

Graph embedding algorithms se the data as a graph. Between the nodes of the graph exist attracting and repelling forces which can be modeled as electrical fields or springs connecting the nodes. The graph is then forced into a lower dimensional representation that tries to represent the forces betweent he nodes accurately by minimizing the total energy of the attracting and repelling forces.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

KamadaKawai can take the following parameters:

ndim The number of dimensions, defaults to 2. Can only be 2 or 3

knn Reduce the graph to keep only the neares neighbors. Defaults to 100.

d The distance function to determine the weights of the graph edges. Defaults to euclidean distances.

Implementation

Wraps around `layout_with_kk`. The parameters maxiter, epsilon and kkconst are set to the default values and cannot be set, this may change in a future release. The DimRed Package adds an extra sparsity parameter by constructing a knn graph which also may improve visualization quality.

References

Kamada, T., Kawai, S., 1989. An algorithm for drawing general undirected graphs. Information Processing Letters 31, 7-15. https://doi.org/10.1016/0020-0190(89)90102-6

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("Swiss Roll", n = 200)
kamada_kawai <- KamadaKawai()
kk <- kamada_kawai@fun(dat, kamada_kawai@stdpars)

plot(kk@data@data)
```

`kPCA-class` *Kernel PCA*

Description

An S4 Class implementing Kernel PCA

Details

Kernel PCA is a nonlinear extension of PCA using kernel methods.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

Kernel PCA can take the following parameters:

ndim the number of output dimensions, defaults to 2

kernel The kernel function, either as a function or a character vector with the name of the kernel. Defaults to `"rbfdot"`

kpar A list with the parameters for the kernel function, defaults to `list(sigma = 0.1)`

The most comprehensive collection of kernel functions can be found in `kpca`. In case the function does not take any parameters `kpar` has to be an empty list.

Implementation

Wraps around `kpca`, but provides additionally forward and backward projections.

References

Sch\"olkopf, B., Smola, A., M\"uller, K.-R., 1998. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation 10, 1299-1319. https://doi.org/10.1162/089976698300017467

See Also

Other dimensionality reduction methods: AutoEncoder-class, DRR-class, DiffusionMaps-class, DrL-class, FastICA-class, FruchtermanReingold-class, HLLE-class, Isomap-class, KamadaKawai-class, LLE-class, MDS-class, NNMF-class, PCA-class, PCA_L1-class, UMAP-class, dimRedMethod-class, dimRedMethodList, nMDS-class, tSNE-class

Examples

```
## Not run:
dat <- loadDataSet("3D S Curve")

## use the S4 class directly:
kpca <- kPCA()
emb <- kpca@fun(dat, kpca@stdpars)

## simpler, use embed():
emb2 <- embed(dat, "kPCA")

plot(emb, type = "2vars")

## End(Not run)
```

`LaplacianEigenmaps-class`

Laplacian Eigenmaps

Description

An S4 Class implementing Laplacian Eigenmaps

Details

Laplacian Eigenmaps use a kernel and were originally developed to separate non-convex clusters under the name spectral clustering.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

`LaplacianEigenmaps` can take the following parameters:

ndim the number of output dimensions.

sparse A character vector specifying hot to make the graph sparse, `"knn"` means that a K-nearest neighbor graph is constructed, `"eps"` an epsilon neighborhood graph is constructed, else a dense distance matrix is used.

knn The number of nearest neighbors to use for the knn graph.

eps The distance for the epsilon neighborhood graph.

t Parameter for the transformation of the distance matrix by $w = exp(-d^2/t)$, larger values give less weight to differences in distance, `t == Inf` treats all distances != 0 equally.

norm logical, should the normed laplacian be used?

Implementation

Wraps around `spec.emb`.

References

Belkin, M., Niyogi, P., 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Computation 15, 1373.

Examples

```
dat <- loadDataSet("3D S Curve")
leim <- LaplacianEigenmaps()
emb <- leim@fun(dat, leim@stdpars)


plot(emb@data@data)
```

`LCMC,dimRedResult-method`
Method LCMC

Description

Calculates the Local Continuity Meta Criterion, which is `Q_NX` adjusted for random overlap inside the K-ary neighborhood.

Usage

```
## S4 method for signature 'dimRedResult'
LCMC(object)
```

Arguments

`object`	of class dimRedResult

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu`
`reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

`LLE-class` *Locally Linear Embedding*

Description

An S4 Class implementing Locally Linear Embedding (LLE)

Details

LLE approximates the points in the manifold by linear combination of its neighbors. These linear combinations are the same inside the manifold and in highdimensional space.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

LLE can take the following parameters:

knn the number of neighbors for the knn graph., defaults to 50.

ndim the number of embedding dimensions, defaults to 2.

Implementation

Wraps around `lle`, only exposes the parameters `k` and `m`.

References

Roweis, S.T., Saul, L.K., 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290, 2323-2326. doi:10.1126/science.290.5500.2323

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("3D S Curve", n = 500)

## directy use the S4 class:
lle <- LLE()
emb <- lle@fun(dat, lle@stdpars)

## using embed():
emb2 <- embed(dat, "LLE", knn = 45)

plot(emb, type = "2vars")
plot(emb2, type = "2vars")
```

`makeKNNgraph` *makeKNNgraph*

Description

Create a K-nearest neighbor graph from data x. Uses `nn2` as a fast way to find the neares neighbors.

Usage

```
makeKNNgraph(x, k, eps = 0, diag = FALSE)
```

Arguments

`x`	data, a matrix, observations in rows, dimensions in columns
`k`	the number of nearest neighbors.
`eps`	number, if `eps > 0` the KNN search is approximate, see `nn2`
`diag`	logical, if `TRUE` every edge of the returned graph will have an edge with weight `0` to itself.

Value

an object of type `igraph` with edge weight being the distances.

`maximize_correlation,dimRedResult-method` *Maximize Correlation with the Axes*

Description

Rotates the data in such a way that the correlation with the first `naxes` axes is maximized.

Usage

```
## S4 method for signature 'dimRedResult'
maximize_correlation(object,
  naxes = ncol(object@data@data), cor_method = "pearson")
```

Arguments

`object`	A dimRedResult object
`naxes`	the number of axes to optimize for.
`cor_method`	which correlation method to use

Details

Methods that do not use eigenvector decomposition, like t-SNE often do not align the data with axes according to the correlation of variables with the data. `maximize_correlation` uses the `optimx` package to rotate the data in such a way that the original variables have maximum correlation with the embedding axes.

`MDS-class` *Metric Dimensional Scaling*

Description

An S4 Class implementing classical scaling (MDS).

Details

MDS tries to maintain distances in high- and low-dimensional space, it has the advantage over PCA that arbitrary distance functions can be used, but it is computationally more demanding.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

MDS can take the following parameters:

ndim The number of dimensions.

d The function to calculate the distance matrix from the input coordinates, defaults to euclidean distances.

Implementation

Wraps around `cmdscale`. The implementation also provides an out-of-sample extension which is not completely optimized yet.

References

Torgerson, W.S., 1952. Multidimensional scaling: I. Theory and method. Psychometrika 17, 401-419. https://doi.org/10.1007/BF02288916

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
## Not run:
dat <- loadDataSet("3D S Curve")

## Use the S4 Class directly:
mds <- MDS()
emb <- mds@fun(dat, mds@stdpars)

## use embed():
emb2 <- embed(dat, "MDS", d = function(x) exp(stats::dist(x)))


plot(emb, type = "2vars")
plot(emb2, type = "2vars")

## End(Not run)
```

mean_R_NX,dimRedResult-method *Method mean_R_NX*

Description

Calculate the mean_R_NX score to assess the quality of a dimensionality reduction.

Usage

```
## S4 method for signature 'dimRedResult'
mean_R_NX(object)
```

Arguments

object	of class dimRedResult

See Also

Other Quality scores for dimensionality reduction: AUC_lnK_R_NX,dimRedResult-method, LCMC,dimRedResult-method, Q_NX,dimRedResult-method, Q_global,dimRedResult-method, Q_local,dimRedResult-method, R_NX,dimRedResult-method, cophenetic_correlation,dimRedResult-method, distance_correlation,dimRedResu plot_R_NX, quality,dimRedResult-method, reconstruction_error,dimRedResult-method, reconstruction_rmse,dimRedResult-method, total_correlation,dimRedResult-method

mixColorRamps *Mixing color ramps*

Description

mix different color ramps

Usage

```
mixColorRamps(vars, ramps)

mixColor1Ramps(vars, ramps = colorRamp(c("blue", "black", "red")))

mixColor2Ramps(vars, ramps = list(colorRamp(c("blue", "green")),
  colorRamp(c("blue", "red"))))

mixColor3Ramps(vars, ramps = list(colorRamp(c("#001A00", "#00E600")),
  colorRamp(c("#00001A", "#0000E6")), colorRamp(c("#1A0000", "#E60000"))))
```

Arguments

`vars`	a list of variables
`ramps`	a list of color ramps, one for each variable.

Details

automatically create colors to represent a varying number of dimensions.

Examples

```
cols <- expand.grid(x = seq(0, 1, length.out = 10),
                    y = seq(0, 1, length.out = 10),
                    z = seq(0, 1, length.out = 10))
mixed <- mixColor3Ramps(cols)

## Not run:
library(rgl)
plot3d(cols$x, cols$y, cols$z, col = mixed, pch = 15)

cols <- expand.grid(x = seq(0, 1, length.out = 10),
                    y = seq(0, 1, length.out = 10))
mixed <- mixColor2Ramps(cols)

## End(Not run)

plot(cols$x, cols$y, col = mixed, pch = 15)
```

`ndims` *Method ndims*

Description

Extract the number of dimensions.

Usage

```
ndims(object, ...)
```

Arguments

`object`	To extract the number of dimensions from.
`...`	Arguments for further methods

`nMDS-class` *Non-Metric Dimensional Scaling*

Description

An S4 Class implementing Non-Metric Dimensional Scaling.

Details

A non-linear extension of MDS using monotonic regression

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

nMDS can take the following parameters:

d A distance function.

ndim The number of embedding dimensions.

Implementation

Wraps around the `monoMDS`. For parameters that are not available here, the standard configuration is used.

References

Kruskal, J.B., 1964. Nonmetric multidimensional scaling: A numerical method. Psychometrika 29, 115-129. https://doi.org/10.1007/BF02289694

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("3D S Curve", n = 300)

## using the S4 classes:
nmds <- nMDS()
emb <- nmds@fun(dat, nmds@stdpars)


## using embed()
emb2 <- embed(dat, "nMDS", d = function(x) exp(dist(x)))


plot(emb, type = "2vars")
plot(emb2, type = "2vars")
```

`NNMF-class` *Non-Negative Matrix Factorization*

Description

S4 Class implementing NNMF.

Details

NNMF is a method for decomposing a matrix into a smaller dimension such that the constraint that the data (and the projection) are not negative is taken into account.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

The method can take the following parameters:

ndim The number of output dimensions.

method character, which algorithm should be used. See `nmf` for possible values. Defaults to "brunet"

nrun integer, the number of times the computations are conducted. See `nmf`

seed integer, a value to control the random numbers used.

options named list, other options to pass to `nmf`

Implementation

Wraps around `nmf`. Note that the estimation uses random numbers. To create reproducible results, set the random number seed in the function call. Also, in many cases, the computations will be conducted in parallel using multiple cores. To disable this, use the option `.pbackend = NULL`.

References

Lee, D.D., Seung, H.S., 1999. Learning the parts of objects by non-negative matrix factorization. Nature 401, 788-791. https://doi.org/10.1038/44565

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("Iris")

set.seed(4646)
factorization <- embed(dat, "NNMF")

proj_dat <- factorization@apply(dat)

plot(proj_dat@data[, 1], proj_dat@data[, 2])

# project new values:

nn_proj <- predict(factorization, iris[1:7, 1:4])
nn_proj
```

`PCA-class` *Principal Component Analysis*

Description

S4 Class implementing PCA.

Details

PCA transforms the data in orthogonal components so that the first axis accounts for the larges variance in the data, all the following axes account for the highest variance under the constraint that they are orthogonal to the preceding axes. PCA is sensitive to the scaling of the variables. PCA is by far the fastest and simples method of dimensionality reduction and should probably always be applied as a baseline if other methods are tested.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

PCA can take the following parameters:

ndim The number of output dimensions.

center logical, should the data be centered, defaults to `TRUE`.

scale. logical, should the data be scaled, defaults to `FALSE`.

Implementation

Wraps around `prcomp`. Because PCA can be reduced to a simple rotation, forward and backward projection functions are supplied.

References

Pearson, K., 1901. On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2, 559-572.

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
dat <- loadDataSet("Iris")

## using the S4 Class
pca <- PCA()
emb <- pca@fun(dat, pca@stdpars)

## using embed()
emb2 <- embed(dat, "PCA")

plot(emb, type = "2vars")
plot(emb@inverse(emb@data), type = "3vars")
```

PCA_L1-class *Principal Component Analysis with L1 error.*

Description

S4 Class implementing PCA with L1 error.

Details

PCA transforms the data so that the L2 reconstruction error is minimized or the variance of the projected data is maximized. This is sensitive to outliers, L1 PCA minimizes the L1 reconstruction error or maximizes the sum of the L1 norm of the projected observations.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

PCA can take the following parameters:

ndim The number of output dimensions.

center logical, should the data be centered, defaults to `TRUE`.

scale. logical, should the data be scaled, defaults to `FALSE`.

fun character or function, the method to apply, see the `pcaL1` package

... other parameters for `fun`

Implementation

Wraps around the different methods is the pcaL1 package. Because PCA can be reduced to a simple rotation, forward and backward projection functions are supplied.

References

Park, Y.W., Klabjan, D., 2016. Iteratively Reweighted Least Squares Algorithms for L1-Norm Principal Component Analysis, in: Data Mining (ICDM), 2016 IEEE 16th International Conference On. IEEE, pp. 430-438.

See Also

Other dimensionality reduction methods: AutoEncoder-class, DRR-class, DiffusionMaps-class, DrL-class, FastICA-class, FruchtermanReingold-class, HLLE-class, Isomap-class, KamadaKawai-class, LLE-class, MDS-class, NNMF-class, PCA-class, UMAP-class, dimRedMethod-class, dimRedMethodList, kPCA-class, nMDS-class, tSNE-class

Examples

```
if(requireNamespace("pcaL1", quietly = TRUE)) {
dat <- loadDataSet("Iris")

## using the S4 Class
pca_l1 <- PCA_L1()
emb <- pca_l1@fun(dat, pca_l1@stdpars)

## using embed()
emb2 <- embed(dat, "PCA_L1")

plot(emb, type = "2vars")
plot(emb@inverse(emb@data), type = "3vars")
}
```

plot *Plotting of dimRed* objects*

Description

Plots a object of class dimRedResult and dimRedData. For the documentation of the plotting function in base see here: plot.default.

Usage

```
plot(x, y, ...)

## S4 method for signature 'dimRedData,ANY'
plot(x, type = "pairs",
  vars = seq_len(ncol(x@data)), col = seq_len(min(3, ncol(x@meta))),
  ...)

## S4 method for signature 'dimRedResult,ANY'
plot(x, type = "pairs",
  vars = seq_len(ncol(x@data@data)), col = seq_len(min(3,
  ncol(x@data@meta))), ...)
```

Arguments

`x`	dimRedResult/dimRedData class, e.g. output of embedded/loadDataSet
`y`	Ignored
`...`	handed over to the underlying plotting function.
`type`	plot type, one of `c("pairs", "parpl", "2vars","3vars", "3varsrgl")`
`vars`	the axes of the embedding to use for plotting
`col`	the columns of the meta slot to use for coloring, can be referenced as the column names or number of x@data

Details

Plotting functions for the classes usind in `dimRed`. they are intended to give a quick overview over the results, so they are somewhat inflexible, e.g. it is hard to modify color scales or plotting parameters.

If you require more control over plotting, it is better to convert the object to a `data.frame` first and use the standard functions for plotting.

Methods (by class)

- `x = dimRedData,y = ANY`: Ploting of dimRedData objects
- `x = dimRedResult,y = ANY`: Ploting of dimRedResult objects.

Examples

```
scurve = loadDataSet("3D S Curve")
plot(scurve, type = "pairs", main = "pairs plot of S curve")
plot(scurve, type = "parpl")
plot(scurve, type = "2vars", vars = c("y", "z"))
plot(scurve, type = "3vars")
```

plot_R_NX *plot_R_NX*

Description

Plot the R_NX curve for different embeddings. Takes a list of `dimRedResult` objects as input. Also the Area under the curve values are computed for a weighted K (see AUC_lnK_R_NX for details) and appear in the legend.

Usage

```
plot_R_NX(x, ndim = NA, weight = "inv")
```

Arguments

`x`	a list of `dimRedResult` objects. The names of the list will appear in the legend with the AUC_lnK value.
`ndim`	the number of dimensions, if `NA` the original number of embedding dimensions is used, can be a vector giving the embedding dimensionality for each single list element of `x`.
`weight`	the weight function used for K when calculating the AUC, one of `c("inv", "log", "log10")`

Value

A ggplot object, the design can be changed by appending `theme(...)`

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResu` `mean_R_NX,dimRedResult-method`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResult-method`, `reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

Examples

```
## define which methods to apply
embed_methods <- c("Isomap", "PCA")
## load test data set
data_set <- loadDataSet("3D S Curve", n = 200)
## apply dimensionality reduction
data_emb <- lapply(embed_methods, function(x) embed(data_set, x))
names(data_emb) <- embed_methods
## plot the R_NX curves:
plot_R_NX(data_emb) +
    ggplot2::theme(legend.title = ggplot2::element_blank(),
                   legend.position = c(0.5, 0.1),
                    legend.justification = c(0.5, 0.1))
```

`print` *Method print*

Description

Imports the print method into the package namespace.

Usage

```
print(x, ...)
```

Arguments

`x`	The object to be printed.
`...`	Other arguments for printing.

`quality,dimRedResult-method`
Quality Criteria for dimensionality reduction.

Description

A collection of functions to compute quality measures on `dimRedResult` objects.

Usage

```
## S4 method for signature 'dimRedResult'
quality(.data, .method = dimRedQualityList(),
  .mute = character(0), ...)

dimRedQualityList()
```

Arguments

`.data`	object of class `dimRedResult`
`.method`	character vector naming one of the methods
`.mute`	what output from the embedding method should be muted.
`...`	the pameters, internally passed as a list to the quality method as `pars = list(...)`

Value

a number

Methods (by class)

- `dimRedResult`: Calculate a quality index from a dimRedResult object.

Implemented methods

Method must be one of `"Q_local", "Q_global","mean_R_NX", "total_correlation","cophenetic_correlation",`

Rank based criteria

`Q_local`, `Q_global`, and `mean_R_nx` are quality criteria based on the Co-ranking matrix. `Q_local` and `Q_global` determine the local/global quality of the embedding, while `mean_R_nx` determines the quality of the overall embedding. They are parameter free and return a single number. The object must include the original data. The number returns is in the range [0, 1], higher values mean a better local/global embedding.

Correlation based criteria

`total_correlation` calculates the sum of the mean squared correlations of the original axes with the axes in reduced dimensions, because some methods do not care about correlations with axes, there is an option to rotate data in reduced space to maximize this criterium. The number may be greater than one if more dimensions are summed up.

`cophenetic_correlation` calculate the correlation between the lower triangles of distance matrices, the correlation and distance methods may be specified. The result is in range [-1, 1].

`distance_correlation` measures the independes of samples by calculating the correlation of distances. For details see `dcor`.

Reconstruction error

`reconstruction_rmse` calculates the root mean squared error of the reconstrucion. `object` requires an inverse function.

Author(s)

Guido Kraemer

References

Lueks, W., Mokbel, B., Biehl, M., Hammer, B., 2011. How to Evaluate Dimensionality Reduction? - Improving the Co-ranking Matrix. arXiv:1110.3917 [cs].

Szekely, G.J., Rizzo, M.L., Bakirov, N.K., 2007. Measuring and testing dependence by correlation of distances. Ann. Statist. 35, 2769-2794. doi:10.1214/009053607000000505

Lee, J.A., Peluffo-Ordonez, D.H., Verleysen, M., 2015. Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. Neurocomputing, 169, 246-261. doi:10.1016/j.neucom.2014.12.095

See Also

Other Quality scores for dimensionality reduction: AUC_lnK_R_NX,dimRedResult-method, LCMC,dimRedResult-method, Q_NX,dimRedResult-method, Q_global,dimRedResult-method, Q_local,dimRedResult-method, R_NX,dimRedResult-method, cophenetic_correlation,dimRedResult-method, distance_correlation,dimRedResu mean_R_NX,dimRedResult-method, plot_R_NX, reconstruction_error,dimRedResult-method, reconstruction_rmse,dimRedResult-method, total_correlation,dimRedResult-method

Examples

```
## Not run:
embed_methods <- dimRedMethodList()
quality_methods <- dimRedQualityList()
scurve <- loadDataSet("Iris")

quality_results <- matrix(NA, length(embed_methods), length(quality_methods),
                              dimnames = list(embed_methods, quality_methods))
embedded_data <- list()

for (e in embed_methods) {
  message("embedding: ", e)
  embedded_data[[e]] <- embed(scurve, e, .mute = c("message", "output"))
  for (q in quality_methods) {
    message("  quality: ", q)
    quality_results[e, q] <- tryCatch(
      quality(embedded_data[[e]], q),
      error = function (e) NA
    )
  }
}

print(quality_results)

## End(Not run)
```

Q_global,dimRedResult-method

Method Q_global

Description

Calculate the Q_global score to assess the quality of a dimensionality reduction.

Usage

```
## S4 method for signature 'dimRedResult'
Q_global(object)
```

Arguments

`object`	of class dimRedResult

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlatio` `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResult-method`, `reconstruction_rmse,dimRedResu` `total_correlation,dimRedResult-method`

`Q_local,dimRedResult-method`
Method Q_local

Description

Calculate the Q_local score to assess the quality of a dimensionality reduction.

Usage

```
## S4 method for signature 'dimRedResult'
Q_local(object, ndim = getNDim(object))
```

Arguments

`object`	of class dimRedResult.
`ndim`	use the first ndim columns of the embedded data for calculation.

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu` `reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

`Q_NX,dimRedResult-method`
Method Q_NX

Description

Calculate the Q_NX score (Chen & Buja 2006, the notation in the publication is M_k). Which is the fraction of points that remain inside the same K-ary neighborhood in high and low dimensional space.

Usage

```
## S4 method for signature 'dimRedResult'
Q_NX(object)
```

Arguments

`object`	of class dimRedResult

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu` `reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

`reconstruction_error,dimRedResult-method`
Method reconstruction_error

Description

Calculate the error using only the first `n` dimensions of the embedded data. `error_fun` can either be one of `c("rmse", "mae")` to calculate the root mean square error or the mean absolute error respectively, or a function that takes to equally sized vectors as input and returns a single number as output.

Usage

```
## S4 method for signature 'dimRedResult'
reconstruction_error(object,
  n = seq_len(ndims(object)), error_fun = "rmse")
```

Arguments

`object`	of class dimRedResult
`n`	a positive integer or vector of integers <= `ndims(object)`
`error_fun`	a function or string indicating an error function, if indication a function it must take to matrices of the same size and return a scalar.

Value

a vector of number with the same length as `n` with the

Author(s)

Guido Kraemer

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResu` `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_rmse,dimRedResult` `total_correlation,dimRedResult-method`

Examples

```
## Not run:
ir <- loadDataSet("Iris")
ir.drr <- embed(ir, "DRR", ndim = ndims(ir))
ir.pca <- embed(ir, "PCA", ndim = ndims(ir))

rmse <- data.frame(
  rmse_drr = reconstruction_error(ir.drr),
  rmse_pca = reconstruction_error(ir.pca)
)

matplot(rmse, type = "l")
plot(ir)
plot(ir.drr)
plot(ir.pca)

## End(Not run)
```

`reconstruction_rmse,dimRedResult-method`

Method reconstruction_rmse

Description

Calculate the reconstruction root mean squared error a dimensionality reduction, the method must have an inverse mapping.

Usage

```
## S4 method for signature 'dimRedResult'
reconstruction_rmse(object)
```

Arguments

`object`	of class dimRedResult

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResu` `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu` `total_correlation,dimRedResult-method`

`R_NX,dimRedResult-method` *Method R_NX*

Description

Calculate the R_NX score from Lee et. al. (2013) which shows the neighborhood preservation for the Kth nearest neighbors, corrected for random point distributions and scaled to range [0, 1].

Usage

```
## S4 method for signature 'dimRedResult'
R_NX(object, ndim = getNDim(object))
```

Arguments

`object`	of class dimRedResult
`ndim`	the number of dimensions to take from the embedded data.

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResult-method`, `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu` `reconstruction_rmse,dimRedResult-method`, `total_correlation,dimRedResult-method`

`total_correlation,dimRedResult-method` *Method total_correlation*

Description

Calculate the total correlation of the variables with the axes to assess the quality of a dimensionality reduction.

Usage

```
## S4 method for signature 'dimRedResult'
total_correlation(object, naxes = ndims(object),
  cor_method = "pearson", is.rotated = FALSE)
```

Arguments

`object`	of class dimRedResult
`naxes`	the number of axes to use for optimization.
`cor_method`	the correlation method to use.
`is.rotated`	if FALSE the object is rotated.

See Also

Other Quality scores for dimensionality reduction: `AUC_lnK_R_NX,dimRedResult-method`, `LCMC,dimRedResult-method`, `Q_NX,dimRedResult-method`, `Q_global,dimRedResult-method`, `Q_local,dimRedResult-method`, `R_NX,dimRedResult-method`, `cophenetic_correlation,dimRedResult-method`, `distance_correlation,dimRedResu` `mean_R_NX,dimRedResult-method`, `plot_R_NX`, `quality,dimRedResult-method`, `reconstruction_error,dimRedResu` `reconstruction_rmse,dimRedResult-method`

`tSNE-class` *t-Distributed Stochastic Neighborhood Embedding*

Description

An S4 Class for t-SNE.

Details

t-SNE is a method that uses Kullback-Leibler divergence between the distance matrices in high and low-dimensional space to embed the data. The method is very well suited to visualize complex structures in low dimensions.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

t-SNE can take the following parameters:

d A distance function, defaults to euclidean distances

perplexity The perplexity parameter, roughly equivalent to neighborhood size.

theta Approximation for the nearest neighbour search, large values are more inaccurate.

ndim The number of embedding dimensions.

Implementation

Wraps around `Rtsne`, which is very well documented. Setting `theta = 0` does a normal t-SNE, larger values for `theta < 1` use the Barnes-Hut algorithm which scales much nicer with data size. Larger values for perplexity take larger neighborhoods into account.

References

Maaten, L. van der, 2014. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15, 3221-3245.

van der Maaten, L., Hinton, G., 2008. Visualizing Data using t-SNE. J. Mach. Learn. Res. 9, 2579-2605.

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `UMAP-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`

Examples

```
## Not run:
dat <- loadDataSet("3D S Curve", n = 300)

## using the S4 class directly:
tsne <- tSNE()
emb <- tsne@fun(dat, tsne@stdpars)
```

```
## using embed()
emb2 <- embed(dat, "tSNE", perplexity = 80)

plot(emb, type = "2vars")
plot(emb2, type = "2vars")

## End(Not run)
```

UMAP-class *Umap embedding*

Description

An S4 Class implementing the UMAP algorithm

Details

Uniform Manifold Approximation is a gradient descend based algorithm that gives results similar to t-SNE, but scales better with the number of points.

Slots

`fun` A function that does the embedding and returns a dimRedResult object.

`stdpars` The standard parameters for the function.

General usage

Dimensionality reduction methods are S4 Classes that either be used directly, in which case they have to be initialized and a full list with parameters has to be handed to the `@fun()` slot, or the method name be passed to the embed function and parameters can be given to the `...`, in which case missing parameters will be replaced by the ones in the `@stdpars`.

Parameters

UMAP can take the follwing parameters:

ndim The number of embedding dimensions.

knn The number of neighbors to be used.

d The distance metric to use.

method `"naive"` for an R implementation, `"python"` for the reference implementation.

Other method parameters can also be passed, see `umap.defaults` for details. The ones above have been standardized for the use with `dimRed` and will get automatically translated for `umap`.

Implementation

The dimRed package wraps the `umap` packages which provides an implementation in pure R and also a wrapper around the original python package `umap-learn` (https://github.com/lmcinnes/umap/)

The `"naive"` implementation is a pure R implementation and considered experimental at the point of writing this, it is also much slower than the python implementation.

The `"python"` implementation is the reference implementation used by McInees et. al. (2018). It requires the `reticulate` package for the interaction with python and the python package `umap-learn` installed (use `pip install umap-learn`).

References

McInnes, Leland, and John Healy. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." https://arxiv.org/abs/1802.03426

See Also

Other dimensionality reduction methods: `AutoEncoder-class`, `DRR-class`, `DiffusionMaps-class`, `DrL-class`, `FastICA-class`, `FruchtermanReingold-class`, `HLLE-class`, `Isomap-class`, `KamadaKawai-class`, `LLE-class`, `MDS-class`, `NNMF-class`, `PCA-class`, `PCA_L1-class`, `dimRedMethod-class`, `dimRedMethodList`, `kPCA-class`, `nMDS-class`, `tSNE-class`

Examples

```
## Not run:
dat <- loadDataSet("3D S Curve", n = 300)

## use the S4 Class directly:
umap <- UMAP()
emb <- umap@fun(dat, umap@stdpars)

## or simpler, use embed():
emb2 <- embed(dat, "UMAP", .mute = NULL, knn = 10)
plot(emb2, type = "2vars")

## End(Not run)
```

Index