Package 'Rdimtools'

May 22, 2020

Type Package

Title Dimension Reduction and Estimation Methods

Version 1.0.3

Description We provide linear and nonlinear dimension reduction techniques.
Intrinsic dimension estimation methods for exploratory analysis are also provided.
For more details on dimensionality techniques, see the paper by
Ma and Zhu (2013) <doi:10.1111/j.1751-5823.2012.00182.x> if you are interested in
statistical approach, or Engel, Huttenberger, and Hamann (2012)
<doi:10.4230/OASIcs.VLUDS.2011.135> for a broader multi-disciplinary overview.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.0.0)

Imports CVXR (>= 1.0), Rcpp (>= 0.12.15), RcppDE, Rcsdp, Rdpack,
RSpectra, graphics, maotai (>= 0.1.5), stats, utils

LinkingTo Rcpp, RcppArmadillo, RcppDist, maotai

RoxygenNote 7.1.0

RdMacros Rdpack

URL http://kyoustat.com/Rdimtools

BugReports http://github.com/kyoustat/Rdimtools/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>),
Changhee Suh [ctb]

Maintainer Kisung You <kyoustat@gmail.com>

Repository CRAN

Date/Publication 2020-05-22 08:50:06 UTC

R topics documented:

`aux.gensamples` *Generate model-based samples*

Description

It generates samples from predefined shapes, set by `dname` parameter. Also incorporated a functionality to add white noise with degree `noise`.

Usage

```
aux.gensamples(
  n = 496,
  noise = 0.01,
  dname = c("swiss", "crown", "helix", "saddle", "ribbon", "bswiss", "cswiss",
    "twinpeaks", "sinusoid", "mobius", "R12in72"),
  ...
)
```

Arguments

`n`	the number of points to be generated.
`noise`	level of additive white noise.
`dname`	name of a predefined shape. Should be one of
	`"swiss"` swiss roll
	`"crown"` crown
	`"helix"` helix

`"saddle"`	manifold near saddle point
`"ribbon"`	ribbon
`"bswiss"`	broken swiss
`"cswiss"`	cut swiss
`"twinpeaks"`	two peaks
`"sinusoid"`	sinusoid on the circle
`"mobius"`	mobius strip embedded in $\mathbf{R}^3$
`"R12in72"`	12-dimensional manifold in $\mathbf{R}^{12}$

`...` extra parameters for the followings #'

parameter	dname	description
`ntwist`	`"mobius"`	number of twists

Value

an $(n \times p)$ matrix of generated data by row. For all methods other than `"R12in72"`, it returns a matrix with $p = 3$.

Author(s)

Kisung You

References

Hein M, Audibert J (2005). "Intrinsic dimensionality estimation of submanifolds in R ^{d}\$." In *Proceedings of the 22nd international conference on Machine learning*, 289–296.

van der Maaten L (2009). "Learning a parametric embedding by preserving local structure." *Proceedings of AI-STATS*.

Examples

```
## generating toy example datasets
set.seed(100)
dat.swiss = aux.gensamples(50, dname="swiss")
dat.crown = aux.gensamples(50, dname="crown")
dat.helix = aux.gensamples(50, dname="helix")
```

---

aux.graphnbd *Construct Nearest-Neighborhood Graph*

Description

Given data, it first computes pairwise distance (`method`) using one of measures defined from `dist` function. Then, `type` controls how nearest neighborhood graph should be constructed. Finally, `symmetric` parameter controls how nearest neighborhood graph should be symmetrized.

Usage

```
aux.graphnbd(
  data,
  method = "euclidean",
  type = c("proportion", 0.1),
  symmetric = "union",
  pval = 2
)
```

Arguments

`data`	an $(n \times p)$ data matrix.
`method`	type of distance to be used. See also `dist`.
`type`	a defining pattern of neighborhood criterion. One of
	c("knn", k) knn with k a positive integer.
	c("enn", radius) enn with a positive radius.
	c("proportion", ratio) takes an `ratio` in (0,1) portion of edges to be connected.
`symmetric`	either "intersect" or "union" for symmetrization, or "asymmetric".
`pval`	a p-norm option for Minkowski distance.

Value

a named list containing

mask a binary matrix of indicating existence of an edge for each element.

dist corresponding distance matrix. `-Inf` is returned for non-connecting edges.

Nearest Neighbor(NN) search

Our package supports three ways of defining nearest neighborhood. First is *knn*, which finds `k` nearest points and flag them as neighbors. Second is *enn* - epsilon nearest neighbor - that connects all the data poinst within a certain radius. Finally, *proportion* flag is to connect proportion-amount of data points sequentially from the nearest to farthest.

Symmetrization

In many graph setting, it starts from dealing with undirected graphs. NN search, however, does not necessarily guarantee if symmetric connectivity would appear or not. There are two easy options for symmetrization; `intersect` for connecting two nodes if both of them are nearest neighbors of each other and `union` for only either of them to be present.

Author(s)

Kisung You

Examples

```
## Generate data
set.seed(100)
X = aux.gensamples(n=100)

## Test three different types of neighborhood connectivity
nn1 = aux.graphnbd(X,type=c("knn",20))         # knn with k=20
nn2 = aux.graphnbd(X,type=c("enn",1))          # enn with radius = 1
nn3 = aux.graphnbd(X,type=c("proportion",0.4)) # connecting 40% of edges

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
image(nn1$mask); title("knn with k=20")
image(nn2$mask); title("enn with radius=1")
image(nn3$mask); title("proportion of ratio=0.4")
par(opar)
```

`aux.kernelcov` *Build a centered kernel matrix K*

Description

From the celebrated Mercer's Theorem, we know that for a mapping ϕ, there exists a kernel function - or, symmetric bilinear form, K such that

$$K(x, y) =< \phi(x), \phi(y) >$$

where $<, >$ is standard inner product. `aux.kernelcov` is a collection of 20 such positive definite kernel functions, as well as centering of such kernel since covariance requires a mean to be subtracted and a set of transformed values $\phi(x_i), i = 1, 2, \ldots, n$ are not centered after transformation. Since some kernels require parameters - up to 2, its usage will be listed in arguments section.

Usage

```
aux.kernelcov(X, ktype)
```

Arguments

`X`	an $(n \times p)$ data matrix
`ktype`	a vector containing the type of kernel and parameters involved. Below the usage is consistent with description **linear** `c("linear",c)` **polynomial** `c("polynomial",c,d)` **gaussian** `c("gaussian",c)`

laplacian `c("laplacian",c)`
anova `c("anova",c,d)`
sigmoid `c("sigmoid",a,b)`
rational quadratic `c("rq",c)`
multiquadric `c("mq",c)`
inverse quadric `c("iq",c)`
inverse multiquadric `c("imq",c)`
circular `c("circular",c)`
spherical `c("spherical",c)`
power/triangular `c("power",d)`
log `c("log",d)`
spline `c("spline")`
Cauchy `c("cauchy",c)`
Chi-squared `c("chisq")`
histogram intersection `c("histintx")`
generalized histogram intersection `c("ghistintx",c,d)`
generalized Student-t `c("t",d)`

Details

There are 20 kernels supported. Belows are the kernels when given two vectors x, y, $K(x, y)$

linear $=< x, y > +c$
polynomial $= (< x, y > +c)^d$
gaussian $= exp(-c\|x-y\|^2), c > 0$
laplacian $= exp(-c\|x-y\|), c > 0$
anova $= \sum_k exp(-c(x_k - y_k)^2)^d, c > 0, d \geq 1$
sigmoid $= tanh(a < x, y > +b)$
rational quadratic $= 1 - (\|x-y\|^2)/(\|x-y\|^2 + c)$
multiquadric $= \sqrt{\|x-y\|^2 + c^2}$
inverse quadric $= 1/(\|x-y\|^2 + c^2)$
inverse multiquadric $= 1/\sqrt{\|x-y\|^2 + c^2}$
circular $= \frac{2}{\pi} arccos(-\frac{\|x-y\|}{c}) - \frac{2}{\pi}\frac{\|x-y\|}{c}\sqrt{1-(\|x-y\|/c)^2}, c > 0$
spherical $= 1 - 1.5\frac{\|x-y\|}{c} + 0.5(\|x-y\|/c)^3, c > 0$
power/triangular $= -\|x-y\|^d, d \geq 1$
log $= -\log(\|x-y\|^d + 1)$
spline $= \prod_i (1 + x_i y_i (1 + min(x_i, y_i)) - \frac{x_i + y_i}{2} min(x_i, y_i)^2 + \frac{min(x_i, y_i)^3}{3})$
Cauchy $= \frac{c^2}{c^2 + \|x-y\|^2}$
Chi-squared $= \sum_i \frac{2x_i y_i}{x_i + y_i}$
histogram intersection $= \sum_i min(x_i, y_i)$
generalized histogram intersection $= sum_i min(|x_i|^c, |y_i|^d)$
generalized Student-t $= 1/(1 + \|x-y\|^d), d \geq 1$

Value

a named list containing

K a $(p \times p)$ kernelizd gram matrix.

Kcenter a $(p \times p)$ centered version of K.

Author(s)

Kisung You

References

Hofmann, T., Scholkopf, B., and Smola, A.J. (2008) *Kernel methods in machine learning*. arXiv:math/0701907.

Examples

```
## generate a toy data
set.seed(100)
X = aux.gensamples(n=100)

## compute a few kernels
Klin = aux.kernelcov(X, ktype=c("linear",0))
Kgau = aux.kernelcov(X, ktype=c("gaussian",1))
Klap = aux.kernelcov(X, ktype=c("laplacian",1))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
image(Klin$K, main="kernel=linear")
image(Kgau$K, main="kernel=gaussian")
image(Klap$K, main="kernel=laplacian")
par(opar)
```

---

aux.pkgstat *Show the number of functions for* **Rdimtools**.

Description

This function is mainly used for tracking progress for this package.

Usage

```
aux.pkgstat()
```

Examples

```
## run with following command
aux.pkgstat()
```

`aux.preprocess` *Preprocessing the data*

Description

`aux.preprocess` can perform one of following operations; `"center"`, `"scale"`, `"cscale"`, `"decorrelate"` and `"whiten"`. See below for more details.

Usage

```
aux.preprocess(
  data,
  type = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`data`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`type`	one of `"center"`, `"scale"`, `"cscale"`, `"decorrelate"` or `"whiten"`.

Value

named list containing:

pX an $(n \times p)$ matrix after preprocessing in accordance with `type` parameter

info a list containing

- `type`: name of preprocessing procedure.
- `mean`: a mean vector of length p.
- `multiplier`: a $(p \times p)$ matrix or 1 for "center".

Operations

we have following operations,

`"center"` subtracts mean of each column so that every variable has mean 0.

`"scale"` turns each column corresponding to variable have variance 1.

`"cscale"` combines `"center"` and `"scale"`.

`"decorrelate"` `"center"` and sets its covariance term having diagonal entries only.

`"whiten"` `"decorrelate"` and sets all diagonal elements be 1.

Author(s)

Kisung You

Examples

```
## Generate data
set.seed(100)
X = aux.gensamples(n=200)

## 5 types of preprocessing
X_center = aux.preprocess(X)
X_scale  = aux.preprocess(X,type="scale")
X_cscale = aux.preprocess(X,type="cscale")
X_decorr = aux.preprocess(X,type="decorrelate")
X_whiten = aux.preprocess(X,type="whiten")
```

---

aux.shortestpath *Find shortest path using Floyd-Warshall algorithm*

Description

This is a fast implementation of Floyd-Warshall algorithm to find the shortest path in a pairwise sense using 'RcppArmadillo'. A logical input is also accepted.

Usage

```
aux.shortestpath(dist)
```

Arguments

`dist`	either an $(n \times n)$ matrix or a `dist` class object.

Value

an $(n \times n)$ matrix containing pairwise shortest path.

Author(s)

Kisung You

References

Floyd, R.W. (1962) *Algorithm 97: Shortest Path*. Commincations of the ACMS, Vol.5(6):345.

Examples

```
## generate a toy data
X = aux.gensamples(n=10)

## Find knn graph with k=5
Xgraph = aux.graphnbd(X,type=c("knn",5))

## Separately use binarized and real distance matrices
W1 = aux.shortestpath(Xgraph$mask)
W2 = aux.shortestpath(Xgraph$dist)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
image(W1, main="from binarized")
image(W2, main="from Euclidean distance")
par(opar)
```

`do.adr` *Adaptive Dimension Reduction*

Description

Adaptive Dimension Reduction (ADR) iteratively finds the best subspace to perform data clustering. It can be regarded as one of remedies for clustering in high dimensional space. Eigenvectors of a between-cluster scatter matrix are used as basis of projection.

Usage

```
do.adr(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  maxiter = 10,
  abstol = 0.001
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`maxiter`	maximum number of iterations allowed.
`abstol`	stopping criterion for incremental change in projection matrix.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Ding C, Xiaofeng He, Hongyuan Zha, Simon H (2002). "Adaptive dimension reduction for clustering high dimensional data." In *Proceedings 2002 IEEE International Conference on Data Mining*, 147–154.

See Also

do.ldakm

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different tolerance level
out1 = do.adr(X, abstol=1e-2)
out2 = do.adr(X, abstol=1e-3)
out3 = do.adr(X, abstol=1e-4)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="tol=1e-2", col=label, pch=19)
plot(out2$Y, main="tol=1e-3", col=label, pch=19)
plot(out3$Y, main="tol=1e-4", col=label, pch=19)
par(opar)
```

do.ammc *Adaptive Maximum Margin Criterion*

Description

Adaptive Maximum Margin Criterion (AMMC) is a supervised linear dimension reduction method. The method uses different weights to characterize the different contributions of the training samples embedded in MMC framework. With the choice of `a=0`, `b=0`, and `lambda=1`, it is identical to standard MMC method.

Usage

```
do.ammc(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  a = 1,
  b = 1,
  lambda = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`a`	tuning parameter for between-class weight in $[0, \infty)$.
`b`	tuning parameter for within-class weight in $[0, \infty)$.
`lambda`	balance parameter for between-class and within-class scatter matrices in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Lu J, Tan Y (2011). "Adaptive maximum margin criterion for image classification." In *2011 IEEE International Conference on Multimedia and Expo*, 1–6.

See Also

do.mmc

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different lambda values
out1 = do.ammc(X, label, lambda=0.1)
out2 = do.ammc(X, label, lambda=1)
out3 = do.ammc(X, label, lambda=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="AMMC::lambda=0.1", pch=19, cex=0.5, col=label)
plot(out2$Y, main="AMMC::lambda=1",   pch=19, cex=0.5, col=label)
plot(out3$Y, main="AMMC::lambda=10",  pch=19, cex=0.5, col=label)
par(opar)
```

---

do.anmm *Average Neighborhood Margin Maximization*

Description

Average Neighborhood Margin Maximization (ANMM) is a supervised method for feature extraction. It aims to find a projection mapping in the following manner; for each data point, the algorithm tries to pull the neighboring points in the same class while pushing neighboring points of different classes far away. It is known that ANMM does suffer less from small sample size problem, which is bottleneck for LDA.

Usage

```
do.anmm(
  X,
  label,
  ndim = 2,
```

```
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  No = ceiling(nrow(X)/10),
  Ne = ceiling(nrow(X)/10)
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
No	neighborhood size for same-class data points; either a constant number or a vector of length-n can be provided, as long as the values reside in $[2, n]$.
Ne	neighborhood size for different-class data points; either a constant number or a vector of length-n can be provided, as long as the values reside in $[2, n]$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Wang F, Zhang C (2007). "Feature Extraction by Maximizing the Average Neighborhood Margin." In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## perform ANMM on different choices of neighborhood size
out1 = do.anmm(X, label, No=6, Ne=6)
out2 = do.anmm(X, label, No=2, Ne=10)
out3 = do.anmm(X, label, No=10,Ne=2)
```

```
## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="(No,Ne)=(6,6)",  pch=19, cex=0.5, col=label)
plot(out2$Y, main="(No,Ne)=(2,10)", pch=19, cex=0.5, col=label)
plot(out3$Y, main="(No,Ne)=(10,2)", pch=19, cex=0.5, col=label)
par(opar)
```

`do.asi` *Adaptive Subspace Iteration*

Description

Adaptive Subspace Iteration (ASI) iteratively finds the best subspace to perform data clustering. It can be regarded as one of remedies for clustering in high dimensional space. Eigenvectors of a within-cluster scatter matrix are used as basis of projection.

Usage

```
do.asi(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  maxiter = 10,
  abstol = 0.001
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`maxiter`	maximum number of iterations allowed.
`abstol`	stopping criterion for incremental change in projection matrix.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Li T, Ma S, Ogihara M (2004). "Document clustering via adaptive subspace iteration." In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 218.

See Also

do.ldakm

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different tolerance level
out1 = do.asi(X, abstol=1e-2)
out2 = do.asi(X, abstol=1e-3)
out3 = do.asi(X, abstol=1e-4)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="ASI::tol=1e-2", pch=19, col=label)
plot(out2$Y, main="ASI::tol=1e-3", pch=19, col=label)
plot(out3$Y, main="ASI::tol=1e-4", pch=19, col=label)
par(opar)
```

---

do.bmds *Bayesian Multidimensional Scaling*

Description

A Bayesian formulation of classical Multidimensional Scaling is presented. Even though this method is based on MCMC sampling, we only return maximum a posterior (MAP) estimate that maximizes the posterior distribution. Due to its nature without any special tuning, increasing `mc.iter` requires much computation. A note on the method is that this algorithm does not return an explicit form of projection matrix so it's classified in our package as a nonlinear method. Also, automatic dimension selection is not supported for simplicity as well as consistency with other methods in the package.

Usage

```
do.bmds(
  X,
  ndim = 2,
  par.a = 5,
  par.alpha = 0.5,
  par.step = 1,
  mc.iter = 50,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  print.progress = FALSE
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
ndim	an integer-valued target dimension.
par.a	hyperparameter for conjugate prior on variance term, i.e., $\sigma^2 \sim IG(a, b)$. Note that b is chosen appropriately as in paper.
par.alpha	hyperparameter for conjugate prior on diagonal term, i.e., $\lambda_j \sim IG(\alpha, \beta_j)$. Note that β_j is chosen appropriately as in paper.
par.step	stepsize for random-walk, which is standard deviation of Gaussian proposal.
mc.iter	the number of MCMC iterations.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
print.progress	a logical; TRUE to show iterations, FALSE otherwise.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Oh M, Raftery AE (2001). "Bayesian Multidimensional Scaling and Choice of Dimension." *Journal of the American Statistical Association*, **96**(455), 1031–1044.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different preprocessing
out1 <- do.bmds(X, ndim=2, preprocess="center")
out2 <- do.bmds(X, ndim=2, preprocess="cscale")
out3 <- do.bmds(X, ndim=2, preprocess="decorrelate")

## embeddings for each procedure
Y1 <- out1$Y; Y2 <- out2$Y; Y3 <- out3$Y

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, main="BMDS::iter=5",  col=label, pch=19)
plot(Y2, main="BMDS::iter=10", col=label, pch=19)
plot(Y3, main="BMDS::iter=50", col=label, pch=19)
par(opar)
```

---

do.bpca *Bayesian Principal Component Analysis*

Description

Bayesian PCA (BPCA) is a further variant of PCA in that it imposes prior and encodes basis selection mechanism. Even though the model is fully Bayesian, `do.bpca` faithfully follows the original paper by Bishop in that it only returns the mode value of posterior as an estimate, in conjunction with ARD-motivated prior as well as consideration of variance to be estimated. Unlike PPCA, it uses full basis and returns relative weight for each base in that the smaller α value is, the more likely corresponding column vector of `mp.W` to be selected as potential basis.

Usage

```
do.bpca(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  reltol = 1e-04,
  maxiter = 123
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
reltol	stopping criterion for iterative update for EM algorithm.
maxiter	maximum number of iterations allowed for EM algorithm.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are principal components.

mp.itercount the number of iterations taken for EM algorithm to converge.

mp.sigma2 estimated σ^2 value via EM algorithm.

mp.alpha length-ndim-1 vector of relative weight for each base in mp.W.

mp.W an $(ndim \times ndim - 1)$ matrix from EM update.

Author(s)

Kisung You

References

Bishop C (1999). "Bayesian PCA." In *Advances in Neural Information Processing Systems*, volume 11, 382–388.

See Also

do.pca, do.ppca

Examples

```
## Not run:
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## compare PCA and BPCA
out1  <- do.pca(X,  ndim=2)
out2  <- do.bpca(X, ndim=2)
```

```
## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=lab, pch=19, cex=0.8, main="PCA")
plot(out2$Y, col=lab, pch=19, cex=0.8, main="BPCA")
par(opar)

## End(Not run)
```

`do.cca` *Canonical Correlation Analysis*

Description

Canonical Correlation Analysis (CCA) is similar to Partial Least Squares (PLS), except for one objective; while PLS focuses on maximizing covariance, CCA maximizes the correlation. This difference sometimes incurs quite distinct results compared to PLS. For algorithm aspects, we used recursive gram-schmidt orthogonalization in conjunction with extracting projection vectors under eigen-decomposition formulation, as the problem dimension matters only up to original dimensionality.

Usage

```
do.cca(data1, data2, ndim = 2)
```

Arguments

`data1`	an $(n \times N)$ data matrix whose rows are observations
`data2`	an $(n \times M)$ data matrix whose rows are observations
`ndim`	an integer-valued target dimension.

Value

a named list containing

Y1 an $(n \times ndim)$ matrix of projected observations from `data1`.

Y2 an $(n \times ndim)$ matrix of projected observations from `data2`.

projection1 a $(N \times ndim)$ whose columns are loadings for `data1`.

projection2 a $(M \times ndim)$ whose columns are loadings for `data2`.

trfinfo1 a list containing information for out-of-sample prediction for `data1`.

trfinfo2 a list containing information for out-of-sample prediction for `data2`.

eigvals a vector of eigenvalues for iterative decomposition.

Author(s)

Kisung You

References

Hotelling H (1936). "RELATIONS BETWEEN TWO SETS OF VARIATES." *Biometrika*, **28**(3-4), 321–377.

See Also

do.pls

Examples

```
## generate 2 normal data matrices
set.seed(100)
mat1 = matrix(rnorm(100*12),nrow=100)+10 # 12-dim normal
mat2 = matrix(rnorm(100*6), nrow=100)-10 # 6-dim normal

## project onto 2 dimensional space for each data
output = do.cca(mat1, mat2, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(output$Y1, main="proj(mat1)")
plot(output$Y2, main="proj(mat2)")
par(opar)
```

---

do.cge *Constrained Graph Embedding*

Description

Constrained Graph Embedding (CGE) is a semi-supervised embedding method that incorporates partially available label information into the graph structure that find embeddings consistent with the labels.

Usage

```
do.cge(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations
label	a length-n vector of data class labels. It should contain NA elements for missing label.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

He X, Ji M, Bao H (2009). "Graph Embedding with Constraints." In *IJCAI*.

Examples

```
## use iris data
data(iris)
X     = as.matrix(iris[,2:4])
label = as.integer(iris[,5])
lcols = as.factor(label)

## copy a label and let 10% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.10)
label_missing = label
label_missing[sample(1:nlabel, nmissing)]=NA

## try different neighborhood sizes
out1 = do.cge(X, label_missing, type=c("proportion",0.10))
out2 = do.cge(X, label_missing, type=c("proportion",0.25))
out3 = do.cge(X, label_missing, type=c("proportion",0.50))

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
```

```
plot(out1$Y, main="10% connected",  pch=19, col=lcols)
plot(out2$Y, main="25% connected", pch=19, col=lcols)
plot(out3$Y, main="50% connected", pch=19, col=lcols)
par(opar)
```

`do.cisomap` *Conformal Isometric Feature Mapping*

Description

Conformal Isomap(C-Isomap) is a variant of a celebrated method of Isomap. It aims at, rather than preserving full isometry, maintaining infinitesimal angles - conformality - in that it alters geodesic distance to reflect scale information.

Usage

```
do.cisomap(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  weight = TRUE,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`weight`	`TRUE` to perform Isomap on weighted graph, or `FALSE` otherwise.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Silva VD, Tenenbaum JB (2003). "Global Versus Local Methods in Nonlinear Dimensionality Reduction." In Becker S, Thrun S, Obermayer K (eds.), *Advances in Neural Information Processing Systems 15*, 721–728. MIT Press.

Examples

```
## generate data
set.seed(100)
X <- aux.gensamples(dname="cswiss",n=100)

## 1. original Isomap
output1 <- do.isomap(X,ndim=2)

## 2. C-Isomap
output2 <- do.cisomap(X,ndim=2)

## 3. C-Isomap on a binarized graph
output3 <- do.cisomap(X,ndim=2,weight=FALSE)

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, main="Isomap")
plot(output2$Y, main="C-Isomap")
plot(output3$Y, main="Binarized C-Isomap")
par(opar)
```

---

do.cnpe *Complete Neighborhood Preserving Embedding*

Description

One of drawbacks of Neighborhood Preserving Embedding (NPE) is the small-sample-size problem under high-dimensionality of original data, where singular matrices to be decomposed suffer from rank deficiency. Instead of applying PCA as a preprocessing step, Complete NPE (CNPE) transforms the singular generalized eigensystem computation of NPE into two eigenvalue decomposition problems.

Usage

```
do.cnpe(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
preprocess	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Wang Y, Wu Y (2010). "Complete neighborhood preserving embedding for face recognition." *Pattern Recognition*, **43**(3), 1008–1015.

Examples

```
## generate data of 3 types with clear difference
dt1  = aux.gensamples(n=20)-50
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+50
lab  = rep(1:3, each=20)

## merge the data
X      = rbind(dt1,dt2,dt3)
```

```
## try different numbers for neighborhood size
out1 = do.cnpe(X, type=c("proportion",0.10))
out2 = do.cnpe(X, type=c("proportion",0.25))
out3 = do.cnpe(X, type=c("proportion",0.50))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=lab, pch=19, main="CNPE::10% connected")
plot(out2$Y, col=lab, pch=19, main="CNPE::25% connected")
plot(out3$Y, col=lab, pch=19, main="CNPE::50% connected")
par(opar)
```

`do.crca` *Curvilinear Component Analysis*

Description

Curvilinear Component Analysis (CRCA) is a type of self-organizing algorithms for manifold learning. Like MDS, it aims at minimizing a cost function (*Stress*) based on pairwise proximity. Parameter `lambda` is a heaviside function for penalizing distance pair of embedded data, and `alpha` controls learning rate similar to that of subgradient method in that at each iteration t the gradient is weighted by α/t.

Usage

```
do.crca(X, ndim = 2, lambda = 1, alpha = 1, maxiter = 1000, tolerance = 1e-06)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`lambda`	threshold value.
`alpha`	initial value for updating.
`maxiter`	maximum number of iterations allowed.
`tolerance`	stopping criterion for maximum absolute discrepancy between two distance matrices.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

niter the number of iterations until convergence.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Demartines P, Herault J (1997). "Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets." *IEEE Transactions on Neural Networks*, **8**(1), 148–154.

Hérault J, Jausions-Picaud C, Guérin-Dugué A (1999). "Curvilinear component analysis for high-dimensional data representation: I. Theoretical aspects and practical use in the presence of noise." In Goos G, Hartmanis J, van Leeuwen J, Mira J, Sánchez-Andrés JV (eds.), *Engineering Applications of Bio-Inspired Artificial Neural Networks*, volume 1607, 625–634. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-66068-2 978-3-540-48772-2, doi: 10.1007/BFb0100530.

See Also

do.crda

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## different initial learning rates
out1 <- do.crca(X,alpha=1)
out2 <- do.crca(X,alpha=5)
out3 <- do.crca(X,alpha=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="alpha=1.0")
plot(out2$Y, col=label, pch=19, main="alpha=5.0")
plot(out3$Y, col=label, pch=19, main="alpha=10.0")
par(opar)
```

---

do.crda *Curvilinear Distance Analysis*

Description

Curvilinear Distance Analysis (CRDA) is a variant of Curvilinear Component Analysis in that the input pairwise distance is altered by curvilinear distance on a data manifold. Like in Isomap, it first generates *neighborhood graph* and finds *shortest path* on a constructed graph so that the shortest-path length plays as an approximate geodesic distance on nonlinear manifolds.

Usage

```
do.crda(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = "union",
  weight = TRUE,
  lambda = 1,
  alpha = 1,
  maxiter = 1000,
  tolerance = 1e-06
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
weight	TRUE to perform CRDA on weighted graph, or FALSE otherwise.
lambda	threshold value.
alpha	initial value for updating.
maxiter	maximum number of iterations allowed.
tolerance	stopping criterion for maximum absolute discrepancy between two distance matrices.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

niter the number of iterations until convergence.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Lee JA, Lendasse A, Verleysen M (2002). "Curvilinear Distance Analysis versus Isomap." In *ESANN*.

Lee JA, Lendasse A, Verleysen M (2004). "Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis." *Neurocomputing*, **57**, 49–76.

See Also

do.isomap, do.crca

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## different settings of connectivity
out1 <- do.crda(X, type=c("proportion",0.10))
out2 <- do.crda(X, type=c("proportion",0.25))
out3 <- do.crda(X, type=c("proportion",0.50))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="10% connected")
plot(out2$Y, col=label, pch=19, main="25% connected")
plot(out3$Y, col=label, pch=19, main="50% connected")
par(opar)
```

---

do.crp *Collaborative Representation-based Projection*

Description

Collaborative Representation-based Projection (CRP) is an unsupervised linear dimension reduction method. Its embedding is based on ℓ_2 graph construction, similar to that of SPP where sparsity constraint is imposed via ℓ_1 optimization problem. Note that though it may be way faster, rank deficiency can pose a great deal of problems, especially when the dataset is large.

Usage

```
do.crp(
  X,
  ndim = 2,
```

```
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  lambda = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
lambda	regularization parameter for constructing ℓ_2 graph.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Yang W, Wang Z, Sun C (2015). "A collaborative representation based projections method for feature extraction." *Pattern Recognition*, **48**(1), 20–27.

See Also

do.spp

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## test different regularization parameters
out1 <- do.crp(X,ndim=2,lambda=0.1)
out2 <- do.crp(X,ndim=2,lambda=1)
out3 <- do.crp(X,ndim=2,lambda=10)

# visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
```

```
plot(out1$Y, col=lab, pch=19, main="CRP::lambda=0.1")
plot(out2$Y, col=lab, pch=19, main="CRP::lambda=1")
plot(out3$Y, col=lab, pch=19, main="CRP::lambda=10")
par(opar)
```

`do.cscore` *Constraint Score*

Description

Constraint Score is a filter-type algorithm for feature selection using pairwise constraints. It first marks all pairwise constraints as same- and different-cluster and construct a feature score for both constraints. It takes ratio or difference of feature score vectors and selects the indices with smallest values.

Usage

```
do.cscore(
  X,
  label,
  ndim = 2,
  score = c("ratio", "difference"),
  lambda = 0.5,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of class labels.
`ndim`	an integer-valued target dimension.
`score`	type of score measures from two score vectors of same- and different-class pairwise constraints; `"ratio"` and `"difference"` method. See the paper from the reference for more details.
`lambda`	a penalty value for different-class pairwise constraints. Only valid for `"difference"` scoring method.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

cscore a length-p vector of constraint scores. Indices with smallest values are selected.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang D, Chen S, Zhou Z (2008). "Constraint Score: A new filter method for feature selection with pairwise constraints." *Pattern Recognition*, **41**(5), 1440–1451.

See Also

do.cscoreg

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid = sample(1:150,50)
iris.dat = as.matrix(iris[subid,1:4])
iris.lab = as.factor(iris[subid,5])

## try different strategy
out1 = do.cscore(iris.dat, iris.lab, score="ratio")
out2 = do.cscore(iris.dat, iris.lab, score="difference", lambda=0)
out3 = do.cscore(iris.dat, iris.lab, score="difference", lambda=0.5)
out4 = do.cscore(iris.dat, iris.lab, score="difference", lambda=1)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(out1$Y, col=iris.lab, main="ratio")
plot(out2$Y, col=iris.lab, main="diff/lambda=0")
plot(out3$Y, col=iris.lab, main="diff/lambda=0.5")
plot(out4$Y, col=iris.lab, main="diff/lambda=1")
par(opar)
```

`do.cscoreg` *Constraint Score using Spectral Graph*

Description

Constraint Score is a filter-type algorithm for feature selection using pairwise constraints. It first marks all pairwise constraints as same- and different-cluster and construct a feature score for both constraints. It takes ratio or difference of feature score vectors and selects the indices with smallest values. Graph laplacian is constructed for approximated nonlinear manifold structure.

Usage

```
do.cscoreg(
  X,
  label,
  ndim = 2,
  score = c("ratio", "difference"),
  lambda = 0.5,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of class labels.
`ndim`	an integer-valued target dimension.
`score`	type of score measures from two score vectors of same- and different-class pairwise constraints; `"ratio"` and `"difference"` method. See the paper from the reference for more details.
`lambda`	a penalty value for different-class pairwise constraints. Only valid for `"difference"` scoring method.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

cscore a length-p vector of constraint scores. Indices with smallest values are selected.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang D, Chen S, Zhou Z (2008). "Constraint Score: A new filter method for feature selection with pairwise constraints." *Pattern Recognition*, **41**(5), 1440–1451.

See Also

do.cscore

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid    = sample(1:150,50)
iris.dat = as.matrix(iris[subid,1:4])
iris.lab = as.factor(iris[subid,5])

## try different strategy
out1 = do.cscoreg(iris.dat, iris.lab, score="ratio")
out2 = do.cscoreg(iris.dat, iris.lab, score="difference", lambda=0)
out3 = do.cscoreg(iris.dat, iris.lab, score="difference", lambda=0.5)
out4 = do.cscoreg(iris.dat, iris.lab, score="difference", lambda=1)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(out1$Y, pch=19, col=iris.lab, main="ratio")
plot(out2$Y, pch=19, col=iris.lab, main="diff/lambda=0")
plot(out3$Y, pch=19, col=iris.lab, main="diff/lambda=0.5")
plot(out4$Y, pch=19, col=iris.lab, main="diff/lambda=1")
par(opar)
```

---

do.dagdne *Double-Adjacency Graphs-based Discriminant Neighborhood Embedding*

Description

Doublue Adjacency Graphs-based Discriminant Neighborhood Embedding (DAG-DNE) is a variant of DNE. As its name suggests, it introduces two adjacency graphs for homogeneous and heterogeneous samples accordaing to their labels.

Usage

```
do.dagdne(
  X,
  label,
  ndim = 2,
  numk = max(ceiling(nrow(X)/10), 2),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
numk	the number of neighboring points for k-nn graph construction.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Ding C, Zhang L (2015). "Double adjacency graphs-based discriminant neighborhood embedding." *Pattern Recognition*, **48**(5), 1734–1742.

See Also

do.dne

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])
```

```
## try different numbers for neighborhood size
out1 = do.dagdne(X, label, numk=5)
out2 = do.dagdne(X, label, numk=10)
out3 = do.dagdne(X, label, numk=20)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="nbd size=5", col=label, pch=19)
plot(out2$Y, main="nbd size=10",col=label, pch=19)
plot(out3$Y, main="nbd size=20",col=label, pch=19)
par(opar)
```

`do.disr` *Diversity-Induced Self-Representation*

Description

Diversity-Induced Self-Representation (DISR) is a feature selection method that aims at ranking features by both representativeness and diversity. Self-representation controlled by `lbd1` lets the most representative features to be selected, while `lbd2` penalizes the degree of inter-feature similarity to enhance diversity from the chosen features.

Usage

```
do.disr(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  lbd1 = 1,
  lbd2 = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`lbd1`	nonnegative number to control the degree of self-representation.
`lbd2`	nonnegative number to control the degree of feature similarity.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Liu Y, Liu K, Zhang C, Wang J, Wang X (2017). "Unsupervised feature selection via Diversity-induced Self-representation." *Neurocomputing*, **219**, 350–363.

See Also

do.rsr

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

#### try different lbd combinations
out1 = do.disr(X, lbd1=1, lbd2=1)
out2 = do.disr(X, lbd1=1, lbd2=5)
out3 = do.disr(X, lbd1=5, lbd2=1)
out4 = do.disr(X, lbd1=5, lbd2=5)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(out1$Y, main="(lbd1,lbd2)=(1,1)", col=label, pch=19)
plot(out2$Y, main="(lbd1,lbd2)=(1,5)", col=label, pch=19)
plot(out3$Y, main="(lbd1,lbd2)=(5,1)", col=label, pch=19)
plot(out4$Y, main="(lbd1,lbd2)=(5,5)", col=label, pch=19)
par(opar)
```

do.dm *Diffusion Maps*

Description

do.dm discovers low-dimensional manifold structure embedded in high-dimensional data space using Diffusion Maps (DM). It exploits diffusion process and distances in data space to find equivalent representations in low-dimensional space.

Usage

```
do.dm(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  bandwidth = 1,
  timescale = 1,
  multiscale = FALSE
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
bandwidth	a scaling parameter for diffusion kernel. Default is 1 and should be a nonnegative real number.
timescale	a target scale whose value represents behavior of heat kernels at time *t*. Default is 1 and should be a positive real number.
multiscale	logical; FALSE is to use the fixed timescale value, TRUE to ignore the given value.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

eigvals a vector of eigenvalues for Markov transition matrix.

Author(s)

Kisung You

References

Nadler B, Lafon S, Coifman RR, Kevrekidis IG (2005). "Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators." In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, 955–962.

Coifman RR, Lafon S (2006). "Diffusion maps." *Applied and Computational Harmonic Analysis*, **21**(1), 5–30.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare different bandwidths
out1 <- do.dm(X,bandwidth=10)
out2 <- do.dm(X,bandwidth=100)
out3 <- do.dm(X,bandwidth=1000)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="DM::bandwidth=10")
plot(out2$Y, pch=19, col=label, main="DM::bandwidth=100")
plot(out3$Y, pch=19, col=label, main="DM::bandwidth=1000")
par(opar)
```

---

do.dne *Discriminant Neighborhood Embedding*

Description

Discriminant Neighborhood Embedding (DNE) is a supervised subspace learning method. DNE tries to move multi-class data points in high-dimensional space in accordance with local intra-class attraction and inter-class repulsion.

Usage

```
do.dne(
  X,
  label,
  ndim = 2,
  numk = max(ceiling(nrow(X)/10), 2),
```

```
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
numk	the number of neighboring points for k-nn graph construction.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang W, Xue X, Lu H, Guo Y (2006). "Discriminant neighborhood embedding for classification." *Pattern Recognition*, **39**(11), 2240–2243.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different numbers for neighborhood size
out1 = do.dne(X, label, numk=5)
out2 = do.dne(X, label, numk=10)
out3 = do.dne(X, label, numk=20)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="DNE::nbd size=5",  col=label, pch=19)
plot(out2$Y, main="DNE::nbd size=10", col=label, pch=19)
plot(out3$Y, main="DNE::nbd size=20", col=label, pch=19)
par(opar)
```

do.dspp *Discriminative Sparsity Preserving Projection*

Description

Discriminative Sparsity Preserving Projection (DSPP) is a supervised dimension reduction method that employs sparse representation model to adaptively build both intrinsic adjacency graph and penalty graph. It follows an integration of global within-class structure into manifold learning under exploiting discriminative nature provided from label information.

Usage

```
do.dspp(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  lambda = 1,
  rho = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
lambda	regularization parameter for constructing sparsely weighted network.
rho	a parameter for balancing the local and global contribution.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Gao Q, Huang Y, Zhang H, Hong X, Li K, Wang Y (2015). "Discriminative sparsity preserving projections for image recognition." *Pattern Recognition*, **48**(8), 2543–2553.

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different rho values
out1 <- do.dspp(X, label, ndim=2, rho=0.01)
out2 <- do.dspp(X, label, ndim=2, rho=0.1)
out3 <- do.dspp(X, label, ndim=2, rho=1)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="rho=0.01", col=label, pch=19)
plot(out2$Y, main="rho=0.1",  col=label, pch=19)
plot(out3$Y, main="rho=1",    col=label, pch=19)
par(opar)

## End(Not run)
```

---

do.dve *Distinguishing Variance Embedding*

Description

Distinguishing Variance Embedding (DVE) is an unsupervised nonlinear manifold learning method. It can be considered as a balancing method between Maximum Variance Unfolding and Laplacian Eigenmaps. The algorithm unfolds the data by maximizing the global variance subject to the locality-preserving constraint. Instead of defining certain kernel, it applies local scaling scheme in that it automatically computes adaptive neighborhood-based kernel bandwidth.

Usage

```
do.dve(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.

ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Wang Q, Li J (2009). "Combining local and global information for nonlinear dimensionality reduction." *Neurocomputing*, **72**(10-12), 2235–2241.

Qinggang W, Jianwei L, Xuchu W (2010). "Distinguishing variance embedding." *Image and Vision Computing*, **28**(6), 872–880.

Examples

```
## generate swiss-roll dataset of size 100
set.seed(100)
X <- aux.gensamples(dname="crown", n=100)

## try different nbd size
out1 <- do.dve(X, type=c("proportion",0.5))
out2 <- do.dve(X, type=c("proportion",0.7))
out3 <- do.dve(X, type=c("proportion",0.9))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="50% connected")
plot(out2$Y, main="70% connected")
plot(out3$Y, main="90% connected")
par(opar)
```

do.elde *Exponential Local Discriminant Embedding*

Description

Local Discriminant Embedding (LDE) suffers from a small-sample-size problem where scatter matrix may suffer from rank deficiency. Exponential LDE (ELDE) provides not only a remedy for the problem using matrix exponential, but also a flexible framework to transform original data into a new space via distance diffusion mapping similar to kernel-based nonlinear mapping.

Usage

```
do.elde(
  X,
  label,
  ndim = 2,
  t = 1,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  k1 = max(ceiling(nrow(X)/10), 2),
  k2 = max(ceiling(nrow(X)/10), 2)
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
t	kernel bandwidth in $(0, \infty)$.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
k1	the number of same-class neighboring points (homogeneous neighbors).
k2	the number of different-class neighboring points (heterogeneous neighbors).

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Dornaika F, Bosaghzadeh A (2013). "Exponential Local Discriminant Embedding and Its Application to Face Recognition." *IEEE Transactions on Cybernetics*, **43**(3), 921–934.

See Also

do.lde

Examples

```
## generate data of 3 types with difference
set.seed(100)
dt1  = aux.gensamples(n=20)-50
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+50

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different kernel bandwidth
out1 = do.elde(X, label, t=1)
out2 = do.elde(X, label, t=10)
out3 = do.elde(X, label, t=100)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="ELDE::bandwidth=1")
plot(out2$Y, pch=19, col=label, main="ELDE::bandwidth=10")
plot(out3$Y, pch=19, col=label, main="ELDE::bandwidth=100")
par(opar)
```

---

do.elpp2 *Enhanced Locality Preserving Projection (2013)*

Description

Enhanced Locality Preserving Projection proposed in 2013 (ELPP2) is built upon a parameter-free philosophy from PFLPP. It further aims to exclude its projection to be uncorrelated in the sense that the scatter matrix is placed in a generalized eigenvalue problem.

Usage

```
do.elpp2(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Dornaika F, Assoum A (2013). "Enhanced and parameterless Locality Preserving Projections for face recognition." *Neurocomputing*, **99**, 448–457.

See Also

do.pflpp

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## compare with PCA and PFLPP
out1 = do.pca(X, ndim=2)
out2 = do.pflpp(X, ndim=2)
out3 = do.elpp2(X, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="PCA")
plot(out2$Y, pch=19, col=lab, main="Parameter-Free LPP")
plot(out3$Y, pch=19, col=lab, main="Enhanced LPP (2013)")
par(opar)
```

`do.enet` *Elastic Net Regularization*

Description

Elastic Net is a regularized regression method by solving

$$\min_\beta \frac{1}{2}\|X\beta - y\|_2^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2$$

where y iis `response` variable in our method. The method can be used in feature selection like LASSO.

Usage

```
do.enet(
  X,
  response,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  ycenter = FALSE,
  lambda1 = 1,
  lambda2 = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`response`	a length-n vector of response variable.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`ycenter`	a logical; `TRUE` to center the response variable, `FALSE` otherwise.
`lambda1`	ℓ_1 regularization parameter in $(0, \infty)$.
`lambda2`	ℓ_2 regularization parameter in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zou H, Hastie T (2005). "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(2), 301–320.

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(100)
n = 123
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try different regularization parameters
out1 = do.enet(X, y, lambda1=0.01)
out2 = do.enet(X, y, lambda1=1)
out3 = do.enet(X, y, lambda1=100)

## extract embeddings
Y1 = out1$Y; Y2 = out2$Y; Y3 = out3$Y

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, pch=19, main="ENET::lambda1=0.01")
plot(Y2, pch=19, main="ENET::lambda1=1")
plot(Y3, pch=19, main="ENET::lambda1=100")
par(opar)
```

`do.eslpp` *Extended Supervised Locality Preserving Projection*

Description

Extended LPP and Supervised LPP are two variants of the celebrated Locality Preserving Projection (LPP) algorithm for dimension reduction. Their combination, Extended Supervised LPP, is a combination of two algorithmic novelties in one that it reflects discriminant information with realistic distance measure via Z-score function.

Usage

```
do.eslpp(
  X,
  label,
  ndim = 2,
  numk = max(ceiling(nrow(X)/10), 2),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`numk`	the number of neighboring points for k-nn graph construction.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zheng Z, Yang F, Tan W, Jia J, Yang J (2007). "Gabor feature-based face recognition using supervised locality preserving projection." *Signal Processing*, **87**(10), 2473–2483.

Shikkenawis G, Mitra SK (2012). "Improving the Locality Preserving Projection for dimensionality reduction." In *2012 Third International Conference on Emerging Applications of Information Technology*, 161–164.

See Also

`do.lpp`, `do.slpp`, `do.extlpp`

Examples

```
## generate data of 2 types with clear difference
set.seed(100)
diff = 50
dt1  = aux.gensamples(n=50)-diff;
dt2  = aux.gensamples(n=50)+diff;

## merge the data and create a label correspondingly
Y      = rbind(dt1,dt2)
label  = rep(1:2, each=50)

## compare LPP, SLPP and ESLPP
outLPP   <- do.lpp(Y)
outSLPP  <- do.slpp(Y, label)
outESLPP <- do.eslpp(Y, label)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(outLPP$Y,   col=label, pch=19, main="LPP")
plot(outSLPP$Y,  col=label, pch=19, main="SLPP")
plot(outESLPP$Y, col=label, pch=19, main="ESLPP")
par(opar)
```

`do.extlpp` *Extended Locality Preserving Projection*

Description

Extended Locality Preserving Projection (EXTLPP) is an unsupervised dimension reduction algorithm with a bit of flavor in adopting discriminative idea by nature. It raises a question on the data points at *moderate* distance in that a Z-shaped function is introduced in defining similarity derived from Euclidean distance.

Usage

```
do.extlpp(
  X,
  ndim = 2,
  numk = max(ceiling(nrow(X)/10), 2),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`ndim`	an integer-valued target dimension.

numk	the number of neighboring points for k-nn graph construction.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Shikkenawis G, Mitra SK (2012). "Improving the Locality Preserving Projection for dimensionality reduction." In *2012 Third International Conference on Emerging Applications of Information Technology*, 161–164.

See Also

do.lpp

Examples

```
## generate data
set.seed(100)
X <- aux.gensamples(n=75)

## run Extended LPP with different neighborhood graph
out1 <- do.extlpp(X, numk=5)
out2 <- do.extlpp(X, numk=10)
out3 <- do.extlpp(X, numk=25)

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="EXTLPP::k=5")
plot(out2$Y, main="EXTLPP::k=10")
plot(out3$Y, main="EXTLPP::k=25")
par(opar)
```

`do.fa` *Exploratory Factor Analysis*

Description

`do.fa` is an optimization-based implementation of a popular technique for Exploratory Data Analysis. It is closely related to principal component analysis.

Usage

```
do.fa(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  maxiter = 1000,
  tolerance = 1e-06
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued number of loading variables, or target dimension.
`preprocess`	an additional option for preprocessing the data. Default is `"center"`. See also `aux.preprocess` for more details.
`maxiter`	maximum number of iterations for updating.
`tolerance`	stopping criterion in a Frobenius norm.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

loadings a $(p \times ndim)$ matrix whose rows are extracted loading factors.

noise a length-p vector of estimated noise.

Author(s)

Kisung You

References

Spearman C (1904). “"General Intelligence," Objectively Determined and Measured.” *The American Journal of Psychology*, **15**(2), 201.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different preprocessing procedure
out1 <- do.fa(X,ndim=2)
out2 <- do.fa(X,ndim=2,preprocess="decorrelate")
out3 <- do.fa(X,ndim=2,preprocess="whiten")

## extract embeddings for each procedure
Y1 <- out1$Y; Y2 <- out2$Y; Y3 <- out3$Y

## visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, pch=19, col=lab, main="FA::centered")
plot(Y2, pch=19, col=lab, main="FA::decorrelated")
plot(Y3, pch=19, col=lab, main="FA::whitened")
par(opar)
```

---

do.fastmap *FastMap*

Description

do.fastmap is an implementation of *FastMap* algorithm. Though it shares similarities with MDS, it is innately a nonlinear method that iteratively updates the projection information using pairwise distance information.

Usage

```
do.fastmap(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.

ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Faloutsos C, Lin K (1995). "FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets." In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data - SIGMOD '95*, 163–174.

Examples

```
## Not run:
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## let's compare with other methods
out1 <- do.pca(X, ndim=2)      # PCA
out2 <- do.mds(X, ndim=2)      # Classical MDS
out3 <- do.fastmap(X, ndim=2)  # FastMap

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="PCA")
plot(out2$Y, pch=19, col=label, main="MDS")
plot(out3$Y, pch=19, col=label, main="FastMap")
par(opar)

## End(Not run)
```

`do.fscore` *Fisher Score*

Description

Fisher Score (FSCORE) is a supervised linear feature extraction method. For each feature/variable, it computes Fisher score, a ratio of between-class variance to within-class variance. The algorithm selects variables with largest Fisher scores and returns an indicator projection matrix.

Usage

```
do.fscore(
  X,
  label,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Fisher RA (1936). "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." *Annals of Eugenics*, **7**(2), 179–188.

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid    = sample(1:150,50)
iris.dat = as.matrix(iris[subid,1:4])
iris.lab = as.factor(iris[subid,5])

## compare Fisher score with LDA
out1 = do.lda(iris.dat, iris.lab)
out2 = do.fscore(iris.dat, iris.lab)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=iris.lab, main="LDA")
plot(out2$Y, col=iris.lab, main="Fisher Score")
par(opar)
```

`do.fssem` *Feature Subset Selection using Expectation-Maximization*

Description

Feature Subset Selection using Expectation-Maximization (FSSEM) takes a wrapper approach to feature selection problem. It iterates over optimizing the selection of variables by incrementally including each variable that adds the most significant amount of scatter separability from a labeling obtained by Gaussian mixture model. This method is quite computation intensive as it pertains to multiple fitting of GMM. Setting smaller `max.k` for each round of EM algorithm as well as target dimension `ndim` would ease the burden.

Usage

```
do.fssem(
  X,
  ndim = 2,
  max.k = 10,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`max.k`	maximum number of clusters for GMM fitting with EM algorithms.

preprocess — an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Dy JG, Brodley CE (2004). "Feature Selection for Unsupervised Learning." *J. Mach. Learn. Res.*, **5**, 845–889.

Examples

```
## run FSSEM with IRIS dataset - select 2 of 4 variables
data(iris)
irismat = as.matrix(iris[,2:4])

## select 50 observations for CRAN-purpose small example
id50 = sample(1:nrow(irismat), 50)
sel.dat = irismat[id50,]
sel.lab = as.factor(iris[id50,5])

## run and visualize
out0 = do.fssem(sel.dat, ndim=2, max.k=3)
opar = par(no.readonly=TRUE)
plot(out0$Y, main="small run", col=sel.lab, pch=19)
par(opar)

## Not run:
## NOT-FOR-CRAN example; run at your machine !
## try different maximum number of clusters
out3 = do.fssem(irismat, ndim=2, max.k=3)
out6 = do.fssem(irismat, ndim=2, max.k=6)
out9 = do.fssem(irismat, ndim=2, max.k=9)

## visualize
cols = as.factor(iris[,5])
opar = par(no.readonly=TRUE)
par(mfrow=c(3,1))
plot(out3$Y, main="max k=3", col=cols)
plot(out6$Y, main="max k=6", col=cols)
plot(out9$Y, main="max k=9", col=cols)
par(opar)
```

```
## End(Not run)
```

`do.ica` *Independent Component Analysis*

Description

`do.ica` is an R implementation of FastICA algorithm, which aims at finding weight vectors that maximize a measure of non-Gaussianity of projected data. FastICA is initiated with pre-whitening of the data. Single and multiple component extraction are both supported. For more detailed information on ICA and FastICA algorithm, see this Wikipedia page.

Usage

```
do.ica(
  X,
  ndim = 2,
  type = "logcosh",
  tpar = 1,
  sym = FALSE,
  tol = 1e-06,
  redundancy = TRUE,
  maxiter = 100
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	nonquadratic function, one of `"logcosh"`,`"exp"`, or `"poly"` be chosen.
`tpar`	a numeric parameter for `logcosh` and `exp` parameters that should be close to 1.
`sym`	a logical value; `FALSE` for not using symmetric decorrelation, `TRUE` otherwise.
`tol`	stopping criterion for iterative update.
`redundancy`	a logical value; `TRUE` for removing `NA` values after prewhitening, `FALSE` otherwise.
`maxiter`	maximum number of iterations allowed.
	Y an $(n \times ndim)$ matrix whose rows are embedded observations.
	trfinfo a list containing information for out-of-sample prediction.
	projection a $(p \times ndim)$ whose columns are basis for projection.

Details

In most of ICA literature, we have

$$S = X * W$$

where W is an unmixing matrix for the given data X. In order to preserve consistency throughout our package, we changed the notation; Y a projected matrix for S, and `projection` for unmixing matrix W.

Author(s)

Kisung You

References

Hyvarinen A, Karhunen J, Oja E (2001). *Independent component analysis*. J. Wiley, New York. ISBN 978-0-471-40540-5.

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## 1. use logcosh function for transformation
output1 <- do.ica(X,ndim=2,type="logcosh")

## 2. use exponential function for transformation
output2 <- do.ica(X,ndim=2,type="exp")

## 3. use polynomial function for transformation
output3 <- do.ica(X,ndim=2,type="poly")

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, col=lab, pch=19, main="ICA::logcosh")
plot(output2$Y, col=lab, pch=19, main="ICA::exp")
plot(output3$Y, col=lab, pch=19, main="ICA::poly")
par(opar)
```

`do.idmap` *Interactive Document Map*

Description

Interactive Document Map originates from text analysis to generate maps of documents by placing similar documents in the same neighborhood. After defining pairwise distance with cosine similarity, authors asserted to use either NNP or FastMap as an engine behind.

Usage

```
do.idmap(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  engine = c("NNP", "FastMap")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
engine	either NNP or FastMap.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

References

Minghim R, Paulovich FV, de Andrade Lopes A (2006). "Content-based text mapping using multidimensional projections for exploration of document collections." In Erbacher RF, Roberts JC, Gröhn MT, Börner K (eds.), *Visualization and Data Analysis*, 60600S.

See Also

do.nnp, do.fastmap

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])
```

```
## let's compare with other methods
out1 <- do.pca(X, ndim=2)
out2 <- do.lda(X, ndim=2, label=lab)
out3 <- do.idmap(X, ndim=2, engine="NNP")

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="PCA")
plot(out2$Y, pch=19, col=lab, main="LDA")
plot(out3$Y, pch=19, col=lab, main="IDMAP")
par(opar)
```

`do.iltsa` *Improved Local Tangent Space Alignment*

Description

Conventional LTSA method relies on PCA for approximating local tangent spaces. Improved LTSA (ILTSA) provides a remedy that can efficiently recover the geometric structure of data manifolds even when data are sparse or non-uniformly distributed.

Usage

```
do.iltsa(
  X,
  ndim = 2,
  type = c("proportion", 0.25),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  t = 10
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.

preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
t	heat kernel bandwidth parameter in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Zhang P, Qiao H, Zhang B (2011). "An improved local tangent space alignment method for manifold learning." *Pattern Recognition Letters*, **32**(2), 181–189.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different bandwidth size
out1 <- do.iltsa(X, t=1)
out2 <- do.iltsa(X, t=10)
out3 <- do.iltsa(X, t=100)

## Visualize two comparisons
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="ILTSA::t=1")
plot(out2$Y, pch=19, col=label, main="ILTSA::t=10")
plot(out3$Y, pch=19, col=label, main="ILTSA::t=100")
par(opar)
```

`do.isomap` *Isometric Feature Mapping*

Description

`do.isomap` is an efficient implementation of a well-known *Isomap* method by Tenenbaum et al (2000). Its novelty comes from applying classical multidimensional scaling on nonlinear manifold, which is approximated as a graph.

Usage

```
do.isomap(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  weight = FALSE,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`weight`	`TRUE` to perform Isomap on weighted graph, or `FALSE` otherwise.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Silva VD, Tenenbaum JB (2003). "Global Versus Local Methods in Nonlinear Dimensionality Reduction." In Becker S, Thrun S, Obermayer K (eds.), *Advances in Neural Information Processing Systems 15*, 721–728. MIT Press.

Examples

```
## generate data
set.seed(100)
X <- aux.gensamples(n=123)

## 1. connecting 10% of data for graph construction.
output1 <- do.isomap(X,ndim=2,type=c("proportion",0.10),weight=FALSE)

## 2. constructing 25%-connected graph
output2 <- do.isomap(X,ndim=2,type=c("proportion",0.25),weight=FALSE)

## 3. constructing 25%-connected with binarization
output3 <- do.isomap(X,ndim=2,type=c("proportion",0.50),weight=FALSE)

## Visualize three different projections
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, main="10%")
plot(output2$Y, main="25%")
plot(output3$Y, main="25%+Binary")
par(opar)
```

---

do.isoproj *Isometric Projection*

Description

Isometric Projection is a linear dimensionality reduction algorithm that exploits geodesic distance in original data dimension and mimicks the behavior in the target dimension. Embedded manifold is approximated by graph construction as of ISOMAP. Since it involves singular value decomposition and guesses intrinsic dimension by the number of positive singular values from the decomposition of data matrix, it automatically corrects the target dimension accordingly.

Usage

```
do.isoproj(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
```

```
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix of projected observations as rows.

projection a $(p \times ndim)$ whose columns are loadings.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Cai D, He X, Han J (2007). "Isometric Projection." In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, 528–533. ISBN 978-1-57735-323-2.

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid <- sample(1:150, 50)
X     <- as.matrix(iris[subid,1:4])
lab   <- as.factor(iris[subid,5])

## try different connectivity levels
output1 <- do.isoproj(X,ndim=2,type=c("proportion",0.50))
output2 <- do.isoproj(X,ndim=2,type=c("proportion",0.70))
output3 <- do.isoproj(X,ndim=2,type=c("proportion",0.90))

## visualize two different projections
```

```
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, main="50%", col=lab, pch=19)
plot(output2$Y, main="70%", col=lab, pch=19)
plot(output3$Y, main="90%", col=lab, pch=19)
par(opar)
```

`do.ispe` *Isometric Stochastic Proximity Embedding*

Description

The isometric SPE (ISPE) adopts the idea of approximating geodesic distance on embedded manifold when two data points are close enough. It introduces the concept of `cutoff` where the learning process is only applied to the pair of data points whose original proximity is small enough to be considered as mutually local whose distance should be close to geodesic distance.

Usage

```
do.ispe(
  X,
  ndim = 2,
  proximity = function(x) {     dist(x, method = "euclidean") },
  C = 50,
  S = 50,
  lambda = 1,
  drate = 0.9,
  cutoff = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`proximity`	a function for constructing proximity matrix from original data dimension.
`C`	the number of cycles to be run; after each cycle, learning parameter
`S`	the number of updates for each cycle.
`lambda`	initial learning parameter.
`drate`	multiplier for `lambda` at each cycle; should be a positive real number in $(0, 1)$.
`cutoff`	cutoff threshold value.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Agrafiotis DK, Xu H (2002). "A self-organizing principle for learning nonlinear manifolds." *Proceedings of the National Academy of Sciences*, **99**(25), 15869–15872.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare with original SPE
outSPE <- do.spe(X, ndim=2)
out1 <- do.ispe(X, ndim=2, cutoff=0.5)
out2 <- do.ispe(X, ndim=2, cutoff=5)
out3 <- do.ispe(X, ndim=2, cutoff=50)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(outSPE$Y, pch=19, col=label, main="SPE")
plot(out1$Y,   pch=19, col=label, main="ISPE::cutoff=0.5")
plot(out2$Y,   pch=19, col=label, main="ISPE::cutoff=5")
plot(out3$Y,   pch=19, col=label, main="ISPE::cutoff=50")
par(opar)
```

---

do.keca *Kernel Entropy Component Analysis*

Description

Kernel Entropy Component Analysis(KECA) is a kernel method of dimensionality reduction. Unlike Kernel PCA(do.kpca), it utilizes eigenbasis of kernel matrix K in accordance with indices of largest Renyi quadratic entropy in which entropy for j-th eigenpair is defined to be $\sqrt{\lambda_j} e_j^T 1_n$, where e_j is j-th eigenvector of an uncentered kernel matrix K.

Usage

```
do.keca(
  X,
  ndim = 2,
  kernel = c("gaussian", 1),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
kernel	a vector containing name of a kernel and corresponding parameters. See also aux.kernelcov for complete description of Kernel Trick.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

entropy a length-ndim vector of estimated entropy values.

Author(s)

Kisung You

References

Jenssen R (2010). "Kernel Entropy Component Analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(5), 847–860.

See Also

aux.kernelcov

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])
```

```
## 1. standard KECA with gaussian kernel
output1 <- do.keca(X,ndim=2)

## 2. gaussian kernel with large bandwidth
output2 <- do.keca(X,ndim=2,kernel=c("gaussian",5))

## 3. use laplacian kernel
output3 <- do.keca(X,ndim=2,kernel=c("laplacian",1))

## Visualize three different projections
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, pch=19, col=label, main="Gaussian kernel")
plot(output2$Y, pch=19, col=label, main="Gaussian, sigma=5")
plot(output3$Y, pch=19, col=label, main="Laplacian kernel")
par(opar)
```

`do.klde` *Kernel Local Discriminant Embedding*

Description

Kernel Local Discriminant Embedding (KLDE) is a variant of Local Discriminant Embedding in that it aims to preserve inter- and intra-class neighborhood information in a nonlinear manner using kernel trick. *Note* that the combination of kernel matrix and its eigendecomposition often suffers from lacking numerical rank. For such case, our algorithm returns a warning message and algorithm stops working any further due to its innate limitations of constructing weight matrix.

Usage

```
do.klde(
  X,
  label,
  ndim = 2,
  t = 1,
  numk = max(ceiling(nrow(X)/10), 2),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  ktype = c("gaussian", 1),
  kcentering = TRUE
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.

t	kernel bandwidth in $(0, \infty)$.
numk	the number of neighboring points for k-nn graph construction.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
ktype	a vector containing name of a kernel and corresponding parameters. See also aux.kernelcov for complete description of Kernel Trick.
kcentering	a logical; TRUE to use centered Kernel matrix, FALSE otherwise.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Hwann-Tzong Chen, Huang-Wei Chang, Tyng-Luh Liu (2005). "Local Discriminant Embedding and Its Variants." In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 846–853.

Examples

```
## generate data of 2 types with clear difference
set.seed(100)
diff = 25
dt1  = aux.gensamples(n=50)-diff;
dt2  = aux.gensamples(n=50)+diff;

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2)
label  = rep(1:2, each=50)

## try different neighborhood size
out1 <- do.klde(X, label, numk=5)
out2 <- do.klde(X, label, numk=10)
out3 <- do.klde(X, label, numk=20)

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="k=5")
plot(out2$Y, col=label, pch=19, main="k=10")
plot(out3$Y, col=label, pch=19, main="k=20")
par(opar)
```

`do.klfda` *Kernel Local Fisher Discriminant Analysis*

Description

Kernel LFDA is a nonlinear extension of LFDA method using kernel trick. It applies conventional kernel method to extend excavation of hidden patterns in a more flexible manner in tradeoff of computational load. For simplicity, only the gaussian kernel parametrized by its bandwidth `t` is supported.

Usage

```
do.klfda(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  localscaling = TRUE,
  t = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`localscaling`	`TRUE` to use local scaling method for construction affinity matrix, `FALSE` for binary affinity.
`t`	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Sugiyama M (2006). "Local Fisher discriminant analysis for supervised dimensionality reduction." In *Proceedings of the 23rd international conference on Machine learning*, 905–912.

Zelnik-manor L, Perona P (2005). "Self-Tuning Spectral Clustering." In Saul LK, Weiss Y, Bottou L (eds.), *Advances in Neural Information Processing Systems 17*, 1601–1608. MIT Press. http://papers.nips.cc/paper/2619-self-tuning-spectral-clustering.pdf.

See Also

do.lfda

Examples

```
## generate 3 different groups of data X and label vector
set.seed(100)
x1 = matrix(rnorm(4*10), nrow=10)-20
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+20
X     = rbind(x1, x2, x3)
label = rep(1:3, each=10)

## try different affinity matrices
out1 = do.klfda(X, label, t=0.1)
out2 = do.klfda(X, label, t=1)
out3 = do.klfda(X, label, t=10)

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="bandwidth=0.1")
plot(out2$Y, pch=19, col=label, main="bandwidth=1")
plot(out3$Y, pch=19, col=label, main="bandwidth=10")
par(opar)
```

`do.klsda` *Kernel Locality Sensitive Discriminant Analysis*

Description

Kernel LSDA (KLSDA) is a nonlinear extension of LFDA method using kernel trick. It applies conventional kernel method to extend excavation of hidden patterns in a more flexible manner in tradeoff of computational load. For simplicity, only the gaussian kernel parametrized by its bandwidth `t` is supported.

Usage

```
do.klsda(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  alpha = 0.5,
  k1 = max(ceiling(nrow(X)/10), 2),
  k2 = max(ceiling(nrow(X)/10), 2),
  t = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`alpha`	balancing parameter for between- and within-class scatter in $[0, 1]$.
`k1`	the number of same-class neighboring points (homogeneous neighbors).
`k2`	the number of different-class neighboring points (heterogeneous neighbors).
`t`	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Cai D, He X, Zhou K, Han J, Bao H (2007). "Locality Sensitive Discriminant Analysis." In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 708–713.

Examples

```
## generate 3 different groups of data X and label vector
x1 = matrix(rnorm(4*10), nrow=10)-50
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+50
X     = rbind(x1, x2, x3)
label = rep(1:3, each=10)

## try different kernel bandwidths
out1 = do.klsda(X, label, t=0.1)
out2 = do.klsda(X, label, t=1)
out3 = do.klsda(X, label, t=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="bandwidth=0.1")
plot(out2$Y, col=label, pch=19, main="bandwidth=1")
plot(out3$Y, col=label, pch=19, main="bandwidth=10")
par(opar)
```

---

do.kmfa *Kernel Marginal Fisher Analysis*

Description

Kernel Marginal Fisher Analysis (KMFA) is a nonlinear variant of MFA using kernel tricks. For simplicity, we only enabled a heat kernel of a form

$$k(x_i, x_j) = \exp(-d(x_i, x_j)^2/2 * t^2)$$

where t is a bandwidth parameter. Note that the method is far sensitive to the choice of t.

Usage

```
do.kmfa(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  k1 = max(ceiling(nrow(X)/10), 2),
  k2 = max(ceiling(nrow(X)/10), 2),
  t = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
k1	the number of same-class neighboring points (homogeneous neighbors).
k2	the number of different-class neighboring points (heterogeneous neighbors).
t	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Yan S, Xu D, Zhang B, Zhang H, Yang Q, Lin S (2007). "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(1), 40–51.

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=20)-100
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+100

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different numbers for neighborhood size
out1 = do.kmfa(X, label, k1=10, k2=10, t=0.001)
out2 = do.kmfa(X, label, k1=10, k2=10, t=0.01)
out3 = do.kmfa(X, label, k1=10, k2=10, t=0.1)

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="bandwidth=0.001")
plot(out2$Y, pch=19, col=label, main="bandwidth=0.01")
```

```
plot(out3$Y, pch=19, col=label, main="bandwidth=0.1")
par(opar)
```

---

do.kmmc *Kernel Maximum Margin Criterion*

Description

Kernel Maximum Margin Criterion (KMMC) is a nonlinear variant of MMC method using kernel trick. For computational simplicity, only the gaussian kernel is used with bandwidth parameter `t`.

Usage

```
do.kmmc(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "decorrelate", "whiten"),
  t = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`t`	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Li H, Jiang T, Zhang K (2006). "Efficient and Robust Feature Extraction by Maximum Margin Criterion." *IEEE Transactions on Neural Networks*, **17**(1), 157–165.

See Also

do.mmc

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,100)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## perform MVP with different preprocessings
out1 = do.kmmc(X, label, t=0.1)
out2 = do.kmmc(X, label, t=1.0)
out3 = do.kmmc(X, label, t=10.0)

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="bandwidth=0.1")
plot(out2$Y, pch=19, col=label, main="bandwidth=1")
plot(out3$Y, pch=19, col=label, main="bandwidth=10.0")
par(opar)
```

---

do.kmvp *Kernel-Weighted Maximum Variance Projection*

Description

Kernel-Weighted Maximum Variance Projection (KMVP) is a generalization of Maximum Variance Projection (MVP). Even though its name contains *kernel*, it is not related to kernel trick well known in the machine learning community. Rather, it generalizes the binary penalization on class discrepancy,

$$S_{ij} = \exp(-\|x_i - x_j\|^2/t) \quad \text{if} \quad C_i \neq C_j$$

where x_i is an i-th data point and t a kernel bandwidth (`bandwidth`). **Note** that when the bandwidth value is too small, it might suffer from numerical instability and rank deficiency due to its formulation.

Usage

```
do.kmvp(
  X,
  label,
  ndim = 2,
```

```
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  bandwidth = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
bandwidth	bandwidth parameter for heat kernel as the equation above.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang T (2007). "Maximum variance projections for face recognition." *Optical Engineering*, **46**(6), 067206.

See Also

do.mvp

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## perform KMVP with different bandwidths
out1 = do.kmvp(X, label, bandwidth=0.1)
out2 = do.kmvp(X, label, bandwidth=1)
out3 = do.kmvp(X, label, bandwidth=10)
```

```
## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="bandwidth=0.1", col=label, pch=19)
plot(out2$Y, main="bandwidth=1",   col=label, pch=19)
plot(out3$Y, main="bandwidth=10",  col=label, pch=19)
par(opar)
```

`do.kpca` *Kernel Principal Component Analysis*

Description

Kernel principal component analysis (KPCA/Kernel PCA) is a nonlinear extension of classical PCA using techniques called kernel trick, a common method of introducing nonlinearity by transforming, usually, covariance structure or other gram-type estimate to make it flexible in Reproducing Kernel Hilbert Space.

Usage

```
do.kpca(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  kernel = c("gaussian", 1)
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`kernel`	a vector containing name of a kernel and corresponding parameters. See also `aux.kernelcov` for complete description of Kernel Trick.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

vars variances of projected data / eigenvalues from kernelized covariance matrix.

Author(s)

Kisung You

References

Schölkopf B, Smola A, Müller K (1997). "Kernel principal component analysis." In Goos G, Hartmanis J, van Leeuwen J, Gerstner W, Germond A, Hasler M, Nicoud J (eds.), *Artificial Neural Networks — ICANN'97*, volume 1327, 583–588. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-63631-1 978-3-540-69620-9, doi: 10.1007/BFb0020217.

See Also

aux.kernelcov

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try out different settings
output1 <- do.kpca(X)                          # default setting
output2 <- do.kpca(X,kernel=c("gaussian",5))  # gaussian kernel with large bandwidth
output3 <- do.kpca(X,kernel=c("laplacian",1)) # laplacian kernel

## visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, col=label, pch=19, main="Gaussian kernel")
plot(output2$Y, col=label, pch=19, main="Gaussian kernel with sigma=5")
plot(output3$Y, col=label, pch=19, main="Laplacian kernel")
par(opar)
```

---

do.kqmi *Kernel Quadratic Mutual Information*

Description

Kernel Quadratic Mutual Information (KQMI) is a supervised linear dimension reduction method. Quadratic Mutual Information is an efficient nonparametric estimation method for Mutual Information for class labels not requiring class priors. The method re-states the estimation procedure in terms of kernel objective in the graph embedding framework.

Usage

```
do.kqmi(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  t = 10
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
t	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Bouzas D, Arvanitopoulos N, Tefas A (2015). "Graph Embedded Nonparametric Mutual Information for Supervised Dimensionality Reduction." *IEEE Transactions on Neural Networks and Learning Systems*, **26**(5), 951–963.

See Also

do.lqmi

Examples

```
## Not run:
## generate 3 different groups of data X and label vector
x1 = matrix(rnorm(4*10), nrow=10)-20
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+20
```

```
X  = rbind(x1, x2, x3)
label = c(rep(1,10), rep(2,10), rep(3,10))

## try different kernel bandwidths
out1 = do.kqmi(X, label, t=0.01)
out2 = do.kqmi(X, label, t=1)
out3 = do.kqmi(X, label, t=100)

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="KQMI::t=0.01")
plot(out2$Y, col=label, main="KQMI::t=1")
plot(out3$Y, col=label, main="KQMI::t=100")
par(opar)

## End(Not run)
```

`do.ksda` *Kernel Semi-Supervised Discriminant Analysis*

Description

Kernel Semi-Supervised Discriminant Analysis (KSDA) is a nonlinear variant of SDA (`do.sda`). For simplicity, we enabled heat/gaussian kernel only. Note that this method is *quite* sensitive to choices of parameters, `alpha`, `beta`, and `t`. Especially when data are well separated in the original space, it may lead to unsatisfactory results.

Usage

```
do.ksda(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  alpha = 1,
  beta = 1,
  t = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.

type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
alpha	balancing parameter between model complexity and empirical loss.
beta	Tikhonov regularization parameter.
t	bandwidth parameter for heat kernel.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Cai D, He X, Han J (2007). "Semi-supervised Discriminant Analysis." In *2007 IEEE 11th International Conference on Computer Vision*, 1–7.

See Also

do.sda

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=20)-100
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+100

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## copy a label and let 10% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.10)
label_missing = label
label_missing[sample(1:nlabel, nmissing)]=NA

## compare true case with missing-label case
out1 = do.ksda(X, label, beta=0, t=0.1)
out2 = do.ksda(X, label_missing, beta=0, t=0.1)
```

```
## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=label, main="true projection")
plot(out2$Y, col=label, main="20% missing labels")
par(opar)
```

---

do.kudp *Kernel-Weighted Unsupervised Discriminant Projection*

Description

Kernel-Weighted Unsupervised Discriminant Projection (KUDP) is a generalization of UDP where proximity is given by weighted values via heat kernel,

$$K_{i,j} = \exp(-\|x_i - x_j\|^2/bandwidth)$$

whence UDP uses binary connectivity. If `bandwidth` is $+\infty$, it becomes a standard UDP problem. Like UDP, it also performs PCA preprocessing for rank-deficient case.

Usage

```
do.kudp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  bandwidth = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`bandwidth`	bandwidth parameter for heat kernel as the equation above.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

interimdim the number of PCA target dimension used in preprocessing.

Author(s)

Kisung You

References

Yang J, Zhang D, Yang J, Niu B (2007). "Globally Maximizing, Locally Minimizing: Unsupervised Discriminant Projection with Applications to Face and Palm Biometrics." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(4), 650–664.

See Also

do.udp

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## use different kernel bandwidth
out1 <- do.kudp(X, bandwidth=0.1)
out2 <- do.kudp(X, bandwidth=10)
out3 <- do.kudp(X, bandwidth=1000)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=lab, pch=19, main="bandwidth=0.1")
plot(out2$Y, col=lab, pch=19, main="bandwidth=10")
plot(out3$Y, col=lab, pch=19, main="bandwidth=1000")
par(opar)
```

do.lamp *Local Affine Multidimensional Projection*

Description

Local Affine Mulditimensional Projection (*LAMP*) can be considered as a nonlinear method even though each datam is projected using locally estimated affine mapping. It first finds a low-dimensional embedding for control points and then locates the rest data using affine mapping. We use $\sqrt{n}$ number of data as controls and Stochastic Neighborhood Embedding is applied as an initial projection of control set. Note that this belongs to the method for visualization so projection onto $\mathbf{R}^2$ is suggested for use.

Usage

```
do.lamp(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Joia P, Paulovich FV, Coimbra D, Cuminato JA, Nonato LG (2011). "Local Affine Multidimensional Projection." *IEEE Transactions on Visualization and Computer Graphics*, **17**(12), 2563–2571.

See Also

do.sne

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## let's compare with PCA
out1 <- do.pca(X, ndim=2)      # PCA
out2 <- do.lamp(X, ndim=2)     # LAMP

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, pch=19, col=label, main="PCA")
plot(out2$Y, pch=19, col=label, main="LAMP")
par(opar)
```

---

do.lapeig *Laplacian Eigenmaps*

Description

do.lapeig performs Laplacian Eigenmaps (LE) to discover low-dimensional manifold embedded in high-dimensional data space using graph laplacians. This is a classic algorithm employing spectral graph theory.

Usage

```
do.lapeig(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  weighted = FALSE,
  kernelscale = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.

type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
weighted	TRUE for weighted graph laplacian and FALSE for combinatorial laplacian where connectivity is represented as 1 or 0 only.
kernelscale	kernel scale parameter. Default value is 1.0.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

eigvals a vector of eigenvalues for laplacian matrix.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Belkin M, Niyogi P (2003). "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation*, **15**(6), 1373–1396.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different levels of connectivity
out1 <- do.lapeig(X, type=c("proportion",0.10), weighted=FALSE)
out2 <- do.lapeig(X, type=c("proportion",0.20), weighted=FALSE)
out3 <- do.lapeig(X, type=c("proportion",0.50), weighted=FALSE)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="10% connected")
plot(out2$Y, pch=19, col=lab, main="20% connected")
```

```
plot(out3$Y, pch=19, col=lab, main="50% connected")
par(opar)
```

`do.lasso` *Least Absolute Shrinkage and Selection Operator*

Description

LASSO is a popular regularization scheme in linear regression in pursuit of sparsity in coefficient vector that has been widely used. The method can be used in feature selection in that given the regularization parameter, it first solves the problem and takes indices of estimated coefficients with the largest magnitude as meaningful features by solving

$$\min_\beta \frac{1}{2}\|X\beta - y\|_2^2 + \lambda\|\beta\|_1$$

where y is `response` in our method.

Usage

```
do.lasso(
  X,
  response,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  ycenter = FALSE,
  lambda = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix whose rows are observations and columns represent independent variables.
`response`	a length-n vector of response variable.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`ycenter`	a logical; `TRUE` to center the response variable, `FALSE` otherwise.
`lambda`	sparsity regularization parameter in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Tibshirani R (1996). "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**(1), 267–288. tex.publisher: [Royal Statistical Society, Wiley].

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(1)
n = 123
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try different regularization parameters
out1 = do.lasso(X, y, lambda=0.1)
out2 = do.lasso(X, y, lambda=1)
out3 = do.lasso(X, y, lambda=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="LASSO::lambda=0.1")
plot(out2$Y, main="LASSO::lambda=1")
plot(out3$Y, main="LASSO::lambda=10")
par(opar)
```

---

do.lda *Linear Discriminant Analysis*

Description

Linear Discriminant Analysis (LDA) originally aims to find a set of features that best separate groups of data. Since we need *label* information, LDA belongs to a class of supervised methods of performing classification. However, since it is based on finding *suitable* projections, it can still

be used to do dimension reduction. We support both binary and multiple-class cases. Note that the target dimension `ndim` should be *less than or equal to* `K-1`, where `K` is the number of classes, or `K=length(unique(label))`. Our code automatically gives bounds on user's choice to correspond to what theory has shown. See the comments section for more details.

Usage

```
do.lda(X, label, ndim = 2)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Limit of Target Dimension Selection

In unsupervised algorithms, selection of `ndim` is arbitrary as long as the target dimension is lower-dimensional than original data dimension, i.e., `ndim < p`. In LDA, it is *not allowed*. Suppose we have `K` classes, then its formulation on S_B, between-group variance, has maximum rank of `K-1`. Therefore, the maximal subspace can only be spanned by at most `K-1` orthogonal vectors.

Author(s)

Kisung You

References

Fisher RA (1936). "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS." *Annals of Eugenics*, **7**(2), 179–188.

Fukunaga K (1990). *Introduction to statistical pattern recognition*, Computer science and scientific computing, 2nd ed edition. Academic Press, Boston. ISBN 978-0-12-269851-4.

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
```

```
lab   = as.factor(iris[subid,5])

## perform onto 2-dimensional space
output = do.lda(X, lab, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
plot(output$Y, col=lab, pch=19, main="3 groups on 2d plane")
par(opar)
```

`do.ldakm` *Combination of LDA and K-means*

Description

`do.ldakm` is an unsupervised subspace discovery method that combines linear discriminant analysis (LDA) and K-means algorithm. It tries to build an adaptive framework that selects the most discriminative subspace. It iteratively applies two methods in that the clustering process is integrated with the subspace selection, and continuously updates its discrimative basis. From its formulation with respect to generalized eigenvalue problem, it can be considered as generalization of Adaptive Subspace Iteration (ASI) and Adaptive Dimension Reduction (ADR).

Usage

```
do.ldakm(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  maxiter = 10,
  abstol = 0.001
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`maxiter`	maximum number of iterations allowed.
`abstol`	stopping criterion for incremental change in projection matrix.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Ding C, Li T (2007). "Adaptive dimension reduction using discriminant analysis and K-means clustering." In *Proceedings of the 24th international conference on Machine learning*, 521–528.

See Also

do.asi, do.adr

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid <- sample(1:150, 50)
X     <- as.matrix(iris[subid,1:4])
lab   <- as.factor(iris[subid,5])

## try different tolerance level
out1 = do.ldakm(X, abstol=1e-2)
out2 = do.ldakm(X, abstol=1e-3)
out3 = do.ldakm(X, abstol=1e-4)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="LDA-KM::tol=1e-2")
plot(out2$Y, pch=19, col=lab, main="LDA-KM::tol=1e-3")
plot(out3$Y, pch=19, col=lab, main="LDA-KM::tol=1e-4")
par(opar)
```

---

do.lde *Local Discriminant Embedding*

Description

Local Discriminant Embedding (LDE) is a supervised algorithm that learns the embedding for the submanifold of each class. Its idea is to same-class data points maintain their original neighborhood information while segregating different-class data distinct from each other.

Usage

```
do.lde(
  X,
  label,
  ndim = 2,
  t = 1,
  numk = max(ceiling(nrow(X)/10), 2),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
t	kernel bandwidth in $(0, \infty)$.
numk	the number of neighboring points for k-nn graph construction.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Hwann-Tzong Chen, Huang-Wei Chang, Tyng-Luh Liu (2005). "Local Discriminant Embedding and Its Variants." In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 846–853.

Examples

```
## generate data of 2 types with clear difference
set.seed(100)
diff = 15
dt1  = aux.gensamples(n=50)-diff;
dt2  = aux.gensamples(n=50)+diff;

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2)
```

```
label  = rep(1:2, each=50)

## try different neighborhood size
out1 <- do.lde(X, label, numk=5)
out2 <- do.lde(X, label, numk=10)
out3 <- do.lde(X, label, numk=25)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="LDE::k=5")
plot(out2$Y, pch=19, col=label, main="LDE::k=10")
plot(out3$Y, pch=19, col=label, main="LDE::k=25")
par(opar)
```

`do.ldp` *Locally Discriminating Projection*

Description

Locally Discriminating Projection (LDP) is a supervised linear dimension reduction method. It utilizes both label/class information and local neighborhood information to discover the intrinsic structure of the data. It can be considered as an extension of LPP in a supervised manner.

Usage

```
do.ldp(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  beta = 10
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`beta`	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhao H, Sun S, Jing Z, Yang J (2006). "Local structure based supervised feature extraction." *Pattern Recognition*, **39**(8), 1546–1550.

Examples

```
## generate data of 3 types with clear difference
dt1  = aux.gensamples(n=20)-100
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+100

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different neighborhood sizes
out1 = do.ldp(X, label, type=c("proportion",0.10))
out2 = do.ldp(X, label, type=c("proportion",0.25))
out3 = do.ldp(X, label, type=c("proportion",0.50))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="10% connectivity")
plot(out2$Y, col=label, pch=19, main="25% connectivity")
plot(out3$Y, col=label, pch=19, main="50% connectivity")
par(opar)
```

`do.lea` *Locally Linear Embedded Eigenspace Analysis*

Description

Locally Linear Embedding (LLE) is a powerful nonlinear manifold learning method. This method, Locally Linear Embedded Eigenspace Analysis - LEA, in short - is a linear approximation to LLE, similar to Neighborhood Preserving Embedding. In our implementation, the choice of weight binarization is removed in order to respect original work. For 1-dimensional projection, which is rarely performed, authors provided a detour for rank correcting mechanism but it is omitted for practical reason.

Usage

```
do.lea(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Fu Y, Huang TS (2005). "Locally Linear Embedded Eigenspace Analysis." *IFP-TR, UIUC*, **2005**, 2–05.

See Also

do.npe

Examples

```
## Not run:
## use iris dataset
data(iris)
set.seed(100)
subid <- sample(1:150, 50)
X     <- as.matrix(iris[subid,1:4])
lab   <- as.factor(iris[subid,5])

## compare LEA with LLE and another approximation NPE
out1 <- do.lle(X, ndim=2)
out2 <- do.npe(X, ndim=2)
out3 <- do.lea(X, ndim=2)

## visual comparison
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="LLE")
plot(out2$Y, pch=19, col=lab, main="NPE")
plot(out3$Y, pch=19, col=lab, main="LEA")
par(opar)

## End(Not run)
```

---

do.lfda *Local Fisher Discriminant Analysis*

Description

Local Fisher Discriminant Analysis (LFDA) is a linear dimension reduction method for supervised case, i.e., labels are given. It reflects *local* information to overcome undesired results of traditional Fisher Discriminant Analysis which results in a poor mapping when samples in a single class form form several separate clusters.

Usage

```
do.lfda(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  localscaling = TRUE
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
localscaling	TRUE to use local scaling method for construction affinity matrix, FALSE for binary affinity.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Sugiyama M (2006). "Local Fisher discriminant analysis for supervised dimensionality reduction." In *Proceedings of the 23rd international conference on Machine learning*, 905–912.

Zelnik-manor L, Perona P (2005). "Self-Tuning Spectral Clustering." In Saul LK, Weiss Y, Bottou L (eds.), *Advances in Neural Information Processing Systems 17*, 1601–1608. MIT Press. http://papers.nips.cc/paper/2619-self-tuning-spectral-clustering.pdf.

Examples

```
## generate 3 different groups of data X and label vector
x1 = matrix(rnorm(4*10), nrow=10)-20
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+20
X     = rbind(x1, x2, x3)
label = rep(1:3, each=10)

## try different affinity matrices
out1 = do.lfda(X, label)
```

```
out2 = do.lfda(X, label, localscaling=FALSE)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=label, main="binary affinity matrix")
plot(out2$Y, col=label, main="local scaling affinity")
par(opar)
```

`do.lisomap` *Landmark Isometric Feature Mapping*

Description

Landmark Isomap is a variant of Isomap in that it first finds a low-dimensional embedding using a small portion of given dataset and graft the others in a manner to preserve as much pairwise distance from all the other data points to landmark points as possible.

Usage

```
do.lisomap(
  X,
  ndim = 2,
  ltype = c("random", "MaxMin"),
  npoints = max(nrow(X)/5, ndim + 1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  weight = TRUE
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`ltype`	on how to select landmark points, either `"random"` or `"MaxMin"`.
`npoints`	the number of landmark points to be drawn.
`preprocess`	an option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`weight`	`TRUE` to perform Landmark Isomap on weighted graph, or `FALSE` otherwise.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Silva VD, Tenenbaum JB (2003). "Global Versus Local Methods in Nonlinear Dimensionality Reduction." In Becker S, Thrun S, Obermayer K (eds.), *Advances in Neural Information Processing Systems 15*, 721–728. MIT Press.

See Also

do.isomap

Examples

```
## use iris data
data(iris)
X   <- as.matrix(iris[,1:4])
lab <- as.factor(iris[,5])

## use different number of data points as landmarks
output1 <- do.lisomap(X, npoints=10, type=c("proportion",0.25))
output2 <- do.lisomap(X, npoints=25, type=c("proportion",0.25))
output3 <- do.lisomap(X, npoints=50, type=c("proportion",0.25))

## visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, pch=19, col=lab, main="10 landmarks")
plot(output2$Y, pch=19, col=lab, main="25 landmarks")
plot(output3$Y, pch=19, col=lab, main="50 landmarks")
par(opar)
```

`do.lle` *Locally Linear Embedding*

Description

Locally-Linear Embedding (LLE) was introduced approximately at the same time as Isomap. Its idea was motivated to describe entire data manifold by making a chain of local patches in that low-dimensional embedding should resemble the connectivity pattern of patches. `do.lle` also provides an automatic choice of regularization parameter based on an optimality criterion suggested by authors.

Usage

```
do.lle(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = "union",
  weight = TRUE,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  regtype = FALSE,
  regparam = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`weight`	`TRUE` to perform LLE on weighted graph, or `FALSE` otherwise.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`regtype`	`TRUE` for automatic regularization parameter selection, `FALSE` otherwise as default.
`regparam`	regularization parameter.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

eigvals a vector of eigenvalues from computation of embedding matrix.

Author(s)

Kisung You

References

Roweis ST (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding." *Science*, **290**(5500), 2323–2326.

Examples

```
## generate swiss-roll data
set.seed(100)
X = aux.gensamples(n=100)

## 1. connecting 10% of data for graph construction.
output1 <- do.lle(X,ndim=2,type=c("proportion",0.10))

## 2. constructing 20%-connected graph
output2 <- do.lle(X,ndim=2,type=c("proportion",0.20))

## 3. constructing 50%-connected with bigger regularization parameter
output3 <- do.lle(X,ndim=2,type=c("proportion",0.5),regparam=10)

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, main="5%")
plot(output2$Y, main="10%")
plot(output3$Y, main="50%+Binary")
par(opar)
```

`do.llle` *Local Linear Laplacian Eigenmaps*

Description

Local Linear Laplacian Eigenmaps is an unsupervised manifold learning method as an extension of Local Linear Embedding (do.lle). It is claimed to be more robust to local structure and noises. It involves the concept of artificial neighborhood in constructing the adjacency graph for reconstruction of the approximated manifold.

Usage

```
do.llle(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  K = round(nrow(X)/2),
  P = max(round(nrow(X)/4), 2),
  bandwidth = 0.2
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
K	size of near neighborhood for each data point.
P	size of artifical neighborhood.
bandwidth	scale parameter for Gaussian kernel. It should be in $(0, 1)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Liu F, Zhang W, Gu S (2016). "Local linear Laplacian eigenmaps: A direct extension of LLE." *Pattern Recognition Letters*, **75**, 30–35.

See Also

do.lle

Examples

```
## Not run:
## use iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.integer(iris$Species)

# see the effect bandwidth
out1 = do.llle(X, bandwidth=0.1, P=20)
out2 = do.llle(X, bandwidth=0.5, P=20)
out3 = do.llle(X, bandwidth=0.9, P=20)

# visualize the results
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="bandwidth=0.1")
plot(out2$Y, col=label, main="bandwidth=0.5")
plot(out3$Y, col=label, main="bandwidth=0.9")
par(opar)

## End(Not run)
```

`do.llp` *Local Learning Projections*

Description

While Principal Component Analysis (PCA) aims at minimizing global estimation error, Local Learning Projection (LLP) approach tries to find the projection with the minimal *local* estimation error in the sense that each projected datum can be well represented based on ones neighbors. For the kernel part, we only enabled to use a gaussian kernel as suggested from the original paper. The parameter `lambda` controls possible rank-deficiency of kernel matrix.

Usage

```
do.llp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  t = 1,
  lambda = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
t	bandwidth for heat kernel in $(0, \infty)$.
lambda	regularization parameter for kernel matrix in $[0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

References

Wu M, Yu K, Yu S, Schölkopf B (2007). "Local learning projections." In *Proceedings of the 24th international conference on Machine learning*, 1039–1046.

Examples

```
## generate data
set.seed(100)
X <- aux.gensamples(n=100, dname="crown")

## test different lambda - regularization - values
out1 <- do.llp(X,ndim=2,lambda=0.1)
out2 <- do.llp(X,ndim=2,lambda=1)
out3 <- do.llp(X,ndim=2,lambda=10)

# visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, main="lambda=0.1")
plot(out2$Y, pch=19, main="lambda=1")
plot(out3$Y, pch=19, main="lambda=10")
par(opar)
```

`do.lltsa` *Linear Local Tangent Space Alignment*

Description

Linear Local Tangent Space Alignment (LLTSA) is a linear variant of the celebrated LTSA method. It uses the tangent space in the neighborhood for each data point to represent the local geometry. Alignment of those local tangent spaces in the low-dimensional space returns an explicit mapping from the high-dimensional space.

Usage

```
do.lltsa(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang T, Yang J, Zhao D, Ge X (2007). "Linear local tangent space alignment and application to face recognition." *Neurocomputing*, **70**(7-9), 1547–1553.

See Also

do.ltsa

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different neighborhood size
out1 <- do.lltsa(X, type=c("proportion",0.25))
out2 <- do.lltsa(X, type=c("proportion",0.50))
out3 <- do.lltsa(X, type=c("proportion",0.75))

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=lab, pch=19, main="LLTSA::25% connected")
plot(out2$Y, col=lab, pch=19, main="LLTSA::50% connected")
plot(out3$Y, col=lab, pch=19, main="LLTSA::75% connected")
par(opar)
```

---

do.lmds *Landmark Multidimensional Scaling*

Description

Landmark MDS is a variant of Classical Multidimensional Scaling in that it first finds a low-dimensional embedding using a small portion of given dataset and graft the others in a manner to preserve as much pairwise distance from all the other data points to landmark points as possible.

Usage

```
do.lmds(
  X,
  ndim = 2,
  npoints = max(nrow(X)/5, ndim + 1),
  preprocess = c("center", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
npoints	the number of landmark points to be drawn.
preprocess	an option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Silva VD, Tenenbaum JB (2002). "Global Versus Local Methods in Nonlinear Dimensionality Reduction." In Thrun S, Obermayer K (eds.), *Advances in Neural Information Processing Systems 15*, 705–712. MIT Press, Cambridge, MA.

Lee S, Choi S (2009). "Landmark MDS ensemble." *Pattern Recognition*, **42**(9), 2045–2053.

See Also

do.mds

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## use 10% and 25% of the data and compare with full MDS
output1 <- do.lmds(X,npoints=round(nrow(X)*0.10))
output2 <- do.lmds(X,npoints=round(nrow(X)*0.25))
output3 <- do.mds(X,ndim=2)

## vsualization
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, pch=19, col=lab, main="10% random points")
```

```
plot(output2$Y, pch=19, col=lab, main="25% random points")
plot(output3$Y, pch=19, col=lab, main="original MDS")
par(opar)
```

`do.lpca2006` *Locally Principal Component Analysis by Yang et al. (2006)*

Description

Locally Principal Component Analysis (LPCA) is an unsupervised linear dimension reduction method. It focuses on the information brought by local neighborhood structure and seeks the corresponding structure, which may contain useful information for revealing discriminative information of the data.

Usage

```
do.lpca2006(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Yang J, Zhang D, Yang J (2006). "Locally principal component learning for face representation and recognition." *Neurocomputing*, **69**(13-15), 1697–1701.

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid = sample(1:150,100)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different neighborhood size
out1 <- do.lpca2006(X, ndim=2, type=c("proportion",0.25))
out2 <- do.lpca2006(X, ndim=2, type=c("proportion",0.50))
out3 <- do.lpca2006(X, ndim=2, type=c("proportion",0.75))

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="LPCA2006::25% connected")
plot(out2$Y, pch=19, col=lab, main="LPCA2006::50% connected")
plot(out3$Y, pch=19, col=lab, main="LPCA2006::75% connected")
par(opar)
```

---

do.lpe *Locality Pursuit Embedding*

Description

Locality Pursuit Embedding (LPE) is an unsupervised linear dimension reduction method. It aims at preserving local structure by solving a variational problem that models the local geometrical structure by the Euclidean distances.

Usage

```
do.lpe(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  numk = max(ceiling(nrow(X)/10), 2)
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
numk	size of k-nn neighborhood in original dimensional space.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Min W, Lu K, He X (2004). "Locality pursuit embedding." *Pattern Recognition*, **37**(4), 781–788.

Examples

```
## generate swiss roll with auxiliary dimensions
set.seed(100)
n     = 100
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## try with different neighborhood sizes
out1 = do.lpe(X, numk=5)
out2 = do.lpe(X, numk=10)
out3 = do.lpe(X, numk=25)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="LPE::numk=5")
plot(out2$Y, main="LPE::numk=10")
plot(out3$Y, main="LPE::numk=25")
```

```
par(opar)
```

`do.lpfda` *Locality Preserving Fisher Discriminant Analysis*

Description

Locality Preserving Fisher Discriminant Analysis (LPFDA) is a supervised variant of LPP. It can also be seemed as an improved version of LDA where the locality structure of the data is preserved. The algorithm aims at getting a subspace projection matrix by solving a generalized eigenvalue problem.

Usage

```
do.lpfda(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  t = 10
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`t`	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhao X, Tian X (2009). "Locality Preserving Fisher Discriminant Analysis for Face Recognition." In Huang D, Jo K, Lee H, Kang H, Bevilacqua V (eds.), *Emerging Intelligent Computing Technology and Applications*, 261–269.

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=20)-50
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+50

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different proportion of connected edges
out1 = do.lpfda(X, label, type=c("proportion",0.10))
out2 = do.lpfda(X, label, type=c("proportion",0.25))
out3 = do.lpfda(X, label, type=c("proportion",0.50))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="10% connectivity")
plot(out2$Y, pch=19, col=label, main="25% connectivity")
plot(out3$Y, pch=19, col=label, main="50% connectivity")
par(opar)
```

---

do.lpmip *Locality-Preserved Maximum Information Projection*

Description

Locality-Preserved Maximum Information Projection (LPMIP) is an unsupervised linear dimension reduction method to identify the underlying manifold structure by learning both the within- and between-locality information. The parameter `alpha` is balancing the tradeoff between two and the flexibility of this model enables an interpretation of it as a generalized extension of LPP.

Usage

```
do.lpmip(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  sigma = 10,
  alpha = 0.5
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
sigma	bandwidth parameter for heat kernel in $(0, \infty)$.
alpha	balancing parameter between two locality information in $[0, 1]$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Haixian Wang, Sibao Chen, Zilan Hu, Wenming Zheng (2008). "Locality-Preserved Maximum Information Projection." *IEEE Transactions on Neural Networks*, **19**(4), 571–585.

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid <- sample(1:150, 50)
X     <- as.matrix(iris[subid,1:4])
```

```
lab   <- as.factor(iris[subid,5])

## try different neighborhood size
out1 <- do.lpmip(X, ndim=2, type=c("proportion",0.10))
out2 <- do.lpmip(X, ndim=2, type=c("proportion",0.25))
out3 <- do.lpmip(X, ndim=2, type=c("proportion",0.50))

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="10% connected")
plot(out2$Y, pch=19, col=lab, main="25% connected")
plot(out3$Y, pch=19, col=lab, main="50% connected")
par(opar)
```

`do.lpp` *Locality Preserving Projection*

Description

`do.lpp` is a linear approximation to Laplacian Eigenmaps. More precisely, it aims at finding a linear approximation to the eigenfunctions of the Laplace-Beltrami operator on the graph-approximated data manifold.

Usage

```
do.lpp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  t = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is `"center"`. See also `aux.preprocess` for more details.
`t`	bandwidth for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

He X (2005). *Locality Preserving Projections*. PhD Thesis, University of Chicago, Chicago, IL, USA.

Examples

```
## use iris dataset
data(iris)
set.seed(100)
subid <- sample(1:150, 50)
X     <- as.matrix(iris[subid,1:4])
lab   <- as.factor(iris[subid,5])

## try different kernel bandwidths
out1 <- do.lpp(X, t=0.1)
out2 <- do.lpp(X, t=1)
out3 <- do.lpp(X, t=10)

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=lab, pch=19, main="LPP::bandwidth=0.1")
plot(out2$Y, col=lab, pch=19, main="LPP::bandwidth=1")
plot(out3$Y, col=lab, pch=19, main="LPP::bandwidth=10")
par(opar)
```

`do.lqmi` *Linear Quadratic Mutual Information*

Description

Linear Quadratic Mutual Information (LQMI) is a supervised linear dimension reduction method. Quadratic Mutual Information is an efficient nonparametric estimation method for Mutual Information for class labels not requiring class priors. For the KQMI formulation, LQMI is a linear equivalent.

Usage

```
do.lqmi(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Bouzas D, Arvanitopoulos N, Tefas A (2015). "Graph Embedded Nonparametric Mutual Information for Supervised Dimensionality Reduction." *IEEE Transactions on Neural Networks and Learning Systems*, **26**(5), 951–963.

See Also

do.kqmi

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare against LDA
```

```
out1 = do.lda(X, label)
out2 = do.lqmi(X, label)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=label, main="LDA projection")
plot(out2$Y, col=label, main="LQMI projection")
par(opar)
```

`do.lscore` *Laplacian Score*

Description

Laplacian Score (LSCORE) is an unsupervised linear feature extraction method. For each feature/variable, it computes Laplacian score based on an observation that data from the same class are often close to each other. Its power of locality preserving property is used, and the algorithm selects variables with smallest scores.

Usage

```
do.lscore(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  t = 10
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`t`	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

lscore a length-p vector of laplacian scores. Indices with smallest values are selected.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

He X, Cai D, Niyogi P (2005). "Laplacian Score for Feature Selection." In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, 507–514.

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid    <- sample(1:150, 50)
iris.dat <- as.matrix(iris[subid,1:4])
iris.lab <- as.factor(iris[subid,5])

## try different kernel bandwidth
out1 = do.lscore(iris.dat, t=0.1)
out2 = do.lscore(iris.dat, t=1)
out3 = do.lscore(iris.dat, t=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=iris.lab, main="bandwidth=0.1")
plot(out2$Y, pch=19, col=iris.lab, main="bandwidth=1")
plot(out3$Y, pch=19, col=iris.lab, main="bandwidth=10")
par(opar)
```

`do.lsda` *Locality Sensitive Discriminant Analysis*

Description

Locality Sensitive Discriminant Analysis (LSDA) is a supervised linear method. It aims at finding a projection which maximizes the margin between data points from different classes at each local area in which the nearby points with the same label are close to each other while the nearby points with different labels are far apart.

Usage

```
do.lsda(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  alpha = 0.5,
  k1 = max(ceiling(nrow(X)/10), 2),
  k2 = max(ceiling(nrow(X)/10), 2)
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`alpha`	balancing parameter for between- and within-class scatter in $[0, 1]$.
`k1`	the number of same-class neighboring points (homogeneous neighbors).
`k2`	the number of different-class neighboring points (heterogeneous neighbors).

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Cai D, He X, Zhou K, Han J, Bao H (2007). "Locality Sensitive Discriminant Analysis." In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 708–713.

Examples

```
## create a data matrix with clear difference
x1 = matrix(rnorm(4*10), nrow=10)-20
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+20
X  = rbind(x1, x2, x3)
label = c(rep(1,10), rep(2,10), rep(3,10))

## try different affinity matrices
out1 = do.lsda(X, label, k1=2, k2=2)
out2 = do.lsda(X, label, k1=5, k2=5)
out3 = do.lsda(X, label, k1=10, k2=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="nbd size 2")
plot(out2$Y, col=label, main="nbd size 5")
plot(out3$Y, col=label, main="nbd size 10")
par(opar)
```

---

do.lsdf *Locality Sensitive Discriminant Feature*

Description

Locality Sensitive Discriminant Feature (LSDF) is a semi-supervised feature selection method. It utilizes both labeled and unlabeled data points in that labeled points are used to maximize the margin between data opints from different classes, while labeled ones are used to discover the geometrical structure of the data space.

Usage

```
do.lsdf(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  gamma = 100
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels. It should contain NA elements for missing label.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
gamma	within-class weight parameter for same-class data.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Cai D, He X, Zhou K, Han J, Bao H (2007). "Locality Sensitive Discriminant Analysis." In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 708–713.

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=20)-50
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+50

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## copy a label and let 20% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.20)
label_missing = label
```

```
label_missing[sample(1:nlabel, nmissing)]=NA

## try different neighborhood sizes
out1 = do.lsdf(X, label_missing, type=c("proportion",0.10))
out2 = do.lsdf(X, label_missing, type=c("proportion",0.25))
out3 = do.lsdf(X, label_missing, type=c("proportion",0.50))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="10% connectivity")
plot(out2$Y, pch=19, col=label, main="25% connectivity")
plot(out3$Y, pch=19, col=label, main="50% connectivity")
par(opar)
```

`do.lsir` *Localized Sliced Inverse Regression*

Description

Localized SIR (SIR) is an extension of celebrated SIR method. As its name suggests, the *locality* concept is brought in that for each slice, only local data points are considered in order to discover intrinsic structure of the data.

Usage

```
do.lsir(
  X,
  response,
  ndim = 2,
  h = max(2, round(nrow(X)/5)),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  ycenter = FALSE,
  numk = max(2, round(nrow(X)/10)),
  tau = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`response`	a length-n vector of response variable.
`ndim`	an integer-valued target dimension.
`h`	the number of slices to divide the range of response vector.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

ycenter	a logical; TRUE to center the response variable, FALSE otherwise.
numk	size of determining neighborhood via k-nearest neighbor selection.
tau	regularization parameter for adjusting rank-deficient scatter matrix.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Wu Q, Liang F, Mukherjee S (2010). "Localized Sliced Inverse Regression." *Journal of Computational and Graphical Statistics*, **19**(4), 843–860.

See Also

do.sir

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(100)
n     = 123
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try different number of neighborhoods
out1 = do.lsir(X, y, numk=5)
out2 = do.lsir(X, y, numk=10)
out3 = do.lsir(X, y, numk=25)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
```

```
plot(out1$Y, main="LSIR::nbd size=5")
plot(out2$Y, main="LSIR::nbd size=10")
plot(out3$Y, main="LSIR::nbd size=25")
par(opar)
```

`do.lsls` *Locality Sensitive Laplacian Score*

Description

Locality Sensitive Laplacian Score (LSLS) is a supervised linear feature extraction method that combines a feature selection framework of laplacian score where the graph laplacian is adjusted as in the scheme of LSDA. The adjustment is taken via decomposed affinity matrices which are separately constructed using the provided class label information.

Usage

```
do.lsls(
  X,
  label,
  ndim = 2,
  alpha = 0.5,
  k = 5,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`alpha`	a weight factor; should be a real number in $[0, 1]$.
`k`	an integer; the size of a neighborhood.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Liao B, Jiang Y, Liang W, Zhu W, Cai L, Cao Z (2014). "Gene Selection Using Locality Sensitive Laplacian Score." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **11**(6), 1146–1156.

See Also

do.lsda, do.lscore

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid    = sample(1:150,50)
iris.dat = as.matrix(iris[subid,1:4])
iris.lab = as.factor(iris[subid,5])

## compare different neighborhood sizes
out1 = do.lsls(iris.dat, iris.lab, k=3)
out2 = do.lsls(iris.dat, iris.lab, k=6)
out3 = do.lsls(iris.dat, iris.lab, k=9)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=iris.lab, pch=19, main="LSLS::k=3")
plot(out2$Y, col=iris.lab, pch=19, main="LSLS::k=6")
plot(out3$Y, col=iris.lab, pch=19, main="LSLS::k=9")
par(opar)
```

---

do.lspe *Locality and Similarity Preserving Embedding*

Description

Locality and Similarity Preserving Embedding (LSPE) is a feature selection method based on Neighborhood Preserving Embedding (do.npe) and Sparsity Preserving Projection (do.spp) by first building a neighborhood graph and then mapping the locality structure to reconstruct coefficients such that data similarity is preserved. Use of $\ell_{2,1}$ norm boosts to impose column-sparsity that enables feature selection procedure.

Usage

```
do.lspe(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  alpha = 1,
  beta = 1,
  bandwidth = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
alpha	nonnegative number to control $\ell_{2,1}$ norm of projection.
beta	nonnegative number to control the degree of local similarity.
bandwidth	positive number for Gaussian kernel bandwidth to define similarity.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Fang X, Xu Y, Li X, Fan Z, Liu H, Chen Y (2014). "Locality and similarity preserving embedding for feature selection." *Neurocomputing*, **128**, 304–315.

See Also

do.rsr

Examples

```
#### generate R12in72 dataset
set.seed(100)
X = aux.gensamples(n=50, dname="R12in72")

#### try different bandwidth values
out1 = do.lspe(X, bandwidth=0.1)
out2 = do.lspe(X, bandwidth=1)
out3 = do.lspe(X, bandwidth=10)

#### visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="LSPE::bandwidth=0.1")
plot(out2$Y, main="LSPE::bandwidth=1")
plot(out3$Y, main="LSPE::bandwidth=10")
par(opar)
```

---

do.lspp *Local Similarity Preserving Projection*

Description

Local Similarity Preserving Projection (LSPP) is a variant of LPP in that it employs a sample-dependent graph generation process as of do.sdlpp. LSPP takes advantage of labeling information to correct local similarity weight in order to make intra-class weight larger than inter-class weight. It uses PCA preprocessing as suggested from the original work.

Usage

```
do.lspp(
  X,
  label,
  ndim = 2,
  t = 1,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
t	kernel bandwidth in $(0, \infty)$.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Huang P, Gao G (2015). "Local similarity preserving projections for face recognition." *AEU - International Journal of Electronics and Communications*, **69**(11), 1724–1732.

See Also

do.sdlpp, do.lpp

Examples

```
## generate data of 2 types with clear difference
diff = 15
dt1  = aux.gensamples(n=50)-diff;
dt2  = aux.gensamples(n=50)+diff;

## merge the data and create a label correspondingly
Y      = rbind(dt1,dt2)
label  = rep(1:2, each=50)

## compare with PCA
out1 <- do.pca(Y, ndim=2)
out2 <- do.slpp(Y, label, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=label, pch=19, main="PCA")
plot(out2$Y, col=label, pch=19, main="LSPP")
par(opar)
```

`do.ltsa` *Local Tangent Space Alignment*

Description

Local Tangent Space Alignment, or LTSA in short, is a nonlinear dimensionality reduction method that mimicks the behavior of low-dimensional manifold embedded in high-dimensional space. Similar to LLE, LTSA computes tangent space using nearest neighbors of a given data point, and a multiple of tangent spaces are gathered to to find an embedding that aligns the tangent spaces in target dimensional space.

Usage

```
do.ltsa(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

eigvals a vector of eigenvalues from the final decomposition.

Author(s)

Kisung You

References

Zhang T, Yang J, Zhao D, Ge X (2007). "Linear local tangent space alignment and application to face recognition." *Neurocomputing*, **70**(7-9), 1547–1553.

Examples

```
## generate data
set.seed(100)
X <- aux.gensamples(dname="cswiss",n=100)

## 1. use 10%-connected graph
output1 <- do.ltsa(X,ndim=2)

## 2. use 25%-connected graph
output2 <- do.ltsa(X,ndim=2,type=c("proportion",0.25))

## 3. use 50%-connected graph
output3 <- do.ltsa(X,ndim=2,type=c("proportion",0.50))

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, main="10%")
plot(output2$Y, main="25%")
plot(output3$Y, main="50%")
par(opar)
```

---

do.mcfs *Multi-Cluster Feature Selection*

Description

Multi-Cluster Feature Selection (MCFS) is an unsupervised feature selection method. Based on a multi-cluster assumption, it aims at finding meaningful features using sparse reconstruction of spectral basis using LASSO.

Usage

```
do.mcfs(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  K = max(round(nrow(X)/5), 2),
  lambda = 1,
  t = 10
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
K	assumed number of clusters in the original dataset.
lambda	ℓ_1 regularization parameter in $(0, \infty)$.
t	bandwidth parameter for heat kernel in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Cai D, Zhang C, He X (2010). "Unsupervised feature selection for multi-cluster data." In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 333–342.

Examples

```
## generate data of 3 types with clear difference
dt1  = aux.gensamples(n=20)-100
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+100

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different regularization parameters
out1 = do.mcfs(X, lambda=0.01)
out2 = do.mcfs(X, lambda=0.1)
```

```
out3 = do.mcfs(X, lambda=1)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="lambda=0.01")
plot(out2$Y, pch=19, col=label, main="lambda=0.1")
plot(out3$Y, pch=19, col=label, main="lambda=1")
par(opar)
```

`do.mds` *(Classical) Multidimensional Scaling*

Description

`do.mds` performs a classical Multidimensional Scaling (MDS) using `Rcpp` and `RcppArmadillo` package to achieve faster performance than `cmdscale`.

Usage

```
do.mds(
  X,
  ndim = 2,
  preprocess = c("center", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an option for preprocessing the data. Default is `"center"`. See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Kruskal JB (1964). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis." *Psychometrika*, **29**(1), 1–27.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different preprocessing
out1 <- do.mds(X,ndim=2)
out2 <- do.mds(X,ndim=2,preprocess="cscale")
out3 <- do.mds(X,ndim=2,preprocess="whiten")

## extract embeddings for each procedure
Y1 <- out1$Y; Y2 <- out2$Y; Y3 <- out3$Y

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, pch=19, col=lab, main="MDS::center")
plot(Y2, pch=19, col=lab, main="MDS::decorrelate")
plot(Y3, pch=19, col=lab, main="MDS::whiten")
par(opar)
```

---

do.mfa *Marginal Fisher Analysis*

Description

Marginal Fisher Analysis (MFA) is a supervised linear dimension reduction method. The intrinsic graph characterizes the intraclass compactness and connects each data point with its neighboring pionts of the same class, while the penalty graph connects the marginal points and characterizes the interclass separability.

Usage

```
do.mfa(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  k1 = max(ceiling(nrow(X)/10), 2),
  k2 = max(ceiling(nrow(X)/10), 2)
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
k1	the number of same-class neighboring points (homogeneous neighbors).
k2	the number of different-class neighboring points (heterogeneous neighbors).

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Yan S, Xu D, Zhang B, Zhang H, Yang Q, Lin S (2007). "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(1), 40–51.

Examples

```
## generate data of 3 types with clear difference
dt1  = aux.gensamples(n=20)-100
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+100

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different numbers for neighborhood size
out1 = do.mfa(X, label, k1=5, k2=5)
out2 = do.mfa(X, label, k1=10,k2=10)
out3 = do.mfa(X, label, k1=25,k2=25)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="MFA::nbd size=5")
plot(out2$Y, main="MFA::nbd size=10")
plot(out3$Y, main="MFA::nbd size=25")
```

```
par(opar)
```

`do.mlie` *Maximal Local Interclass Embedding*

Description

Maximal Local Interclass Embedding (MLIE) is a linear supervised method that the local interclass graph and the intrinsic graph are constructed to find a set of projections that maximize the local interclass scatter and the local intraclass compactness at the same time. It can be deemed an extended version of MFA.

Usage

```
do.mlie(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  k1 = max(ceiling(nrow(X)/10), 2),
  k2 = max(ceiling(nrow(X)/10), 2)
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`k1`	the number of same-class neighboring points (homogeneous neighbors).
`k2`	the number of different-class neighboring points (heterogeneous neighbors).

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

References

Lai Z, Zhao C, Chen Y, Jin Z (2011). "Maximal local interclass embedding with application to face recognition." *Machine Vision and Applications*, **22**(4), 619–627.

See Also

do.mfa

Examples

```
## Not run:
## generate data of 3 types with clear difference
set.seed(100)
diff = 100
dt1  = aux.gensamples(n=20)-diff
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+diff

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different numbers for neighborhood size
out1 = do.mlie(X, label, k1=5, k2=5)
out2 = do.mlie(X, label, k1=10,k2=10)
out3 = do.mlie(X, label, k1=25,k2=25)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="MLIE::nbd size=5")
plot(out2$Y, main="MLIE::nbd size=10")
plot(out3$Y, main="MLIE::nbd size=25")
par(opar)

## End(Not run)
```

---

do.mmc *Maximum Margin Criterion*

Description

Maximum Margin Criterion (MMC) is a linear supervised dimension reduction method that maximizes average margin between classes. The cost function is defined as

$$trace(S_b - S_w)$$

where S_b is an overall variance of class mean vectors, and S_w refers to spread of every class. Note that Principal Component Analysis (PCA) maximizes total scatter, $S_t = S_b + S_w$.

Usage

```
do.mmc(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Li H, Jiang T, Zhang K (2006). "Efficient and Robust Feature Extraction by Maximum Margin Criterion." *IEEE Transactions on Neural Networks*, **17**(1), 157–165.

Examples

```
## generate 3 different groups of data X and label vector
x1 = matrix(rnorm(4*10), nrow=10)-20
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+20
X     = rbind(x1, x2, x3)
label = rep(1:3, each=10)

## perform MVP with different preprocessings
out1 = do.mmc(X, label, ndim=2)
out2 = do.mmc(X, label, ndim=2, preprocess="decorrelate")

## visualize
opar <- par(no.readonly=TRUE)
```

```
par(mfrow=c(1,2))
plot(out1$Y, main="MMC::centering")
plot(out2$Y, main="MMC::decorrelating")
par(opar)
```

---

do.mmp *Maximum Margin Projection*

Description

Maximum Margin Projection (MMP) is a supervised linear method that maximizes the margin between positive and negative examples at each local neighborhood based on same- and different-class neighborhoods depending on class labels.

Usage

```
do.mmp(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  numk = max(ceiling(nrow(X)/10), 2),
  alpha = 0.5,
  gamma = 50
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
numk	the number of neighboring points.
alpha	balancing parameter in $[0, 1]$.
gamma	weight for same-label data points with large magnitude.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Xiaofei He, Deng Cai, Jiawei Han (2008). "Learning a Maximum Margin Subspace for Image Retrieval." *IEEE Transactions on Knowledge and Data Engineering*, **20**(2), 189–201.

Examples

```
## generate data of 3 types with clear difference
dt1  = aux.gensamples(n=20)-100
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+100

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## copy a label and let 20% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.20)
label_missing = label
label_missing[sample(1:nlabel, nmissing)]=NA

## compare with PCA case for full-label case
## for missing label case from MMP computation
out1 = do.pca(X, ndim=2)
out2 = do.mmp(X, label_missing, numk=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=label, main="PCA projection")
plot(out2$Y, col=label, main="20% missing labels")
par(opar)
```

---

do.mmsd *Multiple Maximum Scatter Difference*

Description

Multiple Maximum Scatter Difference (MMSD) is a supervised linear dimension reduction method. It is a variant of MSD in that discriminant vectors are orthonormal. Similar to MSD, it also does not suffer from rank deficiency issue of scatter matrix.

Usage

```
do.mmsd(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  C = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
C	nonnegative balancing parameter for intra- and inter-class scatter.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Fengxi Song, Zhang D, Dayong Mei, Zhongwei Guo (2007). "A Multiple Maximum Scatter Difference Discriminant Criterion for Facial Feature Extraction." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **37**(6), 1599–1606.

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=20)-50
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+50

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)
```

```
## try different balancing parameter
out1 = do.mmsd(X, label, C=0.01)
out2 = do.mmsd(X, label, C=1)
out3 = do.mmsd(X, label, C=100)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="MMSD::C=0.01")
plot(out2$Y, pch=19, col=label, main="MMSD::C=1")
plot(out3$Y, pch=19, col=label, main="MMSD::C=100")
par(opar)
```

---

do.modp *Modified Orthogonal Discriminant Projection*

Description

Modified Orthogonal Discriminant Projection (MODP) is a variant of Orthogonal Discriminant Projection (ODP). Authors argue the assumption in modeling ODP's mechanism to reflect distance and class labeling seem unsound. They propose a modified method to explore the intrinsic structure of original data and enhance the classification ability.

Usage

```
do.modp(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  alpha = 0.5,
  beta = 10
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
alpha	balancing parameter of non-local and local scatter in $[0, 1]$.
beta	scaling control parameter for distant pairs of data in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

References

Zhang S, Lei Y, Wu Y, Yang J (2011). "Modified orthogonal discriminant projection for classification." *Neurocomputing*, **74**(17), 3690–3694.

Examples

```
## generate 3 different groups of data X and label vector
x1 = matrix(rnorm(4*10), nrow=10)-20
x2 = matrix(rnorm(4*10), nrow=10)
x3 = matrix(rnorm(4*10), nrow=10)+20
X     = rbind(x1, x2, x3)
label = rep(1:3, each=10)

## try different beta (scaling control) parameter
out1 = do.modp(X, label, beta=1)
out2 = do.modp(X, label, beta=10)
out3 = do.modp(X, label, beta=100)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="MODP::beta=1")
plot(out2$Y, main="MODP::beta=10")
plot(out3$Y, main="MODP::beta=100")
par(opar)
```

`do.msd` *Maximum Scatter Difference*

Description

Maximum Scatter Difference (MSD) is a supervised linear dimension reduction method. The basic idea of MSD is to use *additive* cost function rather than *multiplicative* trace ratio criterion that was adopted by LDA. Due to such formulation, it can neglect sample-sample-size problem from rank-deficiency of between-class variance matrix.

Usage

```
do.msd(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  C = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`C`	nonnegative balancing parameter for intra- and inter-class variance.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Song F, Zhang D, Chen Q, Wang J (2007). "Face recognition based on a novel linear discriminant criterion." *Pattern Analysis and Applications*, **10**(3), 165–174.

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=20)-50
dt2  = aux.gensamples(n=20)
dt3  = aux.gensamples(n=20)+50

## merge the data and create a label correspondingly
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=20)

## try different balancing parameter
out1 = do.msd(X, label, C=0.01)
out2 = do.msd(X, label, C=1)
out3 = do.msd(X, label, C=100)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="MSD::C=0.01")
plot(out2$Y, pch=19, col=label, main="MSD::C=1")
plot(out3$Y, pch=19, col=label, main="MSD::C=100")
par(opar)
```

---

do.mve *Minimum Volume Embedding*

Description

Minimum Volume Embedding (MVE) is a nonlinear dimension reduction algorithm that exploits semidefinite programming (SDP), like MVU/SDE. Whereas MVU aims at stretching through all direction by maximizing $\sum \lambda_i$, MVE only opts for unrolling the top eigenspectrum and chooses to shrink left-over spectral dimension. For ease of use, unlike kernel PCA, we only made use of Gaussian kernel for MVE. Note that we adopted Rcsdp package in that when given large-scale dataset, it may result in extremely deteriorated computational performance.

Usage

```
do.mve(
  X,
  ndim = 2,
  knn = ceiling(nrow(X)/10),
  kwidth = 1,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  tol = 1e-04,
  maxiter = 1000
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
knn	size of k-nn neighborhood.
kwidth	bandwidth for Gaussian kernel.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
tol	stopping criterion for incremental change.
maxiter	maximum number of iterations allowed.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Shaw B, Jebara T (2007). "Minimum Volume Embedding." In Meila M, Shen X (eds.), *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics March 21-24, 2007, San Juan, Puerto Rico*, volume 2 of JMLR: W\&CP, 460–467.

See Also

do.mvu

Examples

```
## generate ribbon-shaped data
## in order to pass CRAN pretest, n is set to be small.
set.seed(100)
X = aux.gensamples(dname="ribbon",n=25)

## Compare MVU and MVE
#  Note that MVE actually requires much larger number of iterations
#  Here, due to CRAN limit, it was set as 7.
outMVU5  <- do.mvu(X, ndim=2, type=c("knn",5), projtype="kpca")
outMVE5  <- do.mve(X, ndim=2, knn=5, maxiter=7)

## Visualize two comparisons
opar <- par(no.readonly=TRUE)
```

```
par(mfrow=c(1,2))
plot(outMVU5$Y,  main="MVU (k=5)")
plot(outMVE5$Y,  main="MVE (k=5)")
par(opar)
```

`do.mvp` *Maximum Variance Projection*

Description

Maximum Variance Projection (MVP) is a supervised method based on linear discriminant analysis (LDA). In addition to classical LDA, it further aims at preserving local information by capturing the local geometry of the manifold via the following proximity coding,

$$S_{ij} = 1 \quad \text{if} \quad C_i \neq C_j \quad \text{and} = 0 \quad \text{otherwise}$$

, where C_i is the label of an i-th data point.

Usage

```
do.mvp(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang T (2007). "Maximum variance projections for face recognition." *Optical Engineering*, **46**(6), 067206.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## perform MVP with different preprocessings
out1 = do.mvp(X, label)
out2 = do.mvp(X, label, preprocess="decorrelate")
out3 = do.mvp(X, label, preprocess="whiten")

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="centering")
plot(out2$Y, col=label, pch=19, main="decorrelating")
plot(out3$Y, col=label, pch=19, main="whitening")
par(opar)
```

---

do.mvu *Maximum Variance Unfolding / Semidefinite Embedding*

Description

The method of Maximum Variance Unfolding(MVU), also known as Semidefinite Embedding(SDE) is, as its names suggest, to exploit semidefinite programming in performing nonlinear dimensionality reduction by *unfolding* neighborhood graph constructed in the original high-dimensional space. Its unfolding generates a gram matrix K in that we can choose from either directly finding embeddings (`"spectral"`) or use again Kernel PCA technique (`"kpca"`) to find low-dimensional representations. Note that since `do.mvu` depends on Rcsdp, we cannot guarantee its computational efficiency when given a large dataset.

Usage

```
do.mvu(
  X,
```

```
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  projtype = c("spectral", "kpca")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
projtype	type of method for projection; either "spectral" or "kpca" used.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Weinberger KQ, Saul LK (2006). "Unsupervised Learning of Image Manifolds by Semidefinite Programming." *International Journal of Computer Vision*, **70**(1), 77–90.

Examples

```
## generate ribbon-shaped data with the small number of data
set.seed(100)
X = aux.gensamples(dname="ribbon", n=25)

## try different connectivity levels
output1 <- do.mvu(X, type=c("proportion", 0.10))
output2 <- do.mvu(X, type=c("proportion", 0.25))
output3 <- do.mvu(X, type=c("proportion", 0.50))

## visualize three different projections
opar <- par(no.readonly=TRUE)
```

```
par(mfrow=c(1,3))
plot(output1$Y, main="10% connected")
plot(output2$Y, main="25% connected")
plot(output3$Y, main="50% connected")
par(opar)
```

---

do.nnp *Nearest Neighbor Projection*

Description

Nearest Neighbor Projection is an iterative method for visualizing high-dimensional dataset in that a data is sequentially located in the low-dimensional space by maintaining the triangular distance spread of target data with its two nearest neighbors in the high-dimensional space. We extended the original method to be applied for arbitrarily low-dimensional space. Due the generalization, we opted for a global optimization method of *Differential Evolution* (DEoptim) within in that it may add computational burden to certain degrees.

Usage

```
do.nnp(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Tejada E, Minghim R, Nonato LG (2003). "On Improved Projection Techniques to Support Visual Exploration of Multidimensional Data Sets." *Information Visualization*, **2**(4), 218–231.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## let's compare with other methods
out1 <- do.nnp(X, ndim=2)      # NNP
out2 <- do.pca(X, ndim=2)      # PCA
out3 <- do.lamp(X, ndim=2)     # LAMP

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="NNP")
plot(out2$Y, pch=19, col=label, main="PCA")
plot(out3$Y, pch=19, col=label, main="LAMP")
par(opar)
```

---

do.nolpp *Nonnegative Orthogonal Locality Preserving Projection*

Description

Nonnegative Orthogonal Locality Preserving Projection (NOLPP) is a variant of OLPP where projection vectors - or, basis for learned subspace - contain no negative values.

Usage

```
do.nolpp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  t = 1,
  maxiter = 1000,
  reltol = 1e-05
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
t	kernel bandwidth in $(0, \infty)$.
maxiter	number of maximum iteraions allowed.
reltol	stopping criterion for incremental relative error.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zafeiriou S, Laskaris N (2010). "Nonnegative Embeddings and Projections for Dimensionality Reduction and Information Visualization." In *2010 20th International Conference on Pattern Recognition*, 726–729.

See Also

do.olpp

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## use different kernel bandwidths with 20% connectivity
out1 = do.nolpp(X, type=c("proportion",0.5), t=0.01)
out2 = do.nolpp(X, type=c("proportion",0.5), t=0.1)
```

```
out3 = do.nolpp(X, type=c("proportion",0.5), t=1)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="NOLPP::t=0.01")
plot(out2$Y, col=label, main="NOLPP::t=0.1")
plot(out3$Y, col=label, main="NOLPP::t=1")
par(opar)

## End(Not run)
```

---

do.nonpp *Nonnegative Orthogonal Neighborhood Preserving Projections*

Description

Nonnegative Orthogonal Neighborhood Preserving Projections (NONPP) is a variant of ONPP where projection vectors - or, basis for learned subspace - contain no negative values.

Usage

```
do.nonpp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("null", "center", "decorrelate", "whiten"),
  maxiter = 1000,
  reltol = 1e-05
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "center" and other options of "decorrelate" and "whiten" are supported. See also aux.preprocess for more details.
maxiter	number of maximum iteraions allowed.
reltol	stopping criterion for incremental relative error.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zafeiriou S, Laskaris N (2010). "Nonnegative Embeddings and Projections for Dimensionality Reduction and Information Visualization." In *2010 20th International Conference on Pattern Recognition*, 726–729.

See Also

do.onpp

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## use different levels of connectivity
out1 = do.nonpp(X, type=c("proportion",0.1))
out2 = do.nonpp(X, type=c("proportion",0.2))
out3 = do.nonpp(X, type=c("proportion",0.5))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="NONPP::10% connected")
plot(out2$Y, col=label, main="NONPP::20% connected")
plot(out3$Y, col=label, main="NONPP::50% connected")
par(opar)

## End(Not run)
```

do.npca *Nonnegative Principal Component Analysis*

Description

Nonnegative Principal Component Analysis (NPCA) is a variant of PCA where projection vectors - or, basis for learned subspace - contain no negative values.

Usage

```
do.npca(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  maxiter = 1000,
  reltol = 1e-05
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
maxiter	number of maximum iteraions allowed.
reltol	stopping criterion for incremental relative error.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zafeiriou S, Laskaris N (2010). "Nonnegative Embeddings and Projections for Dimensionality Reduction and Information Visualization." In *2010 20th International Conference on Pattern Recognition*, 726–729.

See Also

do.pca

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4]) + 50
label = as.factor(iris[subid,5])

## use different preprocessing
out1 = do.npca(X, preprocess="center")
out2 = do.npca(X, preprocess="cscale")
out3 = do.npca(X, preprocess="whiten")

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="NPCA:: center")
plot(out2$Y, pch=19, col=label, main="NPCA:: cscale")
plot(out3$Y, pch=19, col=label, main="NPCA:: whiten")
par(opar)

## End(Not run)
```

---

do.npe *Neighborhood Preserving Embedding*

Description

do.npe performs a linear dimensionality reduction using Neighborhood Preserving Embedding (NPE) proposed by He et al (2005). It can be regarded as a linear approximation to Locally Linear Embedding (LLE). Like LLE, it is possible for the weight matrix being rank deficient. If regtype is set to TRUE with a proper value of regparam, it will perform Tikhonov regularization as designated. When regularization is needed with regtype parameter to be FALSE, it will automatically find a suitable regularization parameter and put penalty for stable computation. See also do.lle for more details.

Usage

```
do.npe(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = "union",
```

```
  weight = TRUE,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  regtype = FALSE,
  regparam = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
weight	TRUE to perform NPE on weighted graph, or FALSE otherwise.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
regtype	FALSE for not applying automatic Tikhonov Regularization, or TRUE otherwise.
regparam	a positive real number for Regularization. Default value is 1.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

eigval a vector of eigenvalues corresponding to basis expansion in an ascending order.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

He X, Cai D, Yan S, Zhang H (2005). "Neighborhood Preserving Embedding." In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, 1208–1213.

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
```

```
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## use different settings for connectivity
output1 = do.npe(X, ndim=2, type=c("proportion",0.10))
output2 = do.npe(X, ndim=2, type=c("proportion",0.25))
output3 = do.npe(X, ndim=2, type=c("proportion",0.50))

## visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, pch=19, col=label, main="NPE::10% connected")
plot(output2$Y, pch=19, col=label, main="NPE::25% connected")
plot(output3$Y, pch=19, col=label, main="NPE::50% connected")
par(opar)

## End(Not run)
```

`do.nrsr` *Non-convex Regularized Self-Representation*

Description

In the standard, convex RSR problem (`do.rsr`), row-sparsity for self-representation is acquired using matrix $\ell_{2,1}$ norm, i.e, $\|W\|_{2,1} = \sum \|W_{i:}\|_2$. Its non-convex extension aims at achieving higher-level of sparsity using arbitrarily chosen $\|W\|_{2,l}$ norm for $l \in (0, 1)$ and this exploits Iteratively Reweighted Least Squares (IRLS) algorithm for computation.

Usage

```
do.nrsr(
  X,
  ndim = 2,
  expl = 0.5,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  lbd = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`expl`	an exponent in $\ell_{2,l}$ norm for sparsity. Must be in $(0, 1)$, or $l = 1$ reduces to RSR problem.

preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
lbd	nonnegative number to control the degree of self-representation by imposing row-sparsity.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhu P, Zhu W, Wang W, Zuo W, Hu Q (2017). "Non-convex regularized self-representation for unsupervised feature selection." *Image and Vision Computing*, **60**, 22–29.

See Also

do.rsr

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

#### try different exponents for regularization
out1 = do.nrsr(X, expl=0.01)
out2 = do.nrsr(X, expl=0.1)
out3 = do.nrsr(X, expl=0.5)

#### visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="NRSR::expl=0.01")
plot(out2$Y, pch=19, col=label, main="NRSR::expl=0.1")
plot(out3$Y, pch=19, col=label, main="NRSR::expl=0.5")
par(opar)
```

do.odp *Orthogonal Discriminant Projection*

Description

Orthogonal Discriminant Projection (ODP) is a linear dimension reduction method with label information, i.e., *supervised*. The method maximizes weighted difference between local and non-local scatter while local information is also preserved by constructing a neighborhood graph.

Usage

```
do.odp(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  alpha = 0.5,
  beta = 10
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
alpha	balancing parameter of non-local and local scatter in $[0, 1]$.
beta	scaling control parameter for distant pairs of data in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

References

Li B, Wang C, Huang D (2009). "Supervised feature extraction based on orthogonal discriminant projection." *Neurocomputing*, **73**(1-3), 191–196.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different beta (scaling control) parameter
out1 = do.odp(X, label, beta=1)
out2 = do.odp(X, label, beta=10)
out3 = do.odp(X, label, beta=100)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="ODP::beta=1")
plot(out2$Y, col=label, pch=19, main="ODP::beta=10")
plot(out3$Y, col=label, pch=19, main="ODP::beta=100")
par(opar)
```

`do.olda` *Orthogonal Linear Discriminant Analysis*

Description

Orthogonal LDA (OLDA) is an extension of classical LDA where the discriminant vectors are orthogonal to each other.

Usage

```
do.olda(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.

label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Ye J (2005). "Characterization of a Family of Algorithms for Generalized Discriminant Analysis on Undersampled Problems." *J. Mach. Learn. Res.*, **6**, 483–502. ISSN 1532-4435.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare with LDA
out1 = do.lda(X, label)
out2 = do.olda(X, label)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, pch=19, col=label, main="LDA")
plot(out2$Y, pch=19, col=label, main="Orthogonal LDA")
par(opar)
```

do.olpp *Orthogonal Locality Preserving Projection*

Description

Orthogonal Locality Preserving Projection (OLPP) is a variant of do.lpp, which extracts orthogonal basis functions to reconstruct the data in a more intuitive fashion. It adopts PCA as preprocessing step and uses only one eigenvector at each iteration in that it might incur warning messages for solving near-singular system of linear equations.

Usage

```
do.olpp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric"),
  weight = TRUE,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  t = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
symmetric	one of "intersect", "union" or "asymmetric" is supported. Default is "union". See also aux.graphnbd for more details.
weight	TRUE to perform LPP on weighted graph, or FALSE otherwise.
preprocess	an additional option for preprocessing the data. See aux.preprocess for details.
t	bandwidth for heat kernel in $(0, \infty)$

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Cai D, He X, Han J, Zhang H (2006). "Orthogonal Laplacianfaces for Face Recognition." *IEEE Transactions on Image Processing*, **15**(11), 3608–3614.

See Also

do.lpp

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

##  connecting 10% and 25% of data for graph construction each.
output1 <- do.olpp(X,ndim=2,type=c("proportion",0.10))
output2 <- do.olpp(X,ndim=2,type=c("proportion",0.25))

## Visualize
#  In theory, it should show two separated groups of data
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(output1$Y, col=label, pch=19, main="OLPP::10% connected")
plot(output2$Y, col=label, pch=19, main="OLPP::25% connected")
par(opar)

## End(Not run)
```

---

do.onpp *Orthogonal Neighborhood Preserving Projections*

Description

Orthogonal Neighborhood Preserving Projection (ONPP) is an unsupervised linear dimension reduction method. It constructs a weighted data graph from LLE method. Also, it develops LPP method by preserving the structure of local neighborhoods.

Usage

```
do.onpp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Kokiopoulou E, Saad Y (2007). "Orthogonal Neighborhood Preserving Projections: A Projection-Based Dimensionality Reduction Technique." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(12), 2143–2156.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different numbers for neighborhood size
out1 = do.onpp(X, type=c("proportion",0.10))
```

```
out2 = do.onpp(X, type=c("proportion",0.25))
out3 = do.onpp(X, type=c("proportion",0.50))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="ONPP::10% connectivity")
plot(out2$Y, pch=19, col=label, main="ONPP::25% connectivity")
plot(out3$Y, pch=19, col=label, main="ONPP::50% connectivity")
par(opar)
```

`do.opls` *Orthogonal Partial Least Squares*

Description

Also known as multilinear regression or semipenalized CCA, Orthogonal Partial Least Squares (OPLS) was first used to perform multilinear ordinary least squares. In its usage, unlike PLS or CCA, OPLS does not rely on projected variance of response -or, `data2`. Instead, it exploits projected variance of input - covariance of `data1` and relates it under cross-covariance setting. Therefore, OPLS only returns projection information of `data1`, just like any other unsupervised methods in our package.

Usage

```
do.opls(data1, data2, ndim = 2)
```

Arguments

`data1`	an $(n \times N)$ data matrix whose rows are observations.
`data2`	an $(n \times M)$ data matrix whose rows are observations.
`ndim`	an integer-valued target dimension.

Value

a named list containing

Y an $(n \times ndim)$ matrix of projected observations from `data1`.

projection an $(N \times ndim)$ whose columns are loadings for `data1`.

trfinfo a list containing information for out-of-sample prediction for `data1`.

eigvals a vector of eigenvalues for iterative decomposition.

Author(s)

Kisung You

References

Barker M, Rayens W (2003). "Partial least squares for discrimination." *Journal of Chemometrics*, **17**(3), 166–173.

See Also

do.pls

Examples

```
## generate 2 normal data matrices
mat1 = matrix(rnorm(100*12),nrow=100)+10 # 12-dim normal
mat2 = matrix(rnorm(100*6), nrow=100)-10 # 6-dim normal

## compare OPLS and PLS
res_opls = do.opls(mat1, mat2, ndim=2)
res_pls  = do.pls(mat1, mat2, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(res_opls$Y, cex=0.5, main="OPLS result")
plot(res_pls$Y1, cex=0.5, main="PLS result")
par(opar)
```

---

do.pca *Principal Component Analysis*

Description

do.pca performs a classical principal component analysis (PCA) using RcppArmadillo package for faster and efficient computation.

Usage

```
do.pca(
  X,
  ndim = 2,
  cor = FALSE,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.

`cor`	mode of eigendecomposition. `FALSE` for decomposing covariance matrix, and `TRUE` for correlation matrix.
`preprocess`	an option for preprocessing the data. This supports three methods, where default is `"center"`. See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

vars a vector containing variances of projected data onto principal components.

projection a $(p \times ndim)$ whose columns are principal components.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Pearson K (1901). "LIII. On lines and planes of closest fit to systems of points in space." *Philosophical Magazine Series 6*, **2**(11), 559–572.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different preprocessing procedure
out1 <- do.pca(X, ndim=2, preprocess="center")
out2 <- do.pca(X, ndim=2, preprocess="decorrelate")
out3 <- do.pca(X, ndim=2, preprocess="whiten")

## embeddings for each procedure
Y1 <- out1$Y; Y2 <- out2$Y; Y3 <- out3$Y

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, col=lab, pch=19, main="PCA::'center'")
plot(Y2, col=lab, pch=19, main="PCA::'decorrelate'")
plot(Y3, col=lab, pch=19, main="PCA::'whiten'")
par(opar)
```

do.pflpp *Parameter-Free Locality Preserving Projection*

Description

Conventional LPP is known to suffer from sensitivity upon choice of parameters, especially in building neighborhood information. Parameter-Free LPP (PFLPP) takes an alternative step to use normalized Pearson correlation, taking an average of such similarity as a threshold to decide which points are neighbors of a given datum.

Usage

```
do.pflpp(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Dornaika F, Assoum A (2013). "Enhanced and parameterless Locality Preserving Projections for face recognition." *Neurocomputing*, **99**, 448–457.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare with PCA
out1 = do.pca(X, ndim=2)
out2 = do.pflpp(X, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, pch=19, col=label, main="PCA")
plot(out2$Y, pch=19, col=label, main="Parameter-Free LPP")
par(opar)
```

`do.plp` *Piecewise Laplacian-based Projection (PLP)*

Description

`do.plp` is an implementation of Piecewise Laplacian-based Projection (PLP) that adopts two-stage reduction scheme with local approximation.

Usage

```
do.plp(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  type = c("proportion", 0.2)
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.

Details

First step is to select $\sqrt{n}$ number of control points using k-means algorithm. After selecting control points that play similar roles as representatives of the entire data points, it performs classical multidimensional scaling.

For the rest of the data other than control points, Laplacian Eigenmaps (do.lapeig) is then applied to high-dimensional data points lying in neighborhoods of each control point. Embedded low-dimensional local manifold is then aligned to match their coordinates as of their counterparts from classical MDS.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Notes

Random Control Points : The performance of embedding using PLP heavily relies on selection of control points, which is contingent on the performance of k-means clustering.

User Interruption : PLP is actually an interactive algorithm that a user should be able to intervene intermittently. Such functionality is, however, sacrificed in this version.

Author(s)

Kisung You

References

Paulovich F, Eler D, Poco J, Botha C, Minghim R, Nonato L (2011). "Piece wise Laplacian-based Projection for Interactive Data Exploration and Organization." *Computer Graphics Forum*, **30**(3), 1091–1100.

Examples

```
## Not run:
## use iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.integer(iris$Species)

## try with 3 levels of connectivity
out1 = do.plp(X, type=c("proportion", 0.1))
out2 = do.plp(X, type=c("proportion", 0.2))
out3 = do.plp(X, type=c("proportion", 0.5))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="PLP::10% connected")
```

```
plot(out2$Y, col=label, main="PLP::20% connected")
plot(out3$Y, col=label, main="PLP::50% connected")
par(opar)

## End(Not run)
```

`do.pls` *Partial Least Squares*

Description

Given two data sets, Partial Least Squares (PLS) aims at maximizing cross-covariance of latent variables for each data matrix, therefore it can be considered as supervised methods. As we have two input matrices, `do.pls` generates two sets of outputs. Though it is widely used for regression problem, we used it in dimension reduction setting. For algorithm aspects, we used recursive gram-schmidt orthogonalization in conjunction with extracting projection vectors under eigen-decomposition formulation, as the problem dimension matters only up to original dimensionality. For more details, see Wikipedia entry on PLS.

Usage

```
do.pls(data1, data2, ndim = 2)
```

Arguments

`data1`	an $(n \times N)$ data matrix whose rows are observations
`data2`	an $(n \times M)$ data matrix whose rows are observations
`ndim`	an integer-valued target dimension.

Value

a named list containing

Y1 an $(n \times ndim)$ matrix of projected observations from `data1`.

Y2 an $(n \times ndim)$ matrix of projected observations from `data2`.

projection1 an $(N \times ndim)$ whose columns are loadings for `data1`.

projection2 an $(M \times ndim)$ whose columns are loadings for `data2`.

trfinfo1 a list containing information for out-of-sample prediction for `data1`.

trfinfo2 a list containing information for out-of-sample prediction for `data2`.

eigvals a vector of eigenvalues for iterative decomposition.

Author(s)

Kisung You

References

Wold H (1975). "Path Models with Latent Variables: The NIPALS Approach." In *Quantitative Sociology*, 307–357. Elsevier. ISBN 978-0-12-103950-9, doi: 10.1016/B9780121039509.500174.

Rosipal R, Krämer N (2006). "Overview and Recent Advances in Partial Least Squares." In Saunders C, Grobelnik M, Gunn S, Shawe-Taylor J (eds.), *Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop, SLSFS 2005, Bohinj, Slovenia, February 23-25, 2005, Revised Selected Papers*, 34–51. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-34138-3, doi: 10.1007/11752790_2.

See Also

do.cca

Examples

```
## generate 2 normal data matrices
mat1 = matrix(rnorm(100*12),nrow=100)+10 # 12-dim normal
mat2 = matrix(rnorm(100*6), nrow=100)-10 # 6-dim normal

## project onto 2 dimensional space for each data
output = do.pls(mat1, mat2, ndim=2)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(output$Y1, main="proj(mat1)")
plot(output$Y2, main="proj(mat2)")
par(opar)
```

---

do.ppca *Probabilistic Principal Component Analysis*

Description

Probabilistic PCA (PPCA) is a probabilistic framework to explain the well-known PCA model. Using the conjugacy of normal model, we compute MLE for values explicitly derived in the paper. Note that unlike PCA where loadings are directly used for projection, PPCA uses WM^{-1} as projection matrix, as it is relevant to the error model. Also, for high-dimensional problem, it is possible that MLE can have negative values if sample covariance given the data is rank-deficient.

Usage

```
do.ppca(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
preprocess	an option for preprocessing the data. This supports three methods. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are principal components.

mle.sigma2 MLE for σ^2.

mle.W MLE of a $(p \times ndim)$ mapping from latent to observation in column major.

Author(s)

Kisung You

References

Tipping ME, Bishop CM (1999). "Probabilistic Principal Component Analysis." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**(3), 611–622.

See Also

do.pca

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## Compare PCA and PPCA
PCA  <- do.pca(X, ndim=2, preprocess="center")
PPCA <- do.ppca(X, ndim=2, preprocess="center")

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(PCA$Y,  pch=19, col=label, main="PCA")
plot(PPCA$Y, pch=19, col=label, main="PPCA")
par(opar)
```

do.ree *Robust Euclidean Embedding*

Description

Robust Euclidean Embedding (REE) is an embedding procedure exploiting robustness of ℓ_1 cost function. In our implementation, we adopted a generalized version with weight matrix to be applied as well. Its original paper introduced a subgradient algorithm to overcome memory-intensive nature of original semidefinite programming formulation.

Usage

```
do.ree(
  X,
  ndim = 2,
  W = NA,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  initc = 1,
 dmethod = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  maxiter = 100,
  abstol = 0.001
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
W	an $(n \times n)$ weight matrix. Default is uniform weight of 1s.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
initc	initial c value for subgradient iterating stepsize, $c/\sqrt{i}$.
dmethod	a type of distance measure. See dist for more details.
maxiter	maximum number of iterations for subgradient descent method.
abstol	stopping criterion for subgradient descent method.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

niter the number of iterations taken til convergence.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Cayton L, Dasgupta S (2006). "Robust Euclidean Embedding." In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, 169–176.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different distance method
output1 <- do.ree(X, maxiter=50, dmethod="euclidean")
output2 <- do.ree(X, maxiter=50, dmethod="maximum")
output3 <- do.ree(X, maxiter=50, dmethod="canberra")

## visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, col=label, pch=19, main="dmethod-euclidean")
plot(output2$Y, col=label, pch=19, main="dmethod-maximum")
plot(output3$Y, col=label, pch=19, main="dmethod-canberra")
par(opar)
```

---

do.rlda *Regularized Linear Discriminant Analysis*

Description

In small sample case, Linear Discriminant Analysis (LDA) may suffer from rank deficiency issue. Applied mathematics has used Tikhonov regularization - also known as ℓ_2 regularization/shrinkage - to adjust linear operator. Regularized Linear Discriminant Analysis (RLDA) adopts such idea to stabilize eigendecomposition in LDA formulation.

Usage

```
do.rlda(X, label, ndim = 2, alpha = 1)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
alpha	Tikhonow regularization parameter.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Friedman JH (1989). "Regularized Discriminant Analysis." *Journal of the American Statistical Association*, **84**(405), 165.

Examples

```
## Not run:
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different regularization parameters
out1 <- do.rlda(X, label, alpha=0.001)
out2 <- do.rlda(X, label, alpha=0.01)
out3 <- do.rlda(X, label, alpha=100)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="RLDA::alpha=0.1")
plot(out2$Y, pch=19, col=label, main="RLDA::alpha=1")
plot(out3$Y, pch=19, col=label, main="RLDA::alpha=10")
par(opar)

## End(Not run)
```

`do.rndproj` *Random Projection*

Description

`do.rndproj` is a linear dimensionality reduction method based on random projection technique, featured by the celebrated Johnson–Lindenstrauss lemma.

Usage

```
do.rndproj(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  type = c("gaussian", "achlioptas", "sparse"),
  s = max(sqrt(ncol(X)), 3)
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`type`	a type of random projection, one of "gaussian","achlioptas" or "sparse".
`s`	a tuning parameter for determining values in projection matrix. While default is to use $max(log\sqrt{p}, 3)$, it is required for $s \geq 3$.

Details

The Johnson-Lindenstrauss(JL) lemma states that given $0 < \epsilon < 1$, for a set X of m points in R^N and a number $n > 8log(m)/\epsilon^2$, there is a linear map $f : R^N$ to R^n such that

$$(1-\epsilon)|u-v|^2 \leq |f(u)-f(v)|^2 \leq (1+\epsilon)|u-v|^2$$

for all u, v in X.

Three types of random projections are supported for an `(p-by-ndim)` projection matrix R.

1. Conventional approach is to use normalized Gaussian random vectors sampled from unit sphere S^{p-1}.
2. Achlioptas suggested to employ a sparse approach using samples from $\sqrt{3}(1, 0, -1)$ with probability $(1/6, 4/6, 1/6)$.
3. Li et al proposed to sample from $\sqrt{s}(1, 0, -1)$ with probability $(1/2s, 1 - 1/s, 1/2s)$ for $s \geq 3$ to incorporate sparsity while attaining speedup with little loss in accuracy. While the original suggsetion from the authors is to use $\sqrt{p}$ or $p/log(p)$ for s, any user-supported $s \geq 3$ is allowed.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are basis for projection.

epsilon an estimated error ϵ in accordance with JL lemma.

trfinfo a list containing information for out-of-sample prediction.

References

Johnson WB, Lindenstrauss J (1984). "Extensions of Lipschitz mappings into a Hilbert space." In Beals R, Beck A, Bellow A, Hajian A (eds.), *Contemporary Mathematics*, volume 26, 189–206. American Mathematical Society, Providence, Rhode Island. ISBN 978-0-8218-5030-5 978-0-8218-7611-4, doi: 10.1090/conm/026/737400.

Achlioptas D (2003). "Database-friendly random projections: Johnson-Lindenstrauss with binary coins." *Journal of Computer and System Sciences*, **66**(4), 671–687.

Li P, Hastie TJ, Church KW (2006). "Very Sparse Random Projections." In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, 287–296.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## 1. Gaussian projection
output1 <- do.rndproj(X,ndim=2)

## 2. Achlioptas projection
output2 <- do.rndproj(X,ndim=2,type="achlioptas")

## 3. Sparse projection
output3 <- do.rndproj(X,type="sparse")

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(output1$Y, pch=19, col=label, main="RNDPROJ::Gaussian")
plot(output2$Y, pch=19, col=label, main="RNDPROJ::Arclioptas")
plot(output3$Y, pch=19, col=label, main="RNDPROJ::Sparse")
par(opar)
```

`do.rpca` *Robust Principal Component Analysis*

Description

Robust PCA (RPCA) is not like other methods in this package as finding explicit low-dimensional embedding with reduced number of columns. Rather, it is more of a decomposition method of data matrix X, possibly noisy, into low-rank and sparse matrices by solving the following,

$$\text{minimize} \quad \|L\|_* + \lambda\|S\|_1 \quad s.t. L + S = X$$

where L is a low-rank matrix, S is a sparse matrix and $\|\cdot\|_*$ denotes nuclear norm, i.e., sum of singular values. Therefore, it should be considered as *preprocessing* procedure of denoising. Note that after RPCA is applied, L should be used as kind of a new data matrix for any manifold learning scheme to be applied.

Usage

```
do.rpca(
  X,
  mu = 1,
  lambda = sqrt(1/(max(dim(X)))),
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or whose rows are observations and columns represent independent variables.
`mu`	an augmented Lagrangian parameter
`lambda`	parameter for the sparsity term $\|S\|_1$. Default value is given accordingly to the referred paper.
`preprocess`	an additional option for preprocessing the data. Default is `"null"`. See also `aux.preprocess` for more details.

Value

a named list containing

L an $(n \times p)$ low-rank matrix.

S an $(n \times p)$ sparse matrix.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Candès EJ, Li X, Ma Y, Wright J (2011). "Robust principal component analysis?" *Journal of the ACM*, **58**(3), 1–37.

Examples

```
## Load Iris data and put some noise
data(iris)
set.seed(100)
subid = sample(1:150,50)
noise = 0.2
X = as.matrix(iris[subid,1:4])
X = X + matrix(noise*rnorm(length(X)), nrow=nrow(X))
lab = as.factor(iris[subid,5])

## try different regularization parameters
rpca1 = do.rpca(X, lambda=0.1)
rpca2 = do.rpca(X, lambda=1)
rpca3 = do.rpca(X, lambda=10)

## apply identical PCA methods
Y1 = do.pca(rpca1$L, ndim=2)$Y
Y2 = do.pca(rpca2$L, ndim=2)$Y
Y3 = do.pca(rpca3$L, ndim=2)$Y

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, pch=19, col=lab, main="RPCA+PCA::lambda=0.1")
plot(Y2, pch=19, col=lab, main="RPCA+PCA::lambda=1")
plot(Y3, pch=19, col=lab, main="RPCA+PCA::lambda=10")
par(opar)
```

---

do.rpcag *Robust Principal Component Analysis via Geometric Median*

Description

This function robustifies the traditional PCA via an idea of geometric median. To describe, the given data is first split into k subsets for each sample covariance is attained. According to the paper, the median covariance is computed under Frobenius norm and projection is extracted from the largest eigenvectors.

Usage

```
do.rpcag(
  X,
  ndim = 2,
```

```
  k = 5,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
k	the number of subsets for X to be divided.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Minsker S (2015). "Geometric median and robust estimation in Banach spaces." *Bernoulli*, **21**(4), 2308–2335.

Examples

```
## use iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.integer(iris$Species)

## try different numbers for subsets
out1 = do.rpcag(X, ndim=2, k=2)
out2 = do.rpcag(X, ndim=2, k=5)
out3 = do.rpcag(X, ndim=2, k=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="RPCAG::k=2")
plot(out2$Y, col=label, main="RPCAG::k=5")
plot(out3$Y, col=label, main="RPCAG::k=10")
par(opar)
```

`do.rsir` *Regularized Sliced Inverse Regression*

Description

One of possible drawbacks in SIR method is that for high-dimensional data, it might suffer from rank deficiency of scatter/covariance matrix. Instead of naive matrix inversion, several have proposed regularization schemes that reflect several ideas from various incumbent methods.

Usage

```
do.rsir(
  X,
  response,
  ndim = 2,
  h = max(2, round(nrow(X)/5)),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  regmethod = c("Ridge", "Tikhonov", "PCA", "PCARidge", "PCATikhonov"),
  tau = 1,
  numpc = ndim
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`response`	a length-n vector of response variable.
`ndim`	an integer-valued target dimension.
`h`	the number of slices to divide the range of response vector.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`regmethod`	type of regularization scheme to be used.
`tau`	regularization parameter for adjusting rank-deficient scatter matrix.
`numpc`	number of principal components to be used in intermediate dimension reduction scheme.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Chiaromonte F, Martinelli J (2002). "Dimension reduction strategies for analyzing global gene expression data with a response." *Mathematical Biosciences*, **176**(1), 123 – 144. ISSN 0025-5564.

Zhong W, Zeng P, Ma P, Liu JS, Zhu Y (2005). "RSIR: regularized sliced inverse regression for motif discovery." *Bioinformatics*, **21**(22), 4169–4175.

Bernard-Michel C, Gardes L, Girard S (2009). "Gaussian Regularized Sliced Inverse Regression." *Statistics and Computing*, **19**(1), 85–98.

Bernard-Michel C, Douté S, Fauvel M, Gardes L, Girard S (2009). "Retrieval of Mars surface physical properties from OMEGA hyperspectral images using regularized sliced inverse regression." *Journal of Geophysical Research*, **114**(E6).

See Also

do.sir

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(100)
n     = 50
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try with different regularization methods
## use default number of slices
out1 = do.rsir(X, y, regmethod="Ridge")
out2 = do.rsir(X, y, regmethod="Tikhonov")
outsir = do.sir(X, y)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y,   main="RSIR::Ridge")
plot(out2$Y,   main="RSIR::Tikhonov")
plot(outsir$Y, main="standard SIR")
par(opar)
```

`do.rsr` *Regularized Self-Representation*

Description

Given a data matrix X where observations are stacked in a row-wise manner, Regularized Self-Representation (RSR) aims at finding a solution to following optimization problem

$$\min \|X - XW\|_{2,1} + \lambda \|W\|_{2,1}$$

where $\|W\|_{2,1} = \sum_{i=1}^{m} \|W_{i:}\|_2$ is an $\ell_{2,1}$ norm that imposes row-wise sparsity constraint.

Usage

```
do.rsr(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  lbd = 1
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`lbd`	nonnegative number to control the degree of self-representation by imposing row-sparsity.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhu P, Zuo W, Zhang L, Hu Q, Shiu SC (2015). "Unsupervised feature selection by regularized self-representation." *Pattern Recognition*, **48**(2), 438–446.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

#### try different lbd combinations
out1 = do.rsr(X, lbd=0.1)
out2 = do.rsr(X, lbd=1)
out3 = do.rsr(X, lbd=10)

#### visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="RSR::lbd=0.1")
plot(out2$Y, pch=19, col=label, main="RSR::lbd=1")
plot(out3$Y, pch=19, col=label, main="RSR::lbd=10")
par(opar)
```

---

do.sammc *Semi-Supervised Adaptive Maximum Margin Criterion*

Description

Semi-Supervised Adaptive Maximum Margin Criterion (SAMMC) is a semi-supervised variant of AMMC by making use of both labeled and unlabeled data.

Usage

```
do.sammc(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  a = 1,
  b = 1,
  lambda = 1,
  beta = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.
a	tuning parameter for between-class weight in $[0, \infty)$.
b	tuning parameter for within-class weight in $[0, \infty)$.
lambda	balance parameter for between-class and within-class scatter matrices in $(0, \infty)$.
beta	balance parameter for within-class scatter of the labeled data and consistency of the whole data in $(0, \infty)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Lu J, Tan Y (2011). "Adaptive maximum margin criterion for image classification." In *2011 IEEE International Conference on Multimedia and Expo*, 1–6.

See Also

do.mmc, do.ammc

Examples

```
## generate data of 3 types with clear difference
set.seed(100)
dt1  = aux.gensamples(n=33)-50
dt2  = aux.gensamples(n=33)
dt3  = aux.gensamples(n=33)+50

## merge the data and create a label correspondingly
```

```
X      = rbind(dt1,dt2,dt3)
label  = rep(1:3, each=33)

## copy a label and let 20% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.20)
label_missing = label
label_missing[sample(1:nlabel, nmissing)]=NA

## try different balancing
out1 = do.sammc(X, label_missing, beta=0.1)
out2 = do.sammc(X, label_missing, beta=1)
out3 = do.sammc(X, label_missing, beta=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="SAMMC::beta=0.1")
plot(out2$Y, pch=19, col=label, main="SAMMC::beta=1")
plot(out3$Y, pch=19, col=label, main="SAMMC::beta=10")
par(opar)
```

`do.sammon` *Sammon Mapping*

Description

`do.sammon` is an implementation for Sammon mapping, one of the earliest dimension reduction techniques that aims to find low-dimensional embedding that preserves pairwise distance structure in high-dimensional data space.

Usage

```
do.sammon(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  initialize = c("pca", "random")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.
`initialize`	`"random"` or `"pca"`; the former performs fast random projection (see also `do.rndproj`) and the latter performs standard PCA (see also `do.pca`).

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Sammon, J.W. (1969) *A Nonlinear Mapping for Data Structure Analysis*. IEEE Transactions on Computers, C-18 5:401-409.

Sammon J (1969). "A Nonlinear Mapping for Data Structure Analysis." *IEEE Transactions on Computers*, **C-18**(5), 401–409.

Examples

```
## load iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.factor(iris$Species)

## compare two initialization
out1 = do.sammon(X,ndim=2)                   # random projection
out2 = do.sammon(X,ndim=2,initialize="pca")  # pca as initialization

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, pch=19, col=label, main="out1:rndproj")
plot(out2$Y, pch=19, col=label, main="out2:pca")
par(opar)
```

---

do.save *Sliced Average Variance Estimation*

Description

Sliced Average Variance Estimation (SAVE) is a supervised linear dimension reduction method. It is based on sufficiency principle with respect to central subspace concept under the linerity and constant covariance conditions. For more details, see the reference paper.

Usage

```
do.save(
  X,
  response,
  ndim = 2,
  h = max(2, round(nrow(X)/5)),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
response	a length-n vector of response variable.
ndim	an integer-valued target dimension.
h	the number of slices to divide the range of response vector.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Dennis Cook R (2000). "Save: a method for dimension reduction and graphics in regression." *Communications in Statistics - Theory and Methods*, **29**(9-10), 2109–2121.

See Also

do.sir

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(100)
n = 50
theta = runif(n)
h     = runif(n)
```

```
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try with different numbers of slices
out1 = do.save(X, y, h=2)
out2 = do.save(X, y, h=5)
out3 = do.save(X, y, h=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="SAVE::2 slices")
plot(out2$Y, main="SAVE::5 slices")
plot(out3$Y, main="SAVE::10 slices")
par(opar)
```

`do.sda` *Semi-Supervised Discriminant Analysis*

Description

Semi-Supervised Discriminant Analysis (SDA) is a linear dimension reduction method when label is partially missing, i.e., semi-supervised. The labeled data points are used to maximize the separability between classes while the unlabeled ones to estimate the intrinsic structure of the data. Regularization in case of rank-deficient case is also supported via an ℓ_2 scheme via `beta`.

Usage

```
do.sda(X, label, ndim = 2, type = c("proportion", 0.1), alpha = 1, beta = 1)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.

alpha	balancing parameter between model complexity and empirical loss.
beta	Tikhonov regularization parameter.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Cai D, He X, Han J (2007). "Semi-supervised Discriminant Analysis." In *2007 IEEE 11th International Conference on Computer Vision*, 1–7.

Examples

```
## use iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.integer(iris$Species)

## copy a label and let 20% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.20)
label_missing = label
label_missing[sample(1:nlabel, nmissing)]=NA

## compare true case with missing-label case
out1 = do.sda(X, label)
out2 = do.sda(X, label_missing)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, col=label, main="true projection")
plot(out2$Y, col=label, main="20% missing labels")
par(opar)
```

`do.sdlpp` *Sample-Dependent Locality Preserving Projection*

Description

Many variants of Locality Preserving Projection are contingent on graph construction schemes in that they sometimes return a range of heterogeneous results when parameters are controlled to cover a wide range of values. This algorithm takes an approach called *sample-dependent construction* of graph connectivity in that it tries to discover intrinsic structures of data solely based on data.

Usage

```
do.sdlpp(
  X,
  ndim = 2,
  t = 1,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`ndim`	an integer-valued target dimension.
`t`	kernel bandwidth in $(0, \infty)$.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Yang B, Chen S (2010). "Sample-dependent graph construction with application to dimensionality reduction." *Neurocomputing*, **74**(1-3), 301–314.

See Also

`do.lpp`

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare with PCA
out1 <- do.pca(X,ndim=2)
out2 <- do.sdlpp(X, t=0.01)
out3 <- do.sdlpp(X, t=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="PCA")
plot(out2$Y, pch=19, col=label, main="SDLPP::t=1")
plot(out3$Y, pch=19, col=label, main="SDLPP::t=10")
par(opar)
```

`do.sir` *Sliced Inverse Regression*

Description

Sliced Inverse Regression (SIR) is a supervised linear dimension reduction technique. Unlike engineering-driven methods, SIR takes a concept of *central subspace*, where conditional independence after projection is guaranteed. It first divides the range of response variable. Projection vectors are extracted where projected data best explains response variable.

Usage

```
do.sir(
  X,
  response,
  ndim = 2,
  h = max(2, round(nrow(X)/5)),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`response`	a length-n vector of response variable.
`ndim`	an integer-valued target dimension.

h	the number of slices to divide the range of response vector.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Li K (1991). "Sliced Inverse Regression for Dimension Reduction." *Journal of the American Statistical Association*, **86**(414), 316.

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(100)
n = 50
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try with different numbers of slices
out1 = do.sir(X, y, h=2)
out2 = do.sir(X, y, h=5)
out3 = do.sir(X, y, h=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="SIR::2 slices")
plot(out2$Y, main="SIR::5 slices")
plot(out3$Y, main="SIR::10 slices")
par(opar)
```

`do.slpe` *Supervised Locality Pursuit Embedding*

Description

Supervised Locality Pursuit Embedding (SLPE) is a supervised extension of LPE that uses class labels of data points in order to enhance discriminating power in its mapping into a low dimensional space.

Usage

```
do.slpe(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zheng Z, Yang J (2006). "Supervised locality pursuit embedding for pattern classification." *Image and Vision Computing*, **24**(8), 819–826.

See Also

`do.lpe`

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare SLPE with SLPP
out1 <- do.slpp(X, label)
out2 <- do.slpe(X, label)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, pch=19, col=label, main="SLPP")
plot(out2$Y, pch=19, col=label, main="SLPE")
par(opar)
```

---

do.slpp *Supervised Locality Preserving Projection*

Description

As its names suggests, Supervised Locality Preserving Projection (SLPP) is a variant of LPP in that it replaces neighborhood network construction schematic with class information in that if two nodes belong to the same class, it assigns weight of 1, i.e., $S_{ij} = 1$ if x_i and x_j have same class labelings.

Usage

```
do.slpp(X, label, ndim = 2, preprocess = c("center", "decorrelate", "whiten"))
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center" and other options of "decorrelate" and "whiten" are supported. See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zheng Z, Yang F, Tan W, Jia J, Yang J (2007). "Gabor feature-based face recognition using supervised locality preserving projection." *Signal Processing*, **87**(10), 2473–2483.

See Also

do.lpp

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare SLPP with LPP
outLPP  <- do.lpp(X)
outSLPP <- do.slpp(X, label)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(outLPP$Y,  pch=19, col=label, main="LPP")
plot(outSLPP$Y, pch=19, col=label, main="SLPP")
par(opar)
```

---

do.sne *Stochastic Neighbor Embedding*

Description

Stochastic Neighbor Embedding (SNE) is a probabilistic approach to mimick distributional description in high-dimensional - possible, nonlinear - subspace on low-dimensional target space. do.sne fully adopts algorithm details in an original paper by Hinton and Roweis (2002).

Usage

```
do.sne(
  X,
  ndim = 2,
  perplexity = 30,
```

```
  eta = 0.05,
  maxiter = 2000,
  jitter = 0.3,
  jitterdecay = 0.99,
  momentum = 0.5,
  preprocess = c("null", "center", "scale", "scale", "decorrelate", "whiten"),
  pca = TRUE,
  pcascale = FALSE,
  symmetric = FALSE
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
perplexity	desired level of perplexity; ranging [5,50].
eta	learning parameter.
maxiter	maximum number of iterations.
jitter	level of white noise added at the beginning.
jitterdecay	decay parameter in $(0, 1)$. The closer to 0, the faster artificial noise decays.
momentum	level of acceleration in learning.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
pca	whether to use PCA as preliminary step; TRUE for using it, FALSE otherwise.
pcascale	a logical; FALSE for using Covariance, TRUE for using Correlation matrix. See also do.pca for more details.
symmetric	a logical; FALSE to solve it naively, and TRUE to adopt symmetrization scheme.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

vars a vector containing betas used in perplexity matching.

Author(s)

Kisung You

References

Hinton GE, Roweis ST (2003). "Stochastic Neighbor Embedding." In Becker S, Thrun S, Obermayer K (eds.), *Advances in Neural Information Processing Systems 15*, 857–864. MIT Press. http://papers.nips.cc/paper/2276-stochastic-neighbor-embedding.pdf.

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## try different perplexity values
out1 <- do.sne(X, perplexity=5)
out2 <- do.sne(X, perplexity=25)
out3 <- do.sne(X, perplexity=50)

## Visualize two comparisons
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="perplexity=5")
plot(out2$Y, pch=19, col=label, main="perplexity=25")
plot(out3$Y, pch=19, col=label, main="perplexity=50")
par(opar)
```

---

do.spc *Supervised Principal Component Analysis*

Description

Unlike original principal component analysis (do.pca), this algorithm implements a supervised version using response information for feature selection. For each feature/column, its normalized association with `response` variable is computed and the features with large magnitude beyond `threshold` are selected. From the selected submatrix, regular PCA is applied for dimension reduction.

Usage

```
do.spc(
  X,
  response,
  ndim = 2,
  preprocess = c("center", "whiten", "decorrelate"),
  threshold = 0.1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.

response	a length-n vector of response variable.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is center. See also aux.preprocess for more details.
threshold	a threshold value to cut off normalized association between covariates and response.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Bair E, Hastie T, Paul D, Tibshirani R (2006). "Prediction by Supervised Principal Components." *Journal of the American Statistical Association*, **101**(473), 119–137.

Examples

```
## generate swiss roll with auxiliary dimensions
## it follows reference example from LSIR paper.
set.seed(100)
n = 100
theta = runif(n)
h     = runif(n)
t     = (1+2*theta)*(3*pi/2)
X     = array(0,c(n,10))
X[,1] = t*cos(t)
X[,2] = 21*h
X[,3] = t*sin(t)
X[,4:10] = matrix(runif(7*n), nrow=n)

## corresponding response vector
y = sin(5*pi*theta)+(runif(n)*sqrt(0.1))

## try different threshold values
out1 = do.spc(X, y, threshold=2)
out2 = do.spc(X, y, threshold=5)
out3 = do.spc(X, y, threshold=10)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
```

```
plot(out1$Y, main="SPC::threshold=2")
plot(out2$Y, main="SPC::threshold=5")
plot(out3$Y, main="SPC::threshold=10")
par(opar)
```

`do.spca` *Sparse Principal Component Analysis*

Description

Sparse PCA (`do.spca`) is a variant of PCA in that each loading - or, principal component - should be sparse. Instead of using generic optimization package, we opt for formulating a problem as semidefinite relaxation and utilizing ADMM.

Usage

```
do.spca(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  mu = 1,
  rho = 1,
  abstol = 1e-04,
  reltol = 0.01,
  maxiter = 1000
)
```

Arguments

`X`	an $(n \times p)$ matrix whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is `"center"`. See also `aux.preprocess` for more details.
`mu`	an augmented Lagrangian parameter.
`rho`	a regularization parameter for sparsity.
`abstol`	absolute tolerance stopping criterion.
`reltol`	relative tolerance stopping criterion.
`maxiter`	maximum number of iterations.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

projection a $(p \times ndim)$ whose columns are principal components.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Zou H, Hastie T, Tibshirani R (2006). "Sparse Principal Component Analysis." *Journal of Computational and Graphical Statistics*, **15**(2), 265–286.

d'Aspremont A, El Ghaoui L, Jordan MI, Lanckriet GRG (2007). "A Direct Formulation for Sparse PCA Using Semidefinite Programming." *SIAM Review*, **49**(3), 434–448.

Ma S (2013). "Alternating Direction Method of Multipliers for Sparse Principal Component Analysis." *Journal of the Operations Research Society of China*, **1**(2), 253–274.

See Also

do.pca

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## try different regularization parameters for sparsity
out1 <- do.spca(X,ndim=2,rho=0.01)
out2 <- do.spca(X,ndim=2,rho=1)
out3 <- do.spca(X,ndim=2,rho=100)

## embeddings for each procedure
Y1 <- out1$Y; Y2 <- out2$Y; Y3 <- out3$Y

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, col=lab, pch=19, main="SPCA::rho=0.01")
plot(Y2, col=lab, pch=19, main="SPCA::rho=1")
plot(Y3, col=lab, pch=19, main="SPCA::rho=100")
par(opar)
```

---

do.spe *Stochastic Proximity Embedding*

Description

One of drawbacks for Multidimensional Scaling or Sammon mapping is that they have quadratic computational complexity with respect to the number of data. Stochastic Proximity Embedding (SPE) adopts stochastic update rule in that its computational speed is much improved. It performs `C` number of cycles, where for each cycle, it randomly selects two data points and updates their locations correspondingly `S` times. After each cycle, learning parameter λ is multiplied by `drate`, becoming smaller in magnitude.

Usage

```
do.spe(
  X,
  ndim = 2,
  proximity = function(x) {     dist(x, method = "euclidean") },
  C = 50,
  S = 50,
  lambda = 1,
  drate = 0.9
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`proximity`	a function for constructing proximity matrix from original data dimension.
`C`	the number of cycles to be run; after each cycle, learning parameter
`S`	the number of updates for each cycle.
`lambda`	initial learning parameter.
`drate`	multiplier for `lambda` at each cycle; should be a positive real number in $(0, 1)$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Agrafiotis DK (2003). "Stochastic proximity embedding." *Journal of Computational Chemistry*, **24**(10), 1215–1221.

Examples

```
## load iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.factor(iris$Species)

## compare with mds using 2 distance metrics
outM <- do.mds(X, ndim=2)
out1 <- do.spe(X, ndim=2)
out2 <- do.spe(X, ndim=2, proximity=function(x){dist(x, method="manhattan")})

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(outM$Y, col=label, main="MDS")
plot(out1$Y, col=label, main="SPE with L2 norm")
plot(out2$Y, col=label, main="SPE with L1 norm")
par(opar)
```

`do.specs` *Supervised Spectral Feature Selection*

Description

SPEC algorithm selects features from the data via spectral graph approach. Three types of ranking methods that appeared in the paper are available where the graph laplacian is built via class label information.

Usage

```
do.specs(
  X,
  label,
  ndim = 2,
  ranking = c("method1", "method2", "method3"),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of class labels.
`ndim`	an integer-valued target dimension.
`ranking`	types of feature scoring method. See the paper in the reference for more details.
`preprocess`	an additional option for preprocessing the data. Default is "null". See also `aux.preprocess` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

sscore a length-p vector of spectral feature scores.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhao Z, Liu H (2007). "Spectral feature selection for supervised and unsupervised learning." In *Proceedings of the 24th international conference on Machine learning - ICML '07*, 1151–1157.

See Also

do.specu

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid = sample(1:150, 50)
iris.dat = as.matrix(iris[subid,1:4])
iris.lab = as.factor(iris[subid,5])

## try different ranking methods
out1 = do.specs(iris.dat, iris.lab, ranking="method1")
out2 = do.specs(iris.dat, iris.lab, ranking="method2")
out3 = do.specs(iris.dat, iris.lab, ranking="method3")

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=iris.lab, main="SPECS::method1")
plot(out2$Y, pch=19, col=iris.lab, main="SPECS::method2")
plot(out3$Y, pch=19, col=iris.lab, main="SPECS::method3")
par(opar)
```

do.specu *Unsupervised Spectral Feature Selection*

Description

SPEC algorithm selects features from the data via spectral graph approach. Three types of ranking methods that appeared in the paper are available where the graph laplacian is built via RBF kernel.

Usage

```
do.specu(
  X,
  ndim = 2,
  sigma = 1,
  ranking = c("method1", "method2", "method3"),
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
sigma	bandwidth parameter for RBK kernel of type $S_{i,j} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$.
ranking	types of feature scoring method. See the paper in the reference for more details.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

sscore a length-p vector of spectral feature scores.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhao Z, Liu H (2007). "Spectral feature selection for supervised and unsupervised learning." In *Proceedings of the 24th international conference on Machine learning - ICML '07*, 1151–1157.

See Also

do.specs

Examples

```
## use iris data
## it is known that feature 3 and 4 are more important.
data(iris)
set.seed(100)
subid    = sample(1:150,50)
iris.dat = as.matrix(iris[subid,1:4])
iris.lab = as.factor(iris[subid,5])


## try different ranking methods
mysig = 6
out1  = do.specu(iris.dat, sigma=mysig, ranking="method1")
out2  = do.specu(iris.dat, sigma=mysig, ranking="method2")
out3  = do.specu(iris.dat, sigma=mysig, ranking="method3")

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=iris.lab, main="SPECU::method1")
plot(out2$Y, pch=19, col=iris.lab, main="SPECU::method2")
plot(out3$Y, pch=19, col=iris.lab, main="SPECU::method3")
par(opar)
```

---

do.splapeig *Supervised Laplacian Eigenmaps*

Description

Supervised Laplacian Eigenmaps (SPLAPEIG) is a supervised variant of Laplacian Eigenmaps. Instead of setting up explicit neighborhood, it utilizes an adaptive threshold strategy to define neighbors for both within- and between-class neighborhood. It then builds affinity matrices for each information and solves generalized eigenvalue problem. This algorithm may be quite sensitive in the choice of `beta` value.

Usage

```
do.splapeig(
  X,
  label,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
```

```
  beta = 1,
  gamma = 0.5
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
beta	bandwidth parameter for heat kernel in $[0, \infty)$.
gamma	a balancing parameter in $[0, 1]$ between within- and between-class information.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Raducanu B, Dornaika F (2012). "A supervised non-linear dimensionality reduction approach for manifold learning." *Pattern Recognition*, **45**(6), 2432–2444.

See Also

do.lapeig

Examples

```
## load iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.factor(iris[,5])

## try different balancing parameters with beta=50
out1 = do.splapeig(X, label, beta=50, gamma=0.3); Y1=out1$Y
out2 = do.splapeig(X, label, beta=50, gamma=0.6); Y2=out2$Y
out3 = do.splapeig(X, label, beta=50, gamma=0.9); Y3=out3$Y

## visualize
```

```
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(Y1, pch=19, col=label, main="gamma=0.3")
plot(Y2, pch=19, col=label, main="gamma=0.6")
plot(Y3, pch=19, col=label, main="gamma=0.9")
par(opar)
```

`do.spmds` *Spectral Multidimensional Scaling*

Description

`do.spmds` transfers the classical multidimensional scaling problem into the data spectral domain using Laplace-Beltrami operator. Its flexibility to use subsamples and spectral interpolation of non-reference data enables relatively efficient computation for large-scale data.

Usage

```
do.spmds(
  X,
  ndim = 2,
  neigs = max(2, nrow(X)/10),
  ratio = 0.1,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  type = c("proportion", 0.1),
  symmetric = c("union", "intersect", "asymmetric")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`ndim`	an integer-valued target dimension.
`neigs`	number of eigenvectors to be used as *spectral dimension*.
`ratio`	percentage of subsamples as reference points.
`preprocess`	an additional option for preprocessing the data. Default is `"null"`. See also `aux.preprocess` for more details.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`symmetric`	one of `"intersect"`, `"union"` or `"asymmetric"` is supported. Default is `"union"`. See also `aux.graphnbd` for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

Aflalo Y, Kimmel R (2013). "Spectral multidimensional scaling." *Proceedings of the National Academy of Sciences*, **110**(45), 18052–18057.

Examples

```
## Not run:
## Replicate the numerical example from the paper
#  Data Preparation
set.seed(100)
dim.true  = 3     # true dimension
dim.embed = 100   # embedding space (high-d)
npoints   = 1000  # number of samples to be generated

v     = matrix(runif(dim.embed*dim.true),ncol=dim.embed)
coeff = matrix(runif(dim.true*npoints),  ncol=dim.true)
X     = coeff%*%v

# see the effect of neighborhood size
out1  = do.spmds(X, neigs=100, type=c("proportion",0.10))
out2  = do.spmds(X, neigs=100, type=c("proportion",0.25))
out3  = do.spmds(X, neigs=100, type=c("proportion",0.50))

# visualize the results
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, main="10% neighborhood")
plot(out2$Y, main="25% neighborhood")
plot(out3$Y, main="50% neighborhood")
par(opar)

## End(Not run)
```

`do.spp` *Sparsity Preserving Projection*

Description

Sparsity Preserving Projection (SPP) is an unsupervised linear dimension reduction technique. It aims to preserve high-dimensional structure in a sparse manner to find projections that keeps such sparsely-connected pattern in the low-dimensional space. Note that we used **CVXR** for convenient computation, which may lead to slower execution once used for large dataset.

Usage

```
do.spp(
  X,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten"),
  reltol = 1e-04
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations
`ndim`	an integer-valued target dimension.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`reltol`	tolerance level for stable computation of sparse reconstruction weights.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Qiao L, Chen S, Tan X (2010). "Sparsity preserving projections with applications to face recognition." *Pattern Recognition*, **43**(1), 331–341.

Examples

```
## Not run:
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## test different tolerance levels
out1 <- do.spp(X,ndim=2,reltol=0.001)
out2 <- do.spp(X,ndim=2,reltol=0.01)
out3 <- do.spp(X,ndim=2,reltol=0.1)

# visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="SPP::reltol=.001")
plot(out2$Y, pch=19, col=label, main="SPP::reltol=.01")
plot(out3$Y, pch=19, col=label, main="SPP::reltol=.1")
par(opar)

## End(Not run)
```

---

do.spufs *Structure Preserving Unsupervised Feature Selection*

Description

This unsupervised feature selection method is based on self-expression model, which means that the cost function involves difference in self-representation. It does not explicitly require learning the clusterings and different features are weighted individually based on their relative importance. The cost function involves two penalties, sparsity and preservation of local structure.

Usage

```
do.spufs(
  X,
  ndim = 2,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate"),
  alpha = 1,
  beta = 1,
  bandwidth = 1
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.

ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
alpha	nonnegative number to control sparsity in rows of matrix of representation coefficients.
beta	nonnegative number to control the degree of local-structure preservation.
bandwidth	positive number for Gaussian kernel bandwidth to define similarity.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Lu Q, Li X, Dong Y (2018). "Structure preserving unsupervised feature selection." *Neurocomputing*, **301**, 36–45.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

#### try different bandwidth values
out1 = do.spufs(X, bandwidth=0.1)
out2 = do.spufs(X, bandwidth=1)
out3 = do.spufs(X, bandwidth=10)

#### visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="SPUFS::bandwidth=0.1")
plot(out2$Y, pch=19, col=label, main="SPUFS::bandwidth=1")
plot(out3$Y, pch=19, col=label, main="SPUFS::bandwidth=10")
par(opar)
```

`do.ssldp` *Semi-Supervised Locally Discriminant Projection*

Description

Semi-Supervised Locally Discriminant Projection (SSLDP) is a semi-supervised extension of LDP. It utilizes unlabeled data to overcome the small-sample-size problem under the situation where labeled data have the small number. Using two information, it both constructs the within- and between-class weight matrices incorporating the neighborhood information of the data set.

Usage

```
do.ssldp(
  X,
  label,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate"),
  beta = 0.5
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
`label`	a length-n vector of data class labels.
`ndim`	an integer-valued target dimension.
`type`	a vector of neighborhood graph construction. Following types are supported; `c("knn",k)`, `c("enn",radius)`, and `c("proportion",ratio)`. Default is `c("proportion",0.1)`, connecting about 1/10 of nearest data points among all data points. See also `aux.graphnbd` for more details.
`preprocess`	an additional option for preprocessing the data. Default is "center". See also `aux.preprocess` for more details.
`beta`	balancing parameter for intra- and inter-class information in $[0, 1]$.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Zhang S, Lei Y, Wu Y (2011). "Semi-supervised locally discriminant projection for classification and recognition." *Knowledge-Based Systems*, **24**(2), 341–346.

Examples

```
## use iris data
data(iris)
X     = as.matrix(iris[,1:4])
label = as.integer(iris$Species)

## copy a label and let 10% of elements be missing
nlabel = length(label)
nmissing = round(nlabel*0.10)
label_missing = label
label_missing[sample(1:nlabel, nmissing)]=NA

## compute with 3 different levels of 'beta' values
out1 = do.ssldp(X, label_missing, beta=0.1)
out2 = do.ssldp(X, label_missing, beta=0.5)
out3 = do.ssldp(X, label_missing, beta=0.9)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, main="SSLDP::beta=0.1")
plot(out2$Y, col=label, main="SSLDP::beta=0.5")
plot(out3$Y, col=label, main="SSLDP::beta=0.9")
par(opar)
```

---

do.tsne *t-distributed Stochastic Neighbor Embedding*

Description

t-distributed Stochastic Neighbor Embedding (t-SNE) is a variant of Stochastic Neighbor Embedding (SNE) that mimicks patterns of probability distributinos over pairs of high-dimensional objects on low-dimesional target embedding space by minimizing Kullback-Leibler divergence. While conventional SNE uses gaussian distributions to measure similarity, t-SNE, as its name suggests, exploits a heavy-tailed Student t-distribution.

Usage

```
do.tsne(
  X,
  ndim = 2,
  perplexity = 30,
  eta = 0.05,
```

```
  maxiter = 2000,
  jitter = 0.3,
  jitterdecay = 0.99,
  momentum = 0.5,
  preprocess = c("null", "center", "scale", "cscale", "decorrelate", "whiten"),
  pca = TRUE,
  pcascale = FALSE,
  symmetric = FALSE,
  BHuse = TRUE,
  BHtheta = 0.25
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
perplexity	desired level of perplexity; ranging [5,50].
eta	learning parameter.
maxiter	maximum number of iterations.
jitter	level of white noise added at the beginning.
jitterdecay	decay parameter in (0,1). The closer to 0, the faster artificial noise decays.
momentum	level of acceleration in learning.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.
pca	whether to use PCA as preliminary step; TRUE for using it, FALSE otherwise.
pcascale	a logical; FALSE for using Covariance, TRUE for using Correlation matrix. See also do.pca for more details.
symmetric	a logical; FALSE to solve it naively, and TRUE to adopt symmetrization scheme.
BHuse	a logical; TRUE to use Barnes-Hut approximation. See Rtsne for more details.
BHtheta	speed-accuracy tradeoff. If set as 0.0, it reduces to exact t-SNE.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

Author(s)

Kisung You

References

van der Maaten L, Hinton G (2008). "Visualizing data using t-SNE." *The Journal of Machine Learning Research*, **9**(2579-2605), 85.

See Also

do.sne

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
lab   = as.factor(iris[subid,5])

## compare different perplexity
out1 <- do.tsne(X, ndim=2, perplexity=5)
out2 <- do.tsne(X, ndim=2, perplexity=10)
out3 <- do.tsne(X, ndim=2, perplexity=15)

## Visualize three different projections
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="tSNE::perplexity=5")
plot(out2$Y, pch=19, col=lab, main="tSNE::perplexity=10")
plot(out3$Y, pch=19, col=lab, main="tSNE::perplexity=15")
par(opar)
```

---

do.udfs *Unsupervised Discriminative Features Selection*

Description

Though it may sound weird, this method aims at finding discriminative features under the unsupervised learning framework. It assumes that the class label could be predicted by a linear classifier and iteratively updates its discriminative nature while attaining row-sparsity scores for selecting features.

Usage

```
do.udfs(
  X,
  ndim = 2,
  lbd = 1,
  gamma = 1,
  k = 5,
  preprocess = c("null", "center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
lbd	regularization parameter for local Gram matrix to be invertible.
gamma	regularization parameter for row-sparsity via $\ell_{2,1}$ norm.
k	size of nearest neighborhood for each data point.
preprocess	an additional option for preprocessing the data. Default is "null". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

featidx a length-$ndim$ vector of indices with highest scores.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Yang Y, Shen HT, Ma Z, Huang Z, Zhou X (2011). "L2,1-norm Regularized Discriminative Feature Selection for Unsupervised Learning." In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, 1589–1594.

Examples

```
## use iris data
data(iris)
set.seed(100)
subid = sample(1:150, 50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

#### try different neighborhood size
out1 = do.udfs(X, k=5)
out2 = do.udfs(X, k=10)
out3 = do.udfs(X, k=25)

#### visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=label, main="UDFS::k=5")
plot(out2$Y, pch=19, col=label, main="UDFS::k=10")
```

```
plot(out3$Y, pch=19, col=label, main="UDFS::k=25")
par(opar)
```

---

do.udp *Unsupervised Discriminant Projection*

Description

Unsupervised Discriminant Projection (UDP) aims finding projection that balances local and global scatter. Even though the name contains the word *Discriminant*, this algorithm is *unsupervised*. The term there reflects its algorithmic tactic to discriminate distance points not in the neighborhood of each data point. It performs PCA as intermittent preprocessing for rank singularity issue. Authors clearly mentioned that it is inspired by Locality Preserving Projection, which minimizes the local scatter only.

Usage

```
do.udp(
  X,
  ndim = 2,
  type = c("proportion", 0.1),
  preprocess = c("center", "scale", "cscale", "decorrelate", "whiten")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
ndim	an integer-valued target dimension.
type	a vector of neighborhood graph construction. Following types are supported; c("knn",k), c("enn",radius), and c("proportion",ratio). Default is c("proportion",0.1), connecting about 1/10 of nearest data points among all data points. See also aux.graphnbd for more details.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

interimdim the number of PCA target dimension used in preprocessing.

Author(s)

Kisung You

References

Yang J, Zhang D, Yang J, Niu B (2007). "Globally Maximizing, Locally Minimizing: Unsupervised Discriminant Projection with Applications to Face and Palm Biometrics." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(4), 650–664.

See Also

do.lpp

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## use different connectivity level
out1 <- do.udp(X, type=c("proportion",0.05))
out2 <- do.udp(X, type=c("proportion",0.10))
out3 <- do.udp(X, type=c("proportion",0.25))

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, col=label, pch=19, main="connectivity 5%")
plot(out2$Y, col=label, pch=19, main="connectivity 10%")
plot(out3$Y, col=label, pch=19, main="connectivity 25%")
par(opar)
```

---

do.ulda *Uncorrelated Linear Discriminant Analysis*

Description

Uncorrelated LDA (ULDA) is an extension of LDA by using the uncorrelated discriminant transformation and Kahrunen-Loeve expansion of the basis.

Usage

```
do.ulda(
  X,
  label,
  ndim = 2,
  preprocess = c("center", "scale", "cscale", "whiten", "decorrelate")
)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations and columns represent independent variables.
label	a length-n vector of data class labels.
ndim	an integer-valued target dimension.
preprocess	an additional option for preprocessing the data. Default is "center". See also aux.preprocess for more details.

Value

a named list containing

Y an $(n \times ndim)$ matrix whose rows are embedded observations.

trfinfo a list containing information for out-of-sample prediction.

projection a $(p \times ndim)$ whose columns are basis for projection.

Author(s)

Kisung You

References

Jin Z, Yang J, Hu Z, Lou Z (2001). "Face recognition based on the uncorrelated discriminant transformation." *Pattern Recognition*, **34**(7), 1405 – 1416.

See Also

do.lda

Examples

```
## load iris data
data(iris)
set.seed(100)
subid = sample(1:150,50)
X     = as.matrix(iris[subid,1:4])
label = as.factor(iris[subid,5])

## compare with LDA
out1 = do.lda(X, label)
```

```
out2 = do.ulda(X, label)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(out1$Y, pch=19, col=label, main="LDA")
plot(out2$Y, pch=19, col=label, main="Uncorrelated LDA")
par(opar)
```

`est.boxcount` *Box-counting Dimension*

Description

Box-counting dimension, also known as Minkowski-Bouligand dimension, is a popular way of figuring out the fractal dimension of a set in a Euclidean space. Its idea is to measure the number of boxes required to cover the set repeatedly by decreasing the length of each side of a box. It is defined as

$$dim(S) = \lim \frac{\log N(r)}{\log(1/r)}$$

as $r \to 0$, where $N(r)$ is the number of boxes counted to cover a given set for each corresponding r.

Usage

```
est.boxcount(X, nlevel = 50, cut = c(0.1, 0.9))
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`nlevel`	the number of `r` (radius) to be tested.
`cut`	a vector of ratios for computing estimated dimension in $(0, 1)$.

Value

a named list containing containing

estdim estimated dimension using `cut` ratios.

r a vector of radius used.

Nr a vector of boxes counted for each corresponding `r`.

Determining the dimension

Even though we could use arbitrary `cut` to compute estimated dimension, it is also possible to use visual inspection. According to the theory, if the function returns an `output`, we can plot `plot(log(1/output$r),log(output$Nr))` and use the linear slope in the middle as desired dimension of data.

Automatic choice of r

The least value for radius r must have non-degenerate counts, while the maximal value should be the maximum distance among all pairs of data points across all coordinates. `nlevel` controls the number of interim points in a log-equidistant manner.

Author(s)

Kisung You

References

Hentschel H, Procaccia I (1983). "The infinite number of generalized dimensions of fractals and strange attractors." *Physica D: Nonlinear Phenomena*, **8**(3), 435–444.

Ott E (2002). *Chaos in dynamical systems*, 2nd ed edition. Cambridge University Press, Cambridge, U.K. ; New York. ISBN 978-0-521-81196-5 978-0-521-01084-9.

See Also

`est.correlation`

Examples

```
## generate three different dataset
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="twinpeaks")

## compute boxcount dimension
out1 = est.boxcount(X1)
out2 = est.boxcount(X2)
out3 = est.boxcount(X3)

## visually verify : all should have approximate slope of 2.
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(log(1/out1$r), log(out1$Nr), main="swiss roll")
plot(log(1/out2$r), log(out2$Nr), main="ribbon")
plot(log(1/out3$r), log(out3$Nr), main="twinpeaks")
par(opar)
```

`est.clustering` *Intrinsic Dimension Estimation via Clustering*

Description

Instead of directly using neighborhood information, `est.clustering` adopts hierarchical neighborhood information using `hclust` by recursively merging leafs over the range of radii.

Usage

```
est.clustering(X, kmin = round(sqrt(nrow(X))))
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`kmin`	minimal number of neighborhood size to search over.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Eriksson B, Crovella M (2012). "Estimating intrinsic dimension via clustering." In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 760–763.

Examples

```
## create 'swiss' roll dataset
X = aux.gensamples(dname="swiss")

## try different k values
out1 = est.clustering(X, kmin=5)
out2 = est.clustering(X, kmin=25)
out3 = est.clustering(X, kmin=50)

## print the results
line1 = paste0("* est.clustering : kmin=5  gives ",round(out1$estdim,2))
line2 = paste0("* est.clustering : kmin=25 gives ",round(out2$estdim,2))
line3 = paste0("* est.clustering : kmin=50 gives ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.correlation` *Correlation Dimension*

Description

Correlation dimension is a measure of determining the dimension of a given set. It is often referred to as a type of fractal dimension. Its mechanism is somewhat similar to that of box-counting dimension, but has the advantage of being intuitive as well as efficient in terms of computation with some robustness contingent on the lack of availability for large dataset.

$$dim(S) = \lim \frac{\log C(r)}{\log r}$$

as $r \to 0$, where $C(r) = \lim(2/(N-1) * N) \sum_i^N \sum_{j=i+1}^N I(\|x_i - x_j\| \leq r)$.

Usage

```
est.correlation(X, nlevel = 50, method = c("lm", "cut"), cut = c(0.1, 0.9))
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`nlevel`	the number of `r` (radius) to be tested.
`method`	method to estimate the intrinsic dimension; `"lm"` for fitting a linear model for the entire grid of values, and `"cut"` to trim extreme points. `"cut"` method is more robust.
`cut`	a vector of ratios for computing estimated dimension in $(0, 1)$.

Value

a named list containing containing

estdim estimated dimension using `cut` values.

r a vector of radius used.

Cr a vector of $C(r)$ as decribed above.

Determining the dimension

Even though we could use arbitrary `cut` to compute estimated dimension, it is also possible to use visual inspection. According to the theory, if the function returns an `output`, we can plot `plot(log(output$r),log(output$Cr))` and use the linear slope in the middle as desired dimension of data.

Automatic choice of r

The least value for radius r must have non-degenerate counts, while the maximal value should be the maximum distance among all pairs of data points across all coordinates. `nlevel` controls the number of interim points in a log-equidistant manner.

Author(s)

Kisung You

References

Grassberger P, Procaccia I (1983). "Measuring the strangeness of strange attractors." *Physica D: Nonlinear Phenomena*, **9**(1-2), 189–208.

See Also

`est.boxcount`

Examples

```
## generate three different dataset
set.seed(1)
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="twinpeaks")

## compute
out1 = est.correlation(X1)
out2 = est.correlation(X2)
out3 = est.correlation(X3)

## visually verify : all should have approximate slope of 2.
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(log(out1$r), log(out1$Cr), main="swiss roll")
plot(log(out2$r), log(out2$Cr), main="ribbon")
plot(log(out3$r), log(out3$Cr), main="twinpeaks")
par(opar)
```

---

est.danco *Intrinsic Dimensionality Estimation with DANCo*

Description

DANCo exploits the balanced information of both the normalized nearest neighbor distances as well as the angles of data pairs in the neighboring points.

Usage

```
est.danco(X, k = 5)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
k	the neighborhood size used for estimating local intrinsic dimension.

Value

a named list containing containing

estdim estimated dimension via the method.

References

Ceruti C, Bassis S, Rozza A, Lombardi G, Casiraghi E, Campadelli P (2014). "DANCo: An intrinsic dimensionality estimator exploiting angle and norm concentration." *Pattern Recognition*, **47**(8), 2569–2581.

Examples

```
## create 3 datasets of intrinsic dimension 2.
X1 = aux.gensamples(n=50, dname="swiss")
X2 = aux.gensamples(n=50, dname="ribbon")
X3 = aux.gensamples(n=50, dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.danco(X1, k=10)
out2 = est.danco(X2, k=10)
out3 = est.danco(X3, k=10)

## print the results
line1 = paste0("* est.danco : 'swiss'  estiamte is ",round(out1$estdim,2))
line2 = paste0("* est.danco : 'ribbon' estiamte is ",round(out2$estdim,2))
line3 = paste0("* est.danco : 'saddle' estiamte is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

---

est.gdistnn *Intrinsic Dimension Estimation based on Manifold Assumption and Graph Distance*

Description

As the name suggests, this function assumes that the data is sampled from the manifold in that graph representing the underlying manifold is first estimated via k-nn. Then graph distance is employed as an approximation of geodesic distance to locally estimate intrinsic dimension.

Usage

```
est.gdistnn(X, k = 5, k1 = 3, k2 = 10)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
k	the neighborhood size used for constructing a graph. We suggest it to be large enough to build a connected graph.
k1	local neighborhood parameter (smaller radius) for graph distance.
k2	local neighborhood parameter (larger radius) for graph distance.

Value

a named list containing containing

estdim the global estimated dimension, which is averaged local dimension.

estloc a length-n vector of locally estimated dimension at each point.

Author(s)

Kisung You

References

He J, Ding L, Jiang L, Li Z, Hu Q (2014). "Intrinsic dimensionality estimation based on manifold assumption." *Journal of Visual Communication and Image Representation*, **25**(5), 740–747.

Examples

```
## create 3 datasets of intrinsic dimension 2.
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.gdistnn(X1, k=10)
out2 = est.gdistnn(X2, k=10)
out3 = est.gdistnn(X3, k=10)

## print the results
sprintf("* est.gdistnn : estimated dimension for 'swiss'  data is %.2f.",out1$estdim)
sprintf("* est.gdistnn : estimated dimension for 'ribbon' data is %.2f.",out2$estdim)
sprintf("* est.gdistnn : estimated dimension for 'saddle' data is %.2f.",out3$estdim)

line1 = paste0("* est.gdistnn : 'swiss'  estiamte is ",round(out1$estdim,2))
line2 = paste0("* est.gdistnn : 'ribbon' estiamte is ",round(out2$estdim,2))
line3 = paste0("* est.gdistnn : 'saddle' estiamte is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

```
## compare with local-dimension estimate
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
hist(out1$estloc, main="Result-'Swiss'", xlab="local dimension")
abline(v=out1$estdim, lwd=3, col="red")
hist(out2$estloc, main="Result-'Ribbon'", xlab="local dimension")
abline(v=out2$estdim, lwd=3, col="red")
hist(out3$estloc, main="Result-'Saddle'", xlab="local dimension")
abline(v=out2$estdim, lwd=3, col="red")
par(opar)
```

`est.incisingball` *Intrinsic Dimension Estimation with Incising Ball*

Description

Incising ball methods exploits the exponential relationship of the number of samples contained in a ball and the radius of the incising ball.

Usage

```
est.incisingball(X)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Fan M, Qiao H, Zhang B (2009). "Intrinsic dimension estimation of manifolds by incising balls." *Pattern Recognition*, **42**(5), 780–787.

Examples

```
## create an example data with intrinsic dimension 2
X = cbind(aux.gensamples(dname="swiss"),aux.gensamples(dname="swiss"))

## acquire an estimate for intrinsic dimension
output = est.incisingball(X)
sprintf("* est.incisingball : estimated dimension is %d.",output$estdim)
```

`est.made` *Manifold-Adaptive Dimension Estimation*

Description

`do.made` first aims at finding local dimesion estimates using nearest neighbor techniques based on the first-order approximation of the probability mass function and then combines them to get a single global estimate. Due to the rate of convergence of such estimate to be independent of assumed dimensionality, authors claim this method to be *manifold-adaptive*.

Usage

```
est.made(
  X,
  k = round(sqrt(ncol(X))),
  maxdim = min(ncol(X), 15),
  combine = c("mean", "median", "vote")
)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`k`	size of neighborhood for analysis.
`maxdim`	maximum possible dimension allowed for the algorithm to investigate.
`combine`	method to aggregate local estimates for a single global estimate.

Value

a named list containing containing

estdim estimated global intrinsic dimension.

estloc a length-n vector estimated dimension at each point.

Author(s)

Kisung You

References

Farahmand AM, Szepesvári C, Audibert J (2007). "Manifold-adaptive dimension estimation." In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, 265–272.

Examples

```
## create a data set of intrinsic dimension 2.
X = aux.gensamples(dname="swiss")

## compare effect of 3 combining scheme
out1 = est.made(X, combine="mean")
out2 = est.made(X, combine="median")
out3 = est.made(X, combine="vote")

## print the results
line1 = paste0("* est.made : 'mean'   estiamte is ",round(out1$estdim,2))
line2 = paste0("* est.made : 'median' estiamte is ",round(out2$estdim,2))
line3 = paste0("* est.made : 'vote'   estiamte is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

---

est.mindkl *MiNDkl*

Description

It is a minimum neighbor distance estimator of the intrinsic dimension based on Kullback Leibler divergence estimator.

Usage

```
est.mindkl(X, k = 5)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
k	the neighborhood size for defining locality.

Value

a named list containing containing

estdim the global estimated dimension.

Author(s)

Kisung You

References

Lombardi G, Rozza A, Ceruti C, Casiraghi E, Campadelli P (2011). "Minimum Neighbor Distance Estimators of Intrinsic Dimension." In Gunopulos D, Hofmann T, Malerba D, Vazirgiannis M (eds.), *Machine Learning and Knowledge Discovery in Databases*, volume 6912, 374–389. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-23782-9 978-3-642-23783-6, doi: 10.1007/9783642237836_24.

See Also

`est.mindml`

Examples

```
## create 3 datasets of intrinsic dimension 2.
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.mindkl(X1, k=5)
out2 = est.mindkl(X2, k=5)
out3 = est.mindkl(X3, k=5)

## print the results
line1 = paste0("* est.mindkl : 'swiss'  estiamte is ",round(out1$estdim,2))
line2 = paste0("* est.mindkl : 'ribbon' estiamte is ",round(out2$estdim,2))
line3 = paste0("* est.mindkl : 'saddle' estiamte is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.mindml` *MINDml*

Description

It is a minimum neighbor distance estimator of the intrinsic dimension based on Maximum Likelihood principle.

Usage

```
est.mindml(X, k = 5)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`k`	the neighborhood size for defining locality.

Value

a named list containing containing

estdim the global estimated dimension.

Author(s)

Kisung You

References

Lombardi G, Rozza A, Ceruti C, Casiraghi E, Campadelli P (2011). "Minimum Neighbor Distance Estimators of Intrinsic Dimension." In Gunopulos D, Hofmann T, Malerba D, Vazirgiannis M (eds.), *Machine Learning and Knowledge Discovery in Databases*, volume 6912, 374–389. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-23782-9 978-3-642-23783-6, doi: 10.1007/9783642237836_24.

See Also

`est.mindkl`

Examples

```
## create 3 datasets of intrinsic dimension 2.
set.seed(100)
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.mindml(X1, k=10)
out2 = est.mindml(X2, k=10)
out3 = est.mindml(X3, k=10)

## print the results
line1 = paste0("* est.mindml : 'swiss'  estiamte is ",round(out1$estdim,2))
line2 = paste0("* est.mindml : 'ribbon' estiamte is ",round(out2$estdim,2))
line3 = paste0("* est.mindml : 'saddle' estiamte is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.mle1` *Maximum Likelihood Esimation with Poisson Process*

Description

Assuming the density in a hypersphere is constant, authors proposed to build a likelihood structure based on modeling local spread of information via Poisson Process. `est.mle1` requires two parameters that model the reasonable range of neighborhood size to reflect inhomogeneity of distribution across data points.

Usage

```
est.mle1(X, k1 = 10, k2 = 20)
```

Arguments

X	an $(n \times p)$ matrix or data frame whose rows are observations.
k1	minimum neighborhood size, larger than 1.
k2	maximum neighborhood size, smaller than n.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Levina E, Bickel PJ (2005). "Maximum Likelihood Estimation of Intrinsic Dimension." In Saul LK, Weiss Y, Bottou L (eds.), *Advances in Neural Information Processing Systems 17*, 777–784. MIT Press.

Examples

```
## create example data sets with intrinsic dimension 2
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.mle1(X1)
out2 = est.mle1(X2)
out3 = est.mle1(X3)

## print the estimates
line1 = paste0("* est.mle1 : 'swiss'  estiamte is ",round(out1$estdim,2))
line2 = paste0("* est.mle1 : 'ribbon' estiamte is ",round(out2$estdim,2))
line3 = paste0("* est.mle1 : 'saddle' estiamte is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.mle2` *Maximum Likelihood Esimation with Poisson Process and Bias Correction*

Description

Authors argue that the approach proposed in `est.mle1` is empirically bias-prone in that the averaging of sample statistics over all data points is taken to be a harmonic manner.

Usage

```
est.mle2(X, k1 = 10, k2 = 20)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`k1`	minimum neighborhood size, larger than 1.
`k2`	maximum neighborhood size, smaller than n.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

MacKay D, Ghahramani Z (2005). "Comments on 'Maximum Likelihood Estimation of Intrinsic Dimension' by E. Levina and P. Bickel (2004)." http://www.inference.org.uk/mackay/dimension/.

Examples

```
## create example data sets with intrinsic dimension 2
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.mle2(X1)
out2 = est.mle2(X2)
out3 = est.mle2(X3)

line1 = paste0("* est.mle2 : dimension of 'swiss'  data is ",round(out1$estdim,2))
```

```
line2 = paste0("* est.mle2 : dimension of 'ribbon' data is ",round(out2$estdim,2))
line3 = paste0("* est.mle2 : dimension of 'saddle' data is ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.nearneighbor1` *Intrinsic Dimension Estimation with Near-Neighbor Information*

Description

Based on an assumption of data points being locally uniformly distributed, `est.nearneighbor1` estimates the intrinsic dimension based on the local distance information in an iterative manner.

Usage

```
est.nearneighbor1(X, K = max(2, round(ncol(X)/5)))
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`K`	maximum neighborhood size, smaller than p.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Pettis KW, Bailey TA, Jain AK, Dubes RC (1979). "An Intrinsic Dimensionality Estimator from Near-Neighbor Information." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-1**(1), 25–37.

Examples

```
## create an example data with intrinsic dimension 2
X = cbind(aux.gensamples(dname="swiss"),aux.gensamples(dname="swiss"))

## acquire an estimate for intrinsic dimension
output = est.nearneighbor1(X)
sprintf("* est.nearneighbor1 : estimated dimension is %.2f.",output$estdim)
```

`est.nearneighbor2` *Near-Neighbor Information with Bias Correction*

Description

Though similar to `est.nearneighbor1`, authors of the reference argued that there exists innate bias in the method and proposed a non-iterative algorithm to reflect local distance information under a range of neighborhood sizes.

Usage

```
est.nearneighbor2(X, kmin = 2, kmax = max(3, round(ncol(X)/2)))
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`kmin`	minimum neighborhood size, larger than 1.
`kmax`	maximum neighborhood size, smaller than p.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Verveer P, Duin R (1995). "An evaluation of intrinsic dimensionality estimators." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(1), 81–86.

Examples

```
## create an example data with intrinsic dimension 2
X = cbind(aux.gensamples(dname="swiss"),aux.gensamples(dname="swiss"))

## acquire an estimate for intrinsic dimension
output = est.nearneighbor2(X)
sprintf("* est.nearneighbor2 : estimated dimension is %.2f.",output$estdim)
```

`est.packing` *Intrinsic Dimension Estimation using Packing Numbers*

Description

Instead of covering numbers which are expensive to compute in many fractal-based methods, `est.packing` exploits packing numbers as a proxy to describe spatial density. Since it involves random permutation of the dataset at each iteration, every run might have different results.

Usage

```
est.packing(X, eps = 0.01)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`eps`	small positive number for stopping threshold.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Kégl B (2002). "Intrinsic Dimension Estimation Using Packing Numbers." In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, 697–704.

Examples

```
## create 'swiss' roll dataset
X = aux.gensamples(dname="swiss")

## try different eps values
out1 = est.packing(X, eps=0.1)
out2 = est.packing(X, eps=0.01)
out3 = est.packing(X, eps=0.001)

## print the results
line1 = paste0("* est.packing : eps=0.1   gives ",round(out1$estdim,2))
line2 = paste0("* est.packing : eps=0.01  gives ",round(out2$estdim,2))
line3 = paste0("* est.packing : eps=0.001 gives ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.pcathr` *PCA Thresholding with Accumulated Variance*

Description

Principal Component Analysis exploits sample covariance matrix whose eigenvectors and eigenvalues are principal components and projected variance, correspondingly. Given `varratio`, it thresholds the accumulated variance and selects the estimated dimension. Note that other than linear submanifold case, the naive selection scheme from this algorithm lacks flexibility in discovering intrinsic dimension.

Usage

```
est.pcathr(X, varratio = 0.95)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`varratio`	target explainability for accumulated variance in $(0, 1)$.

Value

a named list containing containing

estdim estimated dimension according to `varratio`.

values eigenvalues of sample covariance matrix.

Author(s)

Kisung You

See Also

`do.pca`

Examples

```
## generate 3-dimensional normal data
X = matrix(rnorm(100*3), nrow=100)

## replicate 3 times with translations
Y = cbind(X-10,X,X+10)

## use PCA thresholding estimation with 95% variance explainability
## desired return is for dimension 3.
output   = est.pcathr(Y)
```

```
pmessage = paste("* estimated dimension is ",output$estdim, sep="")
print(pmessage)

## use screeplot
opar <- par(no.readonly=TRUE)
plot(output$values, main="scree plot", type="b")
par(opar)
```

`est.twonn` *Intrinsic Dimension Estimation by a Minimal Neighborhood Information*

Description

Unlike many intrinsic dimension (ID) estimation methods, `est.twonn` only requires two nearest datapoints from a target point and their distances. This extremely minimal approach is claimed to redue the effects of curvature and density variation across different locations in an underlying manifold.

Usage

```
est.twonn(X)
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Facco E, d'Errico M, Rodriguez A, Laio A (2017). "Estimating the intrinsic dimension of datasets by a minimal neighborhood information." *Scientific Reports*, **7**(1).

Examples

```
## create 3 datasets of intrinsic dimension 2.
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.twonn(X1)
out2 = est.twonn(X2)
out3 = est.twonn(X3)

## print the results
line1 = paste0("* est.twonn : 'swiss'  gives ",round(out1$estdim,2))
line2 = paste0("* est.twonn : 'ribbon' gives ",round(out2$estdim,2))
line3 = paste0("* est.twonn : 'saddle' gives ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`est.Ustat` *ID Estimation with Convergence Rate of U-statistic on Manifold*

Description

U-statistic is built upon theoretical arguments with the language of smooth manifold. The convergence rate of the statistic is achieved as a proxy for the estimated dimension by, at least partially, considering the scale and influence of extrinsic curvature. The method returns *integer* valued estimate in that there is no need for rounding the result for practical usage.

Usage

```
est.Ustat(X, maxdim = min(ncol(X), 15))
```

Arguments

`X`	an $(n \times p)$ matrix or data frame whose rows are observations.
`maxdim`	maximum possible dimension allowed for the algorithm to investigate.

Value

a named list containing containing

estdim estimated intrinsic dimension.

Author(s)

Kisung You

References

Hein M, Audibert J (2005). "Intrinsic dimensionality estimation of submanifolds in R ^{d}\$." In *Proceedings of the 22nd international conference on Machine learning*, 289–296.

Examples

```
## create 3 datasets of intrinsic dimension 2.
X1 = aux.gensamples(dname="swiss")
X2 = aux.gensamples(dname="ribbon")
X3 = aux.gensamples(dname="saddle")

## acquire an estimate for intrinsic dimension
out1 = est.Ustat(X1)
out2 = est.Ustat(X2)
out3 = est.Ustat(X3)

## print the results
line1 = paste0("* est.Ustat : 'swiss'  gives ",round(out1$estdim,2))
line2 = paste0("* est.Ustat : 'ribbon' gives ",round(out2$estdim,2))
line3 = paste0("* est.Ustat : 'saddle' gives ",round(out3$estdim,2))
cat(paste0(line1,"\n",line2,"\n",line3))
```

`oos.linear` *Out-Of-Sample Prediction for Linear Methods*

Description

Linear dimensionality reduction methods such as PCA, LPP, or ICA *explicitly* returns a matrix for mapping or projection. When we have new data, therefore, we can simply use the mapping provided. Inputs `projection` and `trfinfo` should be brought from original model you trained.

Usage

```
oos.linear(Xnew, projection, trfinfo)
```

Arguments

`Xnew`	an $(m \times p)$ matrix or data frame whose rows are observations. If a vector is given, it will be considered as an $(1 \times p)$ matrix with single observation.
`projection`	a $(p \times ndim)$ projection matrix.
`trfinfo`	a list containing transformation information generated from manifold learning algorithms. See also `aux.preprocess` for more details.

Value

a named list containing

Ynew an $(m \times ndim)$ matrix whose rows are embedded observations.

Author(s)

Kisung You

Examples

```
## generate sample data and separate them
X = aux.gensamples(n=500)
set.seed(46556)
idxtest  = sample(1:500,20)        # 20% of data for testing
idxtrain = setdiff(1:500,idxtest)  # 80% of data for training

Xtrain = X[idxtrain,]
Xtest  = X[idxtest,]

## run PCA for train data
res_train = do.pca(Xtrain,ndim=2,preprocess="whiten")

## perform OOS.LINEAR on new dataset
## note that inputs should be from a given model you trained
model.projection = res_train$projection
model.trfinfo    = res_train$trfinfo
res_test  = oos.linear(Xtest, model.projection, model.trfinfo)

## let's compare via visualization
xx = c(-2,2) # range of axis 1 for compact visualization
yy = c(-2,2) # range of axis 2 for compact visualization
mm = "black=train / red=test data" # figure title
YY = res_test$Ynew  # out-of-sample projection for test data

opar <- par(no.readonly=TRUE)
plot(res_train$Y, type="p", xlim=xx, ylim=yy,
     main=mm, xlab="axis 1", ylab="axis 2")
points(YY[,1], YY[,2], lwd=3, col="red")
par(opar)
```

`oos.linproj` *OOS : Linear Projection*

Description

The simplest way of out-of-sample extension might be linear regression even though the original embedding is not the linear type by solving

$$\min_\beta \|X_{old}\beta - Y_{old}\|_2^2$$

and use the estimate $\hat{beta}$ to acquire

$$Y_{new} = X_{new}\hat{\beta}$$

. Due to the choice of original preprocessing, `trfinfo` must be brought from the original model you trained.

Usage

```
oos.linproj(Xold, Yold, trfinfo, Xnew)
```

Arguments

`Xold`	an $(n \times p)$ matrix of data in original high-dimensional space.
`Yold`	an $(n \times ndim)$ matrix of data in reduced-dimensional space.
`trfinfo`	a list containing transformation information generated from manifold learning algorithms. See also `aux.preprocess` for more details.
`Xnew`	an $(m \times p)$ matrix for out-of-sample extension.

Value

a named list containing

Ynew an $(m \times ndim)$ matrix whose rows are embedded observations.

Author(s)

Kisung You

Examples

```
## generate sample data and separate them
X = aux.gensamples(n=500)
set.seed(46556)
idxselect  = sample(1:500,20)

Xold = X[setdiff(1:500,idxselect),]  # 80% of data for training
Xnew = X[idxselect,]                 # 20% of data for testing

## run PCA for train data
training = do.pca(Xold,ndim=2,preprocess="whiten")
Yold     = training$Y       # embedded data points
oldinfo  = training$trfinfo # preprocessing information

## perform out-of-sample extension
```

```
output  = oos.linproj(Xold, Yold, oldinfo, Xnew)
Ynew    = output$Ynew

## let's compare via visualization
xx = c(-2,2) # range of axis 1 for compact visualization
yy = c(-2,2) # range of axis 2 for compact visualization
mm = "black=train / red=test data" # figure title

## visualize
opar <- par(no.readonly=TRUE)
plot(Yold, type="p", xlim=xx, ylim=yy, main=mm, xlab="axis 1", ylab="axis 2")
points(Ynew[,1], Ynew[,2], lwd=3, col="red")
par(opar)
```

`package-Rdimtools` *Dimension Reduction and Estimation Methods*

Description

Rdimtools is an R suite of a number of dimension reduction and estimation methods implemented using **RcppArmadillo** for efficient computations. Please see the reference from the package webpage.

Index