Package ‘qkerntool’

October 13, 2022

Title Q-Kernel-Based and Conditionally Negative Definite Kernel-Based Machine Learning Tools

Version 1.19

Description Nonlinear machine learning tool for classification, clustering and dimensionality reduction. It integrates 12 q-kernel functions and 15 conditional negative definite kernel functions and includes the q-kernel and conditional negative definite kernel version of density-based spatial clustering of applications with noise, spectral clustering, generalized discriminant analysis, principal component analysis, multidimensional scaling, locally linear embedding, sammon's mapping and t-Distributed stochastic neighbor embedding.

Depends R (>= 3.0.1)

Imports stats, class, graphics, methods

License GPL (>= 2)

Encoding UTF-8

LazyData true

Maintainer Yusen Zhang <yusenzhang@126.com>

RoxygenNote 6.1.0

NeedsCompilation no

Author Yusen Zhang [aut, cre] (<https://orcid.org/0000-0003-3842-1153>),
Daolin Pang [ctb],
Jinghao Wang [ctb],
Jialin Zhang [ctb]

Repository CRAN

Date/Publication 2019-04-13 23:02:44 UTC

R topics documented:

`as.cndkernmatrix` *Assing cndkernmatrix class to matrix objects*

Description

`as.cndkernmatrix` in package **qkerntool** can be used to create the cndkernmatrix class to matrix objects representing a CND kernel matrix. These matrices can then be used with the cndkernmatrix interfaces which most of the functions in **qkerntool** support.

Usage

```
## S4 method for signature 'matrix'
as.cndkernmatrix(x, center = FALSE)
```

Arguments

`x`	matrix to be assigned the `cndkernmatrix` class
`center`	center the cndkernel matrix in feature space (default: FALSE)

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

cndkernmatrix,qkernmatrix

Examples

```
## Create the data
x <- rbind(matrix(rnorm(10),,2),matrix(rnorm(10,mean=3),,2))
y <- matrix(c(rep(1,5),rep(-1,5)))

### Use as.cndkernmatrix to label the cov. matrix as a CND kernel matrix
### which is eq. to using a linear kernel

K <- as.cndkernmatrix(crossprod(t(x)))

K
```

---

as.qkernmatrix *Assing qkernmatrix class to matrix objects*

Description

`as.qkernmatrix` in package **qkerntool** can be used to create the qkernmatrix class to matrix objects representing a q kernel matrix. These matrices can then be used with the qkernmatrix interfaces which most of the functions in **qkerntool** support.

Usage

```
## S4 method for signature 'matrix'
as.qkernmatrix(x, center = FALSE)
```

Arguments

`x`	matrix to be assigned the `qkernmatrix` class
`center`	center the kernel matrix in feature space (default: FALSE)

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkernmatrix,cndkernmatrix

Examples

```
## Create the data
x <- rbind(matrix(rnorm(10),,2),matrix(rnorm(10,mean=3),,2))
y <- matrix(c(rep(1,5),rep(-1,5)))

### Use as.qkernmatrix to label the cov. matrix as a qkernel matrix
### which is eq. to using a linear kernel

K <- as.qkernmatrix(crossprod(t(x)))

K
```

---

bases *qKernel Functions*

Description

The kernel generating functions provided in qkerntool.
The Non Linear Kernel $k(x,y) = \frac{1}{2(1-q)}(q^{-\alpha||x||^2} + q^{-\alpha||y||^2} - 2q^{-\alpha x'y})$.
The Gaussian kernel $k(x,y) = \frac{1}{1-q}(1 - q^{(||x-y||^2/\sigma)})$.
The Laplacian Kernel $k(x,y) = \frac{1}{1-q}(1 - q^{(||x-y||/\sigma)})$.

The Rational Quadratic Kernel $k(x,y) = \frac{1}{1-q}(1 - q^{\frac{||x-y||^2}{||x-y||^2+c}})$.
The Multiquadric Kernel $k(x,y) = \frac{1}{1-q}(q^c - q^{\sqrt{||x-y||^2+c}})$.
The Inverse Multiquadric Kernel $k(x,y) = \frac{1}{1-q}(q^{-\frac{1}{c}} - q^{-\frac{1}{\sqrt{||x-y||^2+c}}})$.
The Wave Kernel $k(x,y) = \frac{1}{1-q}(q^{-1} - q^{-\frac{\theta}{||x-y||}\sin\frac{||x-y||}{\theta}})$.
The d Kernel $k(x,y) = \frac{1}{1-q}[1 - q^(||x-y||^d)]$.
The Log Kernel $k(x,y) = \frac{1}{1-q}[1 - q^l n(||x-y||^d + 1)]$.
The Cauchy Kernel $k(x,y) = \frac{1}{1-q}(q^{-1} - q^{-\frac{1}{1+||x-y||^2/\sigma}})$.
The Chi-Square Kernel $k(x,y) = \frac{1}{1-q}(1 - q^{\sum 2(x-y)^2/(x+y)\gamma})$.
The Generalized T-Student Kernel $k(x,y) = \frac{1}{1-q}(q^{-1} - q^{-\frac{1}{1+||x-y||^d}})$.

Usage

```
rbfbase(sigma=1,q=0.8)
nonlbase(alpha = 1,q = 0.8)
```

```
laplbase(sigma = 1, q = 0.8)
ratibase(c = 1, q = 0.8)
multbase(c = 1, q = 0.8)
invbase(c = 1, q = 0.8)
wavbase(theta = 1,q = 0.8)
powbase(d = 2, q = 0.8)
logbase(d = 2, q = 0.8)
caubase(sigma = 1, q = 0.8)
chibase(gamma = 1, q = 0.8)
studbase(d = 2, q = 0.8)
```

Arguments

`q`	for all the qkernel function.
`sigma`	for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
`alpha`	for the Non Linear qkernel function "nonlbase".
`c`	for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
`theta`	for the Wave qkernel function "wavbase".
`d`	for the d qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
`gamma`	for the Chi-Square qkernel function "chibase".

Details

The kernel generating functions are used to initialize a kernel function which calculates the kernel function value between two feature vectors in a Hilbert Space. These functions can be passed as a `qkernel` argument on almost all functions in **qkerntool**(e.g., `qkgda`, `qkpca` etc).

Value

Return an S4 object of class `qkernel` which extents the `function` class. The resulting function implements the given kernel calculating the kernel function value between two vectors.

`qpar`	a list containing the kernel parameters (hyperparameters) used.

The kernel parameters can be accessed by the `qpar` function.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

`qkernmatrix`, `cndkernmatrix`

Examples

```
qkfunc <- rbfbase(sigma=1,q=0.8)
qkfunc

qpar(qkfunc)

## create two vectors
x <- rnorm(10)
y <- rnorm(10)

## calculate dot product
qkfunc(x,y)
```

---

blkdiag — *Block diagonal concatenation of matrix*

Description

Y = BLKDIAG(A,B,...) produces diag(A,B,...)

Usage

```
blkdiag(x)
```

Arguments

`x`	a list of matrix

Value

E - Block diagonal concatenation of matrix

Author(s)

Yusen Zhang
<yusenzhang@126.com>

`cndkernel-class` *Class "cndkernel" "nonlkernel" "polykernel" "rbfkernel" "laplkernel"*

Description

The built-in kernel classes in **qkerntool**

Objects from the Class

Objects can be created by calls of the form `new("nonlkernel")`, `new{"polykernel"}`, `new{"rbfkernel"}`, `new{"laplkernel"}`, `new{"anokernel"}`, `new{"ratikernel"}`, `new{"multkernel"}`, `new{"invkernel"}`, `new{"wavkernel"}`, `new{"powkernel"}`, `new{"logkernel"}`, `new{"caukernel"}`, `new{"chikernel"}`, `new{"studkernel"}`,`new{"norkernel"}`

or by calling the `nonlcnd`,`polycnd`, `rbfcnd`, `laplcnd`, `anocnd`, `raticnd`, `multcnd`, `invcnd`, `wavcnd`, `powcnd`, `logcnd`, `caucnd`, `chicnd`, `studcnd`, `norcnd` functions etc..

Slots

`.Data`**:** Object of class `"function"` containing the kernel function

`qpar`**:** Object of class `"list"` containing the kernel parameters

Methods

cndkernmatrix `signature(kernel = "rbfkernel", x ="matrix")`: computes the kernel matrix

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

See Also

`qkernmatrix`,`cndkernmatrix`

Examples

```
cndkfunc <- rbfcnd(gamma = 1)
cndkfunc

qpar(cndkfunc)

## create two vectors
x <- rnorm(10)
y <- rnorm(10)


cndkfunc(x,y)
```

`cndkernmatrix` *CND Kernel Matrix functions*

Description

`cndkernmatrix` calculates the kernel matrix $K_{ij} = k(x_i, x_j)$ or $K_{ij} = k(x_i, y_j)$.

Usage

```
## S4 method for signature 'cndkernel'
cndkernmatrix(cndkernel, x, y = NULL)
```

Arguments

`cndkernel`	the cndkernel function to be used to calculate the CND kernel matrix. This has to be a function of class `cndkernel`, i.e. which can be generated either one of the build in kernel generating functions (e.g., `rbfcnd` `nonlcnd` etc.) or a user defined function of class `cndkernel` taking two vector arguments and returning a scalar.
`x`	a data matrix to be used to calculate the kernel matrix.
`y`	second data matrix to calculate the kernel matrix.

Details

Common functions used during kernel based computations.
The `cndkernel` parameter can be set to any function, of class cndkernel, which computes the kernel function value in feature space between two vector arguments. **qkerntool** provides more than 10 CND kernel functions which can be initialized by using the following functions:

- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Gaussian cndkernel function
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` d cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

(see example.)

Value

`cndkernmatrix` returns a conditionally negative definite matrix with a zero diagonal element.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

nonlbase, rbfbase, laplbase, ratibase, multbase, invbase, wavbase, powbase, logbase, caubase, chibase, studbase

Examples

```
## use the iris data
data(iris)
dt <- as.matrix(iris[ ,-5])

## initialize cndkernel function
lapl <- laplcnd(gamma = 1)
lapl

## calculate cndkernel matrix
cndkernmatrix(lapl, dt)
```

---

cnds *CND Kernel Functions*

Description

The kernel generating functions provided in qkerntool.
The Non Linear Kernel $k(x,y) = [exp(\alpha||x||^2) + exp(\alpha||y||^2) - 2exp(\alpha x'y)]/2$.
The Polynomial kernel $k(x,y) = [(\alpha||x||^2 + c)^d + (\alpha||y||^2 + c)^d - 2(\alpha x'y + c)^d]/2$.
The Gaussian kernel $k(x,y) = 1 - exp(-||x-y||^2/\gamma)$.
The Laplacian Kernel $k(x,y) = 1 - exp(-||x-y||/\gamma)$.
The ANOVA Kernel $k(x,y) = n - \sum exp(-\sigma(x-y)^2)^d$.
The Rational Quadratic Kernel $k(x,y) = ||x-y||^2/(||x-y||^2 + c)$.
The Multiquadric Kernel $k(x,y) = \sqrt{(||x-y||^2 + c^2)} - c$.
The Inverse Multiquadric Kernel $k(x,y) = 1/c - 1/\sqrt{||x-y||^2 + c^2}$.
The Wave Kernel $k(x,y) = 1 - \frac{\theta}{||x-y||} \sin \frac{||x-y||}{\theta}$.
The d Kernel $k(x,y) = ||x-y||^d$.
The Log Kernel $k(x,y) = \log(||x-y||^d + 1)$.
The Cauchy Kernel $k(x,y) = 1 - 1/(1 + ||x-y||^2/\gamma)$.
The Chi-Square Kernel $k(x,y) = \sum 2(x-y)^2/(x+y)$.
The Generalized T-Student Kernel $k(x,y) = 1 - 1/(1 + ||x-y||^d)$.
The normal Kernel $k(x,y) = ||x-y||^2$.

Usage

```
nonlcnd(alpha = 1)
polycnd(d = 2, alpha = 1, c = 1)
rbfcnd(gamma = 1)
laplcnd(gamma = 1)
anocnd(d = 2, sigma = 1)
raticnd(c = 1)
multcnd(c = 1)
invcnd(c = 1)
wavcnd(theta = 1)
powcnd(d = 2)
logcnd(d = 2)
caucnd(gamma = 1)
chicnd( )
studcnd(d = 2)
norcnd()
```

Arguments

`alpha`	for the Non Linear cndkernel function "nonlcnd" and the Polynomial cndkernel function "polycnd".
`gamma`	for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
`sigma`	for the ANOVA cndkernel function "anocnd".
`theta`	for the Wave cndkernel function "wavcnd".
`c`	for the Rational Quadratic cndkernel function "raticnd", the Polynomial cndkernel function "polycnd", the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
`d`	for the Polynomial cndkernel function "polycnd", the ANOVA cndkernel function "anocnd", the cndkernel function "powcnd", the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Details

The kernel generating functions are used to initialize a kernel function which calculates the kernel function value between two feature vectors in a Hilbert Space. These functions can be passed as a `qkernel` argument on almost all functions in **qkerntool**.

Value

Return an S4 object of class `cndkernel` which extents the `function` class. The resulting function implements the given kernel calculating the kernel function value between two vectors.

`qpar`	a list containing the kernel parameters (hyperparameters) used.

The kernel parameters can be accessed by the `qpar` function.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

cndkernmatrix, qkernmatrix

Examples

```
cndkfunc <- rbfcnd(gamma = 1)
cndkfunc

qpar(cndkfunc)

## create two vectors
x <- rnorm(10)
y <- rnorm(10)

## calculate dot product
cndkfunc(x,y)
```

Eucdist *Computes the Euclidean(square Euclidean) distance matrix*

Description

Eucdist Computes the Euclidean(square Euclidean) distance matrix.

Arguments

x	(NxD) matrix (N samples, D features)
y	(MxD) matrix (M samples, D features)
sEuclidean	can be TRUE or FALSE, FALSE to Compute the Euclidean distance matrix.

Value

E - (MxN) Euclidean (square Euclidean) distances between vectors in x and y

Author(s)

Yusen Zhang
<yusenzhang@126.com>

Examples

```
###
data(iris)
testset <- sample(1:150,20)
x <- as.matrix(iris[-testset,-5])
y <- as.matrix(iris[testset,-5])

##
res0 <- Eucdist(x)
res1 <- Eucdist(x, x, sEuclidean = FALSE)
res2 <- Eucdist(x, y = NULL, sEuclidean = FALSE)
res3 <- Eucdist(x, x, sEuclidean = TRUE)
res4 <- Eucdist(x, y = NULL)
res5 <- Eucdist(x, sEuclidean = FALSE)
```

---

mfeat_pix *mfeat_pix dataset*

Description

This dataset consists of features of handwritten numerals (‘0’–‘9’) extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. This dataset is about 240 pixel averages in 2 x 3 windows

Usage

```
data("mfeat_pix")
```

Format

A data frame with 2000 observations on the following 240 variables.

Source

https://archive.ics.uci.edu/ml/datasets/Multiple+Features

Examples

```
data(mfeat_pix)
```

qkdbscan *qKernel-DBSCAN density reachability and connectivity clustering*

Description

Similiar to the Density-Based Spatial Clustering of Applications with Noise(or DBSCAN) algorithm, qKernel-DBSCAN is a density-based clustering algorithm that can be applied under both linear and non-linear situations.

Usage

```
## S4 method for signature 'matrix'
qkdbscan(x, kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
eps = 0.25, MinPts = 5, hybrid = TRUE, seeds = TRUE,  showplot  = FALSE,
countmode = NULL, na.action = na.omit, ...)

## S4 method for signature 'cndkernmatrix'
qkdbscan(x, eps = 0.25, MinPts = 5, seeds = TRUE,
showplot  = FALSE, countmode = NULL, ...)

## S4 method for signature 'qkernmatrix'
qkdbscan(x, eps = 0.25, MinPts = 5, seeds = TRUE,
showplot  = FALSE, countmode = NULL, ...)

## S4 method for signature 'qkdbscan'
predict(object, data, newdata = NULL, predict.max = 1000, ...)
```

Arguments

`x` the data matrix indexed by row, or a kernel matrix of `cndkernmatrix` or `qkernmatrix`.

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` Power qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function

- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` Power cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the Power qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `power, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- `power, sigma` for the ANOVA cndkernel function "anocnd".
- `c` for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- `theta` for the Wave cndkernel function "wavcnd".
- `power` for the Power cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

eps	reachability distance, see Ester et al. (1996). (default:0.25)
MinPts	reachability minimum number of points, see Ester et al.(1996).(default : 5)
hybrid	whether the algothrim expects raw data but calculates partial distance matrices, can be TRUE or FALSE
seeds	can be TRUE or FALSE, FALSE to not include the isseed-vector in the dbscan-object.
showplot	whether to show the plot or not, can be TRUE or FALSE
na.action	a function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
countmode	NULL or vector of point numbers at which to report progress.
object	object of class dbscan.
data	matrix or data.frame.
newdata	matrix or data.frame with raw data to predict.
predict.max	max. batch size for predictions.
...	Further arguments transferred to plot methods.

Details

The data can be passed to the qkdbscan function in a matrix, in addition qkdbscan also supports input in the form of a kernel matrix of class qkernmatrix or class cndkernmatrix.

Value

predict(qkdbscan-method) gives out a vector of predicted clusters for the points in newdata.

qkdbscan gives out an S4 object which is a LIST with components

clust	integer vector coding cluster membership with noise observations (singletons) coded as 0
eps	parameter eps
MinPts	parameter MinPts
kcall	the function call
cndkernf	the kernel function used
xmatrix	the original data matrix

all the slots of the object can be accessed by accessor functions.

Note

The predict function can be used to embed new data on the new space.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

Martin Ester, Hans-Peter Kriegel, Joerg Sander, Xiaowei Xu(1996).
A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise
Institute for Computer Science, University of Munich.
Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)

See Also

`qkernmatrix`, `cndkernmatrix`

Examples

```
# a simple example using the iris
data(iris)
test <- sample(1:150,20)
x<- as.matrix(iris[-test,-5])
ds <- qkdbscan (x,kernel="laplbase",qpar=list(sigma=3.5,q=0.8),eps=0.15,
MinPts=5,hybrid = FALSE)
plot(ds,x)
emb <- predict(ds, x, as.matrix(iris[test,-5]))
points(iris[test,], col= as.integer(1+emb))
```

`qkdbscan-class` *Class "qkdbscan"*

Description

The qkernel-DBSCAN class.

Objects of class "qkdbscan"

Objects can be created by calls of the form `new("qkdbscan", ...)`. or by calling the `qkdbscan` function.

Slots

- `clust`: Object of class `"vector"` containing the cluster membership of the samples
- `eps`: Object of class `"numeric"` containing the reachability distance
- `MinPts`: Object of class `"numeric"` containing the reachability minimum number of points
- `isseed`: Object of class `"logical"` containing the logical vector indicating whether a point is a seed (not border, not noise)

Methods

clust `signature(object = "qkdbscan")`: returns the cluster membership

kcall `signature(object = "qkdbscan")`: returns the performed call

cndkernf `signature(object = "qkdbscan")`: returns the used kernel function

eps `signature(object = "qkdbscan")`: returns the reachability distance

MinPts `signature(object = "qkdbscan")`: returns the reachability minimum number of points

predict `signature(object = "qkdbscan")`: embeds new data

xmatrix `signature(object = "qkdbscan")`: returns the used data matrix

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkernel-class, cndkernel-class

Examples

```
# a simple example using the iris data
x<- as.matrix(iris[,-5])
ds <- qkdbscan (x,kernel="laplbase",qpar=list(sigma=3.5,q=0.8),eps=0.15,
MinPts=5,hybrid = FALSE)
# print the results
clust(ds)
eps(ds)
MinPts(ds)
cndkernf(ds)
xmatrix(ds)
kcall(ds)
```

---

qkernel-class   *Class "qkernel" "rbfqkernel" "nonlqkernel" "laplqkernel" "ratiqkernel"*

Description

The built-in kernel classes in **qkerntool**

Objects from the Class

Objects can be created by calls of the form `new("rbfqkernel")`, `new{"nonlqkernel"}`, `new{"laplqkernel"}`, `new{"ratiqkernel"}`, `new{"multqkernel"}`, `new{"invqkernel"}`, `new{"wavqkernel"}`, `new{"powqkernel"}`, `new{"logqkernel"}`, `new{"cauqkernel"}`, `new{"chiqkernel"}`, `new{"studqkernel"}`

or by calling the `rbfbase`, `nonlbase`, `laplbase`, `ratibase`, `multbase`, `invbase`, `wavbase`, `powbase`, `logbase`, `caubase`, `chibase`, `studbase` functions etc..

Slots

`.Data`**:** Object of class `"function"` containing the kernel function

`qpar`**:** Object of class `"list"` containing the kernel parameters

Methods

qkernmatrix `signature(kernel = "rbfqkernel", x = "matrix")`: computes the qkernel matrix

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

`qkernmatrix`,`cndkernmatrix`

Examples

```
qkfunc <- rbfbase(sigma=1,q=0.8)
qkfunc

qpar(qkfunc)

## create two vectors
x <- rnorm(10)
y <- rnorm(10)

## calculate dot product
qkfunc(x,y)
```

`qkernmatrix` *qKernel Matrix functions*

Description

`qkernmatrix` calculates the qkernel matrix $K_{ij} = k(x_i, x_j)$ or $K_{ij} = k(x_i, y_j)$.

Usage

```
## S4 method for signature 'qkernel'
qkernmatrix(qkernel, x, y = NULL)
```

Arguments

`qkernel`	the kernel function to be used to calculate the qkernel matrix. This has to be a function of class `qkernel`, i.e. which can be generated either one of the build in kernel generating functions (e.g., `rbfbase` etc.) or a user defined function of class `qkernel` taking two vector arguments and returning a scalar.
`x`	a data matrix to be used to calculate the kernel matrix
`y`	second data matrix to calculate the kernel matrix

Details

Common functions used during kernel based computations.
The `qkernel` parameter can be set to any function, of class qkernel, which computes the kernel function value in feature space between two vector arguments. **qkerntool** provides more than 10 qkernel functions which can be initialized by using the following functions:

- `nonlbase` Non Linear qkernel function
- `rbfbase` Gaussian qkernel function
- `laplbase` Laplacian qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` d qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function

(see example.)

Value

`qkernmatrix` returns a conditionally negative definite matrix with a zero diagonal element.

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

See Also

nonlcnd, rbfcnd,polycnd,laplcnd, anocnd, raticnd, multcnd, invcnd, wavcnd, powcnd, logcnd, caucnd, chicnd, studcnd

Examples

```
data(iris)
dt <- as.matrix(iris[ ,-5])

## initialize kernel function
rbf <- rbfbase(sigma = 1.4, q=0.8)
rbf

## calculate qkernel matrix
qkernmatrix(rbf, dt)
```

---

qkgda *qKernel Generalized Discriminant Analysis*

Description

The qkernel Generalized Discriminant Analysis is a method that deals with nonlinear discriminant analysis using kernel function operator.

Usage

```
## S4 method for signature 'matrix'
qkgda(x, label, kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
          features = 0, th = 1e-4, na.action = na.omit, ...)

## S4 method for signature 'cndkernmatrix'
qkgda(x, label, features = 0, th = 1e-4, na.action = na.omit, ...)
## S4 method for signature 'qkernmatrix'
qkgda(x, label, features = 0, th = 1e-4, ...)
```

Arguments

x	the data matrix indexed by row, or a kernel matrix of cndkernmatrix or qkernmatrix.
label	The original labels of the samples.
kernel	the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings: • rbfbase Radial Basis qkernel function "Gaussian"

- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` Power qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` Power cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the Power qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `d, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".

- d, sigma for the ANOVA cndkernel function "anocnd".
- c for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- theta for the Wave cndkernel function "wavcnd".
- d for the Power cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

features	Number of features (principal components) to return. (default: 0 , all)
th	the value of the eigenvalue under which principal components are ignored (only valid when features = 0). (default : 0.0001)
na.action	A function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
...	additional parameters

Details

The qkernel Generalized Discriminant Analysis method provides a mapping of the input vectors into high dimensional feature space, generalizing the classical Linear Discriminant Analysis to non-linear discriminant analysis.
The data can be passed to the qkgda function in a matrix, in addition qkgda also supports input in the form of a kernel matrix of class qkernmatrix or class cndkernmatrix.

Value

An S4 object containing the eigenvectors and their normalized projections, along with the corresponding eigenvalues and the original function.

prj	The normalized projections on eigenvectors)
eVal	The corresponding eigenvalues
eVec	The corresponding eigenvectors
kcall	The formula of the function called
cndkernf	The kernel function used
xmatrix	The original data matrix

all the slots of the object can be accessed by accessor functions.

Note

The predict function can be used to embed new data on the new space

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

1.Baudat, G, and F. Anouar:
Generalized discriminant analysis using a kernel approach
Neural Computation 12.10(2000),2385
2.Deng Cai, Xiaofei He, and Jiawei Han:
Speed Up Kernel Discriminant Analysis
The VLDB Journal,January,2011,vol.20, no.1,21-33.

See Also

qkernmatrix, cndkernmatrix

Examples

```
Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]), Sp = rep(c("1","2","3"), rep(50,3)))
testset <- sample(1:150,20)
train <- as.matrix(iris[-testset,-5])
test <- as.matrix(iris[testset,-5])
Sp = rep(c("1","2","3"), rep(50,3))
labels <-as.numeric(Sp)
trainlabel <- labels[-testset]
testlabel <- labels[testset]

kgda1 <- qkgda(train, label=trainlabel, kernel = "ratibase", qpar = list(c=1,q=0.9),features = 2)

prj(kgda1)
eVal(kgda1)
eVec(kgda1)
kcall(kgda1)
# xmatrix(kgda1)

#print the principal component vectors
prj(kgda1)
#plot the data projection on the components
plot(kgda1@prj,col=as.integer(train), xlab="1st Principal Component",ylab="2nd Principal Component")
```

---

qkgda-class *Class "qkgda"*

Description

The qkernel Generalized Discriminant Analysis class

Objects of class "qkgda"

Objects can be created by calls of the form `new("qkgda", ...)`. or by calling the `qkgda` function.

Slots

- `prj`**:** Object of class `"matrix"` containing the normalized projections on eigenvectors
- `eVal`**:** Object of class `"matrix"` containing the corresponding eigenvalues
- `eVec`**:** Object of class `"matrix"` containing the corresponding eigenvectors
- `label`**:** Object of class `"matrix"` containing the categorical variables that the categorical data be assigned to one of the categories

Methods

- **prj** `signature(object = "qkgda")`: returns the normalized projections
- **eVal** `signature(object = "qkgda")`: returns the eigenvalues
- **eVec** `signature(object = "qkgda")`: returns the eigenvectors
- **kcall** `signature(object = "qkgda")`: returns the performed call
- **cndkernf** `signature(object = "qkgda")`: returns the used kernel function
- **predict** `signature(object = "qkgda")`: embeds new data
- **xmatrix** `signature(object = "qkgda")`: returns the used data matrix

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

See Also

`qkernel-class`, `cndkernel-class`

Examples

```
Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]), Sp = rep(c("1","2","3"), rep(50,3)))
testset <- sample(1:150,20)
train <- as.matrix(iris[-testset,-5])
test <- as.matrix(iris[testset,-5])
Sp = rep(c("1","2","3"), rep(50,3))
labels <-as.numeric(Sp)
trainlabel <- labels[-testset]
testlabel <- labels[testset]

kgda1 <- qkgda(train, label=trainlabel, kernel = "ratibase", qpar = list(c=1,q=0.9),features = 2)

prj(kgda1)
eVal(kgda1)
eVec(kgda1)
cndkernf(kgda1)
kcall(kgda1)
```

qkIsomap *qKernel Isometric Feature Mapping*

Description

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

Usage

```
## S4 method for signature 'matrix'
qkIsomap(x, kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
dims = 2, k, mod = FALSE, plotResiduals = FALSE, verbose = TRUE, na.action = na.omit, ...)

## S4 method for signature 'cndkernmatrix'
qkIsomap(x, dims = 2, k, mod = FALSE, plotResiduals = FALSE,
verbose = TRUE, na.action = na.omit, ...)

## S4 method for signature 'qkernmatrix'
qkIsomap(x, dims = 2, k, mod = FALSE, plotResiduals = FALSE,
verbose = TRUE, na.action = na.omit, ...)
```

Arguments

`x` N x D matrix (N samples, D features) or a kernel matrix of `cndkernmatrix` or `qkernmatrix`.

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` Power qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"

- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` Power cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the Power qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `d, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- `d, sigma` for the ANOVA cndkernel function "anocnd".
- `c` for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- `theta` for the Wave cndkernel function "wavcnd".
- `d` for the Power cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

`dims` vector containing the target space dimension(s)

`k` number of neighbours

`mod` use modified Isomap algorithm

`plotResiduals` show a plot with the residuals between the high and the low dimensional data

`verbose` show a summary of the embedding procedure at the end

`na.action`	A function to specify the action to be taken if `NA`s are found. The default action is `na.omit`, which leads to rejection of cases with missing values on any required variable. An alternative is `na.fail`, which causes an error if `NA` cases are found. (NOTE: If given, this argument must be named.)
`...`	additional parameters

Details

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

Value

`qkIsomap` gives out an S4 object which is a LIST with components

`prj`	a N x dim matrix (N samples, dim features) with the reduced input data (list of several matrices if more than one dimension was specified).
`dims`	the dimension of the target space.
`Residuals`	the residual variances for all dimensions.
`eVal`	the corresponding eigenvalues.
`eVec`	the corresponding eigenvectors.
`cndkernf`	the kernel function used.
`kcall`	The formula of the function called

all the slots of the object can be accessed by accessor functions.

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

References

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

Examples

```
 # another example using the iris
  data(iris)
  testset <- sample(1:150,20)
  train <- as.matrix(iris[-testset,-5])
  labeltrain<- as.integer(iris[-testset,5])
  test <- as.matrix(iris[testset,-5])
  # ratibase(c=1,q=0.8)
  d_low = qkIsomap(train, kernel = "ratibase", qpar = list(c=1,q=0.8),
                  dims=2,  k=5, plotResiduals = TRUE)
  #plot the data projection on the components
 plot(prj(d_low),col=labeltrain, xlab="1st Principal Component",ylab="2nd  Principal Component")

  prj(d_low)
dims(d_low)
Residuals(d_low)
eVal(d_low)
eVec(d_low)
kcall(d_low)
cndkernf(d_low)
```

---

qkIsomap-class *qKernel Isomap embedding*

Description

The qKernel Isometric Feature Mapping class

Objects of class "qkIsomap"

Objects can be created by calls of the form `new("qkIsomap", ...)`. or by calling the `qkIsomap` function.

Slots

`prj`: Object of class "`matrix`" containing the Nxdim matrix (N samples, dim features) with the reduced input data (list of several matrices if more than one dimension specified)

`dims`: Object of class "`numeric`" containing the dimension of the target space (default 2)

`connum`: Object of class "`numeric`" containing the number of connected components in graph

`Residuals`: Object of class "`vector`" containing the residual variances for all dimensions

`eVal`: Object of class "`vector`" containing the corresponding eigenvalues

`eVec`: Object of class "`vector`" containing the corresponding eigenvectors

Methods

prj `signature(object = "qkIsomap")`: returns the Nxdim matrix (N samples, dim features)

dims `signature(object = "qkIsomap")`: returns the dimension

Residuals `signature(object = "qkIsomap")`: returns the residual variances

eVal `signature(object = "qkIsomap")`: returns the eigenvalues

eVec `signature(object = "qkIsomap")`: returns the eigenvectors

xmatrix `signature(object = "qkIsomap")`: returns the used data matrix

kcall `signature(object = "qkIsomap")`: returns the performed call

cndkernf `signature(object = "qkIsomapa")`: returns the used kernel function

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkernel-class, cndkernel-class, qkIsomap

Examples

```
   # another example using the iris data
  data(iris)
  testset <- sample(1:150,20)
  train <- as.matrix(iris[-testset,-5])
  labeltrain<- as.integer(iris[-testset,5])
  test <- as.matrix(iris[testset,-5])
  # ratibase(c=1,q=0.8)
  d_low = qkIsomap(train, kernel = "ratibase", qpar = list(c=1,q=0.8),
                    dims=2,  k=5, plotResiduals = TRUE)
  #plot the data projection on the components
 plot(prj(d_low),col=labeltrain, xlab="1st Principal Component",ylab="2nd  Principal Component")

  prj(d_low)
dims(d_low)
Residuals(d_low)
eVal(d_low)
eVec(d_low)
kcall(d_low)
cndkernf(d_low)
```

qkLLE *qKernel Locally Linear Embedding*

Description

Computes the qkernel Locally Linear Embedding

Usage

```
## S4 method for signature 'matrix'
qkLLE(x, kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
                         dims = 2, k, na.action = na.omit, ...)
## S4 method for signature 'cndkernmatrix'
qkLLE(x, dims = 2, k, na.action = na.omit, ...)
## S4 method for signature 'qkernmatrix'
qkLLE(x, dims = 2, k, na.action = na.omit,...)
```

Arguments

`x` N x D matrix (N samples, D features) or a kernel matrix of `cndkernmatrix` or `qkernmatrix`.

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` Power qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function

- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` Power cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the Power qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `power, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- `power, sigma` for the ANOVA cndkernel function "anocnd".
- `c` for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- `theta` for the Wave cndkernel function "wavcnd".
- `power` for the Power cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

`dims` dimension of the target space

`k` the number of nearest neighbours.

`na.action` A function to specify the action to be taken if `NA`s are found. The default action is `na.omit`, which leads to rejection of cases with missing values on any required variable. An alternative is `na.fail`, which causes an error if `NA` cases are found. (NOTE: If given, this argument must be named.)

`...` additional parameters

Details

The qkernel Locally Linear Embedding (qkLLE) preserves local properties of the data by representing each sample in the data by a linear combination of its k nearest neighbours with each neighbour weighted independently. qkLLE finally chooses the low-dimensional representation that best preserves the weights in the target space. It is an extension of Locally Linear Embedding (LLE) with qkernel method.

Value

It returns an S4 object containing the principal component vectors along with the corresponding eigenvalues.

`prj`	a matrix with the reduced input data
`dims`	dimension of the target space
`eVal`	The corresponding eigenvalues
`eVec`	The corresponding eigenvectors
`cndkernf`	the kernel function used

all the slots of the object can be accessed by accessor functions.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

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

Examples

```
## S4 method for signature 'matrix'
data(iris)
testset <- sample(1:150,20)
train <- as.matrix(iris[-testset,-5])
labeltrain<- as.integer(iris[-testset,5])
test <- as.matrix(iris[testset,-5])
plot(train ,col=labeltrain, xlab="1st Principal Component",ylab="2nd Principal Component")
# ratibase(c=1,q=0.8)
d_low <- qkLLE(train, kernel = "ratibase", qpar = list(c=1,q=0.8), dims=2, k=5)
#plot the data projection on the components
plot(prj(d_low),col=labeltrain, xlab="1st Principal Component",ylab="2nd Principal Component")

## S4 method for signature 'qkernmatrix'
# ratibase(c=0.1,q=0.8)
qkfunc <- ratibase(c=0.1,q=0.8)
ktrain1 <- qkernmatrix(qkfunc,train)
d_low <- qkLLE(ktrain1, dims = 2, k=5)
```

```
#plot the data projection on the components
plot(prj(d_low),col=labeltrain, xlab="1st Principal Component",ylab="2nd Principal Component")
```

---

qkLLE-class *Class "qkLLE"*

Description

The qKernel Locally Linear Embedding class

Objects of class "qkLLE"

Objects can be created by calls of the form `new("qkLLE", ...)`. or by calling the `qkLLE` function.

Slots

`prj`: Object of class `"matrix"` containing the reduced input data

`dims`: Object of class `"numeric"` containing the dimension of the target space (default 2)

`eVal`: Object of class `"vector"` containing the corresponding eigenvalues

`eVec`: Object of class `"matrix"` containing the corresponding eigenvectors

Methods

prj `signature(object = "qkLLE")`: returns the reduced input data

dims `signature(object = "qkLLE")`: returns the dimension

eVal `signature(object = "qkLLE")`: returns the eigenvalues

eVec `signature(object = "qkLLE")`: returns the eigenvectors

xmatrix `signature(object = "qkLLE")`: returns the used data matrix

kcall `signature(object = "qkLLE")`: returns the performed call

cndkernf `signature(object = "qkLLE")`: returns the used kernel function

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkernel-class, cndkernel-class

Examples

```
  ## S4 method for signature 'matrix'
data(iris)
testset <- sample(1:150,20)
train <- as.matrix(iris[-testset,-5])
labeltrain<- as.integer(iris[-testset,5])
test <- as.matrix(iris[testset,-5])
plot(train ,col=labeltrain, xlab="1st Principal Component",ylab="2nd Principal Component")
# ratibase(c=1,q=0.8)
d_low <- qkLLE(train, kernel = "ratibase", qpar = list(c=1,q=0.8), dims=2, k=5)
#plot the data projection on the components
plot(prj(d_low),col=labeltrain,xlab="1st Principal Component",ylab="2nd Principal Component")

## S4 method for signature 'qkernmatrix'
# ratibase(c=0.1,q=0.8)
qkfunc <- ratibase(c=0.1,q=0.8)
ktrain1 <- qkernmatrix(qkfunc,train)
d_low <- qkLLE(ktrain1, dims = 2, k=5)
#plot the data projection on the components
plot(prj(d_low),col=labeltrain,xlab="1st Principal Component",ylab="2nd Principal Component")
```

---

qkMDS *qKernel Metric Multi-Dimensional Scaling*

Description

The qkernel Metric Multi-Dimensional Scaling is a nonlinear form of Metric Multi-Dimensional Scaling

Usage

```
## S4 method for signature 'matrix'
qkMDS(x, kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
dims = 2, plotResiduals = FALSE, verbose = TRUE, na.action = na.omit, ...)

## S4 method for signature 'cndkernmatrix'
qkMDS(x, dims = 2,plotResiduals = FALSE,
verbose = TRUE, na.action = na.omit, ...)

## S4 method for signature 'qkernmatrix'
qkMDS(x, dims = 2,plotResiduals = FALSE,
verbose = TRUE, na.action = na.omit, ...)
```

Arguments

x	N x D matrix (N samples, D features) or a kernel matrix of `cndkernmatrix` or `qkernmatrix`.

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` Power qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` Power cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".

- d, q for the Power qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- alpha for the Non Linear cndkernel function "nonlcnd".
- d, alpha, c for the Polynomial cndkernel function "polycnd".
- gamma for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- d, sigma for the ANOVA cndkernel function "anocnd".
- c for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- theta for the Wave cndkernel function "wavcnd".
- d for the Power cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

dims	vector containing the target space dimension(s)
plotResiduals	show a plot with the residuals between the high and the low dimensional data
verbose	show a summary of the embedding procedure at the end
na.action	A function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
...	additional parameters

Details

There are several versions of non-metric multidimensional scaling in R, but **qkerntool** offers the following unique combination of using qKernel methods

Value

qkMDS gives out an S4 object which is a LIST with components

prj	a N x dim matrix (N samples, dim features) with the reduced input data (list of several matrices if more than one dimension was specified).
dims	the dimension of the target space.
Residuals	the residual variances for all dimensions.
eVal	the corresponding eigenvalues.
eVec	the corresponding eigenvectors.
cndkernf	the kernel function used.
kcall	The formula of the function called

all the slots of the object can be accessed by accessor functions.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

Kruskal, J.B. 1964a. Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *Psychometrika* 29, 1–28.

Examples

```
 # another example using the iris
  data(iris)
  testset <- sample(1:150,20)
  train <- as.matrix(iris[-testset,-5])
  labeltrain<- as.integer(iris[-testset,5])
  test <- as.matrix(iris[testset,-5])
  # ratibase(c=1,q=0.8)
  d_low = qkMDS(train, kernel = "ratibase", qpar = list(c=1,q=0.9),dims = 2,
                plotResiduals = TRUE)
  #plot the data projection on the components
 plot(prj(d_low),col=labeltrain, xlab="1st Principal Component",ylab="2nd Principal Component")

  prj(d_low)
dims(d_low)
Residuals(d_low)
eVal(d_low)
eVec(d_low)
kcall(d_low)
cndkernf(d_low)
```

---

qkMDS-class *qKernel Metric Multi-Dimensional Scaling*

Description

The qkernel Metric Multi-Dimensional Scaling class

Objects of class "qkMDS"

Objects can be created by calls of the form `new("qkMDS", ...)`. or by calling the `qkMDS` function.

Slots

`prj`: Object of class `"matrix"` containing the Nxdim matrix (N samples, dim features) with the reduced input data (list of several matrices if more than one dimension specified)

`dims`: Object of class `"numeric"` containing the dimension of the target space (default 2)

`connum`: Object of class `"numeric"` containing the number of connected components in graph

`Residuals`: Object of class "vector" containing the residual variances for all dimensions

`eVal`: Object of class "vector" containing the corresponding eigenvalues

`eVec`: Object of class "vector" containing the corresponding eigenvectors

Methods

prj `signature(object = "qkMDS")`: returns the Nxdim matrix (N samples, dim features)

dims `signature(object = "qkMDS")`: returns the dimension

Residuals `signature(object = "qkMDS")`: returns the residual variances

eVal `signature(object = "qkMDS")`: returns the eigenvalues

eVec `signature(object = "qkMDS")`: returns the eigenvectors

xmatrix `signature(object = "qkMDS")`: returns the used data matrix

kcall `signature(object = "qkMDS")`: returns the performed call

cndkernf `signature(object = "qkMDS")`: returns the used kernel function

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkernel-class, cndkernel-class, qkMDS

Examples

```
   # another example using the iris
  data(iris)
  testset <- sample(1:150,20)
  train <- as.matrix(iris[-testset,-5])
  labeltrain<- as.integer(iris[-testset,5])
  test <- as.matrix(iris[testset,-5])
  # ratibase(c=1,q=0.8)
  d_low = qkMDS(train, kernel = "ratibase", qpar = list(c=1,q=0.8),
                  dims=2, plotResiduals = TRUE)
  #plot the data projection on the components
 plot(prj(d_low),col=labeltrain, xlab="1st Principal Component",ylab="2nd  Principal Component")

  prj(d_low)
dims(d_low)
Residuals(d_low)
eVal(d_low)
eVec(d_low)
kcall(d_low)
cndkernf(d_low)
```

qkpca *qKernel Principal Components Analysis*

Description

The qkernel Principal Components Analysis is a nonlinear form of principal component analysis.

Usage

```
## S4 method for signature 'formula'
qkpca(x, data = NULL, na.action, ...)
## S4 method for signature 'matrix'
qkpca(x, kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
                        features = 0, th = 1e-4, na.action = na.omit, ...)
## S4 method for signature 'cndkernmatrix'
qkpca(x, features = 0, th = 1e-4, ...)
## S4 method for signature 'qkernmatrix'
qkpca(x, features = 0, th = 1e-4, ...)
```

Arguments

`x` the data matrix indexed by row, a formula describing the model or a kernel matrix of `cndkernmatrix` or `qkernmatrix`.

`data` an optional data frame containing the variables in the model (when using a formula).

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` d qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"

- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` power cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the d qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `d, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- `d, sigma` for the ANOVA cndkernel function "anocnd".
- `c` for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- `theta` for the Wave cndkernel function "wavcnd".
- `d` for the power cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

`features` Number of features (principal components) to return. (default: 0 , all)

`th` the value of the eigenvalue under which principal components are ignored (only valid when features = 0). (default : 0.0001)

`na.action` A function to specify the action to be taken if `NA`s are found. The default action is `na.omit`, which leads to rejection of cases with missing values on any required variable. An alternative is `na.fail`, which causes an error if `NA` cases are found. (NOTE: If given, this argument must be named.)

...	additional parameters

Details

Using kernel functions one can efficiently compute principal components in high-dimensional feature spaces, related to input space by some non-linear map.
The data can be passed to the `qkpca` function in a `matrix`, in addition `qkpca` also supports input in the form of a kernel matrix of class `qkernmatrix` or class `cndkernmatrix`.

Value

An S4 object containing the principal component vectors along with the corresponding eigenvalues.

`pcv`	a matrix containing the principal component vectors (column wise)
`eVal`	The corresponding eigenvalues
`rotated`	The original data projected (rotated) on the principal components
`cndkernf`	the kernel function used
`xmatrix`	The original data matrix

all the slots of the object can be accessed by accessor functions.

Note

The predict function can be used to embed new data on the new space

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

References

Schoelkopf B., A. Smola, K.-R. Mueller :
Nonlinear component analysis as a kernel eigenvalue problem
Neural Computation 10, 1299-1319
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.1366

See Also

`qkernmatrix`, `cndkernmatrix`

Examples

```
# another example using the iris data
data(iris)
test <- sample(1:150,20)
qkpc <- qkpca(~.,data=iris[-test,-5],kernel="rbfbase",
              qpar=list(sigma=50,q=0.8),features=2)

# print the principal component vectors
```

```
pcv(qkpc)
#plot the data projection on the components
plot(rotated(qkpc),col=as.integer(iris[-test,5]),
     xlab="1st Principal Component",ylab="2nd Principal Component")

# embed remaining points
emb <- predict(qkpc,iris[test,-5])
points(emb,col=as.integer(iris[test,5]))
```

---

qkpca-class *Class "qkpca"*

Description

The qkernel Principal Components Analysis class

Objects of class "qkpca"

Objects can be created by calls of the form `new("qkpca", ...)`. or by calling the `qkpca` function.

Slots

`pcv`**:** Object of class `"matrix"` containing the principal component vectors

`eVal`**:** Object of class `"vector"` containing the corresponding eigenvalues

`rotated`**:** Object of class `"matrix"` containing the projection of the data on the principal components

Methods

eVal `signature(object = "qkpca")`: returns the eigenvalues

pcv `signature(object = "qkpca")`: returns the principal component vectors

predict `signature(object = "qkpca")`: embeds new data

rotated `signature(object = "qkpca")`: returns the projected data

xmatrix `signature(object = "qkpca")`: returns the used data matrix

kcall `signature(object = "qkpca")`: returns the performed call

cndkernf `signature(object = "qkpca")`: returns the used kernel function

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

See Also

qkernel-class, cndkernel-class

Examples

```
# another example using the iris data
data(iris)
test <- sample(1:150,20)
qkpc <- qkpca(~.,iris[-test,-5], kernel = "rbfbase",
              qpar = list(sigma = 50, q = 0.8), features = 2)

# print the principal component vectors
pcv(qkpc)
rotated(qkpc)
cndkernf(qkpc)
eVal(qkpc)
xmatrix(qkpc)
names(eVal(qkpc))
```

---

qkprc-class *Class "qkprc"*

Description

The qKernel Prehead class

Objects of class "qkprc"

Objects from the class cannot be created directly but only contained in other classes.

Slots

`cndkernf`: Object of class "`kfunction`" containing the kernel function used

`qpar`: Object of class "`list`" containing the kernel parameters used

`xmatrix`: Object of class "`input`" containing the data matrix used

`ymatrix`: Object of class "`input`" containing the data matrix used

`kcall`: Object of class "`ANY`" containing the function call

`terms`: Object of class "`ANY`" containing the function terms

`n.action`: Object of class "`ANY`" containing the action performed on NA

Methods

cndkernf `signature(object = "qkprc")`: returns the used kernel function

xmatrix `signature(object = "qkprc")`: returns the used data matrix

ymatrix `signature(object = "qkprc")`: returns the used data matrix

kcall `signature(object = "qkprc")`: returns the performed call

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkernel-class, cndkernel-class

---

qkspecc *qkernel spectral Clustering*

Description

A qkernel spectral clustering algorithm. Clustering is performed by embedding the data into the subspace of the eigenvectors of a graph Laplacian matrix.

Usage

```
## S4 method for signature 'matrix'
qkspecc(x,kernel = "rbfbase", qpar = list(sigma = 2, q = 0.9),
          Nocent=NA, normalize="symmetric", maxk=20, iterations=200,
          na.action = na.omit, ...)

## S4 method for signature 'cndkernmatrix'
qkspecc(x, Nocent=NA, normalize="symmetric",
          maxk=20,iterations=200, ...)

## S4 method for signature 'qkernmatrix'
qkspecc(x, Nocent=NA, normalize="symmetric",
          maxk=20,iterations=200, ...)
```

Arguments

x	the matrix of data to be clustered or a kernel Matrix of class `qkernmatrix` or `cndkernmatrix`.

kernel	the kernel function used in computing the affinity matrix. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. kernlab provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function
- `wavbase` Wave qkernel function
- `powbase` d qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function

- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` d cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` a character string or the list of hyper-parameters (kernel parameters). The default character string `list(sigma = 2, q = 0.9)` uses a heuristic to determine a suitable value for the width parameter of the RBF kernel. The second option `"local"` (local scaling) uses a more advanced heuristic and sets a width parameter for every point in the data set. This is particularly useful when the data incorporates multiple scales. A list can also be used containing the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the d qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `d, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- `d, sigma` for the ANOVA cndkernel function "anocnd".
- `c` for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- `theta` for the Wave cndkernel function "wavcnd".

- d for the d cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd". where length is the length of the strings considered, lambda the decay factor and normalized a logical parameter determining if the kernel evaluations should be normalized.

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

Nocent	the number of clusters.
normalize	Normalisation of the Laplacian ("none", "symmetric" or "random-walk").
maxk	If k is NA, an upper bound for the automatic estimation. Defaults to 20.
iterations	the maximum number of iterations allowed.
na.action	the action to perform on NA.
...	additional parameters.

Details

The qkernel spectral clustering works by embedding the data points of the partitioning problem into the subspace of the eigenvectors corresponding to the k smallest eigenvalues of the graph Laplacian matrix. Using a simple clustering method like kmeans on the embedded points usually leads to good performance. It can be shown that qkernel spectral clustering methods boil down to graph partitioning.
The data can be passed to the qkspecc function in a matrix, in addition qkspecc also supports input in the form of a kernel matrix of class qkernmatrix or cndkernmatrix.

Value

An S4 object of class qkspecc which extends the class vector containing integers indicating the cluster to which each point is allocated. The following slots contain useful information

clust	The cluster assignments
eVec	The corresponding eigenvector
eVal	The corresponding eigenvalues
ymatrix	The eigenvectors corresponding to the k smallest eigenvalues of the graph Laplacian matrix.

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

Andrew Y. Ng, Michael I. Jordan, Yair Weiss
On Spectral Clustering: Analysis and an Algorithm
Neural Information Processing Symposium 2001

See Also

qkernmatrix, cndkernmatrix, qkpca

Examples

```
data("iris")
x=as.matrix(iris[,-5])

qspe <- qkspecc(x,kernel = "rbfbase", qpar = list(sigma = 10, q = 0.9),
                Nocent=3, normalize="symmetric", maxk=15, iterations=1200)
plot(x, col = clust(qspe))

qkfunc <- nonlbase(alpha=1/15,q=0.8)
Ktrain <- qkernmatrix(qkfunc, x)
qspe <- qkspecc(Ktrain, Nocent=3, normalize="symmetric", maxk=20)
plot(x, col = clust(qspe))
```

---

qkspecc-class   *Class "qkspecc"*

Description

The qKernel Spectral Clustering Class

Objects from the Class

Objects can be created by calls of the form `new("qkspecc", ...)`. or by calling the function `qkspecc`.

Slots

`clust`**:** Object of class `"vector"` containing the cluster assignments

`eVec`**:** Object of class `"matrix"` containing the corresponding eigenvector in each cluster

`eVal`**:** Object of class `"vector"` containing the corresponding eigenvalue for each cluster

`withinss`**:** Object of class `"vector"` containing the within-cluster sum of squares for each cluster

Methods

clust `signature(object = "qkspecc")`: returns the cluster assignments

eVec `signature(object = "qkspecc")`: returns the corresponding eigenvector in each cluster

eVal `signature(object = "qkspecc")`: returns the corresponding eigenvalue for each cluster

xmatrix `signature(object = "qkspecc")`: returns the original data matrix or a kernel Matrix

ymatrix `signature(object = "qkspecc")`: returns The eigenvectors corresponding to the k smallest eigenvalues of the graph Laplacian matrix.

cndkernf `signature(object = "qkspecc")`: returns the used kernel function

kcall `signature(object = "qkspecc")`: returns the performed call

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

qkspecc, qkernel-class, cndkernel-class

Examples

```
## Cluster the iris data set.
data("iris")
x=as.matrix(iris[,-5])

qspe <- qkspecc(x,kernel = "rbfbase", qpar = list(sigma = 10, q = 0.9),
                Nocent=3, normalize="symmetric", maxk=15, iterations=1200)
clust(qspe)
eVec(qspe)
eVal(qspe)
xmatrix(qspe)
ymatrix(qspe)
cndkernf(qspe)
```

---

qkspeclust — *qkernel spectral Clustering*

Description

This is also a qkernel spectral clustering algorithm which uses three ways to assign labels after the laplacian embedding: kmeans, hclust and dbscan.

Usage

```
## S4 method for signature 'qkspecc'
qkspeclust(x, clustmethod = "kmeans",
         Nocent=NULL,iterations=NULL, hmethod=NULL,eps = NULL, MinPts = NULL)
```

Arguments

x	object of class qkspecc.
clustmethod	the strategy to use to assign labels in the embedding space. There are three ways to assign labels after the laplacian embedding: kmeans, hclust and dbscan.
Nocent	the number of clusters
iterations	the maximum number of iterations allowed for "kmeans".
hmethod	the agglomeration method for "hclust". This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

eps	Reachability distance for "dbscan".
MinPts	Reachability minimum no. of points for "dbscan".

Details

The qkernel spectral clustering works by embedding the data points of the partitioning problem into the subspace of the eigenvectors corresponding to the k smallest eigenvalues of the graph Laplacian matrix. Using the simple clustering methods like kmeans, hclust and dbscan on the embedded points usually leads to good performance. It can be shown that qkernel spectral clustering methods boil down to graph partitioning.

Value

An S4 object of class qkspecc which extends the class vector containing integers indicating the cluster to which each point is allocated. The following slots contain useful information

clust	The cluster assignments
eVec	The corresponding eigenvector
eVal	The corresponding eigenvalues
xmatrix	The original data matrix
ymatrix	The real valued matrix of eigenvectors corresponding to the k smallest eigenvalues of the graph Laplacian matrix
cndkernf	The kernel function used

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

Andrew Y. Ng, Michael I. Jordan, Yair Weiss
On Spectral Clustering: Analysis and an Algorithm
Neural Information Processing Symposium 2001

See Also

qkernmatrix, cndkernmatrix, qkspecc-class, qkspecc

Examples

```
data("iris")
x=as.matrix(iris[ ,-5])

qspe <- qkspecc(x,kernel = "rbfbase", qpar = list(sigma = 90, q = 0.9),
               Nocent=3, normalize="symmetric", maxk=15,iterations=1200)
plot(x, col = clust(qspe))
```

```
qspec <- qkspeclust(qspe,clustmethod = "hclust", Nocent=3, hmethod="ward.D2")
plot(x, col = clust(qspec))
plot(qspec)
```

---

qsammon *qKernel Sammon Mapping*

Description

The qkernel Sammon Mapping 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. qsammon is a nonlinear form of Sammon Mapping.

Usage

```
## S4 method for signature 'matrix'
qsammon(x, kernel = "rbfbase", qpar = list(sigma = 0.5, q = 0.9),
          dims = 2, Initialisation = 'random', MaxHalves = 20,
          MaxIter = 500, TolFun = 1e-7, na.action = na.omit, ...)

## S4 method for signature 'cndkernmatrix'
qsammon(cndkernel, x, k, dims = 2, Initialisation = 'random',
          MaxHalves = 20,MaxIter = 500, TolFun = 1e-7, ...)

## S4 method for signature 'qkernmatrix'
qsammon(qkernel, x, k, dims = 2, Initialisation = 'random',
          MaxHalves = 20, MaxIter = 500, TolFun = 1e-7, ...)
```

Arguments

`x` the data matrix indexed by row or a kernel matrix of `cndkernmatrix` or `qkernmatrix`.

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. qkerntool provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function
- `multbase` Multiquadric qkernel function
- `invbase` Inverse Multiquadric qkernel function

- `wavbase` Wave qkernel function
- `powbase` d qkernel function
- `logbase` Log qkernel function
- `caubase` Cauchy qkernel function
- `chibase` Chi-Square qkernel function
- `studbase` Generalized T-Student qkernel function
- `nonlcnd` Non Linear cndkernel function
- `polycnd` Polynomial cndkernel function
- `rbfcnd` Radial Basis cndkernel function "Gaussian"
- `laplcnd` Laplacian cndkernel function
- `anocnd` ANOVA cndkernel function
- `raticnd` Rational Quadratic cndkernel function
- `multcnd` Multiquadric cndkernel function
- `invcnd` Inverse Multiquadric cndkernel function
- `wavcnd` Wave cndkernel function
- `powcnd` d cndkernel function
- `logcnd` Log cndkernel function
- `caucnd` Cauchy cndkernel function
- `chicnd` Chi-Square cndkernel function
- `studcnd` Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`qpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- `sigma, q` for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" and the Cauchy qkernel function "caubase".
- `alpha, q` for the Non Linear qkernel function "nonlbase".
- `c, q` for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase" and the Inverse Multiquadric qkernel function "invbase".
- `theta, q` for the Wave qkernel function "wavbase".
- `d, q` for the d qkernel function "powbase" , the Log qkernel function "logbase" and the Generalized T-Student qkernel function "studbase".
- `alpha` for the Non Linear cndkernel function "nonlcnd".
- `d, alpha, c` for the Polynomial cndkernel function "polycnd".
- `gamma` for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- `d, sigma` for the ANOVA cndkernel function "anocnd".
- `c` for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".
- `theta` for the Wave cndkernel function "wavcnd".

- d for the d cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".

Hyper-parameters for user defined kernels can be passed through the qpar parameter as well.

`qkernel`	the kernel function to be used to calculate the qkernel matrix.
`cndkernel`	the cndkernel function to be used to calculate the CND kernel matrix.
`k`	the dimension of the original data.
`dims`	Number of features to return. (default: 2)
`Initialisation`	`"random"` or `"pca"`; the former performs fast random projection and the latter performs standard PCA (default : `"random"`)
`MaxHalves`	maximum number of step halvings. (default : 20)
`MaxIter`	the maximum number of iterations allowed. (default : 500)
`TolFun`	relative tolerance on objective function. (default : 1e-7)
`na.action`	A function to specify the action to be taken if `NA`s are found. The default action is `na.omit`, which leads to rejection of cases with missing values on any required variable. An alternative is `na.fail`, which causes an error if `NA` cases are found. (NOTE: If given, this argument must be named.)
`...`	additional parameters

Details

Using kernel functions one can efficiently compute principal components in high-dimensional feature spaces, related to input space by some non-linear map.
The data can be passed to the `qsammon` function in a `matrix`, in addition `qsammon` also supports input in the form of a kernel matrix of class `qkernmatrix` or class `cndkernmatrix`.

Value

`dimRed`	The matrix whose rows are embedded observations.
`kcall`	The function call contained
`cndkernf`	The kernel function used

all the slots of the object can be accessed by accessor functions.

Author(s)

Yusen Zhang
`<yusenzhang@126.com>`

References

Sammon, J.W. (1969) *A Nonlinear Mapping for Data Structure Analysis*. IEEE Transactions on Computers, C-18 5:401-409.

See Also

`qkernmatrix`, `cndkernmatrix`

Examples

```
data(iris)
train <- as.matrix(iris[,1:4])
labeltrain<- as.integer(iris[,5])
## S4 method for signature 'matrix'
kpc2 <- qsammon(train, kernel = "rbfbase", qpar = list(sigma = 2, q = 0.9), dims = 2,
                Initialisation = 'pca', TolFun = 1e-5)
plot(dimRed(kpc2), col = as.integer(labeltrain))
cndkernf(kpc2)
```

---

qsammon-class *Class "qsammon"*

Description

The qKernel Sammon Mapping class

Objects of class "qsammon"

Objects can be created by calls of the form `new("qsammon", ...)`. or by calling the `qsammon` function.

Slots

`dimRed`**:** Object of class `"matrix"` containing the matrix whose rows are embedded observations

`cndkernf`**:** Object of class `"function"` containing the kernel function used

`kcall`**:** Object of class `"ANY"` containing the function call

Methods

dimRed `signature(object = "qsammon")`: returns the matrix whose rows are embedded observations

kcall `signature(object = "qsammon")`: returns the performed call

cndkernf `signature(object = "qsammon")`: returns the used kernel function

Author(s)

Yusen Zhang
<yusenzhang@126.com>

See Also

`qsammon`

Examples

```
data(iris)
train <- as.matrix(iris[,1:4])
labeltrain<- as.integer(iris[,5])
## S4 method for signature 'matrix'
qkpc <- qsammon(train, kernel = "rbfbase", qpar = list(sigma = 0.5, q = 0.9),
                dims = 2, Initialisation = 'pca', MaxHalves = 50)

cndkernf(qkpc)
dimRed(qkpc)
kcall(qkpc)
```

---

qtSNE *qKernel t-Distributed Stochastic Neighbor Embedding*

Description

Wrapper for the qkernel t-distributed stochastic neighbor embeddingg. qtSNE is a method for constructing a low dimensional embedding of high-dimensional data, distances or similarities.

Usage

```
## S4 method for signature 'matrix'
qtSNE(x,kernel = "rbfbase", qpar = list(sigma = 0.1, q = 0.9),
      initial_config = NULL, no_dims=2, initial_dims=30, perplexity=30, max_iter= 1300,
          min_cost=0, epoch_callback=NULL, epoch=100, na.action = na.omit, ...)
## S4 method for signature 'cndkernmatrix'
qtSNE(x,initial_config = NULL, no_dims=2, initial_dims=30,
       perplexity=30, max_iter = 1000, min_cost=0, epoch_callback=NULL,epoch=100)
## S4 method for signature 'qkernmatrix'
qtSNE(x,initial_config = NULL, no_dims=2, initial_dims=30,
       perplexity=30, max_iter = 1000, min_cost=0, epoch_callback=NULL,epoch=100)
```

Arguments

`x` the matrix of data to be clustered or a kernel Matrix of class `qkernmatrix` or `cndkernmatrix`.

`kernel` the kernel function used in computing the affinity matrix. This parameter can be set to any function, of class kernel, which computes a kernel function value between two vector arguments. kernlab provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfbase` Radial Basis qkernel function "Gaussian"
- `nonlbase` Non Linear qkernel function
- `laplbase` Laplbase qkernel function
- `ratibase` Rational Quadratic qkernel function

- multbase Multiquadric qkernel function
- invbase Inverse Multiquadric qkernel function
- wavbase Wave qkernel function
- powbase Power qkernel function
- logbase Log qkernel function
- caubase Cauchy qkernel function
- chibase Chi-Square qkernel function
- studbase Generalized T-Student qkernel function
- nonlcnd Non Linear cndkernel function
- polycnd Polynomial cndkernel function
- rbfcnd Radial Basis cndkernel function "Gaussian"
- laplcnd Laplacian cndkernel function
- anocnd ANOVA cndkernel function
- raticnd Rational Quadratic cndkernel function
- multcnd Multiquadric cndkernel function
- invcnd Inverse Multiquadric cndkernel function
- wavcnd Wave cndkernel function
- powcnd Power cndkernel function
- logcnd Log cndkernel function
- caucnd Cauchy cndkernel function
- chicnd Chi-Square cndkernel function
- studcnd Generalized T-Student cndkernel function

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

qpar a character string or the list of hyper-parameters (kernel parameters). The default character string list(sigma = 2, q = 0.9) uses a heuristic to determine a suitable value for the width parameter of the RBF kernel. The second option "local" (local scaling) uses a more advanced heuristic and sets a width parameter for every point in the data set. This is particularly useful when the data incorporates multiple scales. A list can also be used containing the parameters to be used with the kernel function. Valid parameters for existing kernels are :

- sigma for the Radial Basis qkernel function "rbfbase" , the Laplacian qkernel function "laplbase" the Cauchy qkernel function "caubase" and for the ANOVA cndkernel function "anocnd".
- alpha for the Non Linear qkernel function "nonlbase",for the Non Linear cndkernel function "nonlcnd",and for the Polynomial cndkernel function "polycnd".
- c for the Rational Quadratic qkernel function "ratibase" , the Multiquadric qkernel function "multbase", the Inverse Multiquadric qkernel function "invbase",for the Polynomial cndkernel function "polycnd",for the Rational Quadratic cndkernel function "raticnd" , the Multiquadric cndkernel function "multcnd" and the Inverse Multiquadric cndkernel function "invcnd".

- d for qkernel function "powbase" , the Log qkernel function "logbase", the Generalized T-Student qkernel function "studbase", for the Polynomial cndkernel function "polycnd", for the ANOVA cndkernel function "anocnd",for the d cndkernel function "powcnd" , the Log cndkernel function "logcnd" and the Generalized T-Student cndkernel function "studcnd".
- theta for the Wave qkernel function "wavbase" and for the Wave cndkernel function "wavcnd".
- gamma for the Chi-Square qkernel function "chibase",for the Radial Basis cndkernel function "rbfcnd" and the Laplacian cndkernel function "laplcnd" and the Cauchy cndkernel function "caucnd".
- q For all qkernel Function. where length is the length of the strings considered, lambda the decay factor and normalized a logical parameter determining if the kernel evaluations should be normalized.

Hyper-parameters for user defined kernels can be passed through the qkpar parameter as well.

initial_config	An intitial configure about x (default: NULL)
no_dims	the dimension of the resulting embedding. (default: 2)
initial_dims	The number of dimensions to use in reduction method. (default: 30)
perplexity	Perplexity parameter
max_iter	Number of iterations (default: 1300)
min_cost	The minimum cost for every object after the final iteration
epoch_callback	A callback function used after each epoch (an epoch here means a set number of iterations)
epoch	The interval of the number of iterations displayed (default: 100)
na.action	the action to perform on NA
...	Other arguments that can be passed to qtSNE

Details

When the initial_config argument is specified, the algorithm will automatically enter the final momentum stage. This stage has less large scale adjustment to the embedding, and is intended for small scale tweaking of positioning. This can greatly speed up the generation of embeddings for various similar X datasets, while also preserving overall embedding orientation.

Value

qtSNE gives out an S4 object which is a LIST with components

dimRed	Matrix containing the new representations for the objects after qtSNE
cndkernf	The kernel function used

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

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

van der Maaten, L.J.P. & Hinton, G.E., 2008. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research, 9, pp.2579-2605.

Examples

```
## Not run:
#use iris data set
data(iris)
testset <- sample(1:150,20)
train <- as.matrix(iris[,1:4])

colors = rainbow(length(unique(iris$Species)))
names(colors) = unique(iris$Species)
#for matrix
ecb = function(x,y){
  plot(x,t='n');
  text(x,labels=iris$Species, col=colors[iris$Species])
}
kpc2 <- qtSNE(train, kernel = "rbfbase", qpar = list(sigma=1,q=0.8),
              epoch_callback = ecb, perplexity=10, max_iter = 500)


## End(Not run)
```

---

qtSNE-class *Class "qtSNE"*

Description

An S4 Class for qtSNE.

Details

The qtSNE is a method that uses Qkernel t-Distributed Stochastic Neighborhood Embedding between the distance matrices in high and low-dimensional space to embed the data. The method is very well suited to visualize complex structures in low dimensions.

Objects from the Class

Objects can be created by calls of the form `new("qtSNE", ...)`. or by calling the function `qtSNE`.

Slots

`dimRed` Matrix containing the new representations for the objects after qtSNE

`cndkernf` The kernel function used

Method

`dimRed signature(object="qtSNE")`: return a new representation matrix

`cndkernf signature(object="qtSNE")`: return the kernel used

Author(s)

Yusen Zhang
<yusenzhang@126.com>

References

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

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

See Also

qtSNE

Examples

```
## Not run:
#use iris data set
data(iris)
testset <- sample(1:150,20)
train <- as.matrix(iris[,1:4])

colors = rainbow(length(unique(iris$Species)))
names(colors) = unique(iris$Species)
#for matrix
ecb = function(x,y){
  plot(x,t='n');
  text(x,labels=iris$Species, col=colors[iris$Species])
}
kpc2 <- qtSNE(train, kernel = "rbfbase", qpar = list(sigma=1,q=0.8),
              epoch_callback = ecb, perplexity=10, max_iter = 500)

#cndernf
cndkernf(kpc2)

#dimRed
plot(dimRed(kpc2),col=train)


## End(Not run)
```

Index