Package ‘rdetools’

February 20, 2015

Type Package

Title Relevant Dimension Estimation (RDE) in Feature Spaces

Version 1.0

Date 2008-09-03

Author Jan Saputra Mueller

Maintainer Jan Saputra Mueller <saputra@cs.tu-berlin.de>

Description The package provides functions for estimating the relevant dimension of a data set in feature spaces, applications to model selection, graphical illustrations and prediction.

License GPL-2

Repository CRAN

Date/Publication 2012-10-29 08:59:34

NeedsCompilation no

R topics documented:

rdetools-package *Relevant Dimension Estimation (RDE) in Feature Spaces*

Description

Only a finite number of leading kernel PCA components contain the relevant information of a supervised learning problem if the kernel matches the problem. The package provides functions for estimating the relevant dimension in kernel feature spaces. These functions are also able to calculate denoised versions of your label vectors and to estimate the noise levels in your data sets. RDE can also be used for model selection. The package provides functions for this issue and graphical functions to illustrate the results of RDE and model selection. For making predictions kernel projection regression is available.

Details

Package:	rdetools
Type:	Package
Version:	1.0
Date:	2008-08-03
License:	GPL-2

Author(s)

Jan Saputra Mueller

Maintainer: Jan Saputra Mueller <saputra@cs.tu-berlin.de>

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

Examples

```
## rde on a noisy sinc data set
d <- sincdata(100, 0.1) # generate noisy sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
# estimate relevant dimension, denoised ys and noise level in data set
r <- rde(K, d$y, est_y = TRUE, est_noise = TRUE)
r$rd # relevant dimension
r$yh # denoised ys
r$noise # noise level in data set
drawkpc(r) # draw kernel pca coefficients
```

```
## rde for model selection
d <- sincdata(100, 0.1) # generate sinc data
# do model selection
m <- selectmodel(d$X, d$y, sigma = logspace(-3, 3, 100))
m$best # best model
m$rd # relevant dimension for best model
modelimage(m) # graphical illustration of model selection

## kernel projection regression
d <- sincdata(100, 0.1) # generate sinc data
# do model selection
m <- selectmodel(d$X, d$y, sigma = logspace(-3, 3, 100))
f <- kpr(m) # kernel projection regression
plot(f, -4, 4) # draw predicted function
```

`denoiselabels` *Denoise labels*

Description

The function denoises labels of a dataset by projecting them to the `d` first kernel pca principal directions if `d` is not 0. If `d` is 0 the function returns a matrix containing the projected labels for each dimension in each column. The function is primarily an auxiliary function for the rde functions, and it should not be necessary to call it by hand, because rde will do this for you (see examples).

Usage

```
denoiselabels(d, eigvec, kpc, regression = TRUE)
```

Arguments

`d`	number of leading kernel pca principal directions to project the labels to or 0, if a matrix should be returned which contains the denoised labels for each dimension
`eigvec`	eigenvectors of the kernel matrix
`kpc`	kernel pca coefficients
`regression`	set this to TRUE, if the data should be handled as data of a regression problem and to FALSE in case of a classification problem

Value

denoised version of the labels or a matrix with denoised labels for each dimension in its columns if `d` was 0.

Author(s)

Jan Saputra Mueller

See Also

`rde`, `rde_loocv`, `rde_tcm`

Examples

```
## example with sinc data
d <- sincdata(100, 0.7) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
# rde, return also denoised labels
r <- rde(K, d$y, est_y = TRUE)
r$yh # denoised labels
```

`distimage` *Distance image*

Description

If you've done a model selection with `selectmodel`, this function can draw you a map, in which the distances of the original label vector and the estimated label vectors are shown. This is done by a `filled.contour` plot.

Usage

```
distimage(model,
          color.palette = terrain.colors,
  log = TRUE,
  plottitle = "Distance of Ys",
  ...)
```

Arguments

`model`	list of model selection data as it has been returned by `selectmodel`. `selectmodel` must have been called with `ydist` parameter set to TRUE!
`color.palette`	color palette function to use, see `rainbow`
`log`	leave this TRUE, if the axis of the model parameter should be logarithmically scaled. Set this to FALSE if you want linear scaling.
`plottitle`	title of the plot
`...`	additional parameters for `filled.contour`

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

selectmodel, modelimage, drawkpc, filled.contour, rainbow

Examples

```
## model selection with RBF-kernel and graphical illustration
## of the distances of the labels
d <- sincdata(100, 0.1) # generate sinc data
# do model selection
m <- selectmodel(d$X, d$y, ydist = TRUE, sigma = logspace(-3, 3, 100))
distimage(m) # distance image
```

---

drawkpc *Draw kernel pca coefficients*

Description

The function plots the absolute values of the kernel pca coefficients. The estimated relevant dimension and the estimated noise level (if available) are also drawn. Optionally, it puts a rescaled version of the loo-cv-error/negative-log-likelihood into the plot.

Usage

```
drawkpc(model,
        err = TRUE,
pointcol = "blue",
rdcol = "red",
noisecol = "black",
errcol = "brown",
...)
```

Arguments

model	list of rde data returned by rde or selectmodel
err	leave this TRUE, if you want to have a rescaled version of the the loo-cv-error/negative-log-likelihood in the plot
pointcol	color of the kernel pca coefficients
rdcol	color of the relevant dimension line
noisecol	color of the noise level line
errcol	color of the the loo-cv-error/negative-log-likelihood
...	additional parameters to the plotting functions

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

rde, selectmodel, modelimage, distimage

Examples

```
## draw kernel pca coefficients after calling rde
d <- sincdata(100, 0.1) # generate sinc data
K <- rbfkernel(d$X)
r <- rde(K, d$y, est_noise = TRUE)
drawkpc(r)

## draw kernel pca coefficients after calling selectmodel
d <- sincdata(100, 0.1) # generate sinc data
m <- selectmodel(d$X, d$y, est_noise = TRUE, sigma = logspace(-3, 3, 100))
drawkpc(m)
```

---

estnoise *Estimate noise level*

Description

Estimates the noise level for a label vector 'y' and a denoised version of this label vector 'yh'. Which loss function is used to estimate the noise level depends on the kind of problem (regression problem or classification problem).

Usage

```
estnoise(y, yh, regression = FALSE, nmse = TRUE)
```

Arguments

y	a label vector containg only -1 and 1 for a classification problem, and real numbers in case of regression
yh	a denoised version of y which can be obtained by using e.g. rde
regression	FALSE in case of a classification problem, TRUE in case of a regression problem
nmse	if 'nmse' is TRUE and this is a regression problem, the mean squared error will be normalized

Details

In case of a classification problem, the 0-1-loss is used to estimate the noise level:

$$y = (y_1, ..., y_n)$$

$$L_{01}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{I}_{\{y_i \neq \hat{y}_i\}}$$

In case of a regression problem, the mean squared error (mse) or the normalized mean squared error (nmse) is used, depending on whether 'nmse' is FALSE (mse) or TRUE (nmse):

$$L_{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$L_{nmse}(y, \hat{y}) = \frac{L_{mse}(y, \hat{y})}{\frac{1}{n} \sum_{i=1}^{n} (y_i - \frac{1}{n} \sum_{j=1}^{n} y_j)^2}$$

Value

Estimated noise level

Author(s)

Jan Saputra Mueller

See Also

sincdata, rde_loocv, rde_tcm, rbfkernel, drawkpc

Examples

```
## estimate noise of sinc data explicitly
d <- sincdata(100, 0.7) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
r <- rde(K, d$y, est_y = TRUE) # estimate relevant dimension
noise <- estnoise(d$y, r$yh, regression = TRUE) # estimate noise level

## estimate noise of sinc data implicitly (via rde_loocv)
d <- sincdata(100, 0.7) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
r <- rde(K, d$y, est_y = TRUE) # estimate relevant dimension AND estimate noise
r$noise # estimated noise level
```

isregression — *Estimate from labels whether this is a regression problem*

Description

Estimates whether this is a regression or classification problem by looking at the labels. If all labels are only -1 and 1 a classification problem is assumed, otherwise a regression problem. If the argument 'regression' is TRUE, the function always returns TRUE.

Usage

```
isregression(y, regression = FALSE)
```

Arguments

`y`	label vector for which the kind of problem should be estimated
`regression`	if `regression` is TRUE, the function returns always TRUE, if you want an estimation leave this FALSE

Value

TRUE, if this is a regression problem or the argument `regression` was TRUE, otherwise FALSE

Author(s)

Jan Saputra Mueller

See Also

`rde_loocv`, `rde_tcm`, `estnoise`

Examples

```
## some examples
y_cl <- c(-1, 1, 1, -1, 1) # label vector for classification problem
y_reg <- runif(5) # label vector for regression problem
isregression(y_cl) # FALSE!
isregression(y_cl, regression = TRUE) # Always TRUE!
isregression(y_reg) # TRUE!
```

kpr *Kernel projection regression*

Description

The function does a kernel projection regression. It returns a function which predicts labels for new data points.

Usage

```
kpr(model,
    X = NULL,
    Xname = "X",
    Yname = "Y",
    kernel = NULL,
    regression = TRUE,
    ...)
```

Arguments

`model`	list of rde data returned by `rde` or `selectmodel`
`X`	matrix containing the data points, only needed if `rde` was used
`Xname`	the name of the parameter of the kernel function which should contain the data points, only needed if `rde` was used
`Yname`	the name of the parameter of the kernel function which should contain the 2nd data matrix
`kernel`	kernel function to use, only needed if `rde` was used
`regression`	set this to TRUE in case of a regression problem and to FALSE in case of a classification problem; only needed if `rde` was used
`...`	parameters for the kernel function, only needed if `rde` was used

Value

function which predicts labels for new input data (gets a matrix with one data point per line)

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

`selectmodel`

Examples

```
## kernel projection regression after
## calling selectmodel (recommended)
d <- sincdata(100, 0.1) # generate sinc data
# do model selection
m <- selectmodel(d$X, d$y, sigma = logspace(-3, 3, 100))
f <- kpr(m)
plot(f, -4, 4)
```

`logspace` *Logarithmically spaced sequence generation*

Description

Function generates a logarithmically spaced sequence of `n` values between decades 10^l and 10^u.

Usage

```
logspace(l, u, n)
```

Arguments

`l`	10^l will be the lower value to start from
`u`	10^u will be the upper value to end with
`n`	number of values to generate

Value

Logarithmically spaced sequence of length `n` between 10^l and 10^u.

Author(s)

Jan Saputra Mueller

See Also

`seq`, `selectmodel`

Examples

```
## generate 100 logarithmically spaced values between 10^(-3) and 10^3
logspace(-3, 3, 100)
```

`modelimage`	*Model selection image*

Description

The function produces a graphical illustration of a model selection which has been done with `selectmodel`. Strictly speaking it's a `filled.contour` plot in which additionally the relevant dimensions for the different models are drawn as a black line. `selectmodel` chooses the deepest point in this map, that is the model and the relevant dimension with the smallest loo-cv-error/negative-log-likelihood-value.

Usage

```
modelimage(model,
            color.palette = topo.colors,
   log = TRUE,
   plottitle = "RDE Model Selection",
   ...)
```

Arguments

`model`	list of model selection data as it has been returned by `selectmodel`
`color.palette`	color palette function to use, see `rainbow`
`log`	leave this TRUE, if the axis of the model parameter should be logarithmically scaled. Set this to FALSE if you want linear scaling.
`plottitle`	title of the plot
`...`	additional parameters for `filled.contour`

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

`selectmodel`, `distimage`, `drawkpc`, `filled.contour`, `rainbow`

Examples

```
## model selection with RBF-kernel and graphical illustration
d <- sincdata(100, 0.1) # generate sinc data
# do model selection
m <- selectmodel(d$X, d$y, sigma = logspace(-3, 3, 100))
modelimage(m) # draw model selection image
```

`polykernel` *Calculate polynomial kernel matrix*

Description

Calculates the polynomial kernel matrix for the dataset contained in the matrix `X`, where each row of `X` is a data point. If `Y` is also a matrix (with the same number of columns as `X`), the kernel function is evaluated between all data points of `X` and `Y`.

Usage

```
polykernel(X, d, Y = NULL)
```

Arguments

`X`	matrix containing a data point in each column
`d`	polynomial kernel degree
`Y`	leave this NULL if the kernel function should be evaluated between the data points only contained in `X` (which can be regarded as `Y = X`) or to a matrix with same number of columns as `X` if you want to evaluate the function between the points of `X` and `Y`

Details

Each row of `X` must be a data point, i.e. $X = (x_1, x_2, ..., x_n)$. The kernel matrix `K` is then defined as

$$K = (k(x_i, x_j))_{i,j=1,...,n}$$

If `Y` is not NULL and also contains data points in each row, i.e. $Y = (y_1, y_2, ..., y_m)$, the kernel matrix `K` of `X` and `Y` is defined as

$$K = (k(x_i, y_j))_{i=1,...,n,j=1,...,m}$$

In this case, k is the polynomial kernel, which is defined as

$$k(x, y) = (\langle x, y \rangle + 1)^d$$

where x, y are data points and d is the polynomial kernel degree.

Value

polynomial kernel matrix `K` for the given dataset

Author(s)

Jan Saputra Mueller

See Also

rbfkernel, sincdata

Examples

```
## generate sinc data and calculate polynomial kernel matrix with d = 5
d <- sincdata(100, noise = 0.1)
K <- polykernel(d$X, 5)
```

---

rbfkernel *Calculate RBF kernel matrix*

Description

Calculates the RBF kernel matrix for the dataset contained in the matrix `X`, where each row of `X` is a data point. If `Y` is also a matrix (with the same number of columns as `X`), the kernel function is evaluated between all data points of `X` and `Y`.

Usage

```
rbfkernel(X, sigma = 1, Y = NULL)
```

Arguments

`X`	matrix containing a data point in each row
`sigma`	kernel width of rbf kernel
`Y`	leave this NULL if the kernel function should be evaluated between the data points only contained in `X` (which can be regarded as `Y = X`) or to a matrix with same number of columns as `X` if you want to evaluate the function between the points of `X` and `Y`

Details

Each row of `X` must be a data point, i.e. $X = (x_1, x_2, ..., x_n)$. The kernel matrix `K` is then defined as

$$K = (k(x_i, x_j))_{i,j=1,...,n}$$

If `Y` is not NULL and also contains data points in each row, i.e. $Y = (y_1, y_2, ..., y_m)$, the kernel matrix `K` of `X` and `Y` is defined as

$$K = (k(x_i, y_j))_{i=1,...,n,j=1,...,m}$$

In this case, k is the rbf (radial basis function) kernel, which is defined as

$$k(x, y) = exp(-\frac{||x - y||^2}{2\sigma})$$

where x, y are data points and sigma is the rbf kernel width.

Value

RBF kernel matrix K for the given dataset

Author(s)

Jan Saputra Mueller

See Also

polykernel, sincdata

Examples

```
## generate sinc data and calculate rbf kernel matrix with sigma = 1
d <- sincdata(100, noise = 0.1)
K <- rbfkernel(d$X)
```

rde *Relevant Dimension Estimation (RDE)*

Description

The function estimates the relevant dimension in feature space. By default, this is done by fitting a two-component model, but rde by leave-one-out cross-validation is also available. The function is also able to calculate a denoised version of the labels and to estimate the noise level in the data set.

Usage

```
rde(K, y,
    est_y = FALSE,
    alldim = FALSE,
    est_noise = FALSE,
    regression = FALSE,
    nmse = TRUE,
    dim_rest = 0.5,
    tcm = TRUE)
```

Arguments

K	kernel matrix of the inputs (e.g. rbf kernel matrix)
y	label vector which contains the label for each data point
est_y	set this to TRUE if you want a denoised version of the labels
alldim	if this is TRUE denoised labels for all dimensions are calculated (instead of only for relevant dimension)
est_noise	set this to TRUE if you want an estimated noise level

`regression`	only interesting if one of `est_y`, `alldim`, `est_noise` is TRUE. Set this to TRUE if you want to force the function to handle the data as data for a regression problem. If you leave this FALSE, the function will try to determine itself whether this is a classification or regression problem.
`nmse`	only interesting if `est_noise` is TRUE and the function is handling the data as data of a regression problem. If you leave this TRUE, the normalized mean squared error is used for estimating the noise level, otherwise the conventional mean squared error.
`dim_rest`	percantage of leading dimensions to which the search for the relevant dimensions should be restricted. This is needed due to numerical instabilities. 0.5 should be a good choice in most cases (and is also the default value)
`tcm`	this is TRUE by default; indicates whether rde should be done by TCM or LOO-CV algorithm

Details

If `est_noise` or `alldim` are TRUE, a denoised version of the labels for the relevant dimension will be returned even if `est_y` is FALSE (so e.g. if you want denoised labels and noise approximation it is enough to set `est_noise` to TRUE).

Value

`rd`	estimated relevant dimension
`err`	loo-cv-error/negative-log-likelihood-value for each dimension (the position of the minimum is the relevant dimension)
`yh`	only returned if `est_y`, `alldim` or `est_noise` is TRUE, contains the denoised labels
`Yh`	only returned if `alldim` is TRUE, matrix with denoised labels for each dimension in each column
`noise`	only returned if `est_noise` is TRUE, contains the estimated noise level
`kpc`	kernel pca coefficients
`eigvec`	eigenvectors of the kernel matrix
`eigval`	eigenvalues of the kernel matrix
`tcm`	TRUE if TCM algorithm was used, otherwise (LOO-CV algorithm) FALSE

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

`rde_loocv`, `rde_tcm`, `estnoise`, `isregression`, `rbfkernel`, `polykernel`, `drawkpc`

Examples

```
## example with sinc data using tcm algorithm
d <- sincdata(100, 0.1) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
# rde, return also denoised labels and noise, fit tcm
r <- rde(K, d$y, est_y = TRUE, est_noise = TRUE)
r$rd # estimated relevant dimension
r$noise # estimated noise
drawkpc(r) # draw kernel pca coefficients

## example with sinc data using loo-cv algorithm
d <- sincdata(100, 0.1) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
# rde, return also denoised labels and noise
r <- rde(K, d$y, est_y = TRUE, est_noise = TRUE, tcm = FALSE)
r$rd # estimated relevant dimension
r$noise # estimated noise
drawkpc(r) # draw kernel pca coefficients
```

rde_loocv — *Relevant Dimension Estimation (RDE) by Leave-One-Out Cross-Validation (LOO-CV)*

Description

The function estimates the relevant dimension in feature space by leave-one-out cross-validation. It's also able to calculate a denoised version of the labels and to estimate the noise level in the data set.

Usage

```
rde_loocv(K, y,
          est_y = FALSE,
  alldim = FALSE,
  est_noise = FALSE,
  regression = FALSE,
  nmse = TRUE,
  dim_rest = 0.5)
```

Arguments

`K`	kernel matrix of the inputs (e.g. rbf kernel matrix)
`y`	label vector which contains the label for each data point
`est_y`	set this to TRUE if you want a denoised version of the labels
`alldim`	if this is TRUE denoised labels for all dimensions are calculated (instead of only for relevant dimension)
`est_noise`	set this to TRUE if you want an estimated noise level

`regression`	only interesting if one of `est_y`, `alldim`, `est_noise` is TRUE. Set this to TRUE if you want to force the function to handle the data as data for a regression problem. If you leave this FALSE, the function will try to determine itself whether this is a classification or regression problem.
`nmse`	only interesting if `est_noise` is TRUE and the function is handling the data as data of a regression problem. If you leave this TRUE, the normalized mean squared error is used for estimating the noise level, otherwise the conventional mean squared error.
`dim_rest`	percantage of leading dimensions to which the search for the relevant dimensions should be restricted. This is needed due to numerical instabilities. 0.5 should be a good choice in most cases (and is also the default value)

Details

If `est_noise` or `alldim` are TRUE, a denoised version of the labels for the relevant dimension will be returned even if `est_y` is FALSE (so e.g. if you want denoised labels and noise approximation it is enough to set `est_noise` to TRUE).

Value

`rd`	estimated relevant dimension
`err`	loo-cv error for each dimension (the position of the minimum is the relevant dimension)
`yh`	only returned if `est_y`, `alldim` or `est_noise` is TRUE, contains the denoised labels
`Yh`	only returned if `alldim` is TRUE, matrix with denoised labels for each dimension in each column
`noise`	only returned if `est_noise` is TRUE, contains the estimated noise level
`kpc`	kernel pca coefficients
`eigvec`	eigenvectors of the kernel matrix
`eigval`	eigenvalues of the kernel matrix
`tcm`	always FALSE; used to tell other functions that loo-cv method was used

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

`rde`, `rde_tcm`, `estnoise`, `isregression`, `rbfkernel`, `polykernel`, `drawkpc`

Examples

```
## example with sinc data
d <- sincdata(100, 0.1) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
# rde, return also denoised labels and noise
r <- rde_loocv(K, d$y, est_y = TRUE, est_noise = TRUE)
r$rd # estimated relevant dimension
r$noise # estimated noise
drawkpc(r) # draw kernel pca coefficients
```

---

rde_tcm *Relevant Dimension Estimation (RDE) by Fitting a Two-Component Model (TCM)*

Description

The function estimates the relevant dimension in feature space by fitting a two-component model. It's also able to calculate a denoised version of the labels and to estimate the noise level in the data set.

Usage

```
rde_tcm(K, y,
        est_y = FALSE,
alldim = FALSE,
est_noise = FALSE,
regression = FALSE,
nmse = TRUE,
dim_rest = 0.5)
```

Arguments

`K`	kernel matrix of the inputs (e.g. rbf kernel matrix)
`y`	label vector which contains the label for each data point
`est_y`	set this to TRUE if you want a denoised version of the labels
`alldim`	if this is TRUE denoised labels for all dimensions are calculated (instead of only for relevant dimension)
`est_noise`	set this to TRUE if you want an estimated noise level
`regression`	only interesting if one of `est_y`, `alldim`, `est_noise` is TRUE. Set this to TRUE if you want to force the function to handle the data as data for a regression problem. If you leave this FALSE, the function will try to determine itself whether this is a classification or regression problem.
`nmse`	only interesting if `est_noise` is TRUE and the function is handling the data as data of a regression problem. If you leave this TRUE, the normalized mean squared error is used for estimating the noise level, otherwise the conventional mean squared error.

`dim_rest`	percantage of leading dimensions to which the search for the relevant dimensions should be restricted. This is needed due to numerical instabilities. 0.5 should be a good choice in most cases (and is also the default value)

Details

If `est_noise` or `alldim` are TRUE, a denoised version of the labels for the relevant dimension will be returned even if `est_y` is FALSE (so e.g. if you want denoised labels and noise approximation it is enough to set `est_noise` to TRUE).

Value

`rd`	estimated relevant dimension
`err`	negative log-likelihood for each dimension (the position of the minimum is the relevant dimension)
`yh`	only returned if `est_y`, `alldim` or `est_noise` is TRUE, contains the denoised labels
`Yh`	only returned if `alldim` is TRUE, matrix with denoised labels for each dimension in each column
`noise`	only returned if `est_noise` is TRUE, contains the estimated noise level
`kpc`	kernel pca coefficients
`eigvec`	eigenvectors of the kernel matrix
`eigval`	eigenvalues of the kernel matrix
`tcm`	always TRUE; used to tell other functions that tcm method was used

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

`rde`, `rde_loocv`, `estnoise`, `isregression`, `rbfkernel`, `polykernel`, `drawkpc`

Examples

```
## example with sinc data
d <- sincdata(100, 0.1) # generate sinc data
K <- rbfkernel(d$X) # calculate rbf kernel matrix
# rde, return also denoised labels and noise
r <- rde_tcm(K, d$y, est_y = TRUE, est_noise = TRUE)
r$rd # estimated relevant dimension
r$noise # estimated noise
drawkpc(r) # draw kernel pca coefficients
```

`selectmodel` *Model selection*

Description

The function can be used for selecting the kernel from a number of possible candidates which fits the problem best. You need a parametrized kernel function and a number of possible parameters. A relevant dimension estimation will be done for all parameter combinations and the one with the smallest loo-cv-error/negative-log-likelihood on its estimated relevant dimension will be chosen.

Usage

```
selectmodel(X, y,
            kernel = rbfkernel,
    est_y = FALSE,
    ydist = FALSE,
    est_noise = FALSE,
    regression = FALSE,
    nmse = TRUE,
    tcm = TRUE,
    Xname = "X",
    ...)
```

Arguments

`X`	matrix containing a data point in each row
`y`	label vector which contains the label for each data point
`kernel`	parametrized kernel function which should be used
`est_y`	set this to TRUE if you want a denoised version of the labels for the best model
`ydist`	set this to TRUE if you want a matrix, which contains the distances between the denoised labels and the original labels for all dimensions and all parameter combinations (each line in the matrix contains the distances for one parameter combination. This is needed for `distimage`)
`est_noise`	set this to TRUE if you want an estimated noise level (for the best model)
`regression`	only interesting if `est_y` or `est_noise` is TRUE. Set this to TRUE if you want to force the function to handle the data as data for a regression problem. If you leave this FALSE, the function will try to determine itself whether this is a classification or regression problem.
`nmse`	only interesting if `est_noise` is TRUE and the function is handling the data as data of a regression problem. If you leave this TRUE, the normalized mean squared error is used for estimating the noise level, otherwise the conventional mean squared error.
`tcm`	this is TRUE by default; indicates whether rde should be done by TCM or LOO-CV algorithm

`Xname`	the name of the parameter of the kernel function which should contain the data points. This is `X` by default and can be left as it is if you use `rbfkernel` or `polykernel`.
`...`	for each parameter of the kernel function you should give a list of parameters to select the best parameter combination from (e.g. for `rbfkernel` this is only the parameter `sigma` of for `polykernel` it's only the parameter `d`. See examples.)

Value

`rd`	estimated relevant dimension for best model
`best`	the best parameter combination which has been found through model selection
`yh`	only returned if `est_y`, `alldim` or `est_noise` is TRUE, contains the denoised labels for the best model
`noise`	only returned if `est_noise` is TRUE, contains the estimated noise level for the best model
`Yd`	contains the distances of the denoised labels and the original labels; needed for `distimage`
`rds`	estimated relevant dimensions for each model
`err`	loo-cv-error/negative-log-likelihood-value for each dimension for the best model
`errs`	loo-cv-error/negative-log-likelihood-value for each dimension for all models (in each line is the error for one model)
`kpc`	kernel pca coefficients for best model
`eigvec`	eigenvectors of the kernel matrix for best model
`eigval`	eigenvalues of the kernel matrix for best model
`params`	list of parameters for the kernel function which has been given to the function
`tcm`	TRUE if TCM algorithm was used, otherwise (LOO-CV algorithm) FALSE
`kernel`	kernel function which has been used
`Xname`	the name of the parameter of the kernel function which should contain the data points as it has been given to the function
`X`	matrix with the data points as it has been given to the function
`regression`	TRUE, if the data are data of a regression problem, FALSE in case of a classification problem

Author(s)

Jan Saputra Mueller

References

M. L. Braun, J. M. Buhmann, K. R. Mueller (2008) _On Relevant Dimensions in Kernel Feature Spaces_

See Also

`rde`, `modelimage`, `distimage`, `drawkpc`

Examples

```
## model selection with RBF-kernel
d <- sincdata(100, 0.1) # generate sinc data
# do model selection, calculate also denoised labels
m <- selectmodel(d$X, d$y, est_y = TRUE, sigma = logspace(-3, 3, 100))
m$best # best model
m$rd # relevant dimension for best model
modelimage(m) # draw model selection image

## model selection with polynomial kernel
d <- sincdata(100, 0.1) # generate sinc data
# do model selection, calculate also denoised labels
m <- selectmodel(d$X, d$y, kernel = polykernel, est_y = TRUE, d = 1:20)
m$best # best model
m$rd # relevant dimension for best model
modelimage(m, log = FALSE) # draw model selection image
```

`sinc` *Calculate sinc values*

Description

Calculates (normalized) sinc values for each element of a vector/matrix/... etc.,

$$sinc(x) = \frac{sin(\pi x)}{\pi x}, x \neq 0$$

If $x = 0, sinc(x)$ is defined as 1 (removable singularity in zero).

Usage

```
sinc(X)
```

Arguments

`X`	an arbitrary vector/matrix/... containing numbers

Value

an vector/matrix/... of same size as the argument containing the sinc values for each element

Author(s)

Jan Saputra Mueller

References

http://en.wikipedia.org/wiki/Sinc_function

See Also

sincdata

Examples

```
## calculate sinc values of a vector
v <- 1:10
sinc(v)
```

---

sincdata *Generate random sinc data*

Description

Function draws n points uniformly from the interval $[a, b]$, calculates the sinc (normalized sinc function) values for that points and adds a normal noise with a standard deviation of `noise` to these values.

Usage

```
sincdata(n, noise = 0, a = -4, b = 4)
```

Arguments

`n`	number of points to generate
`noise`	noise level to add to sinc values, i.e. standard deviation of normal noise
`a`	left bound of interval from which the xs are drawn, `a` must be smaller than `b`
`b`	right bound of interval from which the ys are drawn, `b` must be larger than `a`

Value

Randomly generated sinc data

`X`	matrix with one column (i.e. a vector, but returned object is a matrix) containing the x-values
`y`	matrix with one row (i.e. a vector, but returned object is a matrix) containing the y-values

Author(s)

Jan Saputra Mueller

References

http://en.wikipedia.org/wiki/Sinc_function

See Also

sinc

Examples

```
## generate 100 data points with noise level 0
## drawn from the interval [-4,4]
sincdata(100)

## generate 1000 data points with noise level 0.7
## drawn from the interval [-10, 10]
sincdata(100, 0.7, a = -10, b = 10)
```

Index