Package ‘KODAMA’

February 19, 2015

Version 0.0.1
Date 2013-11-20
Author Stefano Cacciatore, Claudio Luchinat, Leonardo Tenori
Maintainer Stefano Cacciatore <tkcaccia@gmail.com>
Title Knowledge discovery by accuracy maximization
Description KODAMA (KnOwledge Discovery by Accuracy MAximization) is an unsupervised and semisupervised learning algorithm that performs feature extraction from noisy and high-dimensional data.
Depends R (>= 2.10.0), e1071, plsgenomics, class
Suggests rgl,
SuggestsNote No suggestions
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2014-11-25 18:05:43

R topics documented:

classprob .......................................................... 2
core .............................................................. 3
dinisurface ......................................................... 4
helicoid .......................................................... 5
kfold .............................................................. 6
KNN.CV ........................................................... 7
knn.dist .......................................................... 8
knn.predict ....................................................... 9
knn.probability .................................................. 10
KODAMA .......................................................... 12
lymphoma ......................................................... 15
majority ......................................................... 16
MetRef ........................................................... 17
normalization ..................................................... 17
classprob

Determines the Prevalence of Each Class

description

This function determines the prevalence or probability for each class of a vector (treated as factor).

Usage

classprob(x)

Arguments

x

A vector.

details

This function treats the input vector as a factor and determines the probability for each level (class) of the factor. The order of the returned probabilities is the order of the levels command, which defaults to numeric or alphabetic order.

Value

A vector whose length is equal to the number of levels in the input. The order is numerically or alphabetically increasing. Note the factors may have levels which are not present in the vector. See examples for details.

Author(s)

Atina Dunlap Brooks

See Also

majority
Examples

```r
# calculate probabilities
x <- sample(c("a","b","c","d","e"), 10, replace=TRUE)
classprob(x)
# labels the probabilities
levels(as.factor(x))

# to see levels which aren't represented in the vector
x <- as.factor(c("a","a","a","b","b","c"))
levels(x)
# now remove the "c"
x <- x[1:5]
# but "c" is still a level
levels(x)
# and the probability is calculated for it
classprob(x)
```

Maximization of Cross-Validateed Accuracy Methods

Description

This function performs the maximization of cross-validated accuracy by an iterative process.

Usage

```r
core(x, clbest, Tcycle=20, FUN=KNN.CV, f.par=list(kn=10),
     constrain=NULL, fix=NULL, shake=FALSE)
```

Arguments

- `x`: a matrix.
- `clbest`: a vector to optimize.
- `Tcycle`: number of interative cycle that leads to the maximization of cross-validated accuracy.
- `FUN`: classifier to be consider. Choices are "KNN.CV", "PLS.SVM.CV", and "PCA.CA.KNN.CV".
- `f.par`: parameters of the classifier.
- `constrain`: a vector of `nrow(data)` elements. Supervised constraints can be imposed by linking some samples in such a way that if one of them is changed the linked ones must change in the same way (i.e., they are forced to belong to the same class) during the maximization of the cross-validation accuracy procedure. Sample with the same identificative constrain will be forced to stay together.
- `fix`: a vector of `nrow(data)` elements. The values of this vector must to be TRUE or FALSE. By default all elements are FALSE. Samples with the TRUE fix value will not change the class label defined in `w` during the maximization of the cross-validation accuracy procedure.
if shake = FALSE the cross-validated accuracy is computed with the class defined in \( \mathbf{W} \) else the it is not, before the maximization of the cross-validation accuracy procedure.

Value

The function returns a list with 3 items:

- \( c \) a classification vector with a maximized cross-validated accuracy.
- \( a \) the maximum cross-validated accuracy achieved.
- \( v \) a vector of all cross-validated accuracy obtained.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References

Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.

See Also

KODAMA

Examples

```
# data(iris)
# u=iris[,,-5]
# s=sample(1:150,150,TRUE)
# results=core(u,s)
# unique(s)
# unique(results$c)
```

---

**Description**

This function creates a data set on the data points are distributed on a Ulisse Dini’s surface.

**Usage**

```
dinisurface(N=1000)
```

**Arguments**

- \( N \) Number of data points.
helicoid

Value
The function returns a three dimensional data set.

Author(s)
Stefano Cacciatore and Leonardo Tenori

References
Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.

See Also
helicoid, swissroll, spirals

Examples
require("rgl")
x=dinisurface()
open3d()
plot3d(x, col=rainbow(1050)[-c(1:50)], box=FALSE, type="s", size=1)

helicoid  Helicoid Data Set Generator

Description
This function creates a data set on the data points are distributed on a Ulisse Dini’s surface.

Usage
helicoid(N=1000)

Arguments
N Number of data points.

Value
The function returns a three dimensional data set.

Author(s)
Stefano Cacciatore and Leonardo Tenori
References
Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.

See Also
swissroll, dinisurface, spirals

Examples
require("rgl")
x=helicoid()
open3d()
plot3d(x, col=rainbow(1050)[-c(1:50)], box=FALSE, type="s", size=1)

---

kfold  $k$-Fold Partitioning

Description
This function computes the k-fold partitioning of a vector. Each record in a vector is randomly
assigned to a group. Group numbers are between 1 and k.

Usage
kfold(constrain, k=10)

Arguments
constrain a vector of labels.
k number of groups.

Value
A vector with group assignments.

Author(s)
Stefano Cacciatore and Leonardo Tenori

Examples
kfold(1:100, k=10)
kfold(rep(1:20, each=5), k=10)
Description

This function performs a 10-fold cross validation on a given data set using k nearest neighbors (kNN) classifier. The output is a vector of predicted labels.

Usage

```
KNN.CV(x, cl, constrain, kn=10)
```

Arguments

- `x`: a matrix.
- `cl`: a classification vector.
- `constrain`: a vector of `nrow(data)` elements. Sample with the same identificative constrain will be split in the training set or in the test test of cross-validation together.
- `kn`: the number of nearest neighbors to consider.

Details

Details are described in Cover, et al. (1967).

Value

The function returns a vector of predicted labels.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References


See Also

`PLS.SVM.CV,PCA.CA,KNN.CV`
knn.dist

Calculates the Distances for KNN Predictions

Description

The distances to be used for K-Nearest Neighbor (KNN) predictions are calculated and returned as a symmetric matrix. Distances are calculated by \texttt{dist}.

Usage

\texttt{knn.dist(x, dist.meth = "euclidean", p = 2)}

Arguments

- \textit{x} a matrix of data.
- \textit{dist.meth} the distance to be used in calculating the neighbors. Any method valid in function \texttt{dist} is valid.
- \textit{p} the power of the Minkowski distance.

Details

This function calculates the distances to be used by \texttt{knn.predict}. Distances are calculated between all cases. In the traditional scenario. The advantage to calculating distances in a separate step prior to prediction, is that these calculations only need to be performed once. So, for example, cross-validation to select \( k \) can be performed on many values of \( k \), with different cross-validation splits, all using a single run of \texttt{knn.dist}. The default method for calculating distances is the \texttt{"euclidean"} distance, which is the method used by the \texttt{knn} function from the \texttt{class} package. Alternative methods may be used here. Any method valid for the function \texttt{dist} is valid here. The parameter \( p \) may be specified with the Minkowski distance to use the \( p \) norm as the distance method.

Value

a square symmetric matrix whose dimensions are the number of rows in the original data. The diagonal contains zeros, the off diagonal entries will be \( >=0 \).

Author(s)

Atina Dunlap Brooks
**knn.predict**

**knn.predict**

**KNN Prediction Routine using Pre-Calculated Distances**

**Description**

K-Nearest Neighbor prediction method which uses the distances calculated by `knn.dist`.

**Usage**

```r
knn.predict(train, test, y, dist.matrix, k=1,
             agg.meth = if (is.factor(y)) "majority" else "mean",
             ties.meth = "min")
```

**Arguments**

- `train`: indexes which specify the rows of the `dist.matrix` to use as training set.
- `test`: indexes which specify the rows of the `dist.matrix` to use as test set.
- `y`: a vector of labels.
- `dist.matrix`: the output from a call to `knn.dist`.
- `k`: the number of nearest neighbors to consider.
- `agg.meth`: method to combine responses of the nearest neighbors, defaults to "majority" for classification and "mean" for continuous responses.
- `ties.meth`: method to handle ties for the k-th neighbor, the default is "min" which uses all ties, alternatives include "max" which uses none if there are ties for the k-th nearest neighbor, "random" which selects among the ties randomly and "first" which uses the ties in their order in the data.

**Examples**

```r
# a quick classification example
x1 <- c(rnorm(20, mean=1), rnorm(20, mean=5))
x2 <- c(rnorm(20, mean=5), rnorm(20, mean=1))
y <- rep(1:2, each=20)
x <- cbind(x1, x2)
train <- sample(1:40, 30)
# plot the training cases
plot(x1[train], x2[train], col=y[train]+1)
# predict the other cases
test <- (1:40)[-train]
dist <- knn.dist(x)
preds <- knn.predict(train, test, y, dist, k=3, agg.meth="majority")
# add the predictions to the plot
points(x1[test], x2[test], col=as.integer(preds)+1, pch="*")
# display the confusion matrix
table(y[test], preds)
```
Details

Predictions are calculated for each test case by aggregating the responses of the k-nearest neighbors among the training cases. k may be specified to be any positive integer less than the number of training cases, but is generally between 1 and 10. The indexes for the training and test cases are in reference to the order of the entire data set as it was passed to knn.dist. The aggregation may be any named function. By default, classification (factored responses) will use the majority class function and non-factored responses will use mean. Other options to consider include min, max, and median. The ties are handled using the rank function. Further information may be found by examining the ties.method there.

Value

A vector of predictions whose length is the number of test cases.

Author(s)

Atina Dunlap Brooks

See Also

knn.dist.dist

Examples

# a quick classification example
x1 <- c(rnorm(20, mean=1), rnorm(20, mean=5))
x2 <- c(rnorm(20, mean=5), rnorm(20, mean=1))
y <- rep(1:2, each=20)
x <- cbind(x1, x2)
train <- sample(1:40, 30)
# plot the training cases
plot(x1[train], x2[train], col=y[train]+1)
# predict the other cases
test <- (1:40)[-train]
kdist <- knn.dist(x)
preds <- knn.predict(train, test, y, kdist, k=3, agg.meth="majority")
# add the predictions to the plot
points(x1[test], x2[test], col=as.integer(preds)+1, pch="+")
# display the confusion matrix
table(y[test], preds)
knn.probability

Usage

knn.probability(train, test, y, dist.matrix, k=1, ties.meth="min")

Arguments

train indexes which specify the rows of the dist.matrix to use as training set.
test indexes which specify the rows of the dist.matrix to use as test set.
y a vector of labels.
dist.matrix the output from a call to knn.dist.
k the number of nearest neighbors to consider.
ties.meth method to handle ties for the k-th neighbor, the default is "min" which uses all ties, alternatives include "max" which uses none if there are ties for the k-th nearest neighbor, "random" which selects among the ties randomly and "first" which uses the ties in their order in the data.

Details

Predictions are calculated for each test case by aggregating the responses of the k-nearest neighbors among the training cases and using the classprob. k may be specified to be any positive integer less than the number of training cases, but is generally between 1 and 10. The indexes for the training and test cases are in reference to the order of the entire data set as it was passed to knn.dist. The ties are handled using the rank function. Further information may be found by examining the ties.meth there.

Value

A matrix of prediction probabilities whose number of columns is the number of test cases and the number of rows is the number of levels in the responses.

Author(s)

Atina Dunlap Brooks

See Also

knn.dist,knn.predict

Examples

# the iris example used by knn(class)
library(class)
data(iris3)
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
# how to get predictions from knn(class)
pred <- knn(train, test, cl, k = 3, prob=TRUE)
# display the confusion matrix
table(pred,cl)
# view probabilities (only the highest probability is returned)
attr(pred, "prob")
# how to get predictions with knn.dist and knn.predict
x <- rbind(train,test)
kdist <- knn.dist(x)
pred <- knn.predict(1:75, 76:150, cl, kdist, k=3)
# display the confusion matrix
table(pred,cl)

# view probabilities (all class probabilities are returned)
knn.probability(1:75, 76:150, cl, kdist, k=3)
# to compare probabilities, rounding done for display purposes
p1 <- knn(train, test, cl, k = 3, prob=TRUE)
p2 <- round(knn.probability(1:75, 76:150, cl, kdist, k=3), digits=2)
table( round(attr(p1,"prob"), digits=2), apply(p2,2,max) )
# note any small differences in predictions are a result of
# both methods breaking ties in majority class randomly

---

**KODAMA**

*Knowledge Discovery by Accuracy Maximization*

---

**Description**

KODAMA (KnOwledge Discovery by Accuracy MAximization) is an unsupervised and semisupervised learning algorithm that performs feature extraction from noisy and high-dimensional data. Unlike other data mining methods, the peculiarity of KODAMA is that it is driven by an integrated procedure of cross validation of the results.

**Usage**

```r
KODAMA(data, M=100, Tcycle=20,
    FUN_VAR=function(x){ceiling(ncol(x))},
    FUN_SAM=function(x){ceiling(nrow(x)*0.75)},
    bagging=FALSE,
    FUN=KNN.CV,
    f.par=list(kn=10),
    W=NULL,
    constrain=NULL,
    fix=rep(FALSE,nrow(data)),
    epsilon=0.05,
    shake=FALSE)
```

**Arguments**

- `data` a matrix.
- `M` number of iterative processes (step I-III).
KODAMA

Tcycle  number of iterative cycle that leads to the maximization of cross-validated accuracy.

FUN_VAR  function to select the number of variable to select randomly. By default all variable are taken.

FUN_SAM  function to select the number of sample to select randomly. By default the 75

bagging  If it Should sampling be with replacement, bagging = TRUE. By default bagging = FALSE

FUN  classifier to be consider. Choices are "knn.cv", "pls.svm.cv", and "pca.ca.knn.cv".

f.par  parameters of the classifier.

W  a vector of nrow(data) elements. The KODAMA procedure can be started by different initializations of the vector W. Without any a priori information the vector W can be initialized with each element being different from the others (i.e., each sample categorized in a one-element class). Alternatively, the vector W can be initialized by a clustering procedure, such as kmeans.

constrain  a vector of nrow(data) elements. Supervised constraints can be imposed by linking some samples in such a way that if one of them is changed the linked ones must change in the same way (i.e., they are forced to belong to the same class) during the maximization of the cross-validation accuracy procedure. Sample with the same identificative constrain will be forced to stay together.

fix  a vector of nrow(data) elements. The values of this vector must to be TRUE or FALSE. By default all elements are FALSE. Samples with the TRUE fix value will not change the class label defined in W during the maximization of the cross-validation accuracy procedure.

epsilon  cut-off value for low proximity. High proximity are typical of intracluster relationships, whereas low proximities are expected for intercluster relationships. Very low proximities between samples are ignored by (default) setting epsilon = 0.05.

shake  if shake = FALSE the cross-validated accuracy is computed with the class defined in W else the it is not, before the maximization of the cross-validation accuracy procedure.

Details

KODAMA consists of five steps. For a simple description of the method, we can divide KODAMA into two parts: (i) the maximization of cross-validated accuracy by an iterative process (step I and II), resulting in the construction of a proximity matrix (step III), and (ii) the definition of a dissimilarity matrix (step IV and V). The first part entails the core idea of KODAMA, that is, the partitioning of data guided by the maximization of the cross-validated accuracy. At the beginning of this part, a fraction of the total samples (defined by FUN_SAM) are randomly selected from the original data. The whole iterative process (step I-III) is repeated M times to average the effects owing to the randomness of the iterative procedure. Each time that this part is repeated, a different fraction of sample is selected. The second part aims at collectioning and processing these results by costructing a dissimilarity matrix to provide a holistic view of the data while maintaining their intrinsic structure (steps IV and V).

Value

The function returns a list with 4 items:
dissimilarity  a dissimilarity matrix.
acc           a vector with the M cross-validated accuracies.
proximity     a proximity matrix.
v             a matrix containing the all classification obtained maximizing the cross-validation accuracy.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References

Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.

See Also

cmdscale

Examples

# data(iris)
# kk=KODAMA(iris[,,-5])
# pp = cmdscale(kk$dissimilarity)
# plot(pp,col=rep(2:4,each=50))
#
# WARNING: The next example is high computational extensive
#
# data(MetRef);
# u=MetRef$data;
# u=u[,which(colSums(u)==0)]
# u=scaling(u)$newXtrain
# class=as.factor(unlist(MetRef$donor))
# kk=KODAMA(u,FUN=PCA.CA.KNN.CV, W=function(x) as.numeric(kmeans(x,50)$cluster))
# pp = cmdscale(kk$dissimilarity)
# plot(pp,col=class)
# pp = cmdscale(kk$dissimilarity)
# plot(pp,col=class)
Description

This dataset consists of gene expression profiles of the three most prevalent adult lymphoid malignancies: diffuse large B-cell lymphoma (DLBCL), follicular lymphoma (FL), and B-cell chronic lymphocytic leukemia (B-CLL). The dataset consists of 4,682 genes in 62 mRNA samples: 42 samples of DLBCL, 9 samples of FL, and 11 samples of B-CLL. Missing value are imputed and data are standardized as described in Dudoit, et al. (2002).

Usage

data(lymphoma)

Value

A list with the following elements:

data  Gene expression data. A matrix with 62 rows and 4,682 columns.
class Class index. A vector with 62 elements.

References

Alizadeh AA, Eisen MB, Davis RE, et al.
Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. 

Dudoit S, Fridlyand J, Speed TP.
Comparison of discrimination methods for the classification of tumors using gene expression data. 

Examples

data(lymphoma)
class=1+as.numeric(as.factor(unlist(lymphoma$class)))
cc=cmdscale(dist(lymphoma$data))
plot(cc,pch=21,bg=class,xlab="First Component",ylab="Second Component");

# WARNING: The next example is high computational extensive
#
# kk=KODAMA(lymphoma$data,FUN=PLS.SVM.CV)
# cc=cmdscale(kk$dissimilarity)
# plot(cc,pch=21,bg=class,xlab="First Component",ylab="Second Component");
**majority**  
*Determines Majority Class*

**Description**

This function determines the majority class of a vector (treated as factor).

**Usage**

```r
majority(x)
```

**Arguments**

- `x`: a vector.

**Details**

This function treats the input vector as a factor and determines which level (class) of the factor is present most often. If two or more levels tie for majority then a random selection is made among the ties.

**Value**

The factor level which occurs most often in `x`.

**Author(s)**

Atina Dunlap Brooks

**See Also**

`factor`

**Examples**

```r
x <- sample(c("a", "b", "c", "d", "e", "f"), 10, replace=TRUE)
majority(x)
```
Nuclear Magnetic Resonance Spectra of Urines

Description

Nuclear magnetic resonance spectra of urines. The data belong to a cohort of 22 healthy donors (11 male and 11 female) where each provided about 40 urine samples over the time course of approximately 2 months, for a total of 873 samples.

Usage

data(MetRef)

Value

A list with the following elements:

data Gene expression data. A matrix with 873 rows and 450 columns.
gender Gender index. A vector with 873 elements.
donor Donor index. A vector with 873 elements.

References


Examples

data(MetRef)
u=MetRef$data;
u=u[,which(colSums(u)==0)]
u=scaling(u)$newXtrain
class=as.numeric(as.factor(MetRef$gender))
c=cmdscale(dist(u))
plot(cc,pch=21,bg=class,xlab="First Component",ylab="Second Component");

Normalization methods

Description

Collection of Different Normalization Methods.

Usage

normalization(Xtrain,Xtest=NULL, method = "pqn",ref=NULL)
Arguments

Xtrain  a matrix of data (training data set).
Xtest   a matrix of data (test data set). (by default = NULL).
method  the normalization method to be used. Choices are "none". "pqn". "sum". "median", "sqrt" (by default = "pqn"). A partial string sufficient to uniquely identify the choice is permitted.
ref     Reference sample for Probabilistic Quotient Normalization. (by default = NULL).

Details

A number of different normalization methods are provided:

- "none": no normalization method is applied.
- "pqn": the Probabilistic Quotient Normalization is computed as described in Dieterle, et al. (2006).
- "sum": samples are normalized to the sum of the absolute value of all variables for a given sample.
- "median": samples are normalized to the median value of all variables for a given sample.
- "sqrt": samples are normalized to the root of the sum of the squared value value of all variables for a given sample.

Value

The function returns a list with 2 items or 4 items (if a test data set is present):

newXtrain   a normalized matrix (training data set).
coeXtrain   a vector of normalization coefficient of the training data set.
newXtest    a normalized matrix (test data set).
coeXtest    a vector of normalization coefficient of the test data set.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References

Dieterle F,Ross A, Schlotterbeck G, Senn H.
Probabilistic Quotient Normalization as Robust Method to Account for Dilution of Complex Biological Mixtures. Application in 1H NMR Metabolomics.

See Also

scaling
Description

This function performs a 10-fold cross validation on a given data set using k nearest neighbors (kNN) classifier. The kNN classifier is performed on the score of the Principal Component Analysis (PCA) and Canonical Analysis (CA). The output is a vector of predicted labels.

Usage

```
PCA.CA.KNN.CV(x, cl, constrain, kn=10, variance=0.9)
```

Arguments

- `x`: a matrix.
- `cl`: a classification vector.
- `constrain`: a vector of `nrow(data)` elements. Sample with the same identificative constrain will be split in the training set or in the test test of cross-validation together.
- `kn`: the number of nearest neighbors to consider.
- `variance`: the number of the principal component of the PCA is selected on the base of the amount of selected variance. (by default = 0.9).

Details

PCA-CA-kNN classifier was used successfully in Wallner-Liebmann, et al. (2012) and Saccenti, et al. (2012) to classify metabolomic data.

Value

The function returns a vector of predicted labels.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References

Wallner-Liebmann S, Gralka E, Tenori L et al.
The impact of free or standardized lifestyle and urine sampling protocol on metabolome recognition accuracy.
*Genes Nutr* 2015;10:441

Saccenti E, Tenori L, Verbruggen P, et al.
Of monkeys and men: a metabolomic analysis of static and dynamic urinary metabolic phenotypes in two species.
PLS.SVM.CV

Cross-Validation with Support Vector Machine.

Description

This is function performs a 10-fold cross validation on a given data set using the Support Vector Machine (SVM) classifier. The SVM classifier is performed on the score of the Partial least squares (PLS). The output is a vector of predicted labels.

Usage

PLS.SVM.CV(x, cl, constrain, ncomp=5, ...)

Arguments

x a matrix.
cl a classification vector.
constrain a vector of nrow(data) elements. Sample with the same identificative constrain will be split in the training set or in the test test of cross-validation together.
ncomp number of component of PLS to consider.
... parameter for svm classifier.

details

PLS-SVM classifier was used successfully in Bertini, et al. (2012) and Aimetti, et al. (2012) to classify metabolomic data.

Value

The function returns a vector of predicted labels.

Author(s)

Stefano Cacciatore and Leonardo Tenori
scaling

References

Bertini I, Cacciatore S, Jensen BV, et al.
Metabolomic NMR fingerprinting to identify and predict survival of patients with metastatic colorectal cancer.

Aimetti M, Cacciatore S, Graziano A, Tenori L.
Metabonomic analysis of saliva reveals generalized chronic periodontitis signature.

See Also

KNN.CV,PCA.CA,KNN.CV

Examples

data(lymphoma)
class=as.factor(unlist(lymphoma$class))
results=PLS.SVM.CV(lymphoma$data,class,1:length(class))
levels(results)=levels(class)
table(results,class)

scaling Scaling methods

Description

Collection of Different Scaling Methods.

Usage

scaling(Xtrain,Xtest=NULL, method = "autoscaling")

Arguments

Xtrain a matrix of data (training data set).
Xtest a matrix of data (test data set). (by default = NULL).
method the scaling method to be used. Choices are "none", "centering", "autoscaling", "rangescaling", "paretoscaling" (by default = "autoscaling"). A partial string sufficient to uniquely identify the choice is permitted.
Details

A number of different scaling methods are provided:

- "none": no scaling method is applied.
- "centering": it centers the mean to zero.
- "autoscaling": it centers the mean to zero and scales data by dividing each variable by the variance.
- "rangescaling": it centers the mean to zero and scales data by dividing each variable by the difference between the minimum and the maximum value.
- "paretoscaling": it centers the mean to zero and scales data by dividing each variable by the square root of the standard deviation. Unit scaling divides each variable by the standard deviation so that each variance equal to 1.

Value

The function returns a list with 1 item or 2 items (if a test data set is present):

- `newXtrain`: a scaled matrix (training data set).
- `newXtest`: a scaled matrix (test data set).

Author(s)

Stefano Cacciatore and Leonardo Tenori

References


See Also

- `normalization`

---

**spirals**  
*Spirals Data Set Generator*

Description

Produce a data set of spiral clusters.

Usage

`spirals(n=c(100,100,100),sd=c(0,0,0))`
Arguments

n a vector of integer. The length of the vector is the number of clusters and each number corresponds to the number of data points in each cluster.

sd amount of noise for each spiral.

Value

The function returns a two dimensional data set.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References

Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.

See Also

helicoid, dinisurface, swissroll

Examples

par(mfrow=c(2,2))
v1=spirals(c(100,100,100),c(0.1,0.1,0.1))
plot(v1,col=rep(2:4,each=100))
v2=spirals(c(100,100,100),c(0.1,0.2,0.3))
plot(v2,col=rep(2:4,each=100))
v3=spirals(c(100,100,100,100),c(0,0.2,0,0))
plot(v3,col=rep(2:6,each=100))
v4=spirals(c(20,40,60,80,100),c(0.1,0.1,0.1,0.1,0.1))
plot(v4,col=rep(2:6,c(20,40,60,80,100)))
Value

The function returns a three dimensional matrix.

Author(s)

Stefano Cacciatore and Leonardo Tenori

References

Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.

Balasubramanian M, Schwartz EL.
The isomap algorithm and topological stability.

Roweis ST, Saul LK.
Nonlinear dimensionality reduction by locally linear embedding.

See Also

helicoid,dinisurface,spirals

Examples

require("rgl")
x=swissroll()
open3d()
plot3d(x, col=rainbow(1050)[-c(1:50)], box=FALSE, type="s", size=1)

---

transformy

Conversion Classification Vector to Matrix

Description

This function converts a classification vector into a classification matrix.

Usage

transformy(y)

Arguments

y  
a vector or factor.
Details

This function converts a classification vector into a classification matrix.

Value

A matrix.

Author(s)

Stefano Cacciatore and Leonardo Tenori

Examples

```r
y <- rep(1:10, 3)
y_transform(y)
```

---

Description

This dataset consists of the spoken, not written, addresses from 1900 until the sixth address by Barack Obama in 2014. Punctuation characters, numbers, words shorter than three characters, and stop-words (e.g., "that", "and", and "which") were removed from the dataset. This resulted in a dataset of 86 speeches containing 834 different meaningful words each. Term frequency-inverse document frequency (TF-IDF) was used to get the feature vectors. It is often used as a weighting factor in information retrieval and text mining. The TF-IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

Usage

```r
data(USA)
```

Value

A list with the following elements:

- `data` Gene expression data. A matrix with 86 rows and 834 columns.
- `year` Year index. A vector with 86 elements.
- `president` President index. A vector with 86 elements.

References

Cacciatore S, Luchinat C, Tenori L.
Knowledge discovery by accuracy maximization.
Examples

# Here is reported the analysis on the State of the Union
# of USA president as shown in Cacciatore, et al. (2014)
# WARNING: This example is high computational extensive
#
# data(USA)
# kk=KODAMA(USA$data)
# cc=cmdscale(kk$dissimilarity)
# par(cex=0.5,mar=c(15,6,2,2));
# plot(USA$year,cc[,1],axes=F,pch=20,xlab="",ylab="First Component");
# axis(1,at=USA$year,labels=rownames(USA$data),las=2);
# axis(2,las=2);
# box()