Package: HiDimDA (via r-universe)

October 25, 2026

Type Package

Title High Dimensional Discriminant Analysis

Version 0.2-7

Date 2024-10-06

Depends R (>= 2.10.0)

Imports splines

Suggests MASS

LazyLoad yes

LazyData yes

Description Performs linear discriminant analysis in high dimensional problems based on reliable covariance estimators for problems with (many) more variables than observations. Includes routines for classifier training, prediction, cross-validation and variable selection.

License GPL (>= 3)

NeedsCompilation yes

Author Antonio Pedro Duarte Silva [aut, cre]

Maintainer Antonio Pedro Duarte Silva <psilva@ucp.pt>

Repository https://cran.r-universe.dev

Date/Publication 2024-10-07 03:02:36 UTC

RemoteUrl https://github.com/cran/HiDimDA

RemoteRef HEAD

RemoteSha 1b9276e9d596e51796a372b0ab670410f3eb1804

Contents

HiDimDA-package 2
AlonDS 5
canldaRes 6
clldaRes 7

CovE 8
DACrossVal 9
Dlda 10
DMat 13
FrobSigAp 13
MatMult 15
Mlda 16
MldaInvE 19
RFlda 20
SelectV 23
ShrnkMat 26
ShrnkMatInv 27
ShrnkSigE 28
SigFq 29
SigFqInv 30
Slda 31
solve 34

Index **35**

`HiDimDA-package` *High Dimensional Discriminant Analysis*

Description

Performs Linear Discriminant Analysis in High Dimensional problems based on reliable covariance estimators for problems with (many) more variables than observations. Includes routines for classifier training, prediction, cross-validation and variable selection.

Details

Package:	HiDimDA
Type:	Package
Version:	0.2-7
Date:	2024-10-06
License:	GPL-3
LazyLoad:	yes
LazyData:	yes

HiDimDA is a package for High-Dimensional Discriminant Analysis aimed at problems with many variables, possibly much more than the number of available observations. Its core consists of the four Linear Discriminant Analyis routines:

Dlda:	Diagonal Linear Discriminant Analysis
Slda:	Shrunken Linear Discriminant Analysis
Mlda:	Maximum-uncertainty Linear Discriminant Analysis

RFlda: Factor-model Linear Discriminant Analysis

and the variable selection routine:

SelectV: High-Dimensional variable selection for supervised classification

that selects variables to be used in a Discriminant classification rule by ranking them according to two-sample t-scores (problems with two-groups), or ANOVA F-scores (problems wih more that two groups), and discarding those with scores below a threshold defined by the Higher Criticism (HC) approach of Donoho and Jin (2008), the Expanded Higher Criticism scheme proposed by Duarte Silva (2011), False Discovery Rate (Fdr) control as suggested by Benjamini and Hochberg (1995), the FAIR approach of Fan and Fan (2008), or simply by fixing the number of retained variables to some pre-defined constant.

All four discriminant routines, 'Dlda', 'Slda', 'Mlda' and 'RFlda', compute Linear Discriminant Functions, by default after a preliminary variable selection step, based on alternative estimators of a within-groups covariance matrix that leads to reliable allocation rules in problems where the number of selected variables is close to, or larger than, the number of available observations.

Consider a Discriminant Analysis problem with k groups, p selected variables, a training sample consisting of $N = \sum_{g=1}^{k} n_g$ observations with group and overall means, $\bar{X}_g$ and $\bar{X}_.$, and a between-groups scatter (scaled by degrees of freedom) matrix, $S_B = \frac{1}{N-k} \sum_{g=1}^{k} n_g (\bar{X}_g - \bar{X}_.)(\bar{X}_g - \bar{X}_.)^T$

Following the two main classical approaches to Linear Discrimant Analysis, the Discriminant Functions returned by HiDimDA discriminant routines are either based on the canonical linear discriminants given by the normalized eigenvectors

$$LD_j = Egvct_j(S_B \hat{\Sigma}_W^{-1})$$
$$j = 1, ..., r = min(p, k-1)$$
$$[LD_1, ..., LD_r]^T \hat{\Sigma}_W [LD_1, ..., LD_r] = I_r$$

or the classification functions

$$CF_g = (\bar{X}_g - \bar{X}_1)\hat{\Sigma}_W^{-1}$$
$$g = 2, ..., k$$

where $\hat{\Sigma}_W^{-1}$ is an estimate of the inverse within-groups covariance.

It is well known that these two approaches are equivalent, in the sense that classification rules that assign new observations to the group with the closest (according to the Euclidean distance) centroid in the space of the canonical variates, $Z = [LD_1 ... LD_r]^T X$, give the same results as the rule that assigns a new observation to group 1 if all classification scores, $Clscr_g = CF_g^T X - CF_g^T \frac{(\bar{X}_1 + \bar{X}_g)}{2}$, are negative, and to the group with the highest classification score otherwise.

The discriminant routines of HiDimDA compute canonical linear discriminant functions by default, and classification functions when the argument 'ldafun' is set to "classification". However, unlike

traditional linear discriminant analysis where Σ_W^{-1} is estimated by the inverse of the sample covariance, which is not well-defined when $p \geq N - k$ and is unreliable if p is close to $N - k$, the routines of HiDimDA use four alternative well-conditioned estimators of Σ_W^{-1} that lead to reliable classification rules if p is larger than, or close to, $N - k$.

In particular, 'Dlda' estimates Σ_W^{-1} by the diagonal matrix of inverse sample variances, 'Slda' by the inverse of an optimally shrunken Ledoit and Wolf's (2004) covariance estimate with the targets and optimal target intensity estimators proposed by Fisher and Sun (2011), 'Mlda' uses a regularized inverse covariance that deemphasizes the importance given to the last eigenvectors of the sample covariance (see Thomaz, Kitani and Gillies (2006) for details), and 'RFlda' uses a factor model estimate of the true inverse correlation (or covariance) matrix based on the approach of Duarte Silva (2011).

The HiDimDA package also includes predict methods for all discriminant routines implemented, a routine ('DACrossVal') for asssessing the quality of the classification results by kfold cross-validation, and utilities for storing, extracting and efficiently handling specialized high-dimensional covariance and inverse covariance matrix estimates.

Author(s)

Antonio Pedro Duarte Silva <psilva@porto.ucp.pt>

Maintainer: Antonio Pedro Duarte Silva <psilva@porto.ucp.pt>

References

Benjamini, Y. and Hochberg, Y. (1995) "Controling the false discovery rate: A practical and powerful approach to multiple testing", *Journal of the Royal Statistical Society* B, 57, 289-300.

Donoho, D. and Jin, J. (2008) "Higher criticism thresholding: Optimal feature selection when useful features are rare and weak", In: *Proceedings National Academy of Sciences*, USA 105, 14790-14795.

Fan, J. and Fan, Y. (2008) "High-dimensional classification using features annealed independence rules", *Annals of Statistics*, 36 (6), 2605-2637.

Fisher, T.J. and Sun, X. (2011) "Improved Stein-type shrinkage estimators for the high-dimensional multivariate normal covariance matrix", *Computational Statistics and Data Analysis*, 55 (1), 1909-1918.

Ledoit, O. and Wolf, M. (2004) "A well-conditioned estimator for large-dimensional covariance matrices.", *Journal of Multivariate Analysis*, 88 (2), 365-411.

Pedro Duarte Silva, A. (2011) "Two Group Classification with High-Dimensional Correlated Data: A Factor Model Approach", *Computational Statistics and Data Analysis*, 55 (1), 2975-2990.

Thomaz, C.E. Kitani, E.C. and Gillies, D.F. (2006) "A maximum uncertainty LDA-based approach for limited sample size problems - with application to face recognition", *Journal of the Brazilian Computer Society*, 12 (2), 7-18

See Also

`Dlda`, `Mlda`, `Slda`,`RFlda`, `predict.canldaRes`, `predict.clldaRes`, `AlonDS`

Examples

```
# train the four main classifiers with their default setings
# on Alon's colon data set (after a logarithmic transformation),
# selecting genes by the Expanded HC scheme

# Pre-process and select the genes to be used in the classifiers

log10genes <- log10(AlonDS[,-1])
SelectionRes <- SelectV(log10genes,AlonDS$grouping)
genesused <- log10genes[SelectionRes$vkpt]

# Train classifiers

DiaglldaRule <- Dlda(genesused,AlonDS$grouping)
FactldaRule <- RFlda(genesused,AlonDS$grouping)
MaxUldaRule <- Mlda(genesused,AlonDS$grouping)
ShrkldaRule <- Slda(genesused,AlonDS$grouping)

# Get in-sample classification results

predict(DiaglldaRule,genesused,grpcodes=levels(AlonDS$grouping))$class
predict(FactldaRule,genesused,grpcodes=levels(AlonDS$grouping))$class
predict(MaxUldaRule,genesused,grpcodes=levels(AlonDS$grouping))$class
predict(ShrkldaRule,genesused,grpcodes=levels(AlonDS$grouping))$class

# Compare classifications with true assignments

cat("Original classes:\n")
print(AlonDS$grouping)

# Show set of selected genes

cat("Genes kept in discrimination rule:\n")
print(colnames(genesused))
cat("Number of selected genes =",SelectionRes$nvkpt,"\n")
```

AlonDS *Alon Colon Cancer Data Set*

Description

This data set was collected by Alon *et. al.* and consists of 2000 genes measured on 62 patients: 40 diagnosed with colon cancer and 22 healthy patients. The patient status is described by the factor 'grouping' and the gene values are given by the numeric variables 'genes.1' through 'genes.2000'.

Usage

```
data(AlonDS)
```

Format

A data frame containing 62 observations on one factor and 2000 numeric variables.

Source

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D. and Levine, A.J. (1999) "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays", In: *Proceedings National Academy of Sciences* USA 96, 6745-6750. The data set is available at http://microarray.princeton.edu/oncology.

`canldaRes` *Class object used for storing the results of a canonical high-dimensional linear discriminant analysis.*

Description

'predict.canldaRes' classifies multivariate observations in conjunction with a 'canldaRes' object.

'print.canldaRes' is the S3 print method for 'canldaRes' objects.

'coef.canldaRes' is the S3 coef method for 'canldaRes' objects.

Usage

```
## S3 method for class 'canldaRes'
predict(object, newdata, prior=object$prior, grpcodes=NULL,
nbvrs=ncol(object$scaling), ...)
```

Arguments

`object`	An object of class 'canldaRes'.
`newdata`	Matrix of cases to be classified.
`prior`	The prior probabilities used.
`grpcodes`	Factor with the class codes. Set to 0:k-1 (k being the number of different classes) by default.
`nbvrs`	Number of canonical discriminant variables used for prediction.
`...`	Further arguments passed to or from other methods.

Value

A list with components

`class`	The MAP classification (a factor)
`ZsqDistances`	A matrix with the squared Euclidean distance, in the discriminant space, of each new observation to the group centroids.
`prior`	The prior probabilities used.

ZsqDprioradj	The adjustments to squared Euclidean distance, induced by the chosen (or estimated) prior probabilities.
Z	A matrix with the values of the canonical discriminant variates.
Zmeans	A matrix with the group centroids in the canonical space.

See Also

Dlda, Mlda, Slda, RFlda, print, coef

---

clldaRes *Class object used for storing the results of a high-dimensional linear discriminant analysis routine (with 'ldafun' argument set to "classification").*

Description

'predict.clldaRes' Classifies multivariate observations in conjunction with a 'clldaRes' object.

'print.clldaRes' is the S3 print method for 'clldaRes' objects.

'coef.clldaRes' is the S3 coef method for 'clldaRes' objects.

Usage

```
## S3 method for class 'clldaRes'
predict(object, newdata, prior=object$prior, grpcodes=NULL,...)
```

Arguments

object	An object of class 'clldaRes'.
newdata	Matrix of cases to be classified.
prior	The prior probabilities used.
grpcodes	Factor with the class codes. Set to 0:k-1 (k being the number of different classes) by default.
...	Further arguments passed to or from other methods.

Value

A list with components

class	The MAP classification (a factor)
x	The classification scores of the test cases.

See Also

Dlda, Mlda, Slda, RFlda, print, coef

CovE *Generic methods for extracting covariance and inverse covariance matrices from objects storing the results of a Linear Discriminant Analysis*

Description

'CovE' Extracts an object with an appropriate representation of a whithin groups covariance matrix from a 'Scanlda', 'Scllda', 'RFcanlda' or a 'RFcllda' object.

'ICovE' Extracts an object with an appropriate representation of a whithin groups inverse covariance matrix from a 'Scanlda', 'Scllda', 'RFcanlda' or a 'RFcllda' object.

Usage

```
## S3 method for class 'Scanlda'
CovE(object)
## S3 method for class 'Scanlda'
ICovE(object)
## S3 method for class 'RFcanlda'
CovE(object)
## S3 method for class 'RFcanlda'
ICovE(object)
## S3 method for class 'Scllda'
CovE(object)
## S3 method for class 'Scllda'
ICovE(object)
## S3 method for class 'RFcllda'
CovE(object)
## S3 method for class 'RFcllda'
ICovE(object)
```

Arguments

object	An object of class 'Scanlda', 'Scllda', 'RFcanlda' or 'RFcllda'.

Value

An object with an appropriate representation of the matrix extracted.

See Also

Slda, RFlda, canldaRes, clldaRes

DACrossVal *Cross Validation for Discriminant Analysis Classification Algorithms*

Description

'DACrossVal' evaluates the performance of a Discriminant Analysis training algorithm by kfold Cross-Validation.

Usage

```
DACrossVal(data, grouping, TrainAlg, EvalAlg=EvalClrule,
Strfolds=TRUE, kfold=10, CVrep=20, prior="proportions", ...)
```

Arguments

data	Matrix or data frame of observations.
grouping	Factor specifying the class for each observation.
TrainAlg	A function with the training algorithm. It should return an object that can be used as input to the argument of 'EValAlg'.
EvalAlg	A function with the evaluation algorithm. By default set to 'EvalClrule' which returns a list with components "err" (estimates of error rates by class) and "Ng" (number of out-sample observations by class). This default can be used for all 'TrainAlg' arguments that return an object with a predict method returning a list with a 'class' component (a factor) containing the classification results.
Strfolds	Boolean flag indicating if the folds should be stratified according to the original class proportions (default), or randomly generated from the whole training sample, ignoring class membership.
kfold	Number of training sample folds to be created in each replication.
CVrep	Number of replications to be performed.
prior	The prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
...	Further arguments to be passed to 'TrainAlg' and 'EvalAlg'.

Value

A three dimensional array with the number of holdout observations, and estimated classification errors for each combination of fold and replication tried. The array dimensions are defined as follows:

The first dimension runs through the different fold-replication combinations.

The second dimension represents the classes.

The third dimension has two named levels representing respectively the number of holdout observations ("Ng"), and the estimated classification errors ("Clerr").

Author(s)

A. Pedro Duarte Silva

See Also

RFlda, predict.clldaRes

Examples

```
# Evaluate the performance of traditional (Fisher's) linear discriminant
# analysis on the iris data set, by ten-fold cross-validation replicated
# three times.

library(MASS)
CrosValRes1 <- DACrossVal(iris[1:4],iris$Species,TrainAlg=lda,CVrep=3)
summary(CrosValRes1[,,"Clerr"])

# Evaluate the performance on Alon's Colon Cancer Data set
# (with a logarithmic transformation), of a one-factor
# linear discriminant rule with the best fifty genes,
# by four-fold cross-validation.

## Not run:

CrosValRes2 <- DACrossVal(log10(AlonDS[,-1]),AlonDS$grouping,TrainAlg=RFlda,
ldafun="classification",Selmethod="fixedp",maxp=50,kfold=4,CVrep=1)
summary(CrosValRes2[,,"Clerr"])


## End(Not run)
```

Dlda *Diagonal Linear Discriminant Analysis.*

Description

'Dlda' finds the coefficients of a linear discriminant rule based on a Diagonal covariance matrix estimator.

Usage

```
## Default S3 method:
Dlda(data, grouping, prior = "proportions", VSelfunct = SelectV,
ldafun=c("canonical","classification"), ...)

## S3 method for class 'data.frame'
Dlda(data, ...)
```

Arguments

data	Matrix or data frame of observations.
grouping	Factor specifying the class for each observation.
prior	The prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
VSelfunct	Variable selection function. Either the string "none" (no selection is to be performed) or a function that takes 'data' and 'grouping' as its first two arguments and returns a list with two components: (i) 'nvkpt' - the number of variables to be used in the Discriminant rule; and (ii) 'vkptInd' - the indices of the variables to be used in the Discriminant rule. The default is the 'SelectV' function that, by default, selects variables by the Expanded HC scheme described in Duarte Silva (2011).
ldafun	Type of discriminant linear functions computed. The alternatives are "canonical" for maximum-discrimination canonical linear functions and "classification" for direct-classification linear functions.
...	Further arguments passed to or from other methods.

Value

If algument 'ldafun' is set to "canonical" an object of class 'canldaRes' with the following components:

prior	The prior probabilities used.
means	The class means.
scaling	A matrix which transforms observations to discriminant functions, normalized so that the within groups covariance matrix is spherical.
svd	The singular values, which give the ratio of the between- and within-group standard deviations on the linear discriminant variables. Their squares are the canonical F-statistics.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL otherwise.
N	The number of observations used.
call	The (matched) function call.

If algument 'ldafun' is set to "classification" an object of class 'clldaRes' with the following components:

prior	The prior probabilities used.
means	The class means.
coef	A matrix with the coefficients of the k-1 classification functions.
cnst	A vector with the thresholds (2nd members of linear classification rules) used in classification rules that assume equal priors.

vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL otherwise.
N	The number of observations used.
call	The (matched) function call.

Author(s)

A. Pedro Duarte Silva

See Also

SelectV, DMat, predict.canldaRes, predict.clldaRes, AlonDS

Examples

```
# train classifier on Alon's Colon Cancer Data Set
#(after a logarithmic transformation).

log10genes <- log10(AlonDS[,-1])


ldarule <- Dlda(log10genes,AlonDS$grouping)

# show classification rule

print(ldarule)

# get in-sample classification results

predict(ldarule,log10genes,grpcodes=levels(AlonDS$grouping))$class

# compare classifications with true assignments

cat("Original classes:\n")
print(AlonDS$grouping)

# Estimate error rates by four-fold cross-validation.
# Note: In cross-validation analysis it is recommended to set
# the argument 'ldafun' to "classification", in order to speed up
# computations by avoiding unecessary eigen-decompositions

## Not run:

CrosValRes <- DACrossVal(log10genes,AlonDS$grouping,TrainAlg=Dlda,
ldafun="classification",kfold=4,CVrep=1)
summary(CrosValRes[,,"Clerr"])


## End(Not run)
```

DMat *DMat objects: diagonal matrices*

Description

Creates a 'DMat' object.

Usage

```
DMat(D)
```

Arguments

D	A vector with the diagonal elements of the matrix.

Value

An object of class 'DMat' for which the generic method 'as.matrix' (converting to a traditional numeric matrix), as well as specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations, are available.

See Also

solve.DMat, LeftMult.DMat, RightMult.DMat

FrobSigAp *Approximation of Covariance Matrices from q-factor models*

Description

'FrobSigAp' finds the parameters of a q-factor model whose covariance is closest to the matrix argument 'Sigma', according to the Frobenius norm.

'FrobSigAp1' finds the parameters of a q-factor model whose covariance is closest to the matrix square `SigmaSr %*% t(SigmaSr)` of the argument 'SigmaSr', according to the Frobenius norm.

Usage

```
FrobSigAp(Sigma, q, nstarts = 1, k0 = NULL, penF = NULL,
atol = 1e-20, rtol = sqrt(.Machine$double.eps))

FrobSigAp1(SigmaSr, SigmaRank, q, nstarts = 1, k0=NULL, penF=NULL,
atol = 1e-20, rtol = 100 * sqrt(.Machine$double.eps))
```

Arguments

Sigma	Square, symmetric and positive-definite matrix to be approximated
q	Number of factors in the assumed factor model.
nstarts	Number of different randomly generated starting points used in the optimization.
k0	Lower bound for the elements of the specific variances. When NULL (default), k0 is set to 0.01 times the minimum diagonal element of 'Sigma'.
penF	Penalty factor, used to forbid specific variances below the k0 bound. When set to NULL (default), a penalty equal to 100 times the maximum diagonal element of 'Sigma' is used.
atol	The absolute convergence tolerance of the local optimizer.
rtol	The relative convergence tolerance of the local optimizer. The local optimizer stops if it is unable to reduce the approximation error (err) by a factor of 'reltol *(abs(err) + reltol)' at a step.
SigmaSr	Matrix square root of the covariance to be approximated.
SigmaRank	Rank of the covariance matrix to be approximated.

Details

The minimization of the error Frobenius norm is performed by the 'nlminb' PORT optimization routine. The actual computations of the errors, and their analytical gradients and hessians, are coded in C in order to speed up the algorithm.
The minimization procedure takes the loadings (B) of the factor model as arguments, computes the optimal specific variances by the analytical formula `D = diag(Sigma) - diag(B %*% t(B))`, and uses the first q eigenvectors of 'Sigma' as starting points of the optimization.
For small values of q (1 or 2), this procedure seems to quickly converge to the global minimum of the approximation error. For larger values of q, the computational time can be much higher and multiple random starting points (that can be specified by the argument 'nstarts') may be required in order to escape local optima.

Value

An object of class 'SigFq' representing the covariance assumed by the closest q-factor model. 'SigFq' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.

Author(s)

A. Pedro Duarte Silva

References

Pedro Duarte Silva, A. (2011) "Two Group Classification with High-Dimensional Correlated Data: A Factor Model Approach", *Computational Statistics and Data Analysis*, 55 (1), 2975-2990.

See Also

RFlda, SigFq, SigFqInv, solve.SigFq, solve.SigFqInv

MatMult *MatMult: Specialized matrix multiplication of 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' and 'SigFqInv' objects.*

Description

'LeftMult' multiplies, on the left, a vector or matrix of compatible dimensions, by a 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' or 'SigFqInv' object.

'RightMult' multiplies, on the right, a vector or matrix of compatible dimensions, by a 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' or 'SigFqInv' object.

Usage

```
## S3 method for class 'DMat'
LeftMult(x, a)
## S3 method for class 'DMat'
RightMult(x, a)
## S3 method for class 'ShrnkMat'
LeftMult(x, a)
## S3 method for class 'ShrnkMat'
RightMult(x, a)
## S3 method for class 'ShrnkMatInv'
LeftMult(x, a)
## S3 method for class 'ShrnkMatInv'
RightMult(x, a)
## S3 method for class 'SigFq'
LeftMult(x, a)
## S3 method for class 'SigFq'
RightMult(x, a)
## S3 method for class 'SigFqInv'
LeftMult(x, a)
## S3 method for class 'SigFqInv'
RightMult(x, a)
```

Arguments

x	An object of class 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' or class 'SigFqInv', with a compact representation of a specialized square symmetric matrix.
a	A vector, or matrix, by which 'x' is to be multiplied.

Value

A vector or a (traditional numeric) matrix with the result of the matrix product.

See Also

DMat, FrobSigAp, ShrnkMat, ShrnkMatInv, SigFq, SigFqInv, solve.DMat, solve.ShrnkMat, solve.ShrnkMatInv, solve.SigFq, solve.SigFqInv

---

Mlda *Maximum uncertainty Linear Discriminant Analysis.*

Description

'Mlda' finds the coefficients of a linear discriminant rule based on the "Maximum uncertainty Linear Discriminant Analysis" approach of Thomaz, Kitani and Gillies (2006).

Usage

```
## Default S3 method:
Mlda(data, grouping, prior = "proportions", StddzData=TRUE,
VSelfunct = SelectV, ldafun=c("canonical","classification"),
PCAstep=FALSE, ...)

## S3 method for class 'data.frame'
Mlda(data, ...)
```

Arguments

data	Matrix or data frame of observations.
grouping	Factor specifying the class for each observation.
prior	The prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
StddzData	A boolean flag indicating whether the data should be standardized first (default) or used in their original scales.
VSelfunct	Variable selection function. Either the string "none" (no selection is to be performed) or a function that takes 'data' and 'grouping' as its first two arguments and returns a list with two components: (i) 'nvkpt' - the number of variables to be used in the Discriminant rule; and (ii) 'vkptInd' - the indices of the variables to be used in the Discriminant rule. The default is the 'SelectV' function that, by default, selects variables by the Expanded HC scheme described in Duarte Silva (2011).
ldafun	Type of discriminant linear functions computed. The alternatives are "canonical" for maximum-discrimination canonical linear functions and "classification" for direct-classification linear functions.

PCAstep	A flag indicating if data should be first projected into the space spanned by its first nrow(data)-1 Principal Components in problems where nrow(data)-1 is less than the number of selected variables. In applications with a very large number of useful variables seting PCAstep to TRUE avoids many potential memory problems and tends to substantially increase the size of the data sets that can be analyzed by Mlda.
...	Further arguments passed to or from other methods.

Value

If algument 'ldafun' is set to "canonical" an object of class 'canldaRes' with the following components:

prior	The prior probabilities used.
means	The class means.
scaling	A matrix which transforms observations to discriminant functions, normalized so that the within groups covariance matrix is spherical.
svd	The singular values, which give the ratio of the between- and within-group standard deviations on the linear discriminant variables. Their squares are the canonical F-statistics.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL otherwise.
N	The number of observations used.
call	The (matched) function call.

If algument 'ldafun' is set to "classification" an object of class 'clldaRes' with the following components:

prior	The prior probabilities used.
means	The class means.
coef	A matrix with the coefficients of the k-1 classification functions.
cnst	A vector with the thresholds (2nd members of linear classification rules) used in classification rules that assume equal priors.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL, otherwise.
N	The number of observations used.
call	The (matched) function call.

Author(s)

A. Pedro Duarte Silva

References

Pedro Duarte Silva, A. (2011) "Two Group Classification with High-Dimensional Correlated Data: A Factor Model Approach", *Computational Statistics and Data Analysis*, 55 (1), 2975-2990.

Thomaz, Kitani and Gillies (2006) "A maximum uncertainty LDA-based approach for limited sample size problems - with application to face recognition", *Journal of the Brazilian Computer Society*, 12 (2), 7-18.

See Also

SelectV, MldaInvE, predict.canldaRes, predict.clldaRes, AlonDS

Examples

```
# train classifier on Alon's Colon Cancer Data Set
# (after a logarithmic transformation).

log10genes <- log10(AlonDS[,-1])

ldarule <- Mlda(log10genes,AlonDS$grouping)

# show classification rule

print(ldarule)

# get in-sample classification results

predict(ldarule,log10genes,grpcodes=levels(AlonDS$grouping))$class

# compare classifications with true assignments

cat("Original classes:\n")
print(AlonDS$grouping)

# Estimate error rates by four-fold cross-validation.
# Note: In cross-validation analysis it is recommended to set
# the argument 'ldafun' to "classification", in order to speed up
# computations by avoiding unecessary eigen-decompositions

## Not run:

CrosValRes <- DACrossVal(log10genes,AlonDS$grouping,TrainAlg=Mlda,
ldafun="classification",kfold=4,CVrep=1)
summary(CrosValRes[,,"Clerr"])


## End(Not run)
```

`MldaInvE` *Maximum uncertainty Linear Discriminant Analysis inverse matrix estimator.*

Description

Builds a well-conditioned estimator for the inverse of a symmetric positive definite matrix, with a bad-conditioned or singular estimate, based on the "Maximum Uncertainty Linear Discriminant Analysis" approach of Thomaz, Kitani and Gillies (2006).

Usage

```
MldaInvE(M, check=TRUE, onlyMinv=TRUE,
numtol=sqrt(.Machine$double.eps))
```

Arguments

`M`	Singular or bad-conditioned estimate of the matrix for which a well-conditioned inverse estimate is sought.
`check`	Boolean flag indicating if the symmetry of M and the sign of its eigenvalues should be check upfront.
`onlyMinv`	Boolean flag indicating if only an estimate of the matrix inverse is sought, or if a well-conditioned approximation to the matrix that M estimates should be returned as well.
`numtol`	Numerical tolerance.

Value

If onlyMinv is set to true a matrix with the inverse estimate sought. Otherwise a list with components ME and MInvE, with a well-conditioned approximation to the matrix that M estimates and its inverse.

Author(s)

A. Pedro Duarte Silva

References

Thomaz, Kitani and Gillies (2006) "A maximum uncertainty LDA-based approach for limited sample size problems - with application to face recognition", *Journal of the Brazilian Computer Society*, 12 (2), 7-18

See Also

`Mlda`

RFlda *High-Dimensional Factor-based Linear Discriminant Analysis.*

Description

'RFlda' finds the coefficients of a linear discriminant rule based on a correlation (or covariance) matrix estimator that tries to approximate the true correlation (covariance) by the closest (according to a Frobenius norm) correlation (covariance) compatible with a q-factor model.

Usage

```
## Default S3 method:
RFlda(data, grouping, q = 1, prior = "proportions",
CorrAp = TRUE, maxq=5, VSelfunct = SelectV,
ldafun=c("canonical","classification"), nstarts = 1,
CVqtrials=1:3, CVqfolds=3, CVqrep=1, CVqStrt=TRUE, ...)

## S3 method for class 'data.frame'
RFlda(data, ...)
```

Arguments

data	Matrix or data frame of observations.
grouping	Factor specifying the class for each observation.
q	Number of factors assumed by the model. This argument can be set to a fixed number between 1 and the argument of 'maxq', or to the string "CVq". In the latter case the number of factors is chosen amongst the values of the argument 'CVqtrials', by minimizing a cross-validated estimate of the error rate.
prior	The prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
CorrAp	A boolean flag indicating whether the approximation error of the correlation (default), or of the covariance matrix , should be minimized.
maxq	Upper limit on the values allowed for argument 'q'.
VSelfunct	Variable selection function. Either the string "none" (no selection is to be performed) or a function that takes 'data' and 'grouping' as its first two arguments and returns a list with two components: (i) 'nvkpt' - the number of variables to be used in the Discriminant rule; and (ii) 'vkptInd' - the indices of the variables to be used in the Discriminant rule. The default is the 'SelectV' function that, by default, selects variables by the Expanded HC scheme described in Duarte Silva (2011).
ldafun	Type of discriminant linear functions computed. The alternatives are "canonical" for maximum-discrimination canonical linear functions and "classification" for direct-classification linear functions.

nstarts	Number of different randomly generated starting points used in the minimization of the Frobenius norm of the correlation (or covariance) matrix approximation.
CVqtrials	Vector of values to be tried for the number of factors assumed by the model, when argument 'q' is set to "CVq".
CVqfolds	Number of training sample folds to be created in each replication of the cross-validation procedure for choosing the number of factors, when argument 'q' is set to "CVq".
CVqrep	Number of replications to be performed in the cross-validation procedure for choosing the number of factors, when argument 'q' is set to "CVq".
CVqStrt	Boolean flag indicating if, in the cross-validation procedure for choosing the number of factors when argument 'q' is set to "CVq", the folds should be stratified according to the original class proportions (default), or randomly generated from the whole training sample ignoring class membership.
...	Further arguments passed to or from other methods.

Value

If algument 'ldafun' is set to "canonical" an object of class 'RFcanlda', which extends class 'canldaRes', with the following components:

prior	The prior probabilities used.
means	The class means.
scaling	A matrix which transforms observations to discriminant functions, normalized so that the within groups covariance matrix is spherical.
svd	The singular values, which give the ratio of the between- and within-group standard deviations on the linear discriminant variables. Their squares are the canonical F-statistics.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL otherwise.
q	The number of o factors used in the factor model chosen.
SigFq	An object of class 'SigFq' with the q-factor model approximation to the within groups covariance matrix. 'SigFq' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
SigFqInv	An object of class 'SigFqInv' with the q-factor model approximation to the within groups precision (inverse covariance) matrix. 'SigFqInv' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
N	The number of observations used.
call	The (matched) function call.

If algument 'ldafun' is set to "classification" an object of class 'RFcllda', which extends class 'clldaRes', with the following components:

prior	The prior probabilities used.
means	The class means.
coef	A matrix with the coefficients of the k-1 classification functions.
cnst	A vector with the thresholds (2nd members of linear classification rules) used in classification rules that assume equal priors.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL, otherwise.
q	The number of o factors used in the factor model chosen.
SigFq	An object of class 'SigFq' with the q-factor model approximation to the within groups covariance matrix. 'SigFq' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
SigFqInv	An object of class 'SigFqInv' with the q-factor model approximation to the within groups precision (inverse covariance) matrix. 'SigFqInv' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
N	The number of observations used.
call	The (matched) function call.

Author(s)

A. Pedro Duarte Silva

References

Pedro Duarte Silva, A. (2011) "Two Group Classification with High-Dimensional Correlated Data: A Factor Model Approach", *Computational Statistics and Data Analysis*, 55 (1), 2975-2990.

See Also

FrobSigAp, SelectV, SigFq, SigFqInv, predict.canldaRes, predict.clldaRes, AlonDS

Examples

```
#train classifier with 10 genes (after a logarithmic transformation)
# on Alon's Colon Cancer Data set.

log10genes <- log10(AlonDS[,-1])

ldarule1 <- RFlda(log10genes,AlonDS$grouping,Selmethod="fixedp",maxp=10)

# get in-sample classification results

predict(ldarule1,log10genes,grpcodes=levels(AlonDS$grouping))$class

# compare classifications with true assignments
```

```
cat("Original classes:\n")
print(AlonDS$grouping)

# Estimate error rates by four-fold cross-validation.
# Note: In cross-validation analysis it is recommended to set
# the argument 'ldafun' to "classification", in order to speed up
# computations by avoiding unecessary eigen-decompositions

## Not run:

CrosValRes1 <- DACrossVal(log10genes,AlonDS$grouping,TrainAlg=RFlda,
Selmethod="fixedp",ldafun="classification",maxp=10,kfold=4,CVrep=1)
summary(CrosValRes1[,,"Clerr"])

# Find the best factor model amongst the choices q=1 or 2

ldarule2 <- RFlda(log10genes,AlonDS$grouping,q="CVq",CVqtrials=1:2,
Selmethod="fixedp",ldafun="classification",maxp=10)
cat("Best error rate estimate found with q =",ldarule2$q,"\n")

# Perform the analysis finding the number of selected genes
# by the Expanded HC scheme

ldarule3 <- RFlda(log10genes,AlonDS$grouping,q=ldarule2$q)
cat("Number of selected genes =",ldarule3$nvkpt,"\n")

# get classification results

predict(ldarule3,log10genes,grpcodes=levels(AlonDS$grouping))$class


## End(Not run)
```

SelectV *Variable Selection for High-Dimensional Supervised Classification.*

Description

Selects variables to be used in a Discriminant Analysis classification rule.

Usage

```
SelectV(data, grouping,
Selmethod=c("ExpHC","HC","Fdr","Fair","fixedp"),
NullDist=c("locfdr","Theoretical"), uselocfdr=c("onlyHC","always"),
minlocfdrp=200, comvar=TRUE, Fdralpha=0.5,
ExpHCalpha=0.5, HCalpha0=0.1, maxp=ncol(data), tol=1E-12, ...)
```

Arguments

data	Matrix or data frame of observations.
grouping	Factor specifying the class for each observation.
Selmethod	The method used to choose the number of variables selected. Current alternatives are: 'ExpHC' (default) for the Expanded Higher Criticism scheme of Duarte Silva (2011) 'HC' for the Higher Criticism (HC) approach of Donoho and Jin (2004, 2008) 'Fdr' for the False Discovery Rate control approach of Benjamini and Hochberg (1995) 'Fair' for the FAIR (Features Annealed Independence Rules) approach of Fan and Fan (2008). This option is only available for two-group classification problems. 'fixedp' for a constant chosen by the user.
NullDist	The Null distribution used to compute pvalues from t-scores or F-scores. Current alternatives are "Theoretical" for the corresponding theoretical distributions, and "locfdr" for an empirical Null of z-scores estimated by the maximum likelihood approach of Efron (2004).
uselocfdr	Flag indicating the statistics for which the Null empirical distribution estimated by the locfdr approach should be used. Current alternatives are "onlyHC" (default) and "always".
minlocfdrp	Minimum number of variables required to estimate empirical Null distributions by the locfdr method. When the number of variables is below 'minlocfdrp', theoretical Nulls are always employed.
comvar	Boolean flag indicating if a common group variance is to be assumed (default) in the computation of the t-scores used for problems with two groups.
Fdralpha	Control level for variable selection based on False Discovery Rate Control (see Benjamini and Hochberg (1995)).
ExpHCalpha	Control level for the first step of the Extended Higher Criticism scheme (see Duarte Silva (2011)).
HCalpha0	Proportion of pvalues used to compute the HC statistic (see Donoho and Jin (2004, 2008)).
maxp	Maximum number of variables to be used in the discriminant rule.
tol	Numerical precision for distinguishing pvalues from 0 and 1. Computed pvalues below 'tol' are set to 'tol', and those above 1-'tol' are set to 1-'tol'.
...	Arguments passed from other methods.

Details

The function 'SelectV' selects variables to be used in a Discriminant classification rule by the Higher Criticism (HC) approach of Donoho and Jin (2004, 2008), the Expanded Higher Criticism scheme proposed by Duarte Silva (2011), False Discovery Rate (Fdr) control as suggested by Benjamini and Hochberg (1995), the FAIR (Features Annealed Independence Rules) approach of Fan and Fan (2008), or simply by fixing the number of selected variables to some pre-defined constant.

The Fdr method is, by default, based on simple p-values derived from t-scores (problems with two groups) or ANOVA F-scores (problems with more than two groups). When the argument 'NullDist' is set to "Theoretical" these values are also used in the HC method. Otherwise, the HC p-values are derived from an empirical Null of z-scores estimated by the maximum likelihood approach of Efron (2004).

The variable rankings are based on absolute-value t-scores or ANOVA F-scores.

Value

A list with two components:

`nvkpt`	the number of variables to be used in the Discriminant rule
`vkptInd`	the indices of the variables to be used in the Discriminant rule

Author(s)

A. Pedro Duarte Silva

References

Benjamini, Y. and Hochberg, Y. (1995) "Controling the false discovery rate: A practical and powerful approach to multiple testing", *Journal of the Royal Statistical Society* B, 57, 289-300.

Donoho, D. and Jin, J. (2004) "Higher criticism for detecting sparse heterogeneous mixtures", *Annals of Statistics* 32, 962-964.

Donoho, D. and Jin, J. (2008) "Higher criticism thresholding: Optimal feature selection when useful features are rare and weak", In: *Proceedings National Academy of Sciences*, USA 105, 14790-14795.

Efron, B. (2004) "Large-scale simultaneous hypothesis testing: the choice of a null hypothesis", *Journal of the American Statistical Association* 99, 96-104.

Fan, J. and Fan, Y. (2008) "High-dimensional classification using features annealed independence rules", *Annals of Statistics*, 36 (6), 2605-2637.

Pedro Duarte Silva, A. (2011) "Two Group Classification with High-Dimensional Correlated Data: A Factor Model Approach", *Computational Statistics and Data Analysis*, 55 (1), 2975-2990.

See Also

`Dlda`, `Mlda`, `Slda`, `RFlda`, `AlonDS`

Examples

```
## Not run:

# Compare the number of variables selected by the four methods
# currently available  on Alon's Colon Cancer Data set
# after a logarithmic transformation.

log10genes <- log10(AlonDS[,-1])
```

```
Res <- array(dim=4)
names(Res) <- c("ExpHC","HC","Fdr","Fair")
Res[1] <- SelectV(log10genes,AlonDS[,1])$nvkpt
Res[2] <- SelectV(log10genes,AlonDS[,1],Selmethod="HC")$nvkpt
Res[3] <- SelectV(log10genes,AlonDS[,1],Selmethod="Fdr")$nvkpt
Res[4] <- SelectV(log10genes,AlonDS[,1],Selmethod="Fair")$nvkpt

print(Res)

## End(Not run)
```

ShrnkMat *ShrnkMat objects: shrunken matrix estimates of a covariance*

Description

Creates a 'ShrnkMat' object.

Usage

```
ShrnkMat(U, D, p, q, Intst, Trgt="Idntty")
```

Arguments

U	A p by q matrix with the orthonomal eigenvectors of the original (unshrunken) covariance estimate.
D	A p-dimensional vector with the eigenvalues of the original (unshrunken) covariance estimate.
p	The dimension of the covariance matrix.
q	The rank of the original (unshrunken) covariance estimate.
Intst	The target intensity used by the shunk estimator.
Trgt	The target used by the shunk estimator. If set to the string "Idntty" (default) a p-dimensional matrix identity target will be assumed. Otherwise a matrix-type object representing a symmetric matrix target.

Value

An object of class 'ShrnkMat' for which the generic method 'as.matrix' (converting to a traditional numeric matrix), as well as specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations, are available.

See Also

solve.ShrnkMat, LeftMult.ShrnkMat, RightMult.ShrnkMat, ShrnkMatInv

ShrnkMatInv — *ShrnkMatInv objects: precision (inverse of covariance) matrices associated with shrunken estimates of a covariance*

Description

Creates a 'ShrnkMatInv' object.

Usage

```
ShrnkMatInv(U, D, p, q, Intst, Trgt="Idntty")
```

Arguments

U	A p by q matrix with the orthonomal eigenvectors of the original (unshrunken) covariance estimate.
D	A p-dimensional vector with the eigenvalues of the original (unshrunken) covariance estimate.
p	The dimension of the covariance matrix.
q	The rank of the original (unshrunken) covariance estimate.
Intst	The target intensity used by the shunk estimator.
Trgt	The target used by the shunk estimator. If set to the string "Idntty" (default) a p-dimensional matrix identity target will be assumed. Otherwise a matrix-type object representing a symmetric matrix target.

Value

An object of class 'ShrnkMatInv' for which the generic method 'as.matrix' (converting to a traditional numeric matrix), as well as specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations, are available.

See Also

ShrnkMat, solve.ShrnkMatInv, LeftMult.ShrnkMatInv,

RightMult.ShrnkMatInv

ShrnkSigE *Shrunken Covariance Estimate.*

Description

Builds a well-conditioned shrunken estimate of a covariance matrix based on Fisher and Sun's (2011) estimates and generalizations of Ledoit and Wolf's (2004) optimal optimal shrunken covariance matrices.

Usage

```
ShrnkSigE(df, p , SigmaRank, Sigma=NULL, SigmaSr=NULL, check=TRUE,
Trgt, minp=20, numtol=sqrt(.Machine$double.eps), ...)
```

Arguments

df	Degrees of freedom of the original (unshrunken) covariance estimate.
p	Dimension of the covariance matrix.
SigmaRank	Rank of the original (unshrunken) covariance estimate.
Sigma	Original (unshrunken) covariance estimate.
SigmaSr	Matrix square-root of the original (unshrunken) covariance estimate, i.e. a matrix, SigmaSr, such that SigmaSr^T SigmaSr = Sigma, where Sigma is the original unshrunken covariance estimate.
Trgt	A string code with the target type used by the shrunken estimator. The alternatives are "CnstDiag" for a Ledoit-Wolf constant diagonal target, "Idntty" for a p-dimensional identity, and "VarDiag" for a diagonal target of empirical variances.
check	Boolean flag indicating if the symmetry and the sign of the Sigma eigenvalues should be check upfront.
minp	Minimum number of variables required for the estimation of the target intensity to be considered reliable. If the dimension of Sigma is below pmin, no shrunken estimate is computed and the original sample covariance is returned.
numtol	Numerical tolerance. All computed eigenvalues below numtol are considered equal to zero, and the rank of original shrunken estimate is adjusted acordingly.
...	Further arguments passed to or from other methods.

Details

ShrnkSigE can take as input an original unshrunken estimate of the covariance matrix or, in alternative, one matrix square-root, SigmaSr (e.g. the original, centered and scaled, data matrix), such that $SigmaSr^T SigmaSr = Sigma$. In problems with more variables than observations it is preferable to use a matrix square-root for reasons of memory and computational efficiency.

Value

An object of class 'ShrnkMat' with a compact representation of the shrunken covariance estimator.

Author(s)

A. Pedro Duarte Silva

References

Ledoit, O. and Wolf, M. (2004) "A well-conditioned estimator for large-dimensional covariance matrices.", *Journal of Multivariate Analysis*, 88 (2), 365-411.

Fisher, T.J. and Sun, X. (2011) "Improved Stein-type shrinkage estimators for the high-dimensional multivariate normal covariance matrix", *Computational Statistics and Data Analysis*, 55 (1), 1909-1918.

See Also

ShrnkMat, Slda

SigFq *SigFq objects: covariance matrices associated with a q-factor model*

Description

Creates a 'SigFq' object.

Usage

```
SigFq(D, B, p, q, optres=NULL)
```

Arguments

D	A p-dimensional vector with the specific variances of the assumed factor model.
B	A p by q matrix with the loadings of assumed factor model.
p	The full dimensionality (i.e., the number of original variables) of the process being modelled.
q	The number of factors in the assumed factor model.
optres	The full optimization results provided by the error minimization algorithm used to find SigFq parameters.

Value

An object of class 'SigFq' for which the generic method 'as.matrix' (converting to a traditional numeric matrix), as well as specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations, are available.

See Also

FrobSigAp, solve.SigFq, LeftMult.SigFq, RightMult.SigFq, SigFqInv

---

SigFqInv — *SigFqInv objects: precision (inverse of covariance) matrices associated with a q-factor model*

Description

Creates a 'SigFqInv' object.

Usage

```
SigFqInv(D, B, p, q, optres=NULL)
```

Arguments

D	A p-dimensional vector with the specific variances of the assumed factor model.
B	A p by q matrix with the loadings of assumed factor model.
p	The full dimensionality (i.e., the number of original variables) of the process being modelled.
q	The number of factors in the assumed factor model.
optres	The full optimization results provided by the error minimization algorithm used to find SigFq parameters.

Value

An object of class 'SigFqInv' for which the generic method 'as.matrix' (converting to a traditional numeric matrix), as well as specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations, are available.

See Also

FrobSigAp, solve.SigFqInv, LeftMult.SigFqInv, RightMult.SigFqInv, SigFq

`Slda` *Shrunken Linear Discriminant Analysis.*

Description

'Slda' finds the coefficients of a linear discriminant rule based on Fisher and Sun's (2011) estimate and generalizations of Ledoit and Wolf's (2004) optimal shrunken covariance matrix.

Usage

```
## Default S3 method:
Slda(data, grouping, prior = "proportions", StddzData=TRUE,
VSelfunct = SelectV, Trgt=c("CnstDiag","Idntty","VarDiag"),
minp=20, ldafun=c("canonical","classification"), ...)

## S3 method for class 'data.frame'
Slda(data, ...)
```

Arguments

`data`	Matrix or data frame of observations.
`grouping`	Factor specifying the class for each observation.
`prior`	The prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
`StddzData`	A boolean flag indicating whether the data should be standardized first (default) or used in their original scales.
`VSelfunct`	Variable selection function. Either the string "none" (no selection is to be performed) or a function that takes 'data' and 'grouping' as its first two arguments and returns a list with two components: (i) 'nvkpt' - the number of variables to be used in the Discriminant rule; and (ii) 'vkptInd' - the indices of the variables to be used in the Discriminant rule. The default is the 'SelectV' function that, by default, selects variables by the Expanded HC scheme described in Duarte Silva (2011).
`Trgt`	A string code with the target type used by the shrunken estimator. The alternatives are "CnstDiag" for a Ledoit-Wolf constant diagonal target, "Idntty" for a p-dimensional identity, and "VarDiag" for a diagonal target of empirical variances.
`minp`	Minimum number of variables required for the estimation of the target intensity to be considered reliable. If the dimension of Sigma is below pmin, no shrunken estimate is computed and the original sample covariance is employed.
`ldafun`	Type of discriminant linear functions computed. The alternatives are "canonical" for maximum-discrimination canonical linear functions and "classification" for direct-classification linear functions.
`...`	Further arguments passed to or from other methods.

Value

If algument 'ldafun' is set to "canonical" an object of class 'Scanlda', which extends class 'canldaRes', with the following components:

prior	The prior probabilities used.
means	The class means.
scaling	A matrix which transforms observations to discriminant functions, normalized so that the within groups covariance matrix is spherical.
svd	The singular values, which give the ratio of the between- and within-group standard deviations on the linear discriminant variables. Their squares are the canonical F-statistics.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL otherwise.
SSig	An object of class 'ShrnkMat' with a compact representation of the within groups covariance matrix. 'ShrnkMat' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
SSigInv	An object of class 'ShrnkMatInv' with a compact representation of the within groups precision (inverse covariance) matrix. 'ShrnkMatInv' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
N	The number of observations used.
call	The (matched) function call.

If algument 'ldafun' is set to "classification" an object of class 'Scllda', which extends class 'clldaRes', with the following components:

prior	The prior probabilities used.
means	The class means.
coef	A matrix with the coefficients of the k-1 classification functions.
cnst	A vector with the thresholds (2nd members of linear classification rules) used in classification rules that assume equal priors.
vkpt	A vector with the indices of the variables kept in the discriminant rule if the number of variables kept is less than 'ncol(data)'. NULL otherwise.
nvkpt	The number of variables kept in the discriminant rule if this number is less than'ncol(data)'. NULL, otherwise.
SSig	An object of class 'ShrnkMat' with a compact representation of the within groups covariance matrix. 'ShrnkMat' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
SSigInv	An object of class 'ShrnkMatInv' with a compact representation of the within groups precision (inverse covariance) matrix. 'ShrnkMatInv' objects have specialized methods for matrix inversion, multiplication, and element-wise arithmetic operations.
N	The number of observations used.
call	The (matched) function call.

Author(s)

A. Pedro Duarte Silva

References

Ledoit, O. and Wolf, M. (2004) "A well-conditioned estimator for large-dimensional covariance matrices.", *Journal of Multivariate Analysis*, 88 (2), 365-411.

Fisher, T.J. and Sun, X. (2011) "Improved Stein-type shrinkage estimators for the high-dimensional multivariate normal covariance matrix", *Computational Statistics and Data Analysis*, 55 (1), 1909-1918.

Pedro Duarte Silva, A. (2011) "Two Group Classification with High-Dimensional Correlated Data: A Factor Model Approach", *Computational Statistics and Data Analysis*, 55 (1), 2975-2990.

See Also

SelectV, ShrnkSigE, ShrnkMat, ShrnkMatInv,

predict.canldaRes, predict.clldaRes, AlonDS

Examples

```
# train classifier on Alon's Colon Cancer Data set
# after a logarithmic transformation
# (selecting genes by the Expanded HC scheme).

ldarule <- Slda(log10(AlonDS[,-1]),AlonDS$grouping)

# show classification rule

print(ldarule)

# get in-sample classification results

predict(ldarule,log10(AlonDS[,-1]),grpcodes=levels(AlonDS$grouping))$class

# compare classifications with true assignments

cat("Original classes:\n")
print(AlonDS[,1])


# Estimate error rates by four-fold cross-validation.
# Note: In cross-validation analysis it is recommended to set
# the argument 'ldafun' to "classification", in order to speed up
# computations by avoiding unecessary eigen-decompositions

## Not run:

CrosValRes <- DACrossVal(log10(AlonDS[,-1]),AlonDS$grouping,TrainAlg=Slda,
ldafun="classification",kfold=4,CVrep=1)
summary(CrosValRes[,,"Clerr"])
```

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

`solve` *Solve methods for 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' and 'SigFqInv' objects.*

Description

These methods solve the equation `a %*% x = b` for 'x', where 'a' is a specialized square symmetric matrix represented by a 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' or 'SigFqInv' object, and 'b' can be either a vector or a matrix.

Usage

```
## S3 method for class 'DMat'
solve(a, b = NULL, ...)
## S3 method for class 'ShrnkMat'
solve(a, b = NULL, ...)
## S3 method for class 'ShrnkMatInv'
solve(a, b = NULL, ...)
## S3 method for class 'SigFq'
solve(a, b = NULL, ...)
## S3 method for class 'SigFqInv'
solve(a, b = NULL, ...)
```

Arguments

`a`	An object of type 'DMat', 'ShrnkMat', 'ShrnkMatInv', 'SigFq' or 'SigFqInv' representing a specialized symmetric square matrix.
`b`	A numeric vector or matrix giving the right-hand side(s) of the linear system. If missing, 'b' is taken to be an identity matrix and solve will return an object representing the inverse of the matrix associated with 'a'.
`...`	Further arguments passed to or from other methods.

Details

The result returned depends on the values of the arguments. When 'b' is not NULL, both functions return a numeric vector or matrix with the solution of the system. When 'b' is NULL, 'solve.DMat' returns a 'DMat' object, 'solve.ShrnkMat' a 'ShrnkMatInv' object, 'solve.ShrnkMatInv' a 'ShrnkMat' object, 'solve.SigFq' a 'SigFqInv' object and 'solve.SigFqInv' returns a 'SigFq' object.

See Also

`DMat`, `ShrnkMat`, `ShrnkMatInv`, `SigFq`, `SigFqInv`, `FrobSigAp`, `solve`

Index

∗ **HiDimDA**
HiDimDA-package, 2
∗ **datasets**
AlonDS, 5
∗.DMat *(*DMat*)*, 13
∗.ShrnkMat *(*ShrnkMat*)*, 26
∗.ShrnkMatInv *(*ShrnkMatInv*)*, 27
∗.SigFq *(*SigFq*)*, 29
∗.SigFqInv *(*SigFqInv*)*, 30
+.DMat *(*DMat*)*, 13
+.ShrnkMat *(*ShrnkMat*)*, 26
+.ShrnkMatInv *(*ShrnkMatInv*)*, 27
+.SigFq *(*SigFq*)*, 29
+.SigFqInv *(*SigFqInv*)*, 30
-.DMat *(*DMat*)*, 13
-.ShrnkMat *(*ShrnkMat*)*, 26
-.ShrnkMatInv *(*ShrnkMatInv*)*, 27
-.SigFq *(*SigFq*)*, 29
-.SigFqInv *(*SigFqInv*)*, 30
/.DMat *(*DMat*)*, 13
/.ShrnkMat *(*ShrnkMat*)*, 26
/.ShrnkMatInv *(*ShrnkMatInv*)*, 27
/.SigFq *(*SigFq*)*, 29
/.SigFqInv *(*SigFqInv*)*, 30

AlonDS, *4*, 5, *12*, *18*, *22*, *25*, *33*
as.matrix.DMat *(*DMat*)*, 13
as.matrix.ShrnkMat *(*ShrnkMat*)*, 26
as.matrix.ShrnkMatInv *(*ShrnkMatInv*)*, 27
as.matrix.SigFq *(*SigFq*)*, 29
as.matrix.SigFqInv *(*SigFqInv*)*, 30

canldaRes, 6, *8*
clldaRes, 7, *8*
coef, *7*
coef.canldaRes *(*canldaRes*)*, 6
coef.clldaRes *(*clldaRes*)*, 7
CovE, 8
CovE.canldaRes *(*canldaRes*)*, 6
CovE.clldaRes *(*clldaRes*)*, 7

DACrossVal, 9
Dlda, *4*, *7*, 10, *25*
DMat, *12*, 13, *16*, *34*

FrobSigAp, 13, *16*, *22*, *30*, *34*
FrobSigAp1 *(*FrobSigAp*)*, 13

HiDimDA *(*HiDimDA-package*)*, 2
HiDimDA-package, 2

ICovE *(*CovE*)*, 8
ICovE.canldaRes *(*canldaRes*)*, 6
ICovE.clldaRes *(*clldaRes*)*, 7
is.Dlda *(*Dlda*)*, 10
is.DMat *(*DMat*)*, 13
is.Mlda *(*Mlda*)*, 16
is.RFlda *(*RFlda*)*, 20
is.ShrnkMat *(*ShrnkMat*)*, 26
is.ShrnkMatInv *(*ShrnkMatInv*)*, 27
is.SigFq *(*SigFq*)*, 29
is.SigFqInv *(*SigFqInv*)*, 30
is.Slda *(*Slda*)*, 31

LeftMult *(*MatMult*)*, 15
LeftMult.DMat, *13*
LeftMult.ShrnkMat, *26*
LeftMult.ShrnkMatInv, *27*
LeftMult.SigFq, *30*
LeftMult.SigFqInv, *30*

MatMult, 15
Mlda, *4*, *7*, 16, *19*, *25*
MldaInvE, *18*, 19

predict.canldaRes, *4*, *12*, *18*, *22*, *33*
predict.canldaRes *(*canldaRes*)*, 6
predict.clldaRes, *4*, *10*, *12*, *18*, *22*, *33*
predict.clldaRes *(*clldaRes*)*, 7
print, *7*
print.canldaRes *(*canldaRes*)*, 6
print.clldaRes *(*clldaRes*)*, 7

print.DMat *(*DMat*)*, 13
print.ShrnkMat *(*ShrnkMat*)*, 26
print.ShrnkMatInv *(*ShrnkMatInv*)*, 27
print.SigFq *(*SigFq*)*, 29
print.SigFqInv *(*SigFqInv*)*, 30

RFlda, *4*, *7, 8*, *10*, *14*, 20, *25*
RightMult *(*MatMult*)*, 15
RightMult.DMat, *13*
RightMult.ShrnkMat, *26*
RightMult.ShrnkMatInv, *27*
RightMult.SigFq, *30*
RightMult.SigFqInv, *30*

SelectV, *12*, *18*, *22*, 23, *33*
ShrnkMat, *16*, 26, *27*, *29*, *33, 34*
ShrnkMatInv, *16*, *26*, 27, *33, 34*
ShrnkSigE, 28, *33*
SigFq, *14*, *16*, *22*, 29, *30*, *34*
SigFqInv, *14*, *16*, *22*, 30, *30*, *34*
Slda, *4*, *7, 8*, *25*, *29*, 31
solve, 34, *34*
solve.DMat, *13*, *16*
solve.ShrnkMat, *16*, *26*
solve.ShrnkMatInv, *16*, *27*
solve.SigFq, *14*, *16*, *30*
solve.SigFqInv, *14*, *16*, *30*