Package ‘TULIP’

June 29, 2020

Title A Toolbox for Linear Discriminant Analysis with Penalties
Version 1.0.1
Description Integrates several popular high-dimensional methods based on Linear Discriminant Analysis (LDA) and provides a comprehensive and user-friendly toolbox for linear, semi-parametric and tensor-variate classification as mentioned in Yuqing Pan, Qing Mai and Xin Zhang (2019) <arXiv:1904.03469>. Functions are included for covariate adjustment, model fitting, cross validation and prediction.
Depends R (>= 3.1.1)
License GPL-2
Encoding UTF-8
LazyData true
Imports tensr, Matrix, MASS, glmnet, methods
NeedsCompilation yes
Author Yuqing Pan <yuqing.pan@stat.fsu.edu>,
Qing Mai <mai@stat.fsu.edu>,
Xin Zhang <henry@stat.fsu.edu>
Maintainer Yuqing Pan <yuqing.pan@stat.fsu.edu>
Repository CRAN
Date/Publication 2020-06-29 04:30:03 UTC

R topics documented:

adjten ................................................................. 2
adjvec ............................................................... 4
catch ................................................................. 5
catch_matrix ......................................................... 9
csa ................................................................. 11
cv.catch ............................................................. 11
cv.dsda .............................................................. 13
cv.msda ............................................................. 14
cv.SeSDA ........................................................... 16
dsda ................................................................. 17
adjten

Adjust tensor for covariates.

Description

Adjusts tensor with respect to covariates to achieve a more accurate performance. Tensor depends on the covariates through a linear regression model. The function returns the coefficients of covariates in regression and adjusted tensor list for further classifier modeling. It estimates coefficients based on training data, and then adjusts training tensor. When testing data is provided, the function will automatically adjust testing data by learned coefficients as well.

Usage

adjten(x, z, y, testx = NULL, testz = NULL, is.centered = FALSE)

Arguments

x
Input tensor or matrix list of length N, where N is the number of observations. Each element of the list is a tensor or matrix. The order of tensor can be any integer not less than 2.

z
Input covariate matrix of dimension N × q, where q < N. Each row of z is an observation.

y
Class label vector of dimension N × 1. For K class problems, y takes values in {1, ⋯, K}.

testx
Input testing tensor or matrix list. Each element of the list is a test case. When testx is not provided, the function will only adjust training data.

testz
Input testing covariate matrix with each row being an observation.

is.centered
Indicates whether the input tensor and covariates have already been centered by their within class mean or not. If is.centered is FALSE, the function adjten will center data by class. If is.centered is TRUE, the function will skip the centering step.
Details

The model CATCH assumes the linear relationship between covariates and tensor as

\[ X = \mu_k + \alpha \overline{X}_{M+1} Z + E, \]

where \( \alpha \) is the matrix of estimated coefficient of covariates. The function removes the effects of covariates on response variable through tensor and obtain \( X - \alpha \overline{X}_{M+1} Z \) as adjusted tensor to fit tensor discriminant analysis model.

In estimating \( \alpha \), which is the alpha in the package, \texttt{adjten} first centers both tensor and covariates within their individual classes, then performs tensor response regression which regresses \( X \) on \( Z \).

Value

- **gamma**: The estimated coefficients of covariates to plug in classifier. \( \gamma_k \) is the \( \gamma_k \) defined function \texttt{catch} of dimension \( q \times (K-1) \), where \( q \) is the size of covariates and \( K \) is the number of classes.
- **xres**: Adjusted training tensor list \( X - \alpha \overline{X}_{M+1} Z \) after adjusting for covariates. The effect of the covariate is removed.
- **testxres**: Adjusted testing tensor list \( X - \alpha \overline{X}_{M+1} Z \) after adjusting for covariates. The effect of the covariate is removed.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References


See Also

- \texttt{catch}

Examples

```r
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars),nrow=n,ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2),nrow=n,ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,],dim=c(p,p,p))
}
obj <- adjten(x, z, y)
```
adjvec

Adjust vector for covariates.

Description

Adjusts vector with respect to covariates. Vector depends on the covariates through a linear regression model. The function returns the coefficients of covariates in regression and adjusted predictor matrix for further classifier modeling. It estimates coefficients based on training data, and then adjusts training tensor. When testing data is provided, the function will automatically adjust testing data by learned coefficients as well.

Usage

adjvec(x, z, y, testx = NULL, testz = NULL, is.centered = FALSE)

Arguments

x  Input matrix of dimension \( N \times p \), where \( N \) is the number of observations and \( p \) is the number of variables. Each row is an observation.

z  Input covariate matrix of dimension \( N \times q \), where \( q < N \). Each row of \( z \) is an observation.

y  Class label vector of dimension \( N \times 1 \). For \( K \) class problems, \( y \) takes values in \( \{1, \cdots, K\} \).

testx Input testing matrix. Each row is a test case. When \( \text{testx} \) is not provided, the function will only adjust training data.

testz Input testing covariate matrix with each row being an observation.

is.centered Indicates whether the input vector and covariates have already been centered by their within class mean or not. If \( \text{is.centered} \) is FALSE, the function \( \text{adjvec} \) will center data by class. If \( \text{is.centered} \) is TRUE, the function will skip the centering step.

Details

Similar as CATCH model, assume the linear relationship between vector predictors and covariates as

\[
X = \mu_k + \alpha \times Z + E,
\]

where \( X \) is a \( N \times p \) matrix and \( \alpha \) is the matrix of estimated coefficient of covariates. The function removes the effects of covariates on response variable through vector and obtain \( X - \alpha \times Z \) as adjusted predictors to fit MSDA and DSDA model.

Value

gamma The estimated coefficients of covariates to plug in classifier. gamma is similar as the \( \gamma_k \) defined function \text{catch} of dimension \( q \times (K - 1) \), where \( q \) is the size of covariates and \( K \) is the number of classes.
The `catch` function solves classification problems and selects variables by fitting a covariate-adjusted tensor classification in high-dimensions (CATCH) model. The input training predictors include two parts: tensor data and low dimensional covariates. The tensor data could be matrix as a special case of tensor. In `catch`, tensor data should be stored in a list form. If the dataset contains no covariate, `catch` can also fit a classifier only based on the tensor predictors. If covariates are provided, the method will adjust tensor for covariates and then fit a classifier based on the adjusted tensor along with the covariates. If users specify testing data at the same time, predicted response will be obtained as well.
catch

catch(x, z = NULL, y, testx = NULL, testz = NULL, nlambda = 100,
lambda.factor = ifelse((nobs - nclass) <= nvars, 0.2, 1E-03),
lambda = NULL, dfmax = nobs, pmax = min(dfmax * 2 + 20, nvars),
 pf = rep(1, nvars), eps = 1e-04, maxit = 1e+05, sml = 1e-06,
verbose = FALSE, perturb = NULL)

Arguments

x  Input tensor (or matrix) list of length \( N \), where \( N \) is the number of observations. Each element of the list is a tensor or matrix. The order of tensor can be any positive integer not less than 2.

z  Input covariate matrix of dimension \( N \times q \), where \( q < N \). \( z \) can be omitted if covariate is absent.

y  Class label. For \( K \) class problems, \( y \) takes values in \{1, \ldots, K\}.

testx  Input testing tensor or matrix list. Each element of the list is a test case. When testx is not provided, the function will only fit the model and return the classifier. When testx is provided, the function will predict response on testx as well.

testz  Input testing covariate matrix. Can be omitted if covariate is absent. However, training covariates \( z \) and testing covariates testz must be provided or not at the same time.

nlambda  The number of tuning values in sequence lambda. If users do not specify lambda values, the package will generate a solution path containing nlambda many tuning values of lambda. Default is 100.

lambda.factor  When lambda is not supplied, catch first finds the largest value in lambda which yields \( \beta = 0 \). Then the minimum value in lambda is obtained by \( \text{largest value \times lambda.factor} \). The sequence of lambda is generated by evenly sampling nlambda numbers within the range. Default value of lambda.factor is 0.2 if \( N < p \) and 0.0001 if \( N > p \).

lambda  A sequence of user-specified lambda values. lambda is the weight of L1 penalty and a smaller lambda allows more variables to be nonzero. If NULL, then the algorithm will generate a sequence of nlambda many potential lambdas according to lambda.factor.

dfmax  The maximum number of selected variables in the model. Default is the number of observations \( N \).

pmax  The maximum number of potential selected variables during iteration. In middle step, the algorithm can select at most pmax variables and then shrink part of them such that the number of final selected variables is less than dfmax. Default is \( \min(\text{dfmax} \times 2 + 20, N) \).

pf  Weight of lasso penalty. Default is a vector of value 1 and length \( p \), representing L1 penalty of length \( p \). Can be modified to use adaptive lasso penalty.

eps  Convergence threshold for coordinate descent difference between iterations. Default value is 1e-04.

maxit  Maximum iteration times for all lambda. Default value is 1e+05.
catch

**sml** Threshold for ratio of loss function change after each iteration to old loss function value. Default value is $1e^{-06}$.

**verbose** Indicates whether print out lambda during iteration or not. Default value is **FALSE**.

**perturb** Perturbation scaler. If it is specified, the value will be added to diagonal of estimated covariance matrix. A small value can be used to accelerate iteration. Default value is **NULL**.

**Details**

The catch function fits a linear discriminant analysis model as follows:

$$Z | (Y = k) \sim N(\phi_k, \psi),$$

$$X | (Z = z, Y = k) \sim TN(\mu_k + \alpha \times M + 1 z, \Sigma_1, \cdots, \Sigma_M).$$

The categorical response is predicted from the estimated Bayes rule:

$$\hat{Y} = \arg \max_{k=1, \cdots, K} a_k + \gamma_k^T Z + < \beta_k, X - \alpha \times M + 1 Z >,$$

where $X$ is the tensor, $Z$ is the covariates, $a_k, \gamma_k$ and $\alpha$ are parameters estimated by CATCH. A detailed explanation can be found in reference. When $Z$ is not **NULL**, the function will first adjust tensor on covariates by modeling

$$X = \mu_k + \alpha \times M + 1 Z + E,$$

where $E$ is an unobservable tensor normal error independent of $Z$. Then catch fits model on the adjusted training tensor $X - \alpha \times M + 1 Z$ and makes predictions on testing data by using the adjusted tensor list. If $Z$ is **NULL**, it reduces to a simple tensor discriminant analysis model.

The coefficient of tensor $\beta$, represented by beta in package, is estimated by

$$\min_{\beta_2, \ldots, \beta_K} \left[ \sum_{k=2}^{K} \left( \langle \beta_k, [\beta_k; \Sigma_1, \ldots, \Sigma_M] \rangle - 2 \langle \beta_k, \bar{\mu}_k - \bar{\mu}_1 \rangle \right) + \lambda \sum_{j_1, \ldots, j_M} \sqrt{\sum_{k=2}^{K} \beta_{k,j_1}^2} \right].$$

When response is multi-class, the group lasso penalty over categories is added to objective function through parameter lambda, and it reduces to a lasso penalty in binary problems.

The function catch will predict categorical response when testing data is provided. If testing data is not provided or if one wishes to perform prediction separately, catch can be used to only fit model with a catch object outcome. The object outcome can be combined with the adjusted tensor list from adjten to perform prediction by predict.catch.

**Value**

**beta** Output variable coefficients for each lambda, which is the estimation of $\beta$ in the Bayes rule. beta is a list of length being the number of lambdas. Each element of beta is a matrix of dimension $nvars \times (nclass - 1)$.

**df** The number of nonzero variables for each value in sequence lambda.

**dim** Dimension of coefficient array.
lambda The actual lambda sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on dfmax and pmax.

obj Objective function value for each value in sequence lambda.

x The tensor list after adjustment in training data. If covariate is absent, this is the original input tensor list.

y Class label in training data.

npasses Total number of iterations.

jerr Error flag.

sigma Estimated covariance matrix on each mode. sigma is a list with the ith element being covariance matrix on ith mode.

delta Estimated delta matrix \( \vec{\mu}_2 - \vec{\mu}_1, \ldots, \vec{\mu}_K - \vec{\mu}_1 \).

mu Estimated mean array of \( X \).

prior Proportion of samples in each class.

call The call that produces this object.

pred Predicted categorical response for each value in sequence lambda when testx is provided.

Author(s)
Yuqing Pan, Qing Mai, Xin Zhang

References

See Also
cv.catch, predict.catch, adjten

Examples

#without prediction
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars), nrow=n, ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2), nrow=n, ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,],dim=c(p,p,p))
}
obj <- catch(x,z,y=y)
catch_matrix

**Catch Matrix**

Fit a CATCH model for matrix and predict categorical response.

**Description**

Fits a classifier for matrix data. catch_matrix is a special case of catch when each observation \( X_j \) is a matrix. Different from catch takes list as input, data need to be formed in an array to call the function (see arguments). The function will perform prediction as well.

**Usage**

```r
catch_matrix(x, z = NULL, y, testx = NULL, testz = NULL, ...)
```

**Arguments**

- **x**: Input matrix array. The array should be organized with dimension \( p_1 \times p_2 \times N \).
- **z**: Input covariate matrix of dimension \( N \times q \), where \( q < N \). \( z \) can be omitted if covariate is absent.
- **y**: Class label. For \( K \) class problems, \( y \) takes values in \( \{1, \cdots, K\} \).
- **testx**: Input testing matrix array. When \( testx \) is not provided, the function will only fit model. When \( testx \) is provided, the function will predict response on \( testx \) as well.
- **testz**: Input testing covariate matrix. Can be omitted if there is no covariate.
- **...**: Other arguments that can be passed to catch.

**Details**

The function fits a matrix classifier as a special case of catch. The fitted model and predictions should be identical to catch when matrix data is provided. Input matrix should be organized as three-way array where sample size is the last dimension. If the matrix is organized in a list, users can either reorganize it or use catch directly to fit model, which takes a matrix or tensor list as input and has the same output as catch_matrix.

**Value**

- **beta**: Output variable coefficients for each \( \lambda \) value. \( beta \) is a list of length being the number of \( \lambda \) values. Each element of \( beta \) is a matrix of dimension \( (p_1 \times p_2) \times (nclass - 1) \).
- **df**: The number of nonzero variables for each value in sequence \( \lambda \).
- **dim**: Dimension of coefficient array.
- **lambda**: The actual \( \lambda \) sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on \( dfmax \) and \( pmax \).
- **obj**: Objective function value for each value in sequence \( \lambda \).
- **x**: The matrix list after adjustment in training data. If covariate is absent, this is the original input matrix.
catch_matrix

y  Class label in training data.
npasses  Total number of iterations.
jerr  Error flag.
sigma  Estimated covariance matrix on each mode. sigma is a list with the ith element being covariance matrix on ith mode.
delta  Estimated delta matrix \((\text{vec}(\hat{\mu}_2 - \hat{\mu}_1), \cdots, \text{vec}(\hat{\mu}_K - \hat{\mu}_1))\).
mu  Estimated mean array.
prior  Prior proportion of observations in each class.
call  The call that produces this object.
pred  Predicted categorical response for each value in sequence lambda when testx is provided.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References


See Also

catch

Examples

# without prediction
n <- 20
p <- 4
k <- 2
nvars <- p*p
x <- array(rnorm(n*nvars), dim=c(p,p,n))
x[,,11:20] <- x[,,11:20] + 0.3
z <- matrix(rnorm(n*2), nrow=n, ncol=2)
z[1:10,] <- z[1:10,] + 0.5
y <- c(rep(1,10), rep(2,10))
obj <- catch_matrix(x, z, y=y)
**csa**

*Colorimetric sensor array (CSA) data*

**Description**

A dataset collected from a series of CSA experiments to identify volatile chemical toxicants (VCT). Chemical dyes were exposed to VCT under different concentration conditions and colors of dyes were recorded to identify the class of VCT. There are two concentration conditions PEL (permissible exposure level) and IDLH (immediately dangerous to life of health).

**Usage**

```r
data(csa)
```

**Format**

Two lists, `PEL` and `IDLH`, and a numeric vector `y`. Each list contains 147 matrices of dimension 36 x 3.

- **PEL**: A list of matrices containing the observations after exposure at PEL.
- **IDLH**: A list of matrices containing the observations after exposure at IDLH level.
- **y**: Class label ranging from 1 to 21.

**Details**

This dataset is provided in the Supplementary material of Zhong (2015). In each concentration case, there are 147 observations and 21 classes. We reorganize the data into a list to be directly called by `catch`. For matrices in the list, each row represents a dye and the three columns correspond to red, green and blue.

**Source**


---

**cv.catch**

*Cross-validation for CATCH*

**Description**

Performs k-fold cross validation for CATCH and returns the best tuning parameter $\lambda$ in the user-specified or automatically generated choices.

**Usage**

```r
cv.catch(x, z = NULL, y, nfolds = 5, lambda = NULL, lambda.opt = "min", ...)
```
Arguments

- **x**: Input tensor or matrix list of length $N$, where $N$ is the number of observations. Each element of the list is a tensor or matrix. The order of tensor can be any number and not limited to three.
- **z**: Input covariate matrix of dimension $N \times q$, where $q < N$. $z$ can be omitted if covariate is absent.
- **y**: Class label. For $K$ class problems, $y$ takes values in $\{1, \ldots, K\}$.
- **nffolds**: Number of folds. Default value is 5.
- **lambda**: User-specified lambda sequence for cross validation. If not specified, the algorithm will generate a sequence of lambdas based on all data and cross validate on the sequence.
- **lambda.opt**: The optimal criteria when multiple elements in lambda return the same minimum classification error. "min" will return the smallest lambda with minimum cross validation error. "max" will return the largest lambda with minimum cross validation error.
- **...**: Other arguments that can be passed to `catch`.

Details

The function `cv.catch` runs function `catch` `nffolds+1` times. The first one fits model on all data. If lambda is specified, it will check if all lambda satisfies the constraints of $df_{max}$ and $p_{max}$ in `catch`. If not, a lambda sequence will be generated according to `lambda.factor` in `catch`. Then the rest `nffolds` many replicates will fit model on `nffolds-1` many folds data and predict on the omitted fold, respectively. Return the lambda with minimum average cross validation error and the largest lambda within one standard error of the minimum.

Value

- **lambda**: The actual lambda sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on $df_{max}$ and $p_{max}$.
- **cvm**: The mean of cross validation errors for each lambda.
- **cvsd**: The standard error of cross validation errors for each lambda.
- **lambda.min**: The lambda with minimum cross validation error. If lambda.opt is min, then returns the smallest lambda with minimum cross validation error. If lambda.opt is max, then returns the largest lambda with minimum cross validation error.
- **lambda.1se**: The largest lambda with cross validation error within one standard error of the minimum.
- **catch.fit**: The fitted catchobj object.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

cv.dsda

See Also
catch

Examples
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars), nrow=n, ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2),nrow=n,ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,], dim=c(p,p,p))
}
objc <- cv.catch(x, z, y=y)

cv.dsda

Cross validation for direct sparse discriminant analysis

Description
Choose the optimal lambda for direct sparse discriminant analysis by cross validation.

Usage
cv.dsda(x, y, nfolds = 5, lambda=lambda, lambda.opt="min",
standardize=FALSE, alpha=1, eps=1e-7)

Arguments

x An n by p matrix containing the predictors.
y An n-dimensional vector containing the class labels.
nfolds The number of folds to be used in cross validation. Default is 5.
lambda A sequence of lambda’s.
lambda.opt Should be either "min" or "max", specifying whether the smallest or the largest
lambda with the smallest cross validation error should be used for the final clas-
sification rule.
standardize A logic object indicating whether x.matrix should be standardized before per-
forming DSDA. Default is FALSE.
alpha The elasticnet mixing parameter, the same as in glmnet. Default is alpha=1 so
that the lasso penalty is used.
eps Convergence threshold for coordinate descent, the same as in glmnet. Default is
1e-7.
Value

- **lambda**: The sequence of lambda's used in cross validation.
- **cvm**: Cross validation errors.
- **cvsd**: The standard error of the cross validation errors.
- **lambda.min**: The optimal lambda chosen by cross validation.
- **model.fit**: The fitted model.

References


See Also

cv.dsdapredict.dsdadsda

cv.msda

Cross-validation for DSDA/MSDA through function msda

Description

Performs K-fold cross validation for msda and returns the best tuning parameter \( \lambda \) in the user-specified or automatically generated choices.

Usage

```r
cv.msda(x, y, model = NULL, nfolds = 5, lambda = NULL, lambda.opt = "min", ...)
```

Arguments

- **x**: Input matrix of predictors. \( x \) is of dimension \( N \times p \); each row is an observation vector.
- **y**: Class label. For \( K \) class problems, \( y \) takes values in \( \{1, \cdots, K\} \).
- **model**: Method type. The model argument can be one of 'binary', 'multi.original', 'multi.modified' and the default is NULL. The function supports fitting DSDA and MSDA models by specifying method type. Without specification, the function will automatically choose one of the methods. If the response variable is binary, the function will fit a DSDA model. If the response variable is multi-class, the function will fit an original MSDA model for dimension \( p \leq 2000 \) and a modified MSDA model for dimension \( p > 2000 \).
- **nfolds**: Number of folds. Default value is 5. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds=3 for multi.original and multi.modified.
lambda
User-specified lambda sequence for cross validation. If not specified, the algorithm will generate a sequence of lambdas based on all data and cross validate on the sequence.

lambda.opt
The optimal criteria when multiple elements in lambda return the same minimum classification error. "min" will return the smallest lambda with minimum cross validation error. "max" will return the largest lambda with the minimum cross validation error.

... other arguments that can be passed to msda.

Details
The function cv.msda runs function msda nfolds+1 times. The first one fits model on all data. If lambda is specified, it will check if all lambda satisfies the constraints of dfmax and pmax in msda. If not, a lambda sequence will be generated according to lambda.factor in msda. Then the rest nfolds many replicates will fit model on nfolds-1 many folds data and predict on the omitted fold, respectively. Return the lambda with minimum average cross validation error and the largest lambda within one standard error of the minimum.

Similar as msda, user can specify which method to use by inputing argument model. Without specification, the function can automatically decide the method by number of classes and variables.

Value
An object of class cv.dsda or cv.msda.original or cv.msda.modified is returned, which is a list with the ingredients of the cross-validation fit.

lambda
The actual lambda sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on dfmax and pmax.

cvm
The mean of cross validation errors for each lambda.

cvsd
The standard error of cross validation errors for each lambda.

lambda.min
The lambda with minimum cross validation error. If lambda.opt is min, then returns the smallest lambda with minimum cross validation error. If lambda.opt is max, then returns the largest lambda with minimum cross validation error.

lambda.1se
The largest value of lambda such that error is within one standard error of the minimum. This argument is only available for object cv.msda.original and cv.msda.modified.

model.fit
A fitted cv.dsda or cv.msda.original or cv.msda.modified object for the full data.

Author(s)
Yuqing Pan, Qing Mai, Xin Zhang

References
URL: https://github.com/emeryyi/msda

See Also
msda

Examples

data(GDS1615)
  x <- GDS1615$x
  y <- GDS1615$y
  obj.cv <- cv.msda(x=x, y=y, nfolds=5, lambda.opt="max")
  lambda.min <- obj.cv$lambda.min
  obj <- msda(x=x, y=y, lambda=lambda.min)
  pred <- predict(obj,x)

---

cv.SeSDA

Cross validation for semiparametric sparse discriminant analysis

Description

Choose the optimal lambda for semiparametric sparse discriminant analysis by cross validation.

Usage

cv.SeSDA(x, y, nfolds = 5, lambda=NULL, lambda.opt="min",
         standardize=FALSE, alpha=1, eps=1e-7)

Arguments

x  An n by p matrix containing the predictors.
y  An n-dimensional vector containing the class labels.
nfolds  The number of folds to be used in cross validation. Default is 5.
lambda  A sequence of lambda’s.
lambda.opt  Should be either "min" or "max", specifying whether the smallest or the largest lambda with the smallest cross validation error should be used for the final classification rule.
standardize  A logic object indicating whether x.matrix should be standardized before performing DSDA. Default is FALSE.
alpha  The elasticnet mixing parameter, the same as in glmnet. Default is alpha=1 so that the lasso penalty is used.
eps  Convergence threshold for coordinate descent, the same as in glmnet. Default is 1e-7.
dsda

Value

transform The transformation functions.
objsda The output of cross validation from cv.dsa on transformed data.

References


See Also
cv.dsa SeSDA

Description

Compute the solution path for direct sparse discriminant analysis (DSDA).

Usage

dsda(x, z=NULL, y, testx=NULL, testz=NULL, standardize=FALSE, lambda=lambda, alpha=1, eps=1e-7)

Arguments

x Input matrix of predictors. x is of dimension $N \times p$; each row is an observation vector.
z Input covariate matrix of dimension $N \times q$, where $q < N$. z can be omitted if covariate is absent.
y An n-dimensional vector containing the class labels. The classes have to be labeled as 1 and 2.
testx Input testing matrix. Each row is a test case. When testx is not provided, the function will only fit the model and return the classifier. When testx is provided, the function will predict response on testx as well.
testz Input testing covariate matrix. Can be omitted if covariate is absent. However, training covariates z and testing covariates testz must be provided or not at the same time.
standardize A logic object indicating whether x should be standardized before performing DSDA. Default is FALSE.
lambda A sequence of lambda's. If lambda is missed, the function will automatically generates a sequence of lambda's to fit model.
alpha The elasticnet mixing parameter, the same as in glmnet. Default is alpha=1 so that the lasso penalty is used.
eps Convergence threshold for coordinate descent, the same as in glmnet. Default is 1e-7.
dsda.all

Value

beta  Output variable coefficients for each lambda. The first element of each solution is the intercept.

lambda  The sequence of lambda's used in computing the solution path.
x  The predictor matrix in training data.
y  The class label in training data.
pred  Predicted categorical response for each value in sequence lambda when testx is provided.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References


Examples

data(GDS1615)  # load the prostate data
x<-GDS1615$x
y<-GDS1615$y
x=x[which(y<3),]
y=y[which(y<3)]
obj.path <- dsda(x, y=y)

Description

Performs direct sparse discriminant analysis, with the optimal lambda chosen by cross validation. The function can perform prediction on test data as well.

Usage

dsda.all(x, y, x.test.matrix=NULL, y.test=NULL, standardize=FALSE, lambda.opt="min", nfolds=10, lambda=lambda, alpha=1, eps=1e-7)
Arguments

x  An n by p matrix containing the predictors.
y  An n-dimensional vector containing the class labels 1 and 2.
x.test.matrix  The predictors of a testing set. (Optional.)
y.test  The class labels of the testing set. (Required if x.test.matrix is supplied, but otherwise optional.)
standardize  A logic object indicating whether x.matrix should be standardized before performing DSDA. Default is FALSE.
lambda.opt  Should be either "min" or "max", specifying whether the smallest or the largest lambda with the smallest cross validation error should be used for the final classification rule.
nfolds  The number of folds to be used in cross validation. Default is 10.
lambda  A sequence of lambda’s.
alpha  The elasticnet mixing parameter, the same as in glmnet. Default is alpha=1 so that the lasso penalty is used.
eps  Convergence threshold for coordinate descent, the same as in glmnet. Default is 1e-7.

Value

error  Testing error if x.test.matrix is supplied.
beta  The coefficients of the classification rule corresponding to the optimal lambda chosen by cross validation.
s  The optimal lambda chosen by cross validation.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References


See Also

dsda

Examples

data(GDS1615)  #load the prostate data
x<-GDS1615$x
y<-GDS1615$y

x=x[which(y<3),]
y=y[which(y<3)]
n <- length(y)  ## split the original dataset to a training set and a testing set
n.test <- round(n/3)
set.seed(20120822)
id <- sample(n, n.test, replace=FALSE)
x.train <- x[-id,]
x.test <- x[id,]
y.train <- y[-id]
y.test <- y[id]

set.seed(123)
# perform direct sparse discriminant analysis
obj <- dsda.all(x.train, y.train, x.test, y.test)
obj$error

---

**GDS1615**

**GDS1615 data introduced in Burczynski et al. (2012).**

**Description**

The dataset is a subset of the dataset available on Gene Expression Omnibus with the accession number GDS1615. The original dataset contains 22,283 gene expression levels and the disease states of the observed subjects. In Mai, Yang and Zou, the dimension of the original dataset was first reduced to 127 by F-test screening.

**Usage**

```r
data(GDS1615)
```

**Value**

This data frame contains the following:

- **x**: Gene expression levels.
- **y**: Disease state that is coded as 1, 2, 3: 1: normal; 2: ulcerative colitis; 3: Crohn’s disease.

**References**


**Examples**

```r
data(GDS1615)
```
Description

Transform the predictors to achieve normality.

Usage

getnorm(x, y, type="pooled")

Arguments

x an n dimensional vector containing n observations for one predictor.
y an n-dimensional vector containing the class labels.
type The type of estimator. Two estimators were proposed in Mai & Zou (2015), the naive estimator and the pooled estimator. The function getnorm() uses the naive estimator if type="naive", and it uses the pooled estimator if type="pooled". The default is "pooled". When the naive estimator is used, it is recommended to label the class with more samples as Class 0.

Value

x.norm Transformed x.
f0 The transformation computed based on observations from Class 0. Not applicable if type="naive".
f1 The transformation computed based on observations from Class 1. Not applicable if type="naive".
mu.hat The sample mean for transformed x from Class 1.
transform The transformation that was actually used to transform x.

References


Examples

data(GDS1615)   # load the prostate data
x<-GDS1615$x
y<-GDS1615$y
x<-exp(x[which(y<3),])
y<-y[which(y<3)]
msda

Fits a regularization path of Sparse Discriminant Analysis and predicts

Description

Fits a regularization path of Sparse Discriminant Analysis at a sequence of regularization parameters lambda. Performs prediction when testing data is provided. The msda function solves classification problem by fitting a sparse discriminant analysis model. When covariates are provided, the function will first make adjustment on the training data. It provides three models: binary for fitting DSDA model to solve binary classification problems, multi.original and multi.modified for fitting MSDA model to solve multi-class classification problems. multi.original runs faster for small dimension case but the computation ability is limited to a relatively large dimension. multi.modified has no such limitation and works in ultra-high dimensions. User can specify method by argument or use the default settings.

Usage

msda(x, z=NULL, y, testx=NULL, testz=NULL, model = NULL, lambda = NULL, standardize=FALSE, alpha=1, nlambda = 100,
lambda.factor = ifelse((nobs - nclass)<= nvars, 0.2, 1e-03), dfmax = nobs,
  pmax = min(dfmax * 2 + 20, nvars), pf = rep(1, nvars), eps = 1e-04,
  maxit = 1e+06, sml = 1e-06, verbose = FALSE, perturb = NULL)

Arguments

x
Input matrix of predictors. x is of dimension N × p; each row is an observation vector.

z
Input covariate matrix of dimension N × q, where q < N. z can be omitted if covariate is absent.

y
Class labl. This argument should be a factor for classification. For model='binary',
y should be a binary variable with values 1 and 2. For model='multi.original'
or 'multi.modified', y should be a multi-class variable starting from 1.

testx
Input testing matrix. Each row is a test case. When testx is not provided,
the function will only fit the model and return the classifier. When testx is
provided, the function will predict response on testx as well.

testz
Input testing covariate matrix. Can be omitted if covariate is absent. However,
training covariates z and testing covariates testz must be provided or not at the
same time.

model
Method type. The model argument can be one of 'binary', 'multi.original',
'multi.modified' and the default is NULL. The function supports fitting DSDA
and MSDA models by specifying method type. Without specification, the func-
tion will automatically choose one of the methods. If the response variable is
binary, the function will fit a DSDA model. If the response variable is multi-
class, the function will fit an original MSDA model for dimension p <= 2000
and a modified MSDA model for dimension p > 2000.

lambda
A user supplied lambda sequence. Typically, by leaving this option unspecified
users can have the program compute its own lambda sequence based on nlambda
and lambda.factor. Supplying a value of lambda overrides this. It is better to
supply a decreasing sequence of lambda values than a single (small) value, if
not, the program will sort user-defined lambda sequence in decreasing order
automatically.

standardize
A logic object indicating whether x should be standardized before performing
DSDA. Default is FALSE. This argument is only valid for model = 'binary'.

alpha
The elasticnet mixing parameter, the same as in glmnet. Default is alpha=1 so
that the lasso penalty is used in DSDA. This argument is only valid for model =
'binary'.

nlambda
The number of tuning values in sequence lambda. If users do not specify lambda
values, the package will generate a solution path containing nlambda many tun-
ing values of lambda. Default is 100 for model = 'multi.original' and 50 for
model = 'multi.modified'.

lambda.factor
The factor for getting the minimal lambda in lambda sequence, where min(lambda)
= lambda.factor * max(lambda). max(lambda) is the smallest value of lambda
for which all coefficients are zero. The default depends on p (the number of
predictors) and its relationship with N (the number of rows in the matrix of pre-
dictors). For Original MSDA, if N > p, the default is 0.0001, close to zero.
If $N < p$, the default is 0.2. For Modified MSDA, if $p \leq 5000$, the default is 0.2. If $5000 < p \leq 30000$, the default is 0.4. If $p > 30000$, the default is 0.5. A very small value of lambda.factor will lead to a saturated fit. It takes no effect if there is user-defined lambda sequence. This argument is only valid for multi.original and multi.modified.

**dfmax**
The maximum number of selected variables in the model. Default is the number of observations $N$. This argument is only valid for multi.original and multi.modified.

**pmax**
The maximum number of potential selected variables during iteration. In middle step, the algorithm can select at most $p_{\text{max}}$ variables and then shrink part of them such that the number of final selected variables is less than $df_{\text{max}}$. Default is $\min(df_{\text{max}} \times 2 + 20, N)$.

**pf**
L1 penalty factor of length $p$. Separate L1 penalty weights can be applied to each coefficient of $\theta$ to allow differential L1 shrinkage. Can be 0 for some variables, which implies no L1 shrinkage, and results in that variable always being included in the model. Default is 1 for all variables (and implicitly infinity for variables listed in exclude). This argument is only valid for multi.original and multi.modified.

**eps**
Convergence threshold for coordinate descent. Each inner coordinate descent loop continues until the relative change in any coefficient. Defaults value is 1e-4.

**maxit**
Maximum number of outer-loop iterations allowed at fixed lambda value. Default is 1e6. If models do not converge, consider increasing maxit. This argument is only valid for multi.original and multi.modified.

**sml**
Threshold for ratio of loss function change after each iteration to old loss function value. Default is 1e-06. This argument is only valid for multi.original and multi.modified.

**verbose**
Whether to print out computation progress. The default is FALSE. This argument is only valid for multi.original and multi.modified.

**perturb**
A scalar number. If it is specified, the number will be added to each diagonal element of the covariance matrix as perturbation. The default is NULL. This argument is only valid for multi.original and multi.modified.

**Details**

The msda function fits a linear discriminant analysis model for vector $X$ as follows:

$$X | Y = k \sim N(\mu_k, \Sigma).$$

The categorical response is predicted from the Bayes rule:

$$\hat{Y} = \arg \max_{k=1, \ldots, K} (X - \frac{\mu_k}{2})^T \beta_k + \log \pi_k.$$

The parameter model specifies which method to use in estimating $\beta$. Users can use binary for binary problems and binary and multi.modified for multi-class problems. In multi.original, the algorithm first computes and stores $\Sigma$, while it doesn’t compute or store the entire covariance matrix in multi.modified. Since the algorithm is element-wise based, multi.modified computes
each element of covariance matrix when needed. Therefore, multi.original is faster for low
dimension but multi.modified can fit model for a much higher dimension case.

Note that for computing speed reason, if models are not converging or running slow, consider
increasing eps and sml, or decreasing nlambda, or increasing lambda.factor before increasing
maxit. Users can also reduce dfmax to limit the maximum number of variables in the model.

The arguments list out all parameters in the three models, but not all of them are necessary in applying
one of the methods. See the specific explanation of each argument for more detail. Meanwhile,
the output of DSDA model only includes beta and lambda.

Value

An object with S3 class dsda or msda.original and msda.modified.

- **beta**: Output variable coefficients for each lambda, which is the estimation of \( \beta \) in the
  Bayes rule. beta is a list of length being the number of lambdas. Each element
  of beta is a matrix of dimension \( nvars \times (nclass - 1) \). For model = 'dsda',
  beta is a vector of length \( nvars + 1 \), where the first element is intercept.
- **df**: The number of nonzero coefficients for each value of lambda.
- **obj**: The fitted value of the objective function for each value of lambda.
- **dim**: Dimension of each coefficient matrix.
- **lambda**: The actual lambda sequence used. The user specified sequence or automatically
generated sequence could be truncated by constraints on dfmax and pmax.
- **x**: The input matrix of predictors for training.
- **y**: Class label in training data.
- **npasses**: Total number of iterations (the most inner loop) summed over all lambda values
- **jerr**: Error flag, for warnings and errors, 0 if no error.
- **sigma**: Estimated sigma matrix. This argument is only available in object msda.original.
- **delta**: Estimated delta matrix. delta[k] = mu[k]-mu[1].
- **mu**: Estimated mu vector.
- **prior**: Prior probability that y belong to class k, estimated by mean(y that belong to k).
- **call**: The call that produced this object
- **pred**: Predicted categorical response for each value in sequence lambda when testx
  is provided.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Mai, Q., Zou, H. and Yuan, M. (2012), "A direct approach to sparse discriminant analysis in ultra-
high dimensions." Biometrika, 99, 29-42.

Mai, Q., Yang, Y., and Zou, H. (2017), "Multiclass sparse discriminant analysis." Statistica Sinica,
in press.

URL: https://github.com/emeryyi/msda
predict.catch

Predict categorical responses for matrix/tensor data.

Description

Predict categorical responses on new matrix/tensor data given the fitted CATCH model input.

Usage

```r
# S3 method for class 'catch'
predict(object, newx, z = NULL, ztest = NULL, gamma = NULL, ...)
```

Arguments

- `object`: Input catchobj class object as fitted model.
- `newx`: Input adjusted testing tensor or matrix list. Each element of the list is a tensor. The tensor should of the same dimension as training data.
- `z`: Input training covariates matrix. `z` can be omitted if there is no covariate.
- `ztest`: Input testing covariates matrix. `ztest` can be omitted if there is no covariate.
- `gamma`: Coefficients of covariates obtained from `adjten`. `gamma` is NULL if there is no covariate.
- `...`: Other arguments that can be passed to `predict`.

Details

The function fits LDA model on selected discriminant vectors. Call `predict` or `predict.catch` to perform predictions.

There are two ways to make predictions. One way is to directly predict at the same time as fitting model by `catch` since `predict.catch` has already been embedded in `catch` and it will predicts response when testing data is provided. The other way is to first use `adjten` to adjuste tensor and `catch` to fit model. `predict.catch` will take the input adjusted tensor list `newx`, covariate coefficient `gamma` from `adjten` and the fitted model from `catch` to perform prediction. The prediction is identical to providing `catch` testing data.

See Also

- `cv.msda`, `predict.msda`

Examples

```r
data(GDS1615)
x<-GDS1615$x
y<-GDS1615$y
obj <- msda(x = x, y = y)
```
Value

Predicted response of newx for each lambda in model object.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References


See Also

catch, adjten

Examples

```r
#generate training data
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars),nrow=n,ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2),nrow=n,ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,],dim=c(p,p,p))
}

#generate testing data
newx <- array(list(),n)
vec_newx <- matrix(rnorm(n*nvars),nrow=n,ncol=nvars)
vec_newx[1:10,] <- vec_newx[1:10,]+2
newz <- matrix(rnorm(n*2),nrow=n,ncol=2)
ewz[1:10,] <- newz[1:10,]+0.5
for (i in 1:n){
  newx[[i]] <- array(vec_newx[i,],dim=c(p,p,p))
}

#Make adjustment and fit model
obj <- adjten(x, z, y, newx, newz)
fit <- catch(x, z, y)
#Predict
pred <- predict(fit, obj$testxres, z, newz, obj$gamma)

#The adjusting, fitting model and predicting step can also be completed
#by one command.
pred <- catch(x, z, y, newx, newz)$pred
```
Description

Predict the class labels by direct sparse discriminant analysis.

Usage

```r
## S3 method for class 'dsda'
predict(object, newx, z=NULL, ztest=NULL, gamma=NULL,...)
```

Arguments

- `object`: An object returned by `dsda` or `msda` with binary setting.
- `newx`: An n by p matrix containing the predictors.
- `z`: Input training covariates matrix. `z` can be omitted if there is no covariate.
- `ztest`: Input testing covariates matrix. `ztest` can be omitted if there is no covariate.
- `gamma`: Coefficients of covariates obtained from `adjvec`. `gamma` is NULL if there is no covariate.
- `...`: Other arguments that can be passed to `predict`.

Value

- `pred`: The the predicted class labels.

References


See Also

ddsa, dsda.all, predict.msda
**predict.msda**

*Predict categorical responses for vector data.*

**Description**

Predict categorical responses on new vector data given the fitted DSDA/MSDA model input.

**Usage**

```r
## S3 method for class 'msda'
predict(object, newx, z = NULL, ztest = NULL, gamma = NULL,...)
```

**Arguments**

- `object` Fitted model object from `msda`. The model object can be anyone of `binary`, `multi.original` and `multi.modified`.
- `newx` The matrix of new values for `x` at which predictions are to be made. If covariates exist, then `newx` should be adjusted matrix.
- `z` Input training covariates matrix. `z` can be omitted if there is no covariate.
- `ztest` Input testing covariates matrix. `ztest` can be omitted if there is no covariate.
- `gamma` Coefficients of covariates obtained from `adjvec`. `gamma` is `NULL` if there is no covariate.
- `...` Other arguments that can be passed to `predict`.

**Details**

The function fits LDA model on selected discriminant vectors. Call `predict` or `predict.msda` to perform prediction. When covariates exist, users could first call `adjvec` to make adjustment and obtain `gamma`. The fitted model from `msda` should also takes adjusted vector as input. The `newx` in `predict.msda` should be adjusted vector as well.

**Value**

Predicted class label(s) at the entire sequence of the penalty parameter `lambda` used to create the model.

**Author(s)**

Yuqing Pan, Qing Mai, Xin Zhang
References


See Also

msda

Examples

data(GDS1615)
x<-GDS1615$x
y<-GDS1615$y
obj <- msda(x = x, y = y)
pred<-predict(obj,x)

predict.SeSDA

Prediction for semiparametric sparse discriminant analysis

Description

Predict the class labels by semiparametric sparse discriminant analysis.

Usage

## S3 method for class 'SeSDA'
predict(object, x.test,...)

Arguments

object An object returned by SeSDA.
x.test An n by p matrix containing the predictors.
... Other arguments that can be passed to predict.

Value

pred The the predicted class labels.

References

ROAD

See Also
dsa, SeSDA

---

**ROAD**

*Solution path for regularized optimal affine discriminant*

**Description**

Compute the solution path for regularized optimal affine discriminant (ROAD).

**Usage**

```r
ROAD(x, y, standardize=FALSE, lambda=NULL, eps=1e-7)
```

**Arguments**

- **x**: Input matrix of predictors. \( x \) is of dimension \( N \times p \); each row is an observation vector.
- **y**: An \( n \)-dimensional vector containing the class labels. The classes have to be labeled as 1 and 2.
- **standardize**: A logic object indicating whether \( x \) should be standardized before performing ROAD. Default is FALSE.
- **lambda**: A sequence of lambda’s. If lambda is missed, the function will automatically generates a sequence of lambda’s to fit model.
- **eps**: Convergence threshold for coordinate descent, the same as in glmnet. Default is 1e-7.

**Details**

The function obtains the solution path of ROAD through dsda.

**Value**

- **beta**: Output variable coefficients for each lambda.
- **lambda**: The sequence of lambda’s used in computing the solution path.

**Author(s)**

Yuqing Pan, Qing Mai, Xin Zhang

**References**

Examples

data(GDS1615)  # load the prostate data
x<-GDS1615$x
y<-GDS1615$y
x=x[which(y<3),]
y=y[which(y<3)]
obj.path <- ROAD(x, y)

Description

Compute the solution path for semiparametric sparse discriminant analysis.

Usage

SeSDA(x, y, standardize=FALSE, lambda=NULL, alpha=1, eps=1e-7)

Arguments

x  Input matrix of predictors. x is of dimension $N \times p$; each row is an observation vector.
y  An n-dimensional vector containing the class labels. The classes have to be labeled as 1 and 2.
standardize  A logic object indicating whether x should be standardized after transformation but before fitting classifier. Default is FALSE.
lambda  A sequence of lambda's. If lambda is missed or NULL, the function will automatically generates a sequence of lambda’s to fit model.
alpha  The elasticnet mixing parameter, the same as in glmnet. Default is alpha=1 so that the lasso penalty is used.
eps  Convergence threshold for coordinate descent, the same as in glmnet. Default is 1e-7.

Value

transform  The transformation functions.
objdsda  A DSDA object fitted on transformed data.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang
References


Examples

```r
data(GDS1615)  # load the prostate data
x<-GDS1615$x
y<-GDS1615$y
x=x[which(y<3),]
y=y[which(y<3)]
obj.path <- SeSDA(x,y)
```

---

**sim.bi.vector**

*Simulate data*

**Description**

Simulate a binary data set with vector predictor.

**Usage**

```r
sim.bi.vector(tesize = 100)
```

**Arguments**

- `tesize` Number of observations in testing data.

**Details**

The function simulates a data set with $p = 500$. Response are binary.

**Value**

- `x` Simulated vector predictor.
- `testx` Simulated testing vector predictor.
- `y` Response corresponding to `x`.
- `testy` Response corresponding to `testx`.

**Author(s)**

Yuqing Pan, Qing Mai, Xin Zhang
sim.tensor.cov  

Simulate data

Description

Simulate a data set with tensor predictor and covariates.

Usage

sim.tensor.cov(tesize = 100)

Arguments

tesize  Number of observations in testing data.

Details

The function simulates a data set with $10 \times 10 \times 10$ tensor and covariate being a two-dimensional vector. Response are binary.

Value

x  Simulated tensor predictor.

z  Simulated covariate.

testx  Simulated testing tensor predictor.

testz  Simualted testing covariate.

vec_x  Vectorization of x.

vec_testx  Vectorization of testx.

y  Response corresponding to x and z.

testy  Response corresponding to testx and testz.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang
SOS

Solution path for sparse discriminant analysis

Description

Compute the solution path for sparse optimal scoring (SOS).

Usage

SOS(x, y, standardize = FALSE, lambda = NULL, eps = 1e-7)

Arguments

x  Input matrix of predictors. x is of dimension N × p; each row is an observation vector.
y  An n-dimensional vector containing the class labels. The classes have to be labeled as 1 and 2.
standardize  A logic object indicating whether x should be standardized before performing SOS. Default is FALSE.
lambda  A sequence of lambda’s. If lambda is missed, the function will automatically generates a sequence of lambda’s to fit model.
eps  Convergence threshold for coordinate descent, the same as in glmnet. Default is 1e-7.

Details

The function obtains the solution path of sparse optimal scoring model through dsda.

Value

beta  Output variable coefficients for each lambda.
lambda  The sequence of lambda’s used in computing the solution path.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Examples

data(GDS1615)  # load the prostate data
x<-GDS1615$x
y<-GDS1615$y
x=x[which(y<3),]
y=y[which(y<3)]
obj.path <- SOS(x, y)
Index

*Topic datasets
  csa, 11
  GDS1615, 20

adjten, 2, 3, 5, 7, 8, 26, 27
adjvec, 4, 28, 29

catch, 3, 4, 5, 7, 9, 10, 12, 13, 26, 27
catch_matrix, 9, 9
csa, 11
cv.catch, 8, 11, 12
cv.dsda, 13
cv.msda, 14, 15, 26
cv.SeSDA, 16

dsda, 17, 19, 28, 31, 35
dsda.all, 18, 28

GDS1615, 20
getnorm, 21

msda, 15, 16, 22, 29, 30

predict, 29
predict.catch, 7, 8, 26, 26
predict.dsda, 28
predict.msda, 26, 28, 29, 29
predict.SeSDA, 30

ROAD, 31

SeSDA, 31, 32
sim.bi.vector, 33
sim.tensor.cov, 34
SOS, 35

x (GDS1615), 20

y (GDS1615), 20