Package 'accSDA'

September 5, 2022

Version 1.1.2

Date 2022-08-11

Title Accelerated Sparse Discriminant Analysis

Imports MASS (>= 7.3.45), ggplot2 (>= 2.1.0), ggthemes (>= 3.2.0), grid (>= 3.2.2), gridExtra (>= 2.2.1)

Depends R (>= 3.2)

Description Implementation of sparse linear discriminant analysis, which is a supervised classification method for multiple classes. Various novel optimization approaches to this problem are implemented including alternating direction method of multipliers ('ADMM'), proximal gradient (PG) and accelerated proximal gradient ('APG') (See Atkins 'et al'. <arXiv:1705.07194>).
Functions for performing cross validation are also supplied along with basic prediction and plotting functions.
Sparse zero variance discriminant analysis ('SZVD') is also included in the package (See Ames and Hong, <arXiv:1401.5492>). See the 'github' wiki for a more extended description.

License GPL (>= 2)

URL https://github.com/gumeo/accSDA/wiki

BugReports https://github.com/gumeo/accSDA/issues

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Gudmundur Einarsson [aut, cre, trl],
Line Clemmensen [aut, ths],
Brendan Ames [aut],
Summer Atkins [aut]

Maintainer Gudmundur Einarsson <gumeo140688@gmail.com>

Repository CRAN

Date/Publication 2022-09-05 14:50:10 UTC

R topics documented:

---

accSDA *accSDA: A package for performing sparse discriminant analysis in various ways.*

Description

The accSDA package provides functions to perform sparse discriminant analysis using a selection of three optimization methods, proximal gradient (PG), accelerated proximal gradient (APG) and alternating direction method of multipliers (ADMM). The package is intended to extend the available tools to perform sparse discriminant analysis in R. The three methods can be called from the function ASDA. Cross validation is also implemented for the L1 regularization parameter. Functions for doing predictions, summary, printing and simple plotting are also provided. The sparse discriminant functions perform lda on the projected data by default, using the lda function in the MASS package. The functions return an object of the same class as the name of the function and provide the lda solution, along with the projected data, thus other kinds of classification algorithms can be employed on the projected data.

ASDA *Accelerated Sparse Discriminant Analysis*

Description

Applies accelerated proximal gradient algorithm, proximal gradient algorithm or alternating direction methods of multipliers algorithm to the optimal scoring formulation of sparse discriminant analysis proposed by Clemmensen et al. 2011.

$$argmin|(Y_t\theta - X_t\beta)|_2^2 + t|\beta|_1 + \lambda|\beta|_2^2$$

Usage

```
ASDA(Xt, ...)

## Default S3 method:
ASDA(
  Xt,
  Yt,
  Om = diag(p),
  gam = 0.001,
  lam = 1e-06,
  q = K - 1,
  method = "SDAAP",
  control = list(),
  ...
)
```

Arguments

Xt	n by p data matrix, (can also be a data.frame that can be coerced to a matrix)
...	Additional arguments for lda function in package MASS.
Yt	n by K matrix of indicator variables (Yij = 1 if i in class j). This will later be changed to handle factor variables as well. Each observation belongs in a single class, so for a given row/observation, only one element is 1 and the rest is 0.
Om	p by p parameter matrix Omega in generalized elastic net penalty.
gam	Regularization parameter for elastic net penalty.
lam	Regularization parameter for l1 penalty, must be greater than zero. If cross-validation is used (CV = TRUE) then this must be a vector of length greater than one.
q	Desired number of discriminant vectors.
method	This parameter selects which optimization method to use. It is specified as a character vector which can be one of the three values SDAP Proximal gradient algorithm. SDAAP Accelerated proximal gradient algorithm. SDAD Alternating directions method of multipliers algorithm. Note that further parameters are passed to the function in the argument control, which is a list with named components.
control	List of control arguments. See Details.

Details

The control list contains the following entries to further tune the algorithms.

PGsteps Maximum number if inner proximal gradient/ADMM algorithm for finding beta. Default value is 1000.

PGtol Stopping tolerance for inner method. If the method is SDAD, then this must be a vector of two values, absolute (first element) and relative tolerance (second element). Default value is 1e-5 for both absolute and relative tolerances.

maxits Number of iterations to run. Default value is 250.

tol Stopping tolerance. Default value is 1e-3.

mu Penalty parameter for augmented Lagrangian term, must be greater than zero and only needs to be specified when using method SDAD. Default value is 1.

CV Logical value which is TRUE if cross validation is supposed to be performed. If cross-validation is performed, then lam should be specified as a vector containing the regularization values to be tested. Default value is FALSE.

folds Integer determining the number of folds in cross-validation. Not needed if CV is not specified. Default value is 5.

feat Maximum fraction of nonzero features desired in validation scheme. Not needed if CV is not specified. Default value is 0.15.

quiet Set to FALSE if status updates are supposed to be printed to the R console. Default value is TRUE. Note that this triggers a lot of printing to the console.

ordinal Set to TRUE if the labels are ordinal. Only available for methods SDAAP and SDAD.

initTheta Option to set the initial theta vector, by default it is a vector of all ones for the first theta.

bt Logical indicating whether backtracking should be used, only applies to the Proximal Gradient based methods. By default, backtracking is not used.

L Initial estimate for Lipshitz constant used for backtracking. Default value is 0.25.

eta Scalar for Lipshitz constant. Default value is 1.25.

rankRed Boolean indicating whether Om is factorized, such that R^t*R=Om, currently only applicable for accelerated proximal gradient.

Value

ASDA returns an object of class "ASDA" including a list with the following named components:

call The matched call.

B p by q matrix of discriminant vectors, i.e. sparse loadings.

Q K by q matrix of scoring vectors, i.e. optimal scores.

varNames Names of the predictors used, i.e. column names of Xt.

origP Number of variables in Xt.

fit Output from function lda on projected data. This is NULL the trivial solution is found, i.e. B is all zeroes. Use lower values of lam if that is the case.

classes The classes in Yt.

lambda The lambda/lam used, best value found by cross- validation if CV is TRUE.

NULL

Note

The input matrix Xt should be normalized, i.e. each column corresponding to a variable should have its mean subtracted and scaled to unit length. The functions normalize and normalizetest are supplied for this purpose in the package.

See Also

SDAAP, SDAP and SDAD

Examples

```
set.seed(123)
# Prepare training and test set
train <- c(1:40,51:90,101:140)
Xtrain <- iris[train,1:4]
nX <- normalize(Xtrain)
Xtrain <- nX$Xc
Ytrain <- iris[train,5]
Xtest <- iris[-train,1:4]
Xtest <- normalizetest(Xtest,nX)
Ytest <- iris[-train,5]

# Define parameters for Alternating Direction Method of Multipliers (SDAD)
Om <- diag(4)+0.1*matrix(1,4,4) #elNet coef mat
gam <- 0.0001
lam <- 0.0001
method <- "SDAD"
q <- 2
control <- list(PGsteps = 100,
                PGtol = c(1e-5,1e-5),
                mu = 1,
                maxits = 100,
                tol = 1e-3,
                quiet = FALSE)

# Run the algorithm
res <- ASDA(Xt = Xtrain,
            Yt = Ytrain,
            Om = Om,
            gam = gam ,
            lam = lam,
            q = q,
            method = method,
            control = control)

# Can also just use the defaults, which is Accelerated Proximal Gradient (SDAAP):
resDef <- ASDA(Xtrain,Ytrain)

# Some example on simulated data
# Generate Gaussian data on three classes with plenty of redundant variables

# This example shows the basic steps on how to apply this to data, i.e.:
```

```
#  1) Setup training data
#  2) Normalize
#  3) Train
#  4) Predict
#  5) Plot projected data
#  6) Accuracy on test set

P <- 300 # Number of variables
N <- 50 # Number of samples per class

# Mean for classes, they are zero everywhere except the first 3 coordinates
m1 <- rep(0,P)
m1[1] <- 3

m2 <- rep(0,P)
m2[2] <- 3

m3 <- rep(0,P)
m3[3] <- 3

# Sample dummy data
Xtrain <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))

Xtest <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))

# Generate the labels
Ytrain <- factor(rep(1:3,each=N))
Ytest <- Ytrain

# Normalize the data
Xt <- accSDA::normalize(Xtrain)
Xtrain <- Xt$Xc # Use the centered and scaled data
Xtest <- accSDA::normalizetest(Xtest,Xt)

# Train the classifier and increase the sparsity parameter from the default
# so we penalize more for non-sparse solutions.
res <- accSDA::ASDA(Xtrain,Ytrain,lam=0.01)

# Plot the projected training data, it is projected to
# 2-dimension because we have 3 classes. The number of discriminant
# vectors is maximum number of classes minus 1.
XtrainProjected <- Xtrain%*%res$beta

plot(XtrainProjected[,1],XtrainProjected[,2],col=Ytrain)

# Predict on the test data
preds <- predict(res, newdata = Xtest)

# Plot projected test data with predicted and correct labels
```

```
XtestProjected <- Xtest%*%res$beta

plot(XtestProjected[,1],XtestProjected[,2],col=Ytest,
     main="Projected test data with original labels")
plot(XtestProjected[,1],XtestProjected[,2],col=preds$class,
     main="Projected test data with predicted labels")

# Calculate accuracy
sum(preds$class == Ytest)/(3*N) # We have N samples per class, so total 3*N
```

ASDABarPlot *barplot for ASDA objects*

Description

This is a function to visualize the discriminant vector from the ASDA method. The plot is constructed as a ggplot barplot and the main purpose of it is to visually inspect the sparsity of the discriminant vectors. The main things to look for are how many parameters are non-zero and if there is any structure in the ones that are non-zero, but the structure is dependent on the order you specify your variables. For time-series data, this could mean that a chunk of variables are non-zero that are close in time, meaning that there is some particular event that is best for discriminating between the classes that you have.

Usage

```
ASDABarPlot(asdaObj, numDVs = 1, xlabel, ylabel, getList = FALSE, main, ...)
```

Arguments

asdaObj	Object from the ASDA function.
numDVs	Number of discriminant vectors (DVs) to plot. This is limited by the number of DVs outputted from the ASDA function or k-1 DVs where k is the number of classes. The first 1 to numDVs are plotted.
xlabel	Label to put under every plot
ylabel	Vector of y-axis labels for each plot, e.g. if there are three DVs, then ylab = c('Discriminant Vector 1', 'Discriminant Vector 2', 'Discriminant Vector 3') is a valid option.
getList	Logical value indicating whether the output should be a list of the plots or the plots stacked in one plot using the gridExtra package. By default the function produces a single plot combining all plots of the DVs.
main	Main title for the plots, this is not used if getList is set to TRUE.
...	Extra arguments to grid.arrange.

Value

barplot.ASDA returns either a single combined plot or a list of individual ggplot objects.

Note

This function is used as a quick diagnostics tool for the output from the ASDA function. Feel free to look at the code to customize the plots in any way you like.

See Also

ASDA

Examples

```
 # Generate and ASDA object with your data, e.g.
 # Prepare training and test set
 # This is a very small data set, I advise you to try it on something with more
# variables, e.g. something from this source: http://www.cs.ucr.edu/~eamonn/time_series_data/
 # or possibly run this on the Gaussian data example from the ASDA function
 train <- c(1:40,51:90,101:140)
 Xtrain <- iris[train,1:4]
 nX <- normalize(Xtrain)
 Xtrain <- nX$Xc
 Ytrain <- iris[train,5]
 Xtest <- iris[-train,1:4]
 Xtest <- normalizetest(Xtest,nX)
 Ytest <- iris[-train,5]
 # Run the method
 resIris <- ASDA(Xtrain,Ytrain)

 # Look at the barplots of the DVs
 ASDABarPlot(resIris)
```

---

genDat *Generate data for ordinal examples in the package*

Description

Given the parameters, the function creates a dataset for testing the ordinal functionality of the package. The data is samples from multivariate Gaussians with different means, where the mean varies along a sinusoidal curve w.r.t. the class label.

Usage

```
genDat(numClasses, numObsPerClass, mu, sigma)
```

Arguments

numClasses	Positive integer specifying the number of classes for the dataset.
numObsPerClass	Number of observations sampled per class.
mu	Mean of the first class.
sigma	2 by 2 covariance matrix

Details

This function is used to demonstrate the usage of the ordinal classifier.

Value

genDat Returns a list with the following attributes:

X A matrix with two columns and numObsPerClass*numClasses rows.

Y Labels for the rows of X.

Author(s)

Gudmundur Einarsson

See Also

ordASDA

Examples

```
set.seed(123)

    # You can play around with these values to generate some 2D data to test one
    numClasses <- 15
    sigma <- matrix(c(1,-0.2,-0.2,1),2,2)
    mu <- c(0,0)
    numObsPerClass <- 5

    # Generate the data, can access with train$X and train$Y
    train <- accSDA::genDat(numClasses,numObsPerClass,mu,sigma)
    test <- accSDA::genDat(numClasses,numObsPerClass*2,mu,sigma)

    # Visualize it, only using the first variable gives very good separation
    plot(train$X[,1],train$X[,2],col = factor(train$Y),asp=1,main="Training Data")
```

---

normalize *Normalize training data*

Description

Normalize a vector or matrix to zero mean and unit length columns.

Usage

```
normalize(X)
```

Arguments

X	a matrix with the training data with observations down the rows and variables in the columns.

Details

This function can e.g. be used for the training data in the ASDA function.

Value

normalize Returns a list with the following attributes:

Xc The normalized data

mx Mean of columns of X.

vx Length of columns of X.

Id Logical vector indicating which variables are included in X. If some of the columns have zero length they are omitted

Author(s)

Line Clemmensen

References

Clemmensen, L., Hastie, T. and Ersboell, K. (2008) "Sparse discriminant analysis", Technical report, IMM, Technical University of Denmark

See Also

normalizetest, predict.ASDA, ASDA

Examples

```
## Data
X<-matrix(sample(seq(3),12,replace=TRUE),nrow=3)

## Normalize data
Nm<-normalize(X)
print(Nm$Xc)

## See if any variables have been removed
which(!Nm$Id)
```

`normalizetest` *Normalize training data*

Description

Normalize test data using output from the `normalize()` of the training data

Usage

```
normalizetest(Xtst, Xn)
```

Arguments

`Xtst`	a matrix with the test data with observations down the rows and variables in the columns.
`Xn`	List with the output from normalize(Xtr) of the training data.

Details

This function can e.g. be used for the test data in the `predict.ASDA` function.

Value

`normalizetest` returns the normalized test data `Xtst`

Author(s)

Line Clemmensen

References

Clemmensen, L., Hastie, T. and Ersboell, K. (2008) "Sparse discriminant analysis", Technical report, IMM, Technical University of Denmark

See Also

`normalize`, `predict.ASDA`, `ASDA`

Examples

```
## Data
Xtr<-matrix(sample(seq(3),12,replace=TRUE),nrow=3)
Xtst<-matrix(sample(seq(3),12,replace=TRUE),nrow=3)

## Normalize training data
Nm<-normalize(Xtr)

## Normalize test data
Xtst<-normalizetest(Xtst,Nm)
```

ordASDA *Ordinal Accelerated Sparse Discriminant Analysis*

Description

Applies accelerated proximal gradient algorithm to the optimal scoring formulation of sparse discriminant analysis proposed by Clemmensen et al. 2011. The problem is further casted to a binary classification problem as described in "Learning to Classify Ordinal Data: The Data Replication Method" by Cardoso and da Costa to handle the ordinal labels. This function serves as a wrapper for the ASDA function, where the appropriate data augmentation is performed. Since the problem is casted into a binary classication problem, only a single discriminant vector comes from the result. The first *p* entries correspond to the variables/coefficients for the predictors, while the following K-1 entries correspond to biases for the found hyperplane, to separate the classes. The resulting object is of class ordASDA and has an accompanying predict function. The paper by Cardoso and dat Costa can be found here: (http://www.jmlr.org/papers/volume8/cardoso07a/cardoso07a.pdf).

Usage

```
ordASDA(Xt, ...)

## Default S3 method:
ordASDA(
  Xt,
  Yt,
  s = 1,
  Om,
  gam = 0.001,
  lam = 1e-06,
  method = "SDAAP",
  control,
  ...
)
```

Arguments

Xt	n by p data matrix, (can also be a data.frame that can be coerced to a matrix)
...	Additional arguments for ASDA and lda function in package MASS.
Yt	vector of length n, equal to the number of samples. The classes should be 1,2,...,K where K is the number of classes. Yt needs to be a numeric vector.
s	We need to find a hyperplane that separates all classes with different biases. For each new bias we define a binary classification problem, where a maximum of s ordinal classes or contained in each of the two classes. A higher value of s means that more data will be copied in the data augmentation step. BY default s is 1.
Om	p by p parameter matrix Omega in generalized elastic net penalty, where p is the number of variables.

gam	Regularization parameter for elastic net penalty, must be greater than zero.
lam	Regularization parameter for l1 penalty, must be greater than zero.
method	String to select method, now either SDAD or SDAAP, see ?ASDA for more info.
control	List of control arguments further passed to ASDA. See ASDA.

Value

ordASDA returns an object of class "ordASDA" including a list with the same components as an ASDA objects and:

h Scalar value for biases.

K Number of classes.

NULL

Note

Remember to normalize the data.

See Also

ASDA.

Examples

```
set.seed(123)

# You can play around with these values to generate some 2D data to test one
numClasses <- 5
sigma <- matrix(c(1,-0.2,-0.2,1),2,2)
mu <- c(0,0)
numObsPerClass <- 5

# Generate the data, can access with train$X and train$Y
train <- accSDA::genDat(numClasses,numObsPerClass,mu,sigma)
test <- accSDA::genDat(numClasses,numObsPerClass*2,mu,sigma)

# Visualize it, only using the first variable gives very good separation
plot(train$X[,1],train$X[,2],col = factor(train$Y),asp=1,main="Training Data")

# Train the ordinal based model
res <- accSDA::ordASDA(train$X,train$Y,s=2,h=1, gam=1e-6, lam=1e-3)
vals <- predict(object = res,newdata = test$X) # Takes a while to run ~ 10 seconds
sum(vals==test$Y)/length(vals) # Get accuracy on test set
#plot(test$X[,1],test$X[,2],col = factor(test$Y),asp=1,
#      main="Test Data with correct labels")
#plot(test$X[,1],test$X[,2],col = factor(vals),asp=1,
#    main="Test Data with predictions from ordinal classifier")
```

predict.ASDA *Predict method for sparse discriminant analysis*

Description

Predicted values based on fit from the function ASDA. This function is used to classify new observations based on their explanatory variables/features.

Usage

```
## S3 method for class 'ASDA'
predict(object, newdata = NULL, ...)
```

Arguments

object	Object of class ASDA. This object is returned from the function ASDA.
newdata	A matrix of new observations to classify.
...	Arguments passed to predict.lda.

Value

A list with components:

class The classification (a factor)

posterior posterior probabilities for the classes

x the scores

Note

The input matrix newdata should be normalized w.r.t. the normalization of the training data

See Also

SDAAP, SDAP and SDAD

Examples

```
# Prepare training and test set
train <- c(1:40,51:90,101:140)
Xtrain <- iris[train,1:4]
nX <- normalize(Xtrain)
Xtrain <- nX$Xc
Ytrain <- iris[train,5]
Xtest <- iris[-train,1:4]
Xtest <- normalizetest(Xtest,nX)
Ytest <- iris[-train,5]

# Define parameters for SDAD
```

```
Om <- diag(4)+0.1*matrix(1,4,4) #elNet coef mat
gam <- 0.01
lam <- 0.01
method <- "SDAD"
q <- 2
control <- list(PGsteps = 100,
                PGtol = c(1e-5,1e-5),
                mu = 1,
                maxits = 100,
                tol = 1e-3,
                quiet = FALSE)

# Run the algorithm
res <- ASDA(Xt = Xtrain,
            Yt = Ytrain,
            Om = Om,
            gam = gam ,
            lam = lam,
            q = q,
            method = method,
            control = control)

# Do the predictions on the test set
preds <- predict(object = res, newdata = Xtest)
```

`predict.ordASDA` *Predict method for ordinal sparse discriminant analysis*

Description

Predicted values based on fit from the function `ordASDA`. This function is used to classify new observations based on their explanatory variables/features. There is no need to normalize the data, the data is normalized based on the normalization data from the ordASDA object.

Usage

```
## S3 method for class 'ordASDA'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class ordASDA. This object is returned from the function `ordASDA`.
`newdata`	A matrix of new observations to classify.
`...`	Arguments passed to `predict.lda`.

Value

A vector of predictions.

See Also

ordASDA

Examples

```
set.seed(123)

# You can play around with these values to generate some 2D data to test one
numClasses <- 5
sigma <- matrix(c(1,-0.2,-0.2,1),2,2)
mu <- c(0,0)
numObsPerClass <- 5

# Generate the data, can access with train$X and train$Y
train <- accSDA::genDat(numClasses,numObsPerClass,mu,sigma)
test <- accSDA::genDat(numClasses,numObsPerClass*2,mu,sigma)

# Visualize it, only using the first variable gives very good separation
plot(train$X[,1],train$X[,2],col = factor(train$Y),asp=1,main="Training Data")

# Train the ordinal based model
res <- accSDA::ordASDA(train$X,train$Y,s=2,h=1, gam=1e-6, lam=1e-3)
vals <- predict(object = res,newdata = test$X) # Takes a while to run ~ 10 seconds
sum(vals==test$Y)/length(vals) # Get accuracy on test set
#plot(test$X[,1],test$X[,2],col = factor(test$Y),asp=1,
#      main="Test Data with correct labels")
#plot(test$X[,1],test$X[,2],col = factor(vals),asp=1,
#     main="Test Data with predictions from ordinal classifier")
```

---

print.ASDA *Print method for ASDA object*

Description

Prints a summary of the output from the ASDA function. The output summarizes the discriminant analysis in human readable format.

Usage

```
## S3 method for class 'ASDA'
print(x, digits = max(3, getOption("digits") - 3), numshow = 5, ...)
```

Arguments

x	Object of class ASDA. This object is returned from the function ASDA.
digits	Number of digits to show in printed numbers.
numshow	Number of best ranked variables w.r.t. to their absolute coefficients.
...	arguments passed to or from other methods.

Value

An invisible copy of x.

See Also

ASDA, predict.ASDA and SDAD

Examples

```
# Prepare training and test set
train <- c(1:40,51:90,101:140)
Xtrain <- iris[train,1:4]
nX <- normalize(Xtrain)
Xtrain <- nX$Xc
Ytrain <- iris[train,5]
Xtest <- iris[-train,1:4]
Xtest <- normalizetest(Xtest,nX)
Ytest <- iris[-train,5]

# Run the algorithm
resDef <- ASDA(Xtrain,Ytrain)

# Print
print(resDef)
```

SZVD *Sparse Zero Variance Discriminant Analysis*

Description

Applies SZVD heuristic for sparse zero-variance discriminant analysis to given training set.

Usage

```
SZVD(train, ...)

## Default S3 method:
SZVD(
  train,
  gamma,
  D,
  penalty = TRUE,
  scaling = TRUE,
  tol = list(abs = 1e-04, rel = 1e-04),
  maxits = 2000,
  beta = 1,
  quiet = TRUE
)
```

Arguments

train	Data matrix where first column is the response class.
...	Parameters passed to SZVD.default.
gamma	Set of regularization parameters controlling l1-penalty.
D	dictionary/basis matrix.
penalty	Controls whether to apply reweighting of l1-penalty (using sigma = within-class std devs).
scaling	Logical indicating whether to scale data such that each feature has variance 1.
tol	Stopping tolerances for ADMM algorithm, must include tol$rel and tol$abs.
maxits	Maximum number of iterations used in the ADMM algorithm.
beta	penalty term controlling the splitting constraint.
quiet	Print intermediate outpur or not.

Details

This function will currently solve as a standalone function in accSDA for time comparison. A wrapper function like ASDA will be created to use the functionality of plots and such. Maybe call it ASZDA. For that purpose the individual ZVD function will need to be implemented.

Value

SZVD returns an object of class "SZVD" including a list with the following named components:

DVs Discriminant vectors.

its Number of iterations required to find DVs.

pen_scal Weights used in reweighted l1-penalty.

N Basis for the null-space of the sample within-class covariance.

means Training class-means.

mus Training meand and variance scaling/centering terms.

w0 unpenalized zero-variance discriminants (initial solutions) plus B and W, etc.

NULL

See Also

Used by: SZVDcv.

Examples

```
set.seed(123)
P <- 300 # Number of variables
N <- 50 # Number of samples per class

# Mean for classes, they are zero everywhere except the first 3 coordinates
m1 <- rep(0,P)
```

```
m1[1] <- 3

m2 <- rep(0,P)
m2[2] <- 3

m3 <- rep(0,P)
m3[3] <- 3

# Sample dummy data
Xtrain <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
               MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))


# Generate the labels
Ytrain <- rep(1:3,each=N)

# Normalize the data
Xt <- accSDA::normalize(Xtrain)
Xtrain <- Xt$Xc

# Train the classifier and increase the sparsity parameter from the default
# so we penalize more for non-sparse solutions.
res <- accSDA::SZVD(cbind(Ytrain,Xtrain),beta=2.5,
                    maxits=1000,tol = list(abs = 1e-04, rel = 1e-04))
```

SZVDcv *Cross-validation of sparse zero variance discriminant analysis*

Description

Applies alternating direction methods of multipliers to solve sparse zero variance discriminant analysis.

Usage

```
SZVDcv(Atrain, ...)

## Default S3 method:
SZVDcv(
  Atrain,
  Aval,
  k,
  num_gammas,
  g_mults,
  D,
  sparsity_pen,
  scaling,
  penalty,
```

```
  beta,
  tol,
  ztol,
  maxits,
  quiet
)
```

Arguments

Atrain	Training data set.
...	Parameters passed to SZVD.default.
Aval	Validation set.
k	Number of classes within training and validation sets.
num_gammas	Number of gammas to train on.
g_mults	Parameters defining range of gammas to train, g_max*(c_min, c_max). Note that it is an array/vector with two elements.
D	Penalty dictionary basis matrix.
sparsity_pen	weight defining validation criteria as weighted sum of misclassification error and cardinality of discriminant vectors.
scaling	Whether to rescale data so each feature has variance 1.
penalty	Controls whether to apply reweighting of l1-penalty (using sigma = within-class std devs)
beta	Parameter for augmented Lagrangian term in the ADMM algorithm.
tol	Stopping tolerances for the ADMM algorithm, must have tol$rel and tol$abs.
ztol	Threshold for truncating values in DVs to zero.
maxits	Maximum number of iterations used in the ADMM algorithm.
quiet	Controls display of intermediate results.

Details

This function might require a wrapper similar to ASDA.

Value

SZVDcv returns an object of class "SZVDcv" including a list with the following named components:

DVs Discriminant vectors for the best choice of gamma.

all_DVs Discriminant vectors for all choices of gamma.

l0_DVs Discriminant vectors for gamma minimizing cardinality.

mc_DVs Discriminant vector minimizing misclassification.

gamma Choice of gamma minimizing validation criterion.

gammas Set of all gammas trained on.

max_g Maximum value of gamma guaranteed to yield a nontrivial solution.

ind Index of best gamma.

w0 unpenalized zero-variance discriminants (initial solutions) plus B and W, etc. from ZVD

NULL

See Also

Non CV version: SZVD.

Examples

```
P <- 300 # Number of variables
N <- 50 # Number of samples per class

# Mean for classes, they are zero everywhere except the first 3 coordinates
m1 <- rep(0,P)
m1[1] <- 3

m2 <- rep(0,P)
m2[2] <- 3

m3 <- rep(0,P)
m3[3] <- 3

# Sample dummy data
Xtrain <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
               MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))
Xval <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
               MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))

# Generate the labels
Ytrain <- rep(1:3,each=N)
Yval <- rep(1:3,each=N)


# Train the classifier and increase the sparsity parameter from the default
# so we penalize more for non-sparse solutions.

res <- accSDA::SZVDcv(cbind(Ytrain,Xtrain),cbind(Yval,Xval),num_gammas=4,
                      g_mults = c(0,1),beta=2.5,
                      D=diag(P), maxits=100,tol=list(abs=1e-3,rel=1e-3), k = 3,
                     ztol=1e-4,sparsity_pen=0.3,quiet=FALSE,penalty=TRUE,scaling=TRUE)
```

SZVD_kFold_cv *Cross-validation of sparse zero variance discriminant analysis*

Description

Applies alternating direction methods of multipliers to solve sparse zero variance discriminant analysis.

Usage

```
SZVD_kFold_cv(X, ...)

## Default S3 method:
SZVD_kFold_cv(
  X,
  Y,
  folds,
  gams,
  beta,
  D,
  q,
  maxits,
  tol,
  ztol,
  feat,
  penalty,
  quiet
)
```

Arguments

X	n by p data matrix, variables should be scaled to by sd
...	Parameters passed to SZVD.default.
Y	n by K indicator matrix.
folds	number of folds to use in K-fold cross-validation.
gams	Number of regularly spaced regularization parameters to try in [0,1]*max_gamma. See details for how max_gamma is computed in the function.
beta	Augmented Lagrangian parameter. Must be greater than zero.
D	Penalty dictionary basis matrix.
q	Desired number of discriminant vectors.
maxits	Number of iterations to run ADMM algorithm.
tol	Stopping tolerances for ADMM, must have tol$rel and tol$abs.
ztol	Rounding tolerance for truncating entries to 0.
feat	Maximum fraction of nonzero features desired in validation scheme.
penalty	Controls whether to apply reweighting of l1-penalty (using sigma = within-class std devs)
quiet	toggles between displaying intermediate statistics.

Details

Add how max_gamma is calculated from the ZVD solution. This function might require a wrapper similar to ASDA.

Value

SZVDcv returns an object of class "SZVDcv" including a list with the named components DVs and gambest. Where DVs are the discriminant vectors for the best l1 regularization parameter and gambest is the best regularization parameter found in the cross-validation.

NULL

See Also

Used by: SZVDcv.

Examples

```
P <- 150 # Number of variables
N <- 20 # Number of samples per class

# Mean for classes, they are zero everywhere except the first 3 coordinates
m1 <- rep(0,P)
m1[1] <- 3

m2 <- rep(0,P)
m2[2] <- 3

m3 <- rep(0,P)
m3[3] <- 3

# Sample dummy data
Xtrain <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
                MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
               MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))


# Generate the labels
Ytrain <- cbind(c(rep(1,N),rep(0,2*N)),
                 c(rep(0,N),rep(1,N),rep(0,N)),
                 c(rep(0,2*N),rep(1,N)))

# Normalize the data
Xt <- accSDA::normalize(Xtrain)
Xtrain <- Xt$Xc

# Train the classifier and increase the sparsity parameter from the default
# so we penalize more for non-sparse solutions.
res <- accSDA::SZVD_kFold_cv(Xtrain,Ytrain,folds=2,gams=2,beta=2.5,q=1, D=diag(P),
                             maxits=50,tol=list(abs=1e-2,rel=1e-2),
                             ztol=1e-3,feat=0.3,quiet=FALSE,penalty=TRUE)
```

ZVD *Zero Variance Discriminant Analysis*

Description

Implements the ZVD algorithm to solve dicriminant vectors.

Usage

```
ZVD(A, ...)

## Default S3 method:
ZVD(A, scaling = FALSE, get_DVs = FALSE)
```

Arguments

A	Matrix, where first column corresponds to class labels.
...	Parameters passed to ZVD.default.
scaling	Logical whether to rescale data so each feature has variance 1.
get_DVs	Logical whether to obtain unpenalized zero-variance discriminant vectors.

Details

This function should potentially be made internal for the release.

Value

SZVDcv returns an object of class "ZVD" including a list with the following named components:

dvs discriminant vectors (optional).

B sample between-class covariance.

W sample within-class covariance.

N basis for the null space of the sample within-class covariance.

mu training mean and variance scaling/centering terms

means vectors of sample class-means.

k number of classes in given data set.

labels list of classes.

obs matrix of data observations.

class_obs Matrices of observations of each class.

NULL

See Also

Used by: SZVDcv.

Examples

```
# Generate Gaussian data on three classes with bunch of redundant variables

P <- 300 # Number of variables
N <- 50 # Number of samples per class

# Mean for classes, they are zero everywhere except the first 3 coordinates
m1 <- rep(0,P)
m1[1] <- 3

m2 <- rep(0,P)
m2[2] <- 3

m3 <- rep(0,P)
m3[3] <- 3

# Sample dummy data
Xtrain <- rbind(MASS::mvrnorm(n=N,mu = m1, Sigma = diag(P)),
            MASS::mvrnorm(n=N,mu = m2, Sigma = diag(P)),
            MASS::mvrnorm(n=N,mu = m3, Sigma = diag(P)))


# Generate the labels
Ytrain <- rep(1:3,each=N)

# Normalize the data
Xt <- accSDA::normalize(Xtrain)
Xtrain <- Xt$Xc

# Train the classifier and increase the sparsity parameter from the default
# so we penalize more for non-sparse solutions.
res <- accSDA::ZVD(cbind(Ytrain,Xtrain))
```

Index