Package 'penalizedLDA'

July 11, 2015

Type Package

Title Penalized Classification using Fisher's Linear Discriminant

Version 1.1

Date 2015-07-09

Author Daniela Witten

Maintainer Daniela Witten <dwitten@u.washington.edu>

Description Implements the penalized LDA proposal of ``Witten and Tibshirani (2011), Penalized classification using Fisher's linear discriminant, to appear in Journal of the Royal Statistical Society, Series B".

License GPL (>= 2)

LazyLoad yes

Depends R (>= 1.9.0)

Imports flsa, graphics, stats, utils

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-11 01:39:15

R topics documented:

penalizedLDA-package *Penalized linear discriminant analysis using lasso and fused lasso penalties.*

Description

This package performs penalized linear discriminant analysis, intended for the high-dimensional setting in which the number of features p exceeds the number of observations n. Fisher's discriminant problem is modified in two ways: 1. A diagonal estimate of the within-class class covariance is used. 2. Lasso or fused lasso penalties are applied to the discriminant vectors in order to encourage sparsity, or sparsity and smoothness.

Details

Package:	penalizedLDA
Type:	Package
Version:	1.1
Date:	2015-07-09
License:	GPL (>=2.0)
LazyLoad:	yes

The main functions are PenalizedLDA, which performs penalized linear discriminant analysis, and PenalizedLDA.cv, which performs cross-validation in order to select the optimal tuning parameters for penalized LDA.

Author(s)

Daniela M. Witten

Maintainer: Daniela M. Witten <dwitten@u.washington.edu>

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discriminant. To appear in Journal of the Royal Statistical Society, Series B.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
out <- PenalizedLDA(x,y,lambda=.14,K=2)
print(out)
```

PenalizedLDA *Perform penalized linear discriminant analysis using L1 or fused lasso penalties.*

Description

Solve Fisher's discriminant problem in high-dimensions using (a) a diagonal estimate of the within-class covariance matrix, and (b) lasso or fused lasso penalties on the discriminant vectors.

Usage

```
PenalizedLDA(x, y, xte=NULL, type = "standard", lambda, K = 2, chrom =
NULL, lambda2 = NULL, standardized = FALSE, wcsd.x = NULL, ymat = NULL,
maxiter = 20, trace=FALSE)
```

Arguments

x	A nxp data matrix; n observations on the rows and p features on the columns.
y	A n-vector containing the class labels. Should be coded as 1, 2, . . . , nclasses, where nclasses is the number of classes.
xte	A mxp data matrix; m test observations on the rows and p features on the columns. Predictions will be made at these test observations. If NULL then no predictions will be output.
type	Either "standard" or "ordered". The former will result in the use of lasso penalties, and the latter will result in fused lasso penalties. "Ordered" is appropriate if the features are ordered and it makes sense for the discriminant vector(s) to preserve that ordering.
lambda	If type="standard" then this is the lasso penalty tuning parameter. If type="ordered" then this is the tuning parameter for the sparsity component of the fused lasso penalty term.
K	The number of discriminant vectors desired. Must be no greater than (number of classes - 1).
chrom	Only applies to type="ordered". Should be used only if the p features correspond to chromosomal locations. In this case, a numeric vector of length p indicating which "chromosome" each feature belongs to. The purpose is to avoid imposing smoothness between chromosomes.
lambda2	If type="ordered", then this penalty controls the smoothness term in the fused lasso penalty. Larger lambda2 will lead to more smoothness.
standardized	Have the features on the data already been standardized to have mean zero and within-class standard deviation 1? In general, set standardized=FALSE.
wcsd.x	If the within-class standard deviation for each feature has already been computed, it can be passed in. Usually will be NULL.
ymat	If y has already been converted into a n x nclasses matrix of indicator variables, it can be passed in. Usually will be NULL.

maxiter	Maximum number of iterations to be performed; default is 20.
trace	Print out progress through iterations?

Details

Assume that the features (columns) of x have been standardized to have mean 0 and standard deviation 1. Then, if type="standard", the optimization problem for the first discriminant vector is

max_b b' Sigma_bet b - lambda*d*sum(abs(b)) s.t. ||b||^2 <= 1

where d is the largest eigenvalue of Sigma_bet.

Alternatively, if type="ordered", the optimization problem for the first discriminant vector is

max_b b' Sigma_bet b - lambda*d*sum(abs(b)) - lambda2*d*sum(abs(bj - b(j-1))) s.t. ||b||^2 <= 1.

For details about the optimization problem for obtaining subsequent discriminant vectors, see the paper below.

Value

ypred	A mxK of predicted class labels for the test observations; output ONLY if xte was passed in. The kth column indicates the test set classifications if the first k discriminant vectors are used. If K=1 then simple a m-vector is output.
discrim	A pxK matrix of penalized discriminant vectors. Note that these discriminant vectors were computed after first scaling each feature of the data to have mean zero and within-class standard deviation one.
xproj	A nxK matrix of the data projected onto the discriminant vectors
xteproj	A mxK matrix of the data projected onto the discriminant vectors; output ONLY if xte was passed in.
crits	The value of the objective at each iteration. If K>1 then this is a list with the value of the objective obtained in.

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discrimint. To appear in JRSSB.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
out <- PenalizedLDA(x,y,lambda=.14,K=2)
```

```
print(out)
plot(out)
# For more examples, try "?PenalizedLDA.cv"
```

PenalizedLDA.cv *Perform cross-validation for penalized linear discriminant analysis.*

Description

Performs cross-validation for PenalizedLDA function.

Usage

```
PenalizedLDA.cv(x, y, lambdas = NULL, K = NULL, nfold = 6, folds = NULL,
    type = "standard", chrom = NULL, lambda2 = NULL)
```

Arguments

x	A nxp data matrix; n is the number of observations and p is the number of features.
y	A n-vector y containing class labels, represented as 1, 2, . . . , nclasses.
lambdas	A vector of lambda values to be considered.
K	The number of discriminant vectors to be used. If K is not specified, then cross-validation will be performed in order to choose the number of discriminant vectors to use.
nfold	Number of cross-validation folds.
folds	Optional - one can pass in a list containing the observations that should be used as the test set in each cross-validation fold.
type	Either "standard" or "ordered". The former will result in the use of lasso penalties, and the latter will result in fused lasso penalties. "Ordered" is appropriate if the features are ordered and it makes sense for the discriminant vector(s) to preserve that ordering.
chrom	Only applies to type="ordered". Should be used only if the p features correspond to chromosomal locations. In this case, a numeric vector of length p indicating which "chromosome" each feature belongs to. The purpose is to avoid imposing smoothness between chromosomes.
lambda2	If type is "ordered", enter the value of lambda2 to be used. Note that cross-validation is performed over lambda (and possibly over K) but not over lambda2.

Value

errs	The mean cross-validation error rates obtained. Either a vector of length equal to length(lambdas) or a length(lambdas)x(length(unique(y))-1) matrix. The former will occur if K is specified and the latter will occur otherwise, in which case cross-validation occurred over K as well as over lambda.

nnonzero	A vector or matrix of the same dimension as "errs". Entries indicate the number of nonzero features involved in the corresponding classifier.
bestK	Value of K(= number of discriminant vectors) that minimizes the cross-validation error.
bestlambda	Value of "lambdas" that minimizes the cross-validation error.
bestlambda.1se	Given that K equals bestK, this is the largest value of lambda such that the corresponding error is within 1 standard error of the minimum. This is the "one standard error" rule for selecting the tuning parameter.
lambdas	Values of lambda considered.
Ks	Values of K considered - only output if K=NULL was input.
folds	Folds used in cross-validation.

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discriminant. To appear in JRSSB.

Examples

```
# Generate data #
set.seed(1)
n <- 20 # number of training obs
m <- 40 # number of test obs
p <- 100 # number of features
x <- matrix(rnorm(n*p), ncol=p)
xte <- matrix(rnorm(m*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,6), rep(4,4))
yte <- rep(1:4, each=10)
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
x[y==3,21:30] <- x[y==3,21:30] - 2.5
xte[yte==1,1:10] <- xte[yte==1,1:10] + 2
xte[yte==2,11:20] <- xte[yte==2,11:20] - 2
xte[yte==3,21:30] <- xte[yte==3,21:30] - 2.5


# Perform cross-validation #
# Use type="ordered" -- that is, we are assuming that the features have
# some sort of spatial structure
cv.out <-
PenalizedLDA.cv(x,y,type="ordered",lambdas=c(1e-4,1e-3,1e-2,.1,1,10),lambda2=.3)
print(cv.out)
plot(cv.out)
# Perform penalized LDA #
out <- PenalizedLDA(x,y,xte=xte,type="ordered", lambda=cv.out$bestlambda,
K=cv.out$bestK, lambda2=.3)
```

```
print(out)
plot(out)
print(table(out$ypred[,out$K],yte))



# Now repeat penalized LDA computations but this time use
# type="standard"  - i.e. don't exploit spatial structure
# Perform cross-validation #
cv.out <-
PenalizedLDA.cv(x,y,lambdas=c(1e-4,1e-3,1e-2,.1,1,10))
print(cv.out)
plot(cv.out)
# Perform penalized LDA #
out <- PenalizedLDA(x,y,xte=xte,lambda=cv.out$bestlambda,K=cv.out$bestK)
print(out)
plot(out)
print(table(out$ypred[,out$K],yte))
```

`plot.penlda` *Plot PenalizedLDA outputs.*

Description

Make some plots!

Usage

```
## S3 method for class 'penlda'
plot(x,...)
```

Arguments

`x`	A "penlda" object; this is the output of the PenalizedLDA function.
`...`	...

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discrimint. To appear in JRSSB.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
xte <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
xte[y==1,1:10] <- xte[y==1,1:10] + 2
xte[y==2,11:20] <- xte[y==2,11:20] - 2
out <- PenalizedLDA(x,y,xte,lambda=.14,K=2)
print(out)
plot(out)
pred.out <- predict(out,xte=xte)
cat("Predictions obtained using PenalizedLDA function and using
predict.penlda function are the same.")
print(cor(pred.out$ypred,out$ypred))
```

---

plot.penldacv *Plot penalized LDA CV results.*

Description

Make a plot of the CV output.

Usage

```
## S3 method for class 'penldacv'
plot(x,...)
```

Arguments

x	A "penldacv" object.
...	...

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discrimint. To appear in JRSSB.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
xte <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
xte[y==1,1:10] <- xte[y==1,1:10] + 2
xte[y==2,11:20] <- xte[y==2,11:20] - 2
out <- PenalizedLDA(x,y,xte,lambda=.14,K=2)
print(out)
plot(out)
pred.out <- predict(out,xte=xte)
cat("Predictions obtained using PenalizedLDA function and using
predict.penlda function are the same.")
print(cor(pred.out$ypred,out$ypred))
```

---

predict.penlda *Make PenalizedLDA predictions on a new data set.*

Description

Given output from the PenalizedLDA function, make predictions on a new data set.

Usage

```
## S3 method for class 'penlda'
predict(object,xte,...)
```

Arguments

object	A "penlda" object; this is the output of the PenalizedLDA function.
xte	A data set on which predictions should be made.
...	...

Value

ypred	A matrix with nrow(xte) rows and K columns where K is the number of discriminant vectors in the "penlda" object passed in. The first column contains predictions obtained if only the 1st discriminant vector is used, the 2nd column contains predictions obtained if the first 2 discriminant vectors are used, and so on. If there is only 1 discriminant vector in the "penlda" object passed in, then just a vector is output.

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discrimint. To appear in JRSSB.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
xte <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
xte[y==1,1:10] <- xte[y==1,1:10] + 2
xte[y==2,11:20] <- xte[y==2,11:20] - 2
out <- PenalizedLDA(x,y,xte,lambda=.14,K=2)
print(out)
plot(out)
pred.out <- predict(out,xte=xte)
cat("Predictions obtained using PenalizedLDA function and using
predict.penlda function are the same.")
print(cor(pred.out$ypred,out$ypred))
```

---

print.penlda *Print PenalizedLDA output.*

Description

A nice way to display the results of PenalizedLDA.

Usage

```
## S3 method for class 'penlda'
print(x,...)
```

Arguments

x	A "penlda" object; this is the output of the PenalizedLDA function.
...	...

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discriminant. To appear in JRSSB.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
xte <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
xte[y==1,1:10] <- xte[y==1,1:10] + 2
xte[y==2,11:20] <- xte[y==2,11:20] - 2
out <- PenalizedLDA(x,y,xte,lambda=.14,K=2)
print(out)
plot(out)
pred.out <- predict(out,xte=xte)
cat("Predictions obtained using PenalizedLDA function and using
predict.penlda function are the same.")
print(cor(pred.out$ypred,out$ypred))
```

---

print.penldacv *Print PenalizedLDA CV results.*

Description

A nice print-out of the results obtained from running the CV function for PenalizedLDA.

Usage

```
## S3 method for class 'penldacv'
print(x,...)
```

Arguments

x	A "penldacv" object; this is the output of the PenalizedLDA CV function.
...	...

Author(s)

Daniela M. Witten

References

D Witten and R Tibshirani (2011) Penalized classification using Fisher's linear discrimint. To appear in JRSSB.

Examples

```
set.seed(1)
n <- 20
p <- 100
x <- matrix(rnorm(n*p), ncol=p)
xte <- matrix(rnorm(n*p), ncol=p)
y <- c(rep(1,5),rep(2,5),rep(3,10))
x[y==1,1:10] <- x[y==1,1:10] + 2
x[y==2,11:20] <- x[y==2,11:20] - 2
xte[y==1,1:10] <- xte[y==1,1:10] + 2
xte[y==2,11:20] <- xte[y==2,11:20] - 2
out <- PenalizedLDA(x,y,xte,lambda=.14,K=2)
print(out)
plot(out)
pred.out <- predict(out,xte=xte)
cat("Predictions obtained using PenalizedLDA function and using
predict.penlda function are the same.")
print(cor(pred.out$ypred,out$ypred))
```

Index