Package ‘DAP’

March 5, 2018

Type Package
Title Discriminant Analysis via Projections
Version 1.0
Date 2018-03-05
Author Tianying Wang and Irina Gaynanova
Maintainer Tianying Wang <tianying@stat.tamu.edu>
License GPL (>= 2)
Imports MASS, stats
LazyData TRUE
URL http://github.com/irinagain/DAP
BugReports http://github.com/irinagain/DAP/issues
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-03-05 20:05:33 UTC

R topics documented:
dap-package .......................................................... 2
apply_DAP ......................................................... 2
classify_DAP .................................................... 4
cv_DAP ........................................................... 5
solve_DAP_C .................................................... 6
solve_DAP_seq ................................................ 8
standardizeData .............................................. 9

Index 11
apply_DAP

Description

Applies Discriminant Analysis via Projections to perform binary classification on the test dataset based on the training data.

Usage

apply_DAP(xtrain, ytrain, xtest, ytest = NULL, lambda_seq = NULL, 
n_lambda = 50, maxmin_ratio = 0.1, nfolds = 5, eps = 1e-04, 
maxiter = 10000, myseed = 1001, prior = TRUE)

Arguments

- **xtrain**: A n x p training dataset; n observations on the rows and p features on the columns.
- **ytrain**: A n vector of training group labels, either 1 or 2.
- **xtest**: A m x p testing dataset; m observations on the rows and p features on the columns.
- **ytest**: An optional m vector of testing group labels, either 1 or 2. If supplied, the function returns misclassification error rate; if NULL, the function returns predicted labels for xtest. Default is NULL.
- **lambda_seq**: An optional sequence of tuning parameters lambda. Default is NULL, and the function generates its own sequence.
apply_DAP

n_lambda Number of lambda values, the default is 50.
maxmin_ratio Smallest value for lambda, as a fraction of maximal value for which all coefficients are zero. The default is 0.1.
nfolds Number of folds for cross-validation, the default is 5.
eps Convergence threshold for the block-coordinate decent algorithm based on the maximum element-wise change in $V$. The default is 1e-4.
maxiter Maximum number of iterations, the default is 10000.
myseed Optional specification of random seed for generating the folds, the default value is 1001.
prior A logical indicating whether to put larger weights to the groups of larger size; the default value is TRUE.

Details

If no feature is selected by DAP, the function will return error of 0.5 and no ypred, indicating that the classifier is no better than random guessing.

Value

A list of

- error Misclassification error rate (if ytest is provided).
- ypred Predicted labels on the test set (if ytest is NULL).
- features Number of selected features.
- feature_id Index of selected features.

Examples

```r
# This is an example for apply_DAP

# Generate data
n_train = 50
n_test = 50
p = 100
mu1 = rep(0, p)
mu2 = rep(3, p)
Sigma1 = diag(p)
Sigma2 = 0.5* diag(p)

# Build training data and test data
x1 = MASS::mvrnorm(n = n_train, mu = mu1, Sigma = Sigma1)
x2 = MASS::mvrnorm(n = n_train, mu = mu2, Sigma = Sigma2)
xtrain = rbind(x1, x2)
x1_test = MASS::mvrnorm(n = n_test, mu = mu1, Sigma = Sigma1)
x2_test = MASS::mvrnorm(n = n_test, mu = mu2, Sigma = Sigma2)
xtest = rbind(x1_test, x2_test)
ytrain = c(rep(1, n_train), rep(2, n_train))
ytest = c(rep(1, n_test), rep(2, n_test))
```
## apply_dap

Given `ytest`, the function will return a misclassification error rate.

```r
ClassificationError = apply_dap(xtrain, ytrain, xtest, ytest)
```

Without `ytest`, the function will return predictions.

```r
ypredict = apply_dap(xtrain, ytrain, xtest)
```

---

### classify_DAP

**Classification via DAP**

**Description**

Classify observations in the test set using the supplied matrix `V` and the training data.

**Usage**

```r
classify_DAP(xtrain, ytrain, xtest, V, prior = TRUE)
```

**Arguments**

- `xtrain`: A `n x p` training dataset; `n` observations on the rows and `p` features on the columns.
- `ytrain`: A `n` vector of training group labels, either 1 or 2.
- `xtest`: A `m x p` testing dataset; `m` observations on the rows and `p` features on the columns.
- `V`: A `p x 2` projection matrix.
- `prior`: A logical indicating whether to put larger weights to the groups of larger size; the default value is `TRUE`.

**Value**

Predicted class labels for the test data.

**Examples**

```r
## This is an example for classify_DAP

## Generate data
n_train = 50
c_test = 50
p = 100
mu1 = rep(0, p)
mu2 = rep(3, p)
Sigma1 = diag(p)
Sigma2 = 0.5* diag(p)
```
```r
## Build training data and test data
x1 = MASS::mvrnorm(n = n_train, mu = mu1, Sigma = Sigma1)
x2 = MASS::mvrnorm(n = n_train, mu = mu2, Sigma = Sigma2)
xtrain = rbind(x1, x2)
x1_test = MASS::mvrnorm(n = n_test, mu = mu1, Sigma = Sigma1)
x2_test = MASS::mvrnorm(n = n_test, mu = mu2, Sigma = Sigma2)
xtest = rbind(x1_test, x2_test)
ytrain = c(rep(1L, n_train), rep(2, n_train))

# Standardize the data
out_s = standardizeData(xtrain, ytrain, center = FALSE)

## Find V
out.proj = solve_DAP_C(X1 = out_s$X1, X2 = out_s$X2, lambda = 0.3)
V = cbind(diag(1/out_s$coef1)%*%out.proj$V[,1], diag(1/out_s$coef2)%*%out.proj$V[,2])

# Predict y using classify_DAP
ypred = classify_DAP(xtrain, ytrain, xtest, V = V)
```

**cv_DAP**

*Cross-validation for DAP*

**Description**

Chooses optimal tuning parameter lambda for DAP based on the k-fold cross-validation to minimize the misclassification error rate

**Usage**

```r
cv_DAP(X, Y, lambda_seq, nfolds = 5, eps = 1e-04, maxiter = 1000, myseed = 1001, prior = TRUE)
```

**Arguments**

- **X**: A n x p training dataset; n observations on the rows and p features on the columns.
- **Y**: A n vector of training group labels, either 1 or 2.
- **lambda_seq**: A sequence of tuning parameters to choose from.
- **nfolds**: Number of folds for cross-validation, the default is 5.
- **eps**: Convergence threshold for the block-coordinate decent algorithm based on the maximum element-wise change in V. The default is 1e-4.
- **maxiter**: Maximum number of iterations, the default is 10000.
- **myseed**: Optional specification of random seed for generating the folds, the default value is 1001.
- **prior**: A logical indicating whether to put larger weights to the groups of larger size; the default value is TRUE.
solve_DAP_C

Solves DAP optimization problem for a given lambda value

Description

Uses block-coordinate descent algorithm to solve DAP problem.

Usage

solve_DAP_C(X1, X2, lambda, Vinit = NULL, eps = 1e-04, maxiter = 10000)
solve_DAP_C

Arguments

- **X1** A n1 x p matrix of group 1 data (scaled).
- **X2** A n2 x p matrix of group 2 data (scaled).
- **lambda** A value of the tuning parameter lambda.
- **Vinit** Optional starting point, the default is NULL, and the algorithm starts with the matrix of zeros.
- **eps** Convergence threshold for the block-coordinate decent algorithm based on the maximum element-wise change in $V$. The default is 1e-4.
- **maxiter** Maximum number of iterations, the default is 10000.

Value

A list of

- **V** A p x 2 projection matrix to be used in DAP classification algorithm.
- **nfeature** Number of nonzero features.
- **iter** Number of iterations until convergence.

Warnings

Please use scaled X1 and X2 for this function, they can be obtained using `standardizeData` to do so.

Examples

```r
## This is an example for solve_DAP_C

## Generate data
n_train = 50
n_test = 50
p = 100
mu1 = rep(0, p)
mu2 = rep(3, p)
Sigma1 = diag(p)
Sigma2 = 0.5* diag(p)

## Build training data
x1 = MASS::mvrnorm(n = n_train, mu = mu1, Sigma = Sigma1)
x2 = MASS::mvrnorm(n = n_train, mu = mu2, Sigma = Sigma2)
xtrain = rbind(x1, x2)
ytrain = c(rep(1, n_train), rep(2, n_train))

## Standardize the data
out_s = standardizeData(xtrain, ytrain, center = FALSE)

## Apply solve_DAP_C
out = solve_DAP_C(X1 = out_s$X1, X2 = out_s$X2, lambda = 0.3)
```
solve_DAP_seq

Solves DAP optimization problem for a given sequence of lambda values

Description

Uses block-coordinate descent algorithm with warm initializations, starts with the maximal supplied lambda value.

Usage

solve_DAP_seq(X1, X2, lambda_seq, eps = 1e-04, maxiter = 10000, feature_max = nrow(X1) + nrow(X2))

Arguments

X1
A n1 x p matrix of group 1 data (scaled).

X2
A n2 x p matrix of group 2 data (scaled).

lambda_seq
A supplied sequence of tuning parameters.

eps
Convergence threshold for the block-coordinate decent algorithm based on the maximum element-wise change in $V$. The default is $1e^{-4}$.

maxiter
Maximum number of iterations, the default is 10000.

feature_max
An upper bound on the number of nonzero features in the solution; the default value is the total sample size. The algorithm trims the supplied lambda_seq to eliminate solutions that exceed feature_max.

Value

A list of

lambda_seq
A sequence of considered lambda values.

V1_mat
A p x m matrix with columns corresponding to the 1st projection vector V1 found at each lambda from lambda_seq.

V2_mat
A p x m matrix with columns corresponding to the 2nd projection vector V2 found at each lambda from lambda_seq.

nfeature_vec
A sequence of corresponding number of selected features for each value in lambda_seq.

Examples

## This is an example for solve_DAP_seq

## Generate data
n_train = 50
n_test = 50
p = 100
standardizeData

mu1 = rep(0, p)
mu2 = rep(3, p)
Sigma1 = diag(p)
Sigma2 = 0.5* diag(p)

## Build training data
x1 = MASS::mvrnorm(n = n_trainL, mu = mu1, Sigma = Sigma1)
x2 = MASS::mvrnorm(n = n_trainL, mu = mu2, Sigma = Sigma2)
xtrain = rbind(x1, x2)
ytrain = c(rep(1L, n_trainL), rep(2, n_trainL))

## Standardize the data
out_s = standardizedata(xtrain, ytrain, center = FALSE)

#### use solve_proj_seq
fit = solve_DAP_seq(X1 = out_s$X1, X2 = out_s$X2, lambda_seq = c(0.2, 0.3, 0.5, 0.7, 0.9))

---

**standardizeData**

Divides the features matrix into two standardized submatrices

**Description**

Given matrix X with corresponding class labels in Y, the function column-centers X, divides it into two submatrices corresponding to each class, and scales the columns of each submatrix to have euclidean norm equal to one.

**Usage**

standardizeData(X, Y, center = TRUE)

**Arguments**

- **X**
  A n x p training dataset; n observations on the rows and p features on the columns.

- **Y**
  A n vector of training group labels, either 1 or 2.

- **center**
  A logical indicating whether X should be centered, the default is TRUE.

**Value**

A list of

- **X1**
  A n1 x p standardized matrix with observations from group 1.

- **X2**
  A n2 x p standardized matrix with observations from group 2.

- **coef1**

- **coef2**
  Back-scaling coefficients for X2.

- **Xmean**
  Column means of the matrix X before centering.
Examples

# An example for the function standardizeData

## Generate data
n_train = 50
n_test = 50
p = 100
mu1 = rep(0, p)
mu2 = rep(3, p)
Sigma1 = diag(p)
Sigma2 = 0.5* diag(p)

## Build training data
x1 = MASS::mvrnorm(n = n_train, mu = mu1, Sigma = Sigma1)
x2 = MASS::mvrnorm(n = n_train, mu = mu2, Sigma = Sigma2)
xtrain = rbind(x1, x2)
ytrain = c(rep(1, n_train), rep(2, n_train))

## Standardize data
out_s = standardizeData(xtrain, ytrain, center = FALSE)
Index

*Topic package
  dap-package, 2

apply_DAP, 2

classify_DAP, 4
cv_DAP, 5
dap-package, 2

solve_DAP_C, 6
solve_DAP_seq, 8
standardizeData, 9