Package ‘mLDA’

December 12, 2018

Type Package

Title Multi-class linear discriminant analysis with ultrahigh-dimensional features

Version 1.0

Date 2018-12-11

Author Yanming Li

Maintainer Yanming Li <liyanmin@umich.edu>

Description The mLDA package implements the multi-class linear discriminant analysis method for classifications with ultrahigh-dimensional data. The method can select both marginally and jointly informative features that are informative for classifications.

License GPL (>= 2)

NeedsCompilation no

R topics documented:

`mLDA-package` *Multi-class linear discriminant analysis with ultrahigh-dimensional features*

Description

The mLDA package implements the multi-class linear discriminant analysis method for classifications with ultrahigh-dimensional data. The method can select both marginally and jointly informative features that are informative for classifications.

Details

The package mLDA conducts a variable selection and a multi-class classification for ultrahigh-dimensional features. The mLDA can select both the marginally informative features and the marginally uninformative but jointly informative features.

Author(s)

Yanming Li <liyanmin@umich.edu>

Maintainer:Yanming Li <liyanmin@umich.edu>

References

Li, Yanming and Hong, Hyokyoung and Li, Yi (2018) Multiclass Linear Discriminant Analysis with Ultrahigh-Dimensional Features. Under revision.

`mLDA.Kclass` *A function for feature selection and K-class classification.*

Description

The `mlda.Kclass` function selects informative features (both marginally and jointly informative features) from ultrahigh-dimensional feature space and use the selected features for binary classifications.

Usage

```
mLDA.Kclass(X, y, X.new, COV = NULL, COR = NULL, K = 3, tau = 200, alpha = 0.5,
 nu = 100, d = 10)
```

Arguments

`X`	An n x p matrix of features. Each row is for a subject and each column is for a feature.
`y`	A class index vector in length of n. Class labels should be coded in 1,...,K
`X.new`	An n.new x p matrix of test data. Each row is for a subject and each column is for a feature.
`COV`	A p x p covariance matrix of the features. By default is set to NULL and will estimated from the training data.
`COR`	A p x p correlation matrix of the features. By default is set to NULL and will estimated from the training data.
`K`	The total number of classes in the training data. By default is set to be 3.
`tau`	The thresholding parameter for the marginal informative features.
`alpha`	The thresolding parameter for the correlation matrix.

`nu` The thresholding parameter for the marginally uninformative but jointly informative features.

`d` An integer specify the number of most highly correlated features to each marginal informative feature to detect. That is, the max size of each connected components.

Details

The `mlda.Kclass` function selects informative features (both marginally and jointly informative features) that differentiate any pair of classes from ultrahigh-dimensional feature space and use the selected features for K-class classifications.

Value

`ISmatrix` An n.new x K matrix of ordered covariance adjusted absolute mean difference statistics, i.e., the inportant scores for a selected feature to differentiate a pair of classes. The columns are in order for class pairs 1<->2, 1<->3, ..., 1<->K, 2<->3, ..., 2<->K, ..., (K-1)<->K.

`screenset` A vector of indices for final selected features.

`Fisher.matrix`
An n.new x K vector of the Fisher discriminant statistics for the test data.

`PredClass` An n.new x 1 vector of predicted classes for the test data.

Author(s)

Yanming Li

References

Li, Yanming and Hong, Hyokyoung and Li, Yi (2018) Multiclass Linear Discriminant Analysis with Ultrahigh-Dimensional Features. Under revision.

Examples

```
tmp <- mLDA.Kclass(X_train, y_train, X.new=X_test, tau=20, alpha=0.05)
```

`mLDA.Kclass.cv` *A function for 5-fold cross validation to select the optimal tuning parameters for feature selection and K-class classification.*

Description

The `mLDA.Kclass.cv` function conducts 5-fold cross-validation to select the optimal tunning parameters to pass to the mLDA.Kclass function.

Usage

```
mLDA.Kclass.cv(X, y, COV = NULL, COR = NULL, K = 3, fold = 5,
tau.list = c(1, 10, 20, 100, 200), alpha = 0.5,
nu.list = c(20, 100, 200), d = 10)
```

Arguments

`X`	An n x p matrix of features. Each row is for a subject and each column is for a feature.
`y`	A class index vector in length of n. Class labels should be coded in 1,...,K
`COV`	A p x p covariance matrix of the features. By default is set to NULL and will estimated from the training data.
`COR`	A p x p correlation matrix of the features. By default is set to NULL and will estimated from the training data.
`K`	The total number of classes in the training data. By default is set to be 3.
`fold`	The total number of fold in the cross validation. By default is set to be 5.
`tau.list`	A vector of thresholding parameters for the marginal informative features.
`alpha`	The thresolding parameter for the correlation matrix. By default is set to be 0.5
`nu.list`	A vector of thresholding parameters for the marginally uninformative but jointly informative features.
`d`	An integer specify the number of most highly correlated features to each marginal informative feature to detect. That is, the max size of each connected components.

Details

The `mLDA.Kclass.cv` function conducts 5-fold cross-validation to select the optimal tunning parameters to pass to the mLDA.Kclass function.

Value

`mLDA.Kclass.cv` returns a matrix of classification errors corresponding to each combination of the tuning parameter values. Each row corresponds to a tau value and each column corresponds to a nu value.

Author(s)

Yanming Li

References

Li, Yanming and Hong, Hyokyoung and Li, Yi (2018) Multiclass Linear Discriminant Analysis with Ultrahigh-Dimensional Features. Under revision.

Examples

```
tmp <- mLDA.Kclass.cv(X_train, y_train, COV=NULL, COR=NULL, K=3, fold=5)
```

`mLDA.pair` *A function for feature selection and binary classification.*

Description

The `mLDA.pair` function selects informative features (both marginally and jointly informative features) from ultrahigh-dimensional feature space and use the selected features for binary classifications.

Usage

```
mLDA.pair(X, y, X.new, COV = NULL, COR = NULL, pair = c(1, 2), tau = 200,
 alpha = 0.5, nu = 100, d = 10)
```

Arguments

`X`	An n x p matrix of features. Each row is for a subject and each column is for a feature.
`y`	A class index vector in length of n. Class labels should be coded in 1,...,K
`X.new`	An n.new x p matrix of test data. Each row is for a subject and each column is for a feature.
`COV`	A p x p covariance matrix of the features. By default is set to NULL and will estimated from the training data.
`COR`	A p x p correlation matrix of the features. By default is set to NULL and will estimated from the training data.
`pair`	A pair of classes in 1,...,K to classify.
`tau`	The thresholding parameter for the marginal informative features.
`alpha`	The thresolding parameter for the correlation matrix.
`nu`	The thresholding parameter for the marginally uninformative but jointly informative features.
`d`	An integer specify the number of most highly correlated features to each marginal informative feature to detect. That is, the max size of each connected components.

Details

The `mlda.pair` function selects informative features (both marginally and jointly informative features) from ultrahigh-dimensional feature space and use the selected features for binary classifications.

Value

iffcond	An n.new vector of ordered covariance adjusted absolute mean difference statistics, which is also an if and only if condition for a feature to be informative.
screenset	A vector of indices for final selected features.
FisherDR	An n.new x 1 vector of the Fisher discriminant statistics for the test data.
PredClass	An n.new x 1 vector of predicted classes for the test data.

Author(s)

Yanming Li

References

Li, Yanming and Hong, Hyokyoung and Li, Yi (2018) Multiclass Linear Discriminant Analysis with Ultrahigh-Dimensional Features. Under revision.

Examples

```
######
library(MASS)
set.seed(12345)

n.train <- 100
n1 <- n.train
n2 <- n.train
n3 <- n.train

mu1 <- c(rep(0,5), rep(1.5,5), rep(0,5), rep(1.5,5), rep(0,5), rep(-1.5,5))
mu2 <- c(rep(0,4), 2.5, rep(-1.5,5), rep(0,4), 2.5, rep(-1.5,5), rep(0,5), rep(-1.5,5))
mu3 <- c(rep(0,5), rep(-1.5,5), rep(0,5), rep(-1.5,5), rep(0,4), 2.5, rep(1.5, 5))

sigma <- 1
Sig0 <- matrix(sigma, 5, 5)
rho <- 0.7
for(i in 2:5){
 for(j in 1:(i-1)){
   Sig0[i,j] <- sigma*rho^(abs(i-j))
   Sig0[j,i] <- Sig0[i,j]
 }
}

Sig <- matrix(rep(0, 900), 30, 30)
Sig[1:5, 1:5] <- Sig0
Sig[6:10, 6:10] <- Sig0
Sig[11:15, 11:15] <- Sig0
Sig[16:20, 16:20] <- Sig0
Sig[21:25, 21:25] <- Sig0
Sig[26:30, 26:30] <- Sig0
```

```
Xinfo1 <- mvrnorm(n1, mu=mu1, Sigma=Sig)
Xinfo2 <- mvrnorm(n2, mu=mu2, Sigma=Sig)
Xinfo3 <- mvrnorm(n3, mu=mu3, Sigma=Sig)

Xinfo <- rbind(Xinfo1, Xinfo2, Xinfo3)

p<- 970
X0 <- matrix(rnorm((n1+n2+n3)*p), (n1+n2+n3), p)
X_train <- cbind(Xinfo,X0)
y_train <- c(rep(1, n1), rep(2, n2), rep(3, n3))

#####
n.test <- 50

n1 <- n.test
n2 <- n.test
n3 <- n.test

mu1 <- c(rep(0,5), rep(1.5,5), rep(0,5), rep(1.5,5), rep(0,5), rep(-1.5,5))
mu2 <- c(rep(0,4), 2.5, rep(-1.5,5), rep(0,4), 2.5, rep(-1.5,5), rep(0,5), rep(-1.5,5))
mu3 <- c(rep(0,5), rep(-1.5,5), rep(0,5), rep(-1.5,5), rep(0,4), 2.5, rep(1.5, 5))

sigma <- 1
Sig0 <- matrix(sigma, 5, 5)
rho <- 0.7
for(i in 2:5){
 for(j in 1:(i-1)){
   Sig0[i,j] <- sigma*rho^(abs(i-j))
   Sig0[j,i] <- Sig0[i,j]
 }
}

Sig <- matrix(rep(0, 900), 30, 30)
Sig[1:5, 1:5] <- Sig0
Sig[6:10, 6:10] <- Sig0
Sig[11:15, 11:15] <- Sig0
Sig[16:20, 16:20] <- Sig0
Sig[21:25, 21:25] <- Sig0
Sig[26:30, 26:30] <- Sig0

Xinfo1 <- mvrnorm(n1, mu=mu1, Sigma=Sig)
Xinfo2 <- mvrnorm(n2, mu=mu2, Sigma=Sig)
Xinfo3 <- mvrnorm(n3, mu=mu3, Sigma=Sig)

Xinfo <- rbind(Xinfo1, Xinfo2, Xinfo3)

X0 <- matrix(rnorm((n1+n2+n3)*p), (n1+n2+n3), p)
X_test <- cbind(Xinfo,X0)
y_test <- c(rep(1,n1), rep(2, n2), rep(3, n3))

try <- mLDA.pair(X_train, y_train, X_test)
```