Package 'RidgeFusion'

February 19, 2015

Type Package

Title R Package for Ridge Fusion in Statistical Learning

Version 1.0-3

Date 2014-02-06

Author Bradley S. Price

Maintainer Bradley S. Price <bprice@bus.miami.edu>

Depends R (>= 3.0.0),mvtnorm, methods

Description This package implements ridge fusion methodology for inverse covariance matrix estimation for use in quadratic discriminant analysis. The package also contains function for model based clustering using ridge fusion for inverse matrix estimation, as well as tuning parameter selection functions. We have also implemented QDA using joint inverse covariance estimation.

ByteCompile TRUE

License MIT + file LICENSE

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-19 01:14:46

R topics documented:

FusedQDA *Quadratic Discriminant Analysis with Ridge Fused Inverse Covariance Estimation*

Description

Calculates the parameter estimates associated with quadratic discriminant analysis

Usage

```
FusedQDA(X,Lambda1,Lambda2,scaleC=FALSE)
```

Arguments

X	A list where each element contains the data of a different class
Lambda1	Ridge tuning parameter, must be greater than or equal to 0
Lambda2	Ridge Fusion tuning parameter, must be greater than or equal to 0
scaleC	If TRUE scale invariant method is used

Value

An object of class RidgeFusedQDA, basically a list including elements

Omega	a list where each element is the inverse covariance matrix estimate for the corresponding element of X
Means	A list of class means
Pi	Class Proportions
Lambda1	
Lambda2	
iter	Number of iterations until convergence

Author(s)

Brad Price

Examples

```
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]
```

```
}
}
Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)
Z=list(X1,Y)
A2=FusedQDA(Z,10,10,scale=TRUE)
names(A2)
```

RidgeFused *Ridged Fused Inverse Covariance Matrix Estimation*

Description

Calculates the ridge fusion precision estimator for multiple classes

Usage

```
RidgeFused(S,lambda1,lambda2,nc,tol=10^(-7), maxiter=1e3,warm.start=NULL,scale=FALSE)
```

Arguments

	A list is returned where the elements are:
	A list of length J that contains the sample covariance estimators of each class
$ambda1	Ridge tuning parameter, must be greater than or equal to 0
lambda2	Ridge Fusion tuning parameter, must be greater than or equal to 0
nc	A vector of length J that contains the sample size of each class
tol	Convergence tolerance for blockwise coordinate descent algorithm
maxiter	The number of iterations the algorithm will run if convergence tolerance is not met
warm.start	If NULL no warm start is used. If initialized with a list of positive definite inverse covariance matrix estimates of length J, will use them as initialization for the algorithm.
scale	If FALSE scale invariant method is used

Value

An object of class RidgeFusion, basically a list including elements

Omega	a list where each element is the inverse covariance matrix estimate for the corresponding element of S
Ridge	lambda1
FusedRidge	lambda2
iter	Number of iterations until convergence

Author(s)

Brad Price

Examples

```
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]

}
}
    Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)
## Creating a list to use as S
S=list(0,0)
S[[1]]=(99/100)*cov(X1)
S[[2]]=(99/100)*cov(Y)

## Creating the vector of sample sizes
nc2=c(100,100)

## Running RidgeFused scale invariant method for tuning parameters lambda1=1 ,lambda2=2
A=RidgeFused(S,1,2,nc2,scale=TRUE)
A
names(A)
```

RidgeFusedCV *Ridged Fused Validation Likelihood*

Description

Calculates the Valdiation Likelihood Score for candidate tuning parameters

Usage

```
RidgeFusedCV(X,lambda1,lambda2,Fold,tol=10^-6,warm.start=TRUE,scaleCV=FALSE,INF=FALSE)
```

Arguments

X	A list of length J that contains the data for each class
lambda1	A vector with all possible Ridge tuning parameters
lambda2	A vector with all possible Ridge Fusion tuning parameters

Fold	A list of length K, the number of folds, where each element is a list of length of the number of classes that contains the indices for the kth fold of the jth class. Fold[[1]][[1]] contains the indices of the first fold in class 1, Fold[[1]][[2]] contains the indices of the first fold of class 2
tol	Convergence tolerance for blockwise coordinate descent algorithm at each grid point
warm.start	A True/False variable, that indicates if warm.starts should be used
scaleCV	If TRUE scale invariant method is used
INF	If TRUE sets all inverse covariance matrices equal in the result

Value

An object of class RidgeFusionCV, basically a list including elements

Omega	a list where each element is the precision matrix estimate for the corresponding element of S
BestRidge	The ridge grid point that minimizes the validation likelihood score
BestFusedRidge	The fused ridge grid point that minimizes the validation likelihood score
CV	The matrix of validation likelihood scores and the grid points they match

Author(s)

Brad Price

Examples

```
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]

}
}
    Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)

## Creating a list of the data for each class
Z=list(X1,Y)

Samp=list(0,0)
Samp[[1]]=sample(1:100)
Samp[[2]]=sample(1:100)
```

```
## Creating Fold list
Fold1=list(0,0)
for(i in 1:5){
Fold1[[i]]=list(0,0)
for(j in 1:2){
Fold1[[i]][[j]]=Samp[[j]][((20*(i-1))+1):(i*20)]
}
}

## Calculating Validation likelihood scores for
##tuning parameter grid 10^(-1:1) for Ridge, and 10^(2:3) for Ridge Fusion
Tell=RidgeFusedCV(Z,10^(-1:1),10^(2:3),Fold1,scaleCV=TRUE)
Tell
names(Tell)
```

`RidgeFusedQDA-class` *Class* "`RidgeFusedQDA`"

Description

A class for implementing quadratic discriminant analysis with joint precision matrix estimation using ridge fusion

Usage

```
RidgeFusedQDA(...)
predict.RidgeFusedQDA(object,newdata,class=TRUE,...)
```

Arguments

`...`	Optional Arguments
`object`	An object of RidgeFusedQDA
`newdata`	data to be predicted
`class`	if TRUE then predicted classes are returned if false QDA scores are returned

Objects from the Class

Objects can be created by calls of the form `RidgeFusedQDA(...)`.

Slots

`Omega`: Object of class "`list`" ~~

`Means`: Object of class "`list`" ~~

`Pi`: Object of class "`vector`" ~~

`lambda1`: Object of class "`numeric`" ~~

`lambda2`: Object of class "`numeric`" ~~

Methods

predict signature(object = "RidgeFusedQDA"): ...

print signature(x = "RidgeFusedQDA"): ...

Author(s)

Brad Price

Examples

```
showClass("RidgeFusedQDA")
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]

}
}
Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)
Z=list(X1,Y)
A2=FusedQDA(Z,10,10,scale=TRUE)
names(A2)
Class=predict(A2,X1,class=TRUE)
Score=predict(A2,X1,class=FALSE)
```

RidgeFusion-class *Class* "RidgeFusion"

Description

A class for jointly estimating the precision matrix with ridge fusion

Usage

```
RidgeFusion(...)
```

Arguments

```
...
```

Objects from the Class

Objects can be created by calls of the form `RidgeFusion(...)`.

Slots

`Omega`: Object of class `"list"` ~~

`Ridge`: Object of class `"numeric"` ~~

`FusedRidge`: Object of class `"numeric"` ~~

`iter`: Object of class `"numeric"` ~~

Methods

print `signature(x = "RidgeFusion")`: ...

Author(s)

Brad Price

Examples

```
showClass("RidgeFusion")
```

RidgeFusionCV-class *Class* `"RidgeFusionCV"`

Description

A class for preforming the validation likelihood for joint precision matrix estimation using ridge fusion

Usage

```
RidgeFusionCV(...)
```

Arguments

`...`

Objects from the Class

Objects can be created by calls of the form `RidgeFusionCV(...)`.

Slots

`BestRidge`: Object of class `"numeric"` ~~

`BestFusedRidge`: Object of class `"numeric"` ~~

`CV`: Object of class `"matrix"` ~~

Methods

print `signature(x = "RidgeFusionCV")`: ...

Author(s)

Brad Price

Examples

```
showClass("RidgeFusionCV")
```

`SSRidgeFused` *Semis Supervised Ridge Fusion Model Based Clustering*

Description

Calculates parameters for model based clustering using ridge fusion estimation of precision matrix

Usage

```
SSRidgeFused(Z, Xu, lambda1, lambda2, Scale=FALSE, warm=NULL,tol=.001)
```

Arguments

`Z`	A list of length J that contains the labeled data for each class
`Xu`	The unlabeled data
`lambda1`	A vector with all possible Ridge tuning parameters
`lambda2`	A vector with all possible Ridge Fusion tuning parameters
`Scale`	If `TRUE` scale invariant method is used
`warm`	Default is `NULL`, if initialized with mixing distributions for each of the unlabeled data, will use in initialization of parameters
`tol`	tolerence for convergence criterion of the alphas

Value

An object of class `SSRidgeFusion`, basically a list including elements

`Omega`	a list where each element is the precision matrix estimate for the corresponding element of S
`Ridge`	lambda1
`FusedRidge`	lambda2
`iter`	The number of iterations until the EM algorithm converged
`Alpha`	Mixing coefficients for each of the unlabeled data points
`Means`	Class/Cluster Means
`Pi`	Probability Mass Function for the classes

Author(s)

Brad Price

Examples

```
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]

}
}
    Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)

## Creating a list of the data for each class
Z=list(X1,Y)

##Creating Unlabeled data set
Z1=rmvnorm(250,rep(2*log(p)/p,p),Sig1)
Z2=rmvnorm(250,,Sig2)
ZU=rbind(Z1,Z2)
## Running Semi-Supervised Ridge Fused Model based clustering
Hi=SSRidgeFused(Z,ZU,1,1,Scale=TRUE,warm=NULL)
## Showing example of a warm.start
Hi2=SSRidgeFused(Z,ZU,1,1,Scale=TRUE,warm=Hi$Alphas)
```

SSRidgeFusedCV — *Tuning Parameter Selection For Semi-Supervised Ridge Fusion Model Based Clustering via EM Validation Likelihood*

Description

Calculates validation scores for possible tuning parameters for Semi-Supervised Ridge Fusion Model Based Clustering

Usage

```
SSRidgeFusedCV(X,Xu,Lam1,Lam2,Fold,FoldU,scaleCV=FALSE,tolCV=0.01)
```

Arguments

X	A list of length J that contains the labeled data for each class
Xu	The unlabeled data
Lam1	A vector with all possible Ridge tuning parameters
Lam2	A vector with all possible Ridge Fusion tuning parameters
scaleCV	If TRUE scale invariant method is used
Fold	see Ridge Fused CV usage
FoldU	A list of length of the number of validation sets containing the indices of each set for the unlabeled data
tolCV	Covergence tolerance for each iteration of the cross validation via validation likelihood

Value

An object of class RidgeFusionCV, basically a list including elements

Omega	a list where each element is the inverse covariance matrix estimate for the corresponding element of S
BestRidge	The grid point of lambda1 that minimizes the validation score
BestFusedRidge	The grid point of lambda2 that minimizes the validation score
CV	Matrix containing the full grid of points that were input and the validation scores

Author(s)

Brad Price

Examples

```
## Not run:
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]

}
}
    Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)

## Creating a list of the data for each class
Z=list(X1,Y)
```

```
##Creating Unlabeled data set
Z1=rmvnorm(250,rep(2*log(p)/p,p),Sig1)
Z2=rmvnorm(250,,Sig2)
ZU=rbind(Z1,Z2)

Samp=list(0,0)
Samp[[1]]=sample(1:100)
Samp[[2]]=sample(1:100)

## Creating Fold list
Fold1=list(0,0)
for(i in 1:5){
Fold1[[i]]=list(0,0)
for(j in 1:2){
Fold1[[i]][[j]]=Samp[[j]][((20*(i-1))+1):(i*20)]
}
}


## Creating Validation sets for unlabeled data
SampU=sample(1:500)
FoldU1=list(0,0)
for(i in 1:5){
FoldU1[[i]]=SampU[((100*(i-1)+1)):(i*100)]
}


Hello=SSRidgeFusedCV(Z,ZU,10^(-2:-1),10^(-3:1),Fold1,FoldU1,scaleCV=FALSE)

## End(Not run)
```

SSRidgeFusion-class *Class* "SSRidgeFusion"

Description

A class to implement semi-supervised model based clustering with ridge fusion precision matrix estimation

Usage

```
SSRidgeFusion(...)
predict.SSRidgeFusion(object,newdata,class=TRUE,...)
```

Arguments

...	Optional Arguments
object	An object of RidgeFusedQDA

newdata	data to be predicted
class	if TRUE then predicted classes are returned if false QDA scores are returned

Objects from the Class

Objects can be created by calls of the form SSRidgeFusion(...).

Slots

Alphas: Object of class "matrix" ~~

Means: Object of class "list" ~~

Pi: Object of class "vector" ~~

Omega: Object of class "list" ~~

Ridge: Object of class "numeric" ~~

FusedRidge: Object of class "numeric" ~~

iter: Object of class "numeric" ~~

Extends

Class "RidgeFusion", directly.

Methods

predict signature(object = "SSRidgeFusion"): ...

print signature(x = "SSRidgeFusion"): ...

Author(s)

Brad Price

Examples

```
showClass("SSRidgeFusion")
## Creating a toy example with 5 variables
library(mvtnorm)
set.seed(526)
p=5
    Sig1=matrix(0,p,p)
for(j in 1:p){
for(i in j:p){
            Sig1[j,i]=.7^abs(i-j)
            Sig1[i,j]=Sig1[j,i]

}
}
    Sig2=diag(c(rep(2,p-5),rep(1,5)),p,p)
X1=rmvnorm(100,rep(2*log(p)/p,p),Sig1)
Y=rmvnorm(100,,Sig2)
```

```
## Creating a list of the data for each class
Z=list(X1,Y)

##Creating Unlabeled data set
Z1=rmvnorm(250,rep(2*log(p)/p,p),Sig1)
Z2=rmvnorm(250,,Sig2)
ZU=rbind(Z1,Z2)
## Running Semi-Supervised Ridge Fused Model based clustering
Hi=SSRidgeFused(Z,ZU,1,1,Scale=TRUE,warm=NULL)
Class=predict(Hi,Z1,class=TRUE)
Score=predict(Hi,Z1,class=FALSE)
```

Index