Package 'groupRemMap'

April 9, 2015

Version 0.1-0

Date 2015-03-25

Title Regularized Multivariate Regression for Identifying Master Predictors Using the GroupRemMap Penalty

Author Xianlong Wang <xwan2@fhcrc.org>, Li Qin, Hexin Zhang, Yuzheng Zhang, Li Hsu, Pei Wang <pei.wang@mssm.edu>

Maintainer Xianlong Wang <xwan2@fhcrc.org>

Description An implementation of the GroupRemMap penalty for fitting regularized multivariate response regression models under the high-dimension-low-sample-size setting. When the predictors naturally fall into groups, the GroupRemMap penalty encourages procedure to select groups of predictors, while control for the overall sparsity of the final model.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-04-09 01:07:16

R topics documented:

`group.remmap` *A function to fit the regularized multivariate regression model using the GroupRemMap penalty.*

Description

A function to fit regularized multivariate regression model using the GroupRemMap penalty.

Usage

```
group.remmap(X, Y,G,lam1,lam2,gamma=0.5, phi0=NULL, C.m=NULL)
```

Arguments

X	numeric matrix (n by p): columns correspond to predictor variables and rows correspond to samples. Missing values are not allowed.
G	numeric (typically integer) vector of length p: the memberships of the predictors.
Y	numeric matrix (n by q): columns correspond to response variables and rows correspond to samples. Missing values are not allowed.
lam1	numeric value: the l_1 norm penalty parameter.
lam2	numeric value: the bridge penalty parameter.
gamma	numeric value: the bridge degree
phi0	numeric matrix (p by q): an initial estimate of the coefficient matrix of the multivariate regression model; default(=NULL): univariate estimates are used as initial estimates.
C.m	numeric matrix (p by q): $C_m[i,j] = 0$ means the corresponding coefficient beta[i,j] is set to be zero in the model;$C_m[i,j] = 1$ means the corresponding beta[i,j] is included in the MAP penalty; $C_m[i,j] = 2$ means the corresponding beta[i,j] is not included in the MAP penalty; the default(=NULL) sets all $C_m[i,j]$ to be 1.

Details

group.remmap uses a computationally efficient approach for performing multivariate regression under the high-dimension-low-sample-size setting. The approach allows for correlation among the predictors within each group. (Wang et al., 2013).

Value

A list with two components

phi	the estimated coefficient matrix (p by q) of the regularized multivariate regression model.
rss.v	a vector of length q recording the RSS values of the q regressions.

Author(s)

X Wang, L Qin, H Zhang, Y Zhang, L Hsu, P Wang

References

X Wang, L Qin, H Zhang, Y Zhang, L Hsu, P Wang (2013) "A regularized multivariate regression approach for eQTL analysis" Statistics in Biosciences

Examples

```
##########################################
############# Generate an example data set
##########################################
n=20
#number of predictors
p=9
#number of response variables
q=4
set.seed(1)

###assume predictors falling into 3 groups, and the sizes of group 1
#group 2, and group 3 to be 3, 2, 4, respectively
G=c(1,1,1,2,2,3,3,3,3)

### generate X matrix
rho=0.5; Sig<-matrix(0,p,p)
for(i in 2:p){ for(j in 1: (i-1)){
    Sig[i,j]<-rho^(abs(i-j))
    Sig[j,i]<-Sig[i,j]
}}
diag(Sig)<-1
R<-chol(Sig)
X.m<-matrix(rnorm(n*p),n,p)
X.m<-X.m


#########################################################
#############Create coefficient matrix, B#############
#########################################################
B=matrix(0, p, q)

set.seed(100)
#number of predictors in each group
sz.group=c(3,2,4)
cumsum.group.sz=cumsum(sz.group)
#number of significant predictors in each group
num.sig.in.group=c(2,1,3)

###--Generate coefficients for group 1, allowing
###--every predictor in this group to predict the
###--the same subset of responses
#number of significant preditors, 2
num.sig.x=num.sig.in.group[1]
#numbers of responses predicted for each significant predictor
num.reg.y=c(3,1)
#indices of the significant predictors in group 1
ids.sig.x.tmp=sample(sz.group[1], num.sig.x)
#indices of all the responses that are predicted by predictors in group 1
ids.reg.y.tmp=sample(q, max(num.reg.y))
```

```
#indices of the responses that are predicted by each predictor in group 1
idxs.reg.y.tmp=sapply(num.reg.y, function (x) sample(ids.reg.y.tmp, x))

#Generate coefficient for each row of the coefficient matrix
for(idx.tmp in 1:num.sig.x)
{
    r=ids.sig.x.tmp[idx.tmp]
    c=idxs.reg.y.tmp[[idx.tmp]]
    B[r,c]=runif(num.reg.y[idx.tmp], min=1,max=4) #
}
#

###--Generate coefficients for group 2

num.sig.x=num.sig.in.group[2]
num.reg.y=2
ids.sig.x.tmp=sample(sz.group[2], num.sig.x)
ids.reg.y.tmp=sample(q, max(num.reg.y))
idxs.reg.y.tmp=list(as.numeric(sapply(num.reg.y, function (x) sample(ids.reg.y.tmp, x))))

#Generate coefficient for each row of the coefficient matrix
for(idx.tmp in 1:num.sig.x)
{
    r=cumsum.group.sz[1]+ids.sig.x.tmp[idx.tmp]
    c=idxs.reg.y.tmp[[idx.tmp]]

    B[r,c]=runif(num.reg.y[idx.tmp], min=1,max=4) #
}
#



###--Generate coefficients for group 3,

num.sig.x=num.sig.in.group[3]
num.reg.y=c(1,2,2)
ids.sig.x.tmp=sample(sz.group[3], num.sig.x)
ids.reg.y.tmp=sample(q, max(num.reg.y))
idxs.reg.y.tmp=sapply(num.reg.y, function (x) sample(ids.reg.y.tmp, x))

#Generate coefficient for each row of the coefficient matrix
for(idx.tmp in 1:num.sig.x)
{
    r=cumsum.group.sz[2]+ids.sig.x.tmp[idx.tmp]
    c=idxs.reg.y.tmp[[idx.tmp]]

    B[r, c]=runif(num.reg.y[idx.tmp], min=1,max=4) #
}
#



##############################################
```

```
### generate responses
#############################################
E.m<-matrix(rnorm(n*q),n,q)
Y.m<-X.m


#############################################
############ perform analysis
#############################################

#############################################
## 1. ## fit model for one pair of (lamL1, lamL2)
#############################################

try1=group.remmap(X.m, Y.m, G=G, lam1=100, lam2=50, gamma=0.5, phi0=NULL, C.m=NULL)


#############################################
## 2. ## Select tuning parameters with v-fold cross-validation;
##    cv based on unshrinked estimator (ols.cv) is recommended over cv
##    based on shrinked estimator (rss.cv);
###   ## the latter tends to select large models.
#############################################

lamL1.v=exp(seq(log(8), log(15), length=5))
lamL2.v=seq(10, 20, length=5)
try2=group.remmap.cv(X=X.m, Y=Y.m, G=G, lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)

############# use CV based on unshrinked estimator (ols.cv)
pick=which.min(as.vector(try2$ols.cv))
lamL1.pick=try2$l.index[1,pick]    ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try2$l.index[2,pick]
##fit the GroupRemMap model under the optimal (LamL1,LamL2).
result=group.remmap(X.m, Y.m, G=G, lam1=lamL1.pick, lam2=lamL2.pick, phi0=NULL, C.m=NULL)

## number of false positives at the individual predictor level
FP=sum(B[result$phi!=0]==0)
## number of false negatives at the individual predictor level
FN=sum(B[result$phi==0]!=0)
##CV (unshrinked) selected tuning parameters
#print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep=""))
print(paste("False Postive=", FP, "; False Negative=", FN, sep=""))

## number of errors at the group level
group.selection.matrix=aggregate(result$phi, list(G), function (x) any(x>=1e-6)+0)
true.group.selection.in.B=aggregate(B, list(G), function (x) any(x>=1e-3)+0)
## number of false positives at the group level
FP.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==1]==0)
## number of false negatives at the group level
FN.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==0]!=0)
print(paste("Group level FP=", FP.group, "; group level FN=", FN.group, sep=""))
```

```
############## use CV based on shrinked estimator (rss.cv); it tends to
##############select very large models: thus is not recommended in general
pick=which.min(as.vector(try2$rss.cv))
lamL1.pick=try2$l.index[1,pick]    ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try2$l.index[2,pick]
##fit the GroupRemMap model under the optimal (LamL1,LamL2).
result=group.remmap(X.m, Y.m, G=G, lam1=lamL1.pick, lam2=lamL2.pick, phi0=NULL, C.m=NULL)
## number of false positives
FP=sum(B[result$phi!=0]==0)
FN=sum(B[result$phi==0]!=0)  ## number of false negatives
##CV (shrinked) selected tuning parameters
#print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep=""))
print(paste("False Positive=", FP, "; False Negative=", FN, sep=""))



## number of false positive at the group level
FP.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==1]==0)
## number of false positive at the group level
FN.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==0]!=0)
print(paste("Group level FP=", FP.group, "; group level FN=", FN.group, sep=""))
```

`group.remmap.cv` *Fit GroupRemMap models for a series of tuning parameters and return the corresponding v-fold cross-validation scores.*

Description

Fit GroupRemMap models for a series of tuning parameters and return the corresponding v-fold cross-validation scores. Two types of cross-validation scores are computed: cv based on unshrinked estimator (ols.cv); and cv based on shrinked estimator (rss.cv); ols.cv is recommended. rss.cv tends to select very large models and thus is not recommended in general (especially for very sparse models).

Usage

```
group.remmap.cv(X,Y,G,lam1.v, lam2.v,gamma=0.5, C.m=NULL,fold=10, seed=1)
```

Arguments

`X`	numeric matrix (n by p): columns correspond to predictor variables and rows correspond to samples. Missing values are not allowed.
`G`	numeric (typically integer) vector of length p: the memberships of the predictors.
`Y`	numeric matrix (n by q): columns correspond to response variables and rows correspond to samples. Missing values are not allowed.

lam1.v	numeric value: the l_1 norm penalty parameter.
lam2.v	numeric value: the bridge penalty parameter.
gamma	numeric value: the bridge degree
C.m	numeric matrix (p by q): $C_m[i,j] = 0$ means the corresponding coefficient beta[i,j] is set to be zero in the model;$C_m[i,j] = 1$ means the corresponding beta[i,j] is included in the MAP penalty; $C_m[i,j] = 2$ means the corresponding beta[i,j] is not included in the MAP penalty; the default(=NULL) sets all $C_m[i,j]$ to be 1.
fold	numeric value: the number of folds in cross validation.
seed	numeric value: set the seed for creating the subsamples for cross valiation.

Details

group.remmap.cv is used to perform two-dimensional grid search of the tuning parameters (lamL1.v, lamL2.v) based on v-fold cross-validation scores. (Wang et.al., 2013).

Value

A list with four components

ols.cv	a numeric matrix recording the cross validation scores based on unshrinked OLS estimators for each pair of (lamL1, lamL2).
rss.cv	a numeric matrix recording the cross validation scores based on shrinked GroupRemMap estimators for each pair of (lamL1, lamL2).
phi.cv	a list recording the fitted GroupRemMap coefficients on cross validation training subsets. Each component corresponds to one cv fold and it is again a list with each component corresponding to the estimated GroupRemMap coefficients for one pair of (lamL1, lamL2) on that training subset.
l.index	numeric matrix with two rows: each column is a pair of (lamL1, lamL2) and the kth column corresponds to the kth cv score in as.vector(ols.cv) and as.vector(rss.cv).

Author(s)

X Wang, L Qin, H Zhang, Y Zhang, L Hsu, P Wang

References

X Wang, L Qin, H Zhang, Y Zhang, L Hsu, P Wang (2013) "A regularized multivariate regression approach for eQTL analysis" Statistics in Biosciences

Examples

```
##########################################
############# Generate an example data set
##########################################
n=20
```

```
#number of predictors
p=9
#number of response variables
q=4
set.seed(1)

###assume predictors falling into 3 groups, and the sizes of group 1
#group 2, and group 3 to be 3, 2, 4, respectively
G=c(1,1,1,2,2,3,3,3,3)

### generate X matrix
rho=0.5; Sig<-matrix(0,p,p)
for(i in 2:p){ for(j in 1: (i-1)){
    Sig[i,j]<-rho^(abs(i-j))
    Sig[j,i]<-Sig[i,j]
}}
diag(Sig)<-1
R<-chol(Sig)
X.m<-matrix(rnorm(n*p),n,p)
X.m<-X.m


#########################################################
#############Create coefficient matrix, B#############
#########################################################
B=matrix(0, p, q)

set.seed(100)
#number of predictors in each group
sz.group=c(3,2,4)
cumsum.group.sz=cumsum(sz.group)
#number of significant predictors in each group
num.sig.in.group=c(2,1,3)

###--Generate coefficients for group 1, allowing
###--every predictor in this group to predict the
###--the same subset of responses
#number of significant preditors, 2
num.sig.x=num.sig.in.group[1]
#numbers of responses predicted for each significant predictor
num.reg.y=c(3,1)
#indices of the significant predictors in group 1
ids.sig.x.tmp=sample(sz.group[1], num.sig.x)
#indices of all the responses that are predicted by predictors in group 1
ids.reg.y.tmp=sample(q, max(num.reg.y))
#indices of the responses that are predicted by each predictor in group 1
idxs.reg.y.tmp=sapply(num.reg.y, function (x) sample(ids.reg.y.tmp, x))

#Generate coefficient for each row of the coefficient matrix
for(idx.tmp in 1:num.sig.x)
{
    r=ids.sig.x.tmp[idx.tmp]
    c=idxs.reg.y.tmp[[idx.tmp]]
```

```
    B[r,c]=runif(num.reg.y[idx.tmp], min=1,max=4) #
}
#

###--Generate coefficients for group 2

num.sig.x=num.sig.in.group[2]
num.reg.y=2
ids.sig.x.tmp=sample(sz.group[2], num.sig.x)
ids.reg.y.tmp=sample(q, max(num.reg.y))
idxs.reg.y.tmp=list(as.numeric(sapply(num.reg.y, function (x) sample(ids.reg.y.tmp, x))))

#Generate coefficient for each row of the coefficient matrix
for(idx.tmp in 1:num.sig.x)
{
    r=cumsum.group.sz[1]+ids.sig.x.tmp[idx.tmp]
    c=idxs.reg.y.tmp[[idx.tmp]]

    B[r,c]=runif(num.reg.y[idx.tmp], min=1,max=4) #
}
#



###--Generate coefficients for group 3,

num.sig.x=num.sig.in.group[3]
num.reg.y=c(1,2,2)
ids.sig.x.tmp=sample(sz.group[3], num.sig.x)
ids.reg.y.tmp=sample(q, max(num.reg.y))
idxs.reg.y.tmp=sapply(num.reg.y, function (x) sample(ids.reg.y.tmp, x))

#Generate coefficient for each row of the coefficient matrix
for(idx.tmp in 1:num.sig.x)
{
    r=cumsum.group.sz[2]+ids.sig.x.tmp[idx.tmp]
    c=idxs.reg.y.tmp[[idx.tmp]]

    B[r, c]=runif(num.reg.y[idx.tmp], min=1,max=4) #
}
#



##############################################
### generate responses
##############################################
E.m<-matrix(rnorm(n*q),n,q)
Y.m<-X.m


##############################################
############ perform analysis
```

```
##############################################

##############################################
## 1. ## fit model for one pair of (lamL1, lamL2)
##############################################

try1=group.remmap(X.m, Y.m, G=G, lam1=100, lam2=50, gamma=0.5, phi0=NULL, C.m=NULL)


##############################################
## 2. ## Select tuning parameters with v-fold cross-validation;
##    cv based on unshrinked estimator (ols.cv) is recommended over cv
##    based on shrinked estimator (rss.cv);
###   ## the latter tends to select large models.
##############################################

lamL1.v=exp(seq(log(8), log(15), length=5))
lamL2.v=seq(10, 20, length=5)
try2=group.remmap.cv(X=X.m, Y=Y.m, G=G, lamL1.v, lamL2.v, C.m=NULL, fold=10, seed=1)

############# use CV based on unshrinked estimator (ols.cv)
pick=which.min(as.vector(try2$ols.cv))
lamL1.pick=try2$l.index[1,pick]    ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try2$l.index[2,pick]
##fit the GroupRemMap model under the optimal (LamL1,LamL2).
result=group.remmap(X.m, Y.m, G=G, lam1=lamL1.pick, lam2=lamL2.pick, phi0=NULL, C.m=NULL)

## number of false positives at the individual predictor level
FP=sum(B[result$phi!=0]==0)
## number of false negatives at the individual predictor level
FN=sum(B[result$phi==0]!=0)
##CV (unshrinked) selected tuning parameters
#print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep=""))
print(paste("False Postive=", FP, "; False Negative=", FN, sep=""))

## number of errors at the group level
group.selection.matrix=aggregate(result$phi, list(G), function (x) any(x>=1e-6)+0)
true.group.selection.in.B=aggregate(B, list(G), function (x) any(x>=1e-3)+0)
## number of false positives at the group level
FP.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==1]==0)
## number of false negatives at the group level
FN.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==0]!=0)
print(paste("Group level FP=", FP.group, "; group level FN=", FN.group, sep=""))



############# use CV based on shrinked estimator (rss.cv); it tends to
#############select very large models: thus is not recommended in general
pick=which.min(as.vector(try2$rss.cv))
lamL1.pick=try2$l.index[1,pick]    ##find the optimal (LamL1,LamL2) based on the cv score
lamL2.pick=try2$l.index[2,pick]
##fit the GroupRemMap model under the optimal (LamL1,LamL2).
result=group.remmap(X.m, Y.m, G=G, lam1=lamL1.pick, lam2=lamL2.pick, phi0=NULL, C.m=NULL)
```

```
## number of false positives
FP=sum(B[result$phi!=0]==0)
FN=sum(B[result$phi==0]!=0)  ## number of false negatives
##CV (shrinked) selected tuning parameters
#print(paste("lamL1=", round(lamL1.pick,3), "; lamL2=", round(lamL2.pick,3), sep=""))
print(paste("False Positive=", FP, "; False Negative=", FN, sep=""))



## number of false positive at the group level
FP.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==1]==0)
## number of false positive at the group level
FN.group=sum(true.group.selection.in.B[,-1][group.selection.matrix[,-1]==0]!=0)
print(paste("Group level FP=", FP.group, "; group level FN=", FN.group, sep=""))
```

Index