Package 'bootpenal'

February 22, 2016

Title Bootstrap Penalization for variable selection with big data

Version 1.0

Author Kuangnan Fang [aut,cre], Shuangge Ma[aut,cre]

Maintainer Kuangnan Fang <xmufkn@xmu.edu.cn>

Description

Bootstrap penalization for big data with large p (number of covariates) and/or large n (sample size).It fits block random lasso and block random adaptive lasso for data with large p. It fits split lasso and split adaptive lasso for data with large n. It fits split block random lasso and split block random adaptive lasso for data with large p and n.It is strongly recommended to conduct functions on High performance computer using parallel computing.

Depends R (>= 3.0.3)

Imports glmnet,ClustOfVar,clValid,msgps,MASS

Suggests parallel

License GPL (>= 2)

LazyData true

URL https://www.r-project.org

R topics documented:

`bootpenal-package` *Bootstrap penalization for variable selection with big data*

Description

With the cost of data collection decreasing fast, data with both large p (number of covariates) and/or large n (sample size) are now commonly encountered. For many problems, penalization is needed for regularized estimation and variable selection. Straightforward application of penalization to large datasets may demand a !0big computer!1 with high computational power and lead to high computational cost. To tackle this problem, Fang and Ma propose using bootstrap penalization, which dissects a big penalized estimation into a set of small ones, each of which only demands a !0small computer!1, and hence is computationally more feasible. The proposed methodological development is achieved via multiple steps. First for data with large p and small to moderate n, a block bootstrap penalization approach is developed. Covariates are first clustered to generate (relatively) homogeneous blocks. Then two sequential steps are carried out, where in each step and for each bootstrap sample, we select blocks of covariates and run penalization.

This package conduct variable selection for large datasets using bootstrap penalization. It fits block random lasso and block random adaptive lasso for data with large p but small to moderate n. It fits split lasso and split adaptive lasso for data with large n but small to moderate p. It fits split block random lasso and split block random adaptive lasso for data with large p and n. It is strongly recommended to conduct functions on High performance computer using parallel computing.

Details

Package:	bootpenal
Type:	Package
Version:	1.0
Date:	2016-02-10
License:	What license is it under?

It is very simple to use. Accepts `x,y` data for regression models, and produces the regularization path over a grid of values for the tuning parameter `tun`.It is strongly recommended to conduct functions on High performance computer using parallel computing. Only 6 functions:
`generate`
`Brlasso`
`Bradap`
`split_lasso`
`split_Brlasso`
`FDNR`

Author(s)

Kuangnan Fang, Shuangge Ma
Maintainer: Kuangnan Fang <xmufkn@xmu.edu.cn>

References

Fang, K. Ma, S. (2016) *Analyzing large datasets with bootstrap penalization*
Dunn J.C. (1974) *Well separated clusters and fuzzy partitions,Journal on Cybernetics, Vol. 4(1),*

95-104
Wang, S., Nan, B., Rosset, S., Zhu, J. (2010) *Random Lasso, The Annals of Applied Statistics, Vol. 5(1),468-485*

Examples

```
##independent covariates
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=split_Brlasso(X=x,y,B=30,nc=3,sb=3)
print(fit1)
fit1$coef.min # extract coefficients at a single value of lambda
fit2=split_Brlasso(X=x,y,sb=2,nc=3,B=30,q1=3,q2=3,method="Bradap")
fit2
fit3=Brlasso(X=x,y,nc=3,B=30,q1=3,q2=3,tun=seq(0.3,0.5,by=0.1))
fit3
## Auto correlation structure covariates
x<-generate(n=100,p=20,r=0.95,pair=FALSE)
betatrue<-c(rep(0,10),rep(1,10))
y<-x%*%betatrue +rnorm(100)
fit4=Brlasso(X=x,y,nc=3,B=30,q1=3,q2=3,tun=0.5)
##return FDR and FNR results
FDNR(beta=fit4$coef.min,betatrue)
```

`Bradap` *Bootstrap block adaptive random lasso for data with large p*

Description

Bootstrap block adaptive random lasso for data with large p

Usage

```
Bradap(X,y,nc=5,B=100,q1=4,q2=4,tun=0.5)
```

Arguments

`X`	input matrix, of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class `"sparseMatrix"` as in package `Matrix`
`y`	response variable, should be quantitative; not yet for qualitative response
`nc`	The maximum number of covariate clusters- default is 5. if number of variables `nvar<50`, conduct covariate hirachical clustering, if `nvar>=50`, conduct covariate kmeans clustering.
`B`	The number of bootstrap samples- default is 100. Draw `B` bootstrap samples each with sample size `nrows` by sampling with replacement from the original data
`q1`	The number of candidate blocks selected randomly from the `K` blocks in step 2
`q2`	The number of candidate blocks selected randomly from the `K` blocks in step 3
`tun`	The tunning parameter for stability-based selection- default is 0.5

Details

With data with large p, the key is to deal with p. The proposed method is realized in three steps.

Step 1: Cluster the p covariates into K non-overlapping blocks.
Multiple clustering techniques are applicable. As to be discussed in detail below, the goal of clustering is to generate covariate blocks with weak or even no correlation. Thus the clustering should be based on correlation. When the number of covariates less than 50, we take the hirachical clustering as the default. When the the number of covariates larger than or equal to 50, we take K-means clustering as the default. When $K = p$, the proposed method simplifies to the random Lasso(Wang et al., 2010).

Step 2: Generate an importance measure for each block.
Draw B bootstrap samples each with sample size n by sampling with replacement from the original data. For $b_2 = 1$ to B:
Select q_1 candidate blocks at random from the K blocks.
Apply adaptive Lasso to the bootstrap sample to obtain the estimated coefficients.
Compute the importance measure of block I_k
This step generates the importance measures for the K blocks. Each bootstrap dataset contains the same number of subjects as the original dataset but a smaller number of covariates. For a single bootstrap dataset, the computational cost can be considerably lower than that of the original dataset if $k_1 << K$. Although the bootstrap needs to be conducted B_2 times, as it can be executed in a highly parallel manner, it is still possible that the resulted computer time can be significantly lowered. With penalization, the relative importance of a covariate is measured by the magnitude of its estimate. We average over multiple bootstrap samples to generate the overall importance measure.

Step 3: Generate the final selection and estimation results. Draw B bootstrap samples each with sample size n by sampling with replacement from the original data.
For $b_3 = 1$ to B:
Select q_2 candidate blocks from the K blocks with selection probabilities proportional to the importance measure I_k's obtained in Step 2.
Apply adaptive Lasso to the bootstrap sample to obtain the estimated coefficients.
Conduct stability-based selection.That is, set $\hat{\beta}_j = 0$ if

$$B_3^{-1} \sum_{b_3=1}^{B_3} I(\hat{\beta}_j^{(b_3)} \neq 0) < \pi_1, j = 1, 2, \cdots, p$$

If $\hat{\beta}_j \neq 0$ from the previous step, then calculate the final estimate as

$$\hat{\beta}_j = B_3^{-1} \sum_{b_3=1}^{B_3} \hat{\beta}_j^{(b_3)}, j = 1, 2, \cdots, p$$

. This step shares a similar strategy as Step 2. One major difference is that the blocks are not bootstrapped at random. As we are more interested in the important blocks, they are assigned higher probabilities of being selected.

For `nc` this is maximum number of clusters of variables. The range of number of cluster condcucted in `2:nc`. The best number of cluster `K` is chosen as minimizing the Dunn index (Dunn, 1974). For `tun` this is the vector of tunnig parameter π to conduct stability-based selection. It is estimated using a BIC criterion. We set a coefficient estimate equal to zero if the estimates in more than π percentage of all bootstrap samples are zero. Here π can be viewed as a tuning parameter and estimated using a BIC criterion as

$$\hat{\pi}^{BIC} = argmin_{\pi}(nlog(n^{-1}\|y - \hat{y}\|^2) + log(n)|\hat{A}(\pi)|)$$

,
where $|\hat{A}(\pi)|$ is the number of nonzero estimates with cutoff π.

Value

`coef`	a `nvars x length(tun)` matrix of coefficients
`coef.min`	the coefficients of the model that has the min value of BIC
`nclust`	the number of clusters(blocks) of covariates chosen by Dunn index
`clustf`	a vector of integers indicating the cluster to which each variable is allocated
`IM`	a vector of important measure of each variable
`BIM`	a vector of important measure of each blocks

Author(s)

Kuangnan Fang, Shuangge Ma
Maintainer: Kuangnan Fang <xmufkn@xmu.edu.cn>

References

Dunn J.C. (1974) *Well separated clusters and fuzzy partitions,Journal on Cybernetics, Vol. 4(1), 95-104*

Wang, S., Nan, B., Rosset, S., Zhu, J. (2010) *Random Lasso, The Annals of Applied Statistics, Vol. 5(1),468-485*

See Also

`Brlasso`, `summary`

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=Bradap(X=x,y,nc=10,B=100,q1=4,q2=4,tun=c(0.4,0.5))
print(fit1)
fit1$coef.min # extract coefficients at a single value of lambda
```

`Brlasso` *Bootstrap block random lasso for big data with large p but small to moderate n*

Description

Bootstrap block random lasso for big data with large p but small to moderate n

Usage

```
Brlasso(X,y,nc=5,B=100,q1=4,q2=4,tun=0.5)
```

Arguments

X	input matrix, of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class "sparseMatrix" as in package Matrix
y	response variable, should be quantitative now; not yet for qualitative response
nc	The maximum number of covariate clusters- default is 5. if number of variables nvar<50, conduct covariate hirachical clustering, if nvar>=50, conduct covariate kmeans clustering.
B	The number of bootstrap samples- default is 100. Draw B bootstrap samples each with sample size nrows by sampling with replacement from the original data
q1	The number of candidate blocks selected randomly from the K blocks in step 2
q2	The number of candidate blocks selected randomly from the K blocks in step 3
tun	The tunning parameter for stability-based selection- default is 0.5

Details

With data with large p, the key is to deal with p. The proposed method is realized in three steps.

Step 1: Cluster the p covariates into K non-overlapping blocks.
Multiple clustering techniques are applicable. As to be discussed in detail below, the goal of clustering is to generate covariate blocks with weak or even no correlation. Thus the clustering should be based on correlation. When the number of covariates less than 50, we take the hirachical clustering as the default. When the the number of covariates larger than or equal to 50, we take K-means clustering as the default. When $K = p$, the proposed method simplifies to the random Lasso(Wang et al., 2010).

Step 2: Generate an importance measure for each block.
Draw B bootstrap samples each with sample size n by sampling with replacement from the original data. For $b_2 = 1$ to B:
Select q_1 candidate blocks at random from the K blocks.
Apply Lasso to the bootstrap sample to obtain the estimated coefficients.
Compute the importance measure of block I_k
This step generates the importance measures for the K blocks. Each bootstrap dataset contains the same number of subjects as the original dataset but a smaller number of covariates. For a single bootstrap dataset, the computational cost can be considerably lower than that of the original dataset if $k_1 << K$. Although the bootstrap needs to be conducted B_2 times, as it can be executed in a highly parallel manner, it is still possible that the resulted computer time can be significantly lowered. With penalization, the relative importance of a covariate is measured by the magnitude of its estimate. We average over multiple bootstrap samples to generate the overall importance measure.

Step 3: Generate the final selection and estimation results. Draw B bootstrap samples each with sample size n by sampling with replacement from the original data.
For $b_3 = 1$ to B:
Select q_2 candidate blocks from the K blocks with selection probabilities proportional to the importance measure I_k's obtained in Step 2.
Apply Lasso to the bootstrap sample to obtain the estimated coefficients.
Conduct stability-based selection.That is, set $\hat{\beta}_j = 0$ if

$$B_3^{-1} \sum_{b_3=1}^{B_3} I(\hat{\beta}_j^{(b_3)} \neq 0) < \pi_1, j = 1, 2, \cdots, p$$

If $\hat{\beta}_j \neq 0$ from the previous step, then calculate the final estimate as

$$\hat{\beta}_j = B_3^{-1} \sum_{b_3=1}^{B_3} \hat{\beta}_j^{(b_3)}, j = 1, 2, \cdots, p$$

. This step shares a similar strategy as Step 2. One major difference is that the blocks are not bootstrapped at random. As we are more interested in the important blocks, they are assigned higher probabilities of being selected.

For `nc` this is maximum number of clusters of variables. The range of number of cluster condcucted in `2:nc`. The best number of cluster `K` is chosen as minimizing the Dunn index (Dunn, 1974). For `tun` this is the vector of tunnig parameter π to conduct stability-based selection. It is estimated using a BIC criterion. We set a coefficient estimate equal to zero if the estimates in more than π percentage of all bootstrap samples are zero. Here π can be viewed as a tuning parameter and estimated using a BIC criterion as

$$\hat{\pi}^{BIC} = argmin_{\pi}(nlog(n^{-1}\|y - \hat{y}\|^2) + log(n)|\hat{A}(\pi)|)$$

,
where $|\hat{A}(\pi)|$ is the number of nonzero estimates with cutoff π.

Value

`coef`	a `nvars x length(tun)` matrix of coefficients
`coef.min`	the coefficients of the model that has the min value of BIC
`nclust`	the number of clusters(blocks) of covariates chosen by Dunn index
`clustf`	a vector of integers indicating the cluster to which each variable is allocated
`IM`	a vector of important measure of each variable
`BIM`	a vector of important measure of each blocks

Author(s)

Kuangnan Fang, Shuangge Ma
Maintainer: Kuangnan Fang <xmufkn@xmu.edu.cn>

References

Dunn J.C. (1974) *Well separated clusters and fuzzy partitions,Journal on Cybernetics, Vol. 4(1), 95-104*

Wang, S., Nan, B., Rosset, S., Zhu, J. (2010) *Random Lasso, The Annals of Applied Statistics, Vol. 5(1),468-485*

See Also

`Bradap`, `summary`

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=Brlasso(X=x,y,nc=3,B=30,q1=3,q2=3,tun=c(0.4,0.5))
print(fit1)
fit1$coef.min # extract coefficients at a single value of lambda
```

FDNR *Compute FDR and FNR of results for simulation*

Description

Compute FDR and FNR of results for simulation

Usage

```
FDNR(beta,betatrue)
```

Arguments

beta	estimated coefficients
betatrue	true value of coefficients

Details

False discovery rate (FDR) is computed by `FDR=fp/(tp+fp)`, where `tp` is the number of variables with nonzero true coefficients, `fn` is the number of variables with zero true coefficients but nonzero estimated coefficients, `fp` is the number of variables with nonzero true coefficients but zero estimated coefficients. False negative rate (FNR) is computed by `FNR=fn/(tp+fn)`.

Value

FDR	false discovery rate
FNR	false negative rate

Author(s)

Kuangnan Fang, Shuangge Ma
Maintainer: Kuangnan Fang <xmufkn@xmu.edu.cn>

Examples

```
x=matrix(rnorm(100*20),100,20)
betatrue<-c(rep(0,10),rep(1,10))
y=x%*%betatrue +rnorm(100)
fit1=Brlasso(x,y,nc=2,B=20,q1=2,q2=2)
FDNR(beta=fit1$coef.min,betatrue=betatrue)
```

`generate` *generate Auto correlation or pairwise correlation structure random varibales*

Description

generate Auto correlation or pairwise correlation structure random varibales.

Usage

```
generate(n=100,p=4,r=0.95,pair=FALSE)
```

Arguments

`n`	number of observation
`p`	number of covariates
`r`	The number of correlation `r` value-default is 0.95
`pair`	logical; if TRUE, generate pairwise correlation structure random variables.if FALSE(default),generate autocorrelation structure variables.Default is FALSE

Details

For auto correlation structure, the correlation coefficient between x_j and x_k is $r(j,k) = r^{(|j-k|)}$. For pairwise correlation structure, the pairwise correlation coefficient is r, that is the correlation coefficient between x_j and x_k is $r(j,k) = r$.

See Also

`mvrnorm`

Examples

```
# generate auto correlation structure
generate(n=100,p=3,r=0.95,pair=FALSE)
generate(n=100,p=4,r=0.5,pair=FALSE)

#generate paiwise correlation structure
generate(n=100,p=4,r=0.9,pair=TRUE)
```

`prostate` *Factors associated with prostate specific antigen*

Description

Data from a study by by Stamey et al. (1989) to examine the association between prostate specific antigen (PSA) and several clinical measures that are potentially associated with PSA in men who were about to receive a radical prostatectomy. The variables are as follows:

- lcavol: Log cancer volume
- lweight: Log prostate weight
- age: The man's age
- lbph: Log of the amount of benign hyperplasia
- svi: Seminal vesicle invasion; 1=Yes, 0=No
- lcp: Log of capsular penetration
- gleason: Gleason score
- pgg45: Percent of Gleason scores 4 or 5
- lpsa: Log PSA

Usage

```
data(prostate)
```

Format

A data frame with 97 observations on 9 variables

Source

http://statweb.stanford.edu/~tibs/ElemStatLearn/

References

- Hastie T, Tibshirani R, and Friedman J. (2001). *The Elements of Statistical Learning*. Springer.
- Stamey T, et al. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients. *Journal of Urology*, **16**: 1076-1083.

`split_Brlasso` *Bootstrap penalization for data with large n and large p*

Description

Bootstrap penalization for data with large n and and large p

Usage

```
split_Brlasso(X,y,sb=4,nc=5,B=100,q1=3,q2=3,tun_s=0.5,tun=0.5,method=c("Brlasso","Bradap"))
```

Arguments

X	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable,should be quantitative now; not yet for qualitative response
sb	the number of sample blocks
nc	the maximum number of cluster of variables
B	The number of bootstrap samples- default is 100. Draw B bootstrap samples each with sample size nrows by sampling with replacement from the original data
q1	The number of candidate blocks selected randomly from the K blocks in step 2
q2	The number of candidate blocks selected randomly from the K blocks in step 3
tun_s	The tunning parameter for sample(observation) stability-based selection- default is 0.5
tun	the tunning parameter for variables stability-based selection
method	The penalty mehtod. The Brlasso indicates the block random lasso penalty. The Bradap is the block random adaptive lasso. The default is Brlasso penalty.

Details

When both the number of covariates and sample size are large, the methods described in the previous two sections can be naturally combined to conduct bootstrap in both dimensions. The proposed method consists of the following steps.

Step 1: Generate S bootstrap samples, each with size m, by sampling without replacement from the original data. Here $m << n$. Denote the subject index of the sth bootstrap sample as I_s.

Step 2: For $s = 1$ to S: Apply the block bootstrap penalization method(block random lasso or block adaptive random lasso) to the sth bootstrap sample, and obtain the estimate as $\hat{\beta}^{(s)}$. Compute the prediction mean square error $PMSE^s$.

Step 3: Conduct stability-based selection. That is, set $\hat{\beta}_j = 0$ if $S^{-1}\sum_{s=1}^{S} I(\hat{\beta}_j^{(s)} \neq 0) < \pi$ for $j = 1, \ldots, p$.

Step 4: If $\hat{\beta}_j \neq 0$ from the previous step, then calculate the final estimate as

$$\hat{\beta}_j = \frac{1}{S}\sum_{s=1}^{S} w^s \hat{\beta}_j^{(s)}$$

. The weight w^s is computed as

$$w^s = \frac{max(\sqrt{PMSE_0^s} - \sqrt{PMSE^s}, 0)}{\sum_{s=1}^{S} max(\sqrt{PMSE_0^s} - \sqrt{PMSE^s}, 0)}$$

. In Step 3, π can be chosen using a BIC-type criterion as described above.

For tun this is the vector of tunnig parameter π to conduct stability-based selection. It can be estimated by a BIC criterion. We set a coefficient estimate equal to zero if the estimates in more than π percentage of all bootstrap samples are zero.

Value

coef.min	the coefficients of the model that has the min value of BIC
PMSE	a vector of Predict mean square erro(PMSE) of each sample blocks

Author(s)

Kuangnan Fang, Shuangge Ma
Maintainer: Kuangnan Fang <xmufkn@xmu.edu.cn>

References

Fang, K. Ma, S. (2016) *Analyzing large datasets with bootstrap penalization*

See Also

Brlasso,Bradap, split_lasso, summary

Examples

```
x=matrix(rnorm(100*15),100,15)
y=rnorm(100)
fit1=split_Brlasso(X=x,y,sb=2,nc =3,B=30,q1=3,q2=3)
print(fit1)
fit1$coef.min # extract coefficients at a single value of lambda
fit2=split_Brlasso(X=x,y,sb=2,nc=3,B=30,q1=3,q2=3,method="Bradap")
fit2
```

---

split_lasso *Bootstrap penalization for data with large n but small to moderate p*

Description

Bootstrap penalization for data with large n but small to moderate p

Usage

```
split_lasso(X,y,sb=4,tun_s=0.5,method=c("lasso","adap"))
```

Arguments

X	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable,should be quantitative now; not yet for qualitative response
sb	the number of sample blocks
tun_s	The tunning parameter for sample(observation) stability-based selection- default is 0.5
method	The penalty mehtod. The lasso indicates the lasso penalty. The adap is the adaptive lasso. The default is lasso penalty.

Details

For data with large n but small to moderate p, we resort to the " m out of n" bootstrap to improve computational feasibility. The proposed method consists of the following steps.

Step 1: Generate S bootstrap samples, each with size m, by sampling without replacement from the original data. Here $m << n$. When there are a large number of "small computers" (as opposed to a big one), small m and large S can lead to significantly reduced computer time. The results are not sensitive to the value of m as long as it is not too small. Denote the subject index of the sth bootstrap sample as I_s.

Step 2: For $s = 1$ to S: Apply Lasso or adaptive Lasso to subjects in I_s and obtain the estimate Compute the prediction mean square error as $PMSE^s$

Step 3: Conduct stability-based selection. That is, set $\hat{\beta}_j = 0$ if

$$S^{-1} \sum_{s=1}^{S} I(\hat{\beta}_j^{(s)} \neq 0) < \pi_2, j = 1, \ldots, p$$

.

Step 4: If $\hat{\beta}_j \neq 0$ from the previous step, then calculate the final estimate as

$$\hat{\beta}_j = \frac{1}{S} \sum_{s=1}^{S} w^s \hat{\beta}_j^{(s)}$$

.

When it is difficult to manipulate and analyze data with a large sample size, we use bootstrap to generate multiple small datasets, each of which can be analyzed with low computational cost. The overall computer time can be significantly reduced with parallel computing. The final estimate is taken as the weighted average over bootstrap runs. In Step 3, the tuning parameter π can be chosen as

$$\hat{\pi}^{BIC} = argmin_{\pi}(nlog(n^{-1}\|y - \hat{y}\|^2) + log(n)|\hat{A}(\pi)|)$$

, in the same manner as in the previous section. In Step 4, when averaging over bootstrap estimates, we assign larger weights to models with better prediction performance on independent subjects.

For `tun_s` this is the vector of tunnig parameter π to conduct stability-based selection. It can be estimated by a BIC criterion. We set a coefficient estimate equal to zero if the estimates in more than π percentage of all bootstrap samples are zero.

Value

`coef.min`	the coefficients of the model that has the min value of BIC
`PMSE`	a vector of Predict mean square erro(PMSE) of each sample blocks

Author(s)

Kuangnan Fang, Shuangge Ma
Maintainer: Kuangnan Fang <xmufkn@xmu.edu.cn>

References

Fang, K. Ma, S. (2016) *Analyzing large datasets with bootstrap penalization*

See Also

split_Brlasso, summary

Examples

```
x=matrix(rnorm(500*20),500,20)
y=rnorm(500)
fit1=split_lasso(X=x,y)
print(fit1)
fit1$coef.min # extract coefficients with min BIC
```

Index