Package ‘ISBF’

February 19, 2015

Type Package

Title Iterative Selection of Blocks of Features - ISBF

Version 0.2.1

Date 2014/11/10

Author Pierre Alquier

Maintainer Pierre Alquier <pierre.alquier@ensae.fr>

Description Selection of features for sparse regression estimation (like the LASSO). Selection of blocks of features when the regression parameter is sparse and constant by blocks (like the Fused-LASSO). Application to cgh arrays.

License GPL

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-11-10 14:24:12

R topics documented:

ISBF-package *Iterative Selection of Blocks of Features - ISBF*

Description

Selection of features for sparse regression estimation (like the LASSO). Selection of blocks of features when the regression parameter is sparse and constant by blocks (like the Fused-LASSO). Application to CGH data. TEST VERSION.

Details

Package:	ISBF
Type:	Package
Version:	0.1
Date:	2014-11-10
License:	GPL
LazyLoad:	yes

The package contains the following functions:

1) isbfReg - performs regression estimation in the model Y = Xb + e where b is sparse. > isbfReg(X,Y) If b is also constant by block, the function may find blocks up to a given size K. > isbf(X,Y,K=...)

2) isbf - particular case where X is the identity matrix, Y = b + e, and b is sparse. This function is much faster than isbfReg. > isbf(Y) If b is also constant by block, the function may find blocks up to size K. > isbf(Y,K=...)

[Be careful, for functions 1 and 2, the computation time and the memory used grows with K!!!!]

3) cghISBF - applies isfb to every chromosome in a cgh array. The object returned has the type cghFLasso used in the package cghFLasso (see Tibshirani and Wang, 2008). Therefore, it can be plotted using the package cghFLasso.

Finally, CGHDisease1 is an example of cgh array taken from the cghFLasso package.

Author(s)

Pierre Alquier <alquier@ensae.fr>

References

P. Alquier, An Algorithm for Iterative Selection of Blocks of Features, Proceedings of ALT'10, 2010, M. Hutter, F. Stephan, V. Vovk and T. Zeugmann Eds., Lecture Notes in Artificial Intelligence, pp. 35-49, Springer.

P. Alquier, Iterative Feature Selection in Least Square Regression Estimation, Annales de l'IHP, B (Proba. Stat.), 2008, vol. 44, no. 1, pp 47-88.

R. Tibshirani and P. Wang, Spatial Smoothing and Hot Spot Detection for CGH Data Using the Fused Lasso, Biostatistics, 2008, vol. 9, no. 1, pp 18-29.

`CGHDisease1` *An example of CGH array.*

Description

An example of CGH array for the package ISBF. Taken from the package cghFLasso.

Usage

```
data(CGHDisease1)
```

Format

A list of 3 (data, chr, nucposi).

Details

The package cghFLasso contains several CGH arrays, in two categories (normal, and disease). This is just a copy of the first array in the disease category. Quoting the package cghFLasso: the value of each entry is the log fluorescence ratio resulted from the CGH experiment. The order of the genes/clones in the rows is the same as the order of the genes/clones on the genome. chr and nucposi provide chromosome number and nucleotide position for each gene/clone.

Source

http://www-stat.stanford.edu/~tibs/software.html

References

R. Tibshirani and P. Wang, Spatial Smoothing and Hot Spot Detection for CGH Data Using the Fused Lasso, Biostatistics, 2008, vol. 9, no. 1, pp 18-29.

Examples

```
data(CGHDisease1)
cgh = cghISBF(CGHDisease1$data,CGHDisease1$chromosome,CGHDisease1$nucposi,s=1,K=100)
```

cghISBF *Iterative Selection of Blocks of Features for CGH arrays*

Description

A function to process reference CGH arrays using the isbf function. Note that this function takes the same input than the function cghFLasso of the package cghFLasso, see Tibshirani and Wang (2008). The output is an object of the type cghFLasso. Therefore, it can be plotted using the package cghFLasso.

Usage

```
cghISBF(CGH.Array, chromosome, nucleotide.position, epsilon = 0.05,
K = 1, impmin = 1/100, s = NULL, v = NULL)
```

Arguments

`CGH.Array` Numeric vector. The result of one or multiple CGH experiments. Each column is the log2 ratios returned from one array experiment and is ordered according to the gene/clones position on the genome. No missing values allowed.

`chromosome` Numeric vector. Length should be the same as CGH.Array. The chromosome number of each gene/clone.

`nucleotide.position` Numeric vector. Length should be the same as CGH.Array. The nucleotide position of each gene/clone. This information is used mainly for plot.

`epsilon` The confidence level when running ISBF. The theoretical guarantees in Alquier (2010) is that each iteration of the ISBF procedure gets closer to the real parameter b with probability at least 1-epsilon. When epsilon is very small, the procedure becomes very conservative. When epsilon is too large, there is a risk of overfitting. If not specified, epsilon = 5%.

`K` The maximal length of blocks checked in the iterations. If not specified, K=1. If K is larger than the length of a chromosome, then it will we adjusted when the function isbf is called on this chromosome.

`impmin` Criterion for the end of the iterations. When no more iteration can provide an improvement of Xb larger than impmin, the algorithm stops. If not speficief, impmin=1/100.

`s` The threshold used in the iterations. If not specified, the theoretical value of Alquier (2010) is used: s = sqrt(2*v*log(p*K/epsilon)).

`v` The variance of e, if it is known. If not specified, estimated on the data (by a MA(10)-smoothing).

Value

`Esti.CopyN` data vector reporting the estimated DNA copy numbers for seleted genes/clones of all the samples.

`CGH.Array` a copy of the input data CGH.Array.

`chromosome` a copy of the input chromosome.

`nucleotide.position` of copy of the input nucleotide.position.

`FDR` NULL, FDR is not computed.

Author(s)

Pierre Alquier <alquier@ensae.fr>

References

P. Alquier, An Algorithm for Iterative Selection of Blocks of Features, Proceedings of ALT'10, 2010, M. Hutter, F. Stephan, V. Vovk and T. Zeugmann Eds., Lecture Notes in Artificial Intelligence, pp. 35-49, Springer.

R. Tibshirani and P. Wang, Spatial Smoothing and Hot Spot Detection for CGH Data Using the Fused Lasso, Biostatistics, 2008, vol. 9, no. 1, pp 18-29.

Examples

```
data(CGHDisease1)
cgh = cghISBF(CGHDisease1$data,CGHDisease1$chromosome,CGHDisease1$nucposi,s=1,K=100)
```

isbf *Iterative Selection of Blocks of Features - isbf*

Description

isbfperforms estimation in the model Y = b + e where the unknown parameter b is sparse, or sparse and constant by blocks. Y is a vector of size p, b a vector of size p and e is the noise.

When b is sparse and constant by blocks, one can use isbf(Y,K=...) where K is the expected maximal size for a block. The method used is Iterative Selection of Blocks of Features procedure of Alquier (2010). When b is only sparse, one can use isbf(Y), as the default value for K is 1. Of course, one can always set K=p, but be careful, the computation time and the memory used is directly proportional to p*K.

NOTE: one can used isbfReg(X,Y) with X the identity matrix instead, but isbf(Y) is really faster.

Usage

```
isbf(Y, epsilon = 0.05, K = 1, impmin = 1/100,
s = NULL, v = NULL)
```

Arguments

`Y` The data. A vector of size p.

`epsilon` The confidence level. The theoretical guarantees in Alquier (2010) is that each iteration of the ISBF procedure gets closer to the real parameter b with probability at least 1-epsilon. When epsilon is very small, the procedure becomes very conservative. When epsilon is too large, there is a risk of overfitting. If not specified, epsilon = 5%.

`K` The maximal length of blocks. If not specified, K=1, this means we seek for a sparse (not constant by block) parameter b. One should take a larger K is b is really expected to be constant by blocks. If p is quite small (up to 1000), K=p is a reasonnable choice. For larger values of p, please take into account that the computation time and the memory used is directly proportional to p*K.

`impmin` Criterion for the end of the iterations. When no more iteration can provide an improvement of Xb larger than impmin, the algorithm stops. If not speficied, impmin=1/100.

`s` The threshold used in the iterations. If not specified, the theoretical value of Alquier (2010) is used: s = sqrt(2*v*log(p*K/epsilon)).

`v` The variance of e, if it is known. If not specified, estimated on the data (by a MA(10)-smoothing).

Value

`beta` The estimated parameter b.

`s` The value of s.

`impmin` The value of impmin.

`K` The value of K.

Author(s)

Pierre Alquier <alquier@ensae.fr>

References

P. Alquier, An Algorithm for Iterative Selection of Blocks of Features, Proceedings of ALT'10, 2010, M. Hutter, F. Stephan, V. Vovk and T. Zeugmann Eds., Lecture Notes in Artificial Intelligence, pp. 35-49, Springer.

Examples

```
# generating data
b = c(rep(0,100),rep(2,40),rep(0,60))
e = rnorm(200,0,0.3)
Y = b + e

# call of isbf
A = isbf(Y,K=200,v=0.3)
```

```
# visualization of the results
plot(Y)
lines(A$beta,col="red")
```

---

isbfReg *Iterative Selection of Blocks of Features in Regression Estimation - isbfReg*

Description

isbfReg performs regression estimation in the model Y = Xb + e where the unknown parameter b is sparse, or sparse and constant by blocks. Y is a vector of size n, X a (n,p) matrix, b a vector of size p and e is the noise.

When b is sparse, one can basically use isbfReg(X,Y), the method used is the Iterative Feature Selection procedure of Alquier (2008). When b is sparse and constant by blocks, one can use isbfReg(X,Y,K=...) where K is the expected maximal size for a block. The method used is Iterative Selection of Blocks of Features procedure of Alquier (2010). Of course, one can always set K=p, but be careful, the computation time and the memory used is directly proportional to p*K.

Usage

```
isbfReg(X, Y, epsilon = 0.05, K = 1, impmin = 1/100, favgroups = 0,
centX = TRUE, centY = TRUE, s = NULL, v = NULL)
```

Arguments

`X`	The data: the matrix of inputs. Size (n,p).
`Y`	The data: the vector of outputs. Size n.
`epsilon`	The confidence level. The theoretical guarantees in Alquier (2010) is that each iteration of the ISBF procedure gets closer to the real parameter b with probability at least 1-epsilon. When epsilon is very small, the procedure becomes very conservative. When epsilon is too large, there is a risk of overfitting. If not specified, epsilon = 5%.
`K`	The maximal length of blocks checked in the iterations. If not specified, K=1, this means we seek for a sparse (not constant by block) parameter b, as in Alquier (2008). One should take a larger K is b is really expected to be constant by blocks. If p is quite small (up to 1000), K=p is a reasonnable choice. For larger values of p, please take into account that the computation time and the memory used is directly proportional to p*K.
`impmin`	Criterion for the end of the iterations. When no more iteration can provide an improvement of Xb larger than impmin, the algorithm stops. If not speficied, impmin=1/100.
`favgroups`	In case of noisy input data, one may want to favor larger groups in order to stabilize estimation. By default, this variable is taken to 0, but take it larger for noisy input data.

`centX` If TRUE, the function centers the variables in X before processing.

`centY` If TRUE, the function centers the variable Y before processing.

`s` The threshold used in the iterations. If not specified, the theoretical value of Alquier (2010) is used: s = sqrt(2*v*log(p*K/epsilon)).

`v` The variance of e, if it is known. If not specified, this parameter is VERY roughly estimated by var(Y)/2.

Value

`beta` The estimated parameter b.

`s` The value of s.

`impmin` The value of impmin.

`K` The value of K.

Author(s)

Pierre Alquier <alquier@ensae.fr>

References

P. Alquier, An Algorithm for Iterative Selection of Blocks of Features, Proceedings of ALT'10, 2010, M. Hutter, F. Stephan, V. Vovk and T. Zeugmann Eds., Lecture Notes in Artificial Intelligence, pp. 35-49, Springer.

P. Alquier, Iterative Feature Selection in Least Square Regression Estimation, Annales de l'IHP, B (Proba. Stat.), 2008, vol. 44, no. 1, pp 47-88.

Examples

```
# generating data
X = matrix(data=rnorm(5000),nr=50,nc=100)
b = c(rep(0,50),rep(-3,30),rep(0,20))
e = rnorm(50,0,0.3)
Y = X%*%b + e

# call of isbfReg
A = isbfReg(X,Y,K=100,v=0.3)

# visualization of the results
plot(b)
lines(A$beta,col="red")
```

Index