Package 'grpSLOPE'

November 20, 2016

Type Package

Title Group Sorted L1 Penalized Estimation

Version 0.2.1

Description Group SLOPE is a penalized linear regression method that is used for adaptive selection of groups of significant predictors in a high-dimensional linear model.
The Group SLOPE method can control the (group) false discovery rate at a user-specified level (i.e., control the expected proportion of irrelevant among all selected groups of predictors).

License GPL-3

URL https://github.com/agisga/grpSLOPE.git

BugReports https://github.com/agisga/grpSLOPE/issues

LazyData TRUE

Imports SLOPE (>= 0.1.3)

RoxygenNote 5.0.1.9000

Suggests testthat, knitr, rmarkdown, pander

VignetteBuilder knitr

NeedsCompilation no

Author Alexej Gossmann [aut, cre],
Damian Brzyski [aut],
Weijie Su [aut],
Malgorzata Bogdan [aut],
Ewout van den Berg [ctb] (A part of the optimization code was obtained from http://statweb.stanford.edu/~candes/SortedL1/software.html under GNU GPL-3),
Emmanuel Candes [ctb] (A part of the optimization code was obtained from http://statweb.stanford.edu/~candes/SortedL1/software.html under GNU GPL-3)

Maintainer Alexej Gossmann <alexej.go@googlemail.com>

Repository CRAN

Date/Publication 2016-11-20 09:18:04

R topics documented:

coef.grpSLOPE 2
getGroupID 3
grpSLOPE 4
lambdaGroupSLOPE 6
predict.grpSLOPE 7
proxGroupSortedL1 8
proximalGradientSolverGroupSLOPE 9
sigma 10

Index **12**

`coef.grpSLOPE` *Extract model coefficients*

Description

Extract the regression coefficients from a `grpSLOPE` object, either on the scale of the normalized design matrix (i.e., columns centered and scaled to unit norm), or on the original scale.

Usage

```
## S3 method for class 'grpSLOPE'
coef(object, scaled = TRUE, ...)
```

Arguments

`object`	A `grpSLOPE` object
`scaled`	Should the coefficients be returned for the normalized version of the design matrix?
`...`	Potentially further arguments passed to and from methods

Details

If `scaled` is set to `TRUE`, then the coefficients are returned for the normalized version of the design matrix, which is the scale on which they were computed. If `scaled` is set to `FALSE`, then the coefficients are transformed to correspond to the original (unaltered) design matrix. In case that `scaled = FALSE`, an estimate for the intercept term is returned with the other coefficients. In case that `scaled = TRUE`, the estimate of the intercept is always equal to zero, and is not explicitly provided.

Value

A named vector of regression coefficients where the names signify the group that each entry belongs to

Examples

```
set.seed(1)
A   <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(letters[1:20]), each=5)
b   <- c(rep(1, 20), rep(0, 80))
y   <- A %*% b + rnorm(10)
result <- grpSLOPE(X=A, y=y, group=grp, fdr=0.1)
head(coef(result), 8)
#       a_1       a_2       a_3       a_4       a_5       b_1       b_2       b_3
#  7.942177  7.979269  8.667013  8.514861 10.026664  8.963364 10.037355 10.448692
head(coef(result, scaled = FALSE), 8)
# (Intercept)        a_1        a_2        a_3        a_4        a_5        b_1        b_2
#  -0.4418113  0.8886878  0.8372108  0.8422089  0.8629597  0.8615827  0.9323849  0.9333445
```

`getGroupID` *Get a groupID object*

Description

Mostly intended for internal use.

Usage

```
getGroupID(group)
```

Arguments

`group`	A vector describing the grouping structure. It should contain a group id for each predictor variable.

Value

An object of class groupID, which is a list, whose members are vectors of indices corresponding to each group. The names of the list members are the corresponding group names.

Examples

```
group  <- c("A", "A", 2, 9, "A", 9, 9, 2, "A")
group.id <- getGroupID(group)
group.id
# $A
# [1] 1 2 5 9
#
# $`2`
# [1] 3 8
#
# $`9`
# [1] 4 6 7
```

```
#
# attr(,"class")
# [1] "groupID"
```

---

grpSLOPE *Group SLOPE (Group Sorted L-One Penalized Estimation)*

Description

Performs selection of significant groups of predictors and estimation of the corresponding coefficients using the Group SLOPE method (see Brzyski et. al., 2016).

Usage

```
grpSLOPE(X, y, group, fdr, lambda = "corrected", sigma = NULL,
  verbose = FALSE, orthogonalize = NULL, normalize = TRUE,
  max.iter = 10000, dual.gap.tol = 1e-06, infeas.tol = 1e-06,
  x.init = vector(), ...)
```

Arguments

X	The model matrix
y	The response variable
group	A vector describing the grouping structure. It should contain a group id for each predictor variable.
fdr	Target group false discovery rate (gFDR)
lambda	Method used to obtain the regularizing sequence lambda. Possible values are "max", "mean", and "corrected" (default). See lambdaGroupSLOPE for detail. Alternatively, any non-increasing sequence of the correct length can be passed.
sigma	Noise level. If ommited, estimated from the data, using Procedure 2 in Brzyski et. al. (2016).
verbose	A logical specifying whether to print output or not
orthogonalize	Whether to orthogonalize the model matrix within each group. Do not set manually unless you are certain that your data is appropriately pre-processed.
normalize	Whether to center the input data and re-scale the columns of the design matrix to have unit norms. Do not disable this unless you are certain that your data are appropriately pre-processed.
max.iter	See proximalGradientSolverGroupSLOPE.
dual.gap.tol	See proximalGradientSolverGroupSLOPE.
infeas.tol	See proximalGradientSolverGroupSLOPE.
x.init	See proximalGradientSolverGroupSLOPE.
...	Options passed to prox_sorted_L1

Details

Multiple methods are available to generate the regularizing sequence `lambda`, see `lambdaGroupSLOPE` for detail. The model matrix is transformed by orthogonalization within each group (see Section 2.1 in Brzyski et. al., 2016), and penalization is imposed on $\|X_{I_i}\beta_{I_i}\|$. When `orthogonalize = TRUE`, due to within group orthogonalization, the solution vector `beta` cannot be computed, if a group submatrix does not have full column rank (e.g., if there are more predictors in a selected group than there are observations). In that case only the solution vector `c` of the transformed (orthogonalized) model is returned. Additionally, in any case the vector `group.norms` is returned with its ith entry being $\|X_{I_i}\beta_{I_i}\|$, i.e., the overall effect of each group. Note that all of these results are returned on the scale of the normalized versions of `X` and `y`. However, `original.scale` contains the regression coefficients transformed to correspond to the original (unaltered) `X` and `y`. In that case, an estimate for the intercept term is also returned with the other coefficients in `original.scale` (while on the normalized scale the estimate of the intercept is always equal to zero, and is not explicitly provided in the `grpSLOPE` output).

Value

A list with members:

beta Solution vector. See Details.

c Solution vector of the transformed model. See Details.

group.norms Overall effect of each group. See Details.

selected Names of selected groups (i.e., groups of predictors with at least one non-zero coefficient estimate)

optimal Convergence status

iter Iterations of the proximal gradient method

lambda Regularizing sequence

lambda.method Method used to construct the regularizing sequence

sigma (Estimated) noise level

group The provided grouping structure (corresponding to `beta`)

group.c Grouping structure of the transformed model (corresponding to `c`)

original.scale A list containing the estimated intercept and regression coefficients on the original scale. See Details.

References

D. Brzyski, A. Gossmann, W. Su, and M. Bogdan (2016) *Group SLOPE – adaptive selection of groups of predictors*, https://arxiv.org/abs/1610.04960

Examples

```
# generate some data
set.seed(1)
A   <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(1:20), each=5)
b   <- c(runif(20), rep(0, 80))
```

```
# (i.e., groups 1, 2, 3, 4, are truly significant)
y   <- A %*% b + rnorm(10)
fdr <- 0.1 # target false discovery rate
# fit a Group SLOPE model
result <- grpSLOPE(X=A, y=y, group=grp, fdr=fdr)
result$selected
# [1] "1"  "2"  "3"  "4"  "14"
result$sigma
# [1] 0.7968632
head(result$group.norms)
#         1         2         3         4         5         6
#  2.905449  5.516103  8.964201 10.253792  0.000000  0.000000
```

`lambdaGroupSLOPE` *Regularizing sequence for Group SLOPE*

Description

Generate the regularizing sequence `lambda` for the Group SLOPE problem according to one of multiple methods (see Details).

Usage

```
lambdaGroupSLOPE(method, fdr, group, wt, n.obs = NULL)
```

Arguments

`method`	Possible values are "max", "mean", and "corrected". See under Details.
`fdr`	Target group false discovery rate (gFDR)
`group`	A vector describing the grouping structure. It should contain a group id for each predictor variable.
`wt`	A named vector of weights, one weight per group of predictors (named according to names as in vector `group`)
`n.obs`	Number of observations (i.e., number of rows in `A`); required only if method is "corrected"

Details

Multiple methods are available to generate the regularizing sequence `lambda`:

- "max" – lambdas as in Theorem 2.5 in Brzyski et. al. (2016). Provalby controls gFDR in orthogonal designs.
- "mean" – lambdas of equation (2.16) in Brzyski et. al. (2016). Applicable for gFDR control in orthogonal designs. Less conservative than "max".
- "corrected" – lambdas of Procedure 1 in Brzyski et. al. (2016); in the special case that all group sizes are equal and `wt` is a constant vector, Procedure 6 of Brzyski et. al. (2016) is applied. Applicable for gFDR control when predictors from different groups are stochastically independent.

References

D. Brzyski, A. Gossmann, W. Su, and M. Bogdan (2016) *Group SLOPE – adaptive selection of groups of predictors*, https://arxiv.org/abs/1610.04960

Examples

```
# specify 6 groups of sizes 2, 3, and 4
group <- c(1, 1, 2, 2, 2, 3, 3, 3, 3,
           4, 4, 5, 5, 5, 6, 6, 6, 6)
# set the weight for each group to the square root of the group's size
wt <- rep(c(sqrt(2), sqrt(3), sqrt(4)), 2)
names(wt) <- 1:6
# compute different lambda sequences
lambda.max <- lambdaGroupSLOPE(method="max", fdr=0.1, group=group, wt=wt)
lambda.mean <- lambdaGroupSLOPE(method="mean", fdr=0.1, group=group, wt=wt)
lambda.corrected <- lambdaGroupSLOPE(method="corrected", fdr=0.1,
                                     group=group, wt=wt, n.obs=1000)
rbind(lambda.max, lambda.mean, lambda.corrected)
#                     [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
# lambda.max       2.023449 1.844234 1.730818 1.645615 1.576359 1.517427
# lambda.mean      1.880540 1.723559 1.626517 1.554561 1.496603 1.447609
# lambda.corrected 1.880540 1.729811 1.637290 1.568971 1.514028 1.467551
```

`predict.grpSLOPE` *Obtain predictions*

Description

Obtain predictions from a `grpSLOPE` model on new data

Usage

```
## S3 method for class 'grpSLOPE'
predict(object, newdata, ...)
```

Arguments

`object`	A `grpSLOPE` object
`newdata`	Predictor variables arranged in a matrix
`...`	Potentially further arguments passed to and from methods

Details

Note that `newdata` must have the same shape, and must contain the same predictor variables as columns in the same order as the design matrix `X` that was used for the `grpSLOPE` model fit.

Examples

```
set.seed(1)
A   <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(1:20), each = 5)
b   <- c(rep(1, 20), rep(0, 80))
y   <- A %*% b + rnorm(10)
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1)
newdata <- matrix(rnorm(800), 8, 100)
# group SLOPE predictions:
predict(result, newdata)
# [1] -5.283385 -6.313938 -3.173068  1.901488  9.796677 -0.144516 -0.611164 -5.167620
# true mean values:
as.vector(newdata %*% b)
# [1] -5.0937160 -6.5814111 -3.5776124  2.7877449 11.0668777  1.0253236 -0.4261076 -4.8622940
```

`proxGroupSortedL1` *Prox for group SLOPE*

Description

Evaluate the proximal mapping for the group SLOPE problem.

Usage

```
proxGroupSortedL1(y, group, lambda, ...)
```

Arguments

`y`	The response vector
`group`	Either a vector or an object of class `groupID` (e.g. as produced by `getGroupID`), which is describing the grouping structure. If it is a vector, then it should contain a group id for each predictor variable.
`lambda`	A decreasing sequence of regularization parameters λ
`...`	Options passed to `prox_sorted_L1`

Details

`proxGroupSortedL1` evaluates the proximal mapping of the group SLOPE problem by reducing it to the prox for the (regular) SLOPE and then applying the fast prox algorithm for the Sorted L1 norm.

References

M. Bogdan, E. van den Berg, C. Sabatti, W. Su, E. Candes (2015), *SLOPE – Adaptive variable selection via convex optimization*, http://arxiv.org/abs/1407.3824

Examples

```
grp <- c(0,0,0,1,1,0,2,1,0,2)
proxGroupSortedL1(y = 1:10, group = grp, lambda = c(10, 9, 8))
#  [1] 0.2032270 0.4064540 0.6096810 0.8771198 1.0963997 1.2193620 1.3338960
#  [8] 1.7542395 1.8290430 1.9055657
```

`proximalGradientSolverGroupSLOPE`

Proximal gradient method for Group SLOPE

Description

Compute the coefficient estimates for the Group SLOPE problem.

Usage

```
proximalGradientSolverGroupSLOPE(y, A, group, wt, lambda, max.iter = 10000,
  verbose = FALSE, dual.gap.tol = 1e-06, infeas.tol = 1e-06,
  x.init = vector(), ...)
```

Arguments

`y`	the response vector
`A`	the model matrix
`group`	A vector describing the grouping structure. It should contain a group id for each predictor variable.
`wt`	A vector of weights (per coefficient)
`lambda`	A decreasing sequence of regularization parameters λ
`max.iter`	Maximal number of iterations to carry out
`verbose`	A `logical` specifying whether to print output or not
`dual.gap.tol`	The tolerance used in the stopping criteria for the duality gap
`infeas.tol`	The tolerance used in the stopping criteria for the infeasibility
`x.init`	An optional initial value for the iterative algorithm
`...`	Options passed to `prox_sorted_L1`

Details

`proximalGradientSolverGroupSLOPE` computes the coefficient estimates for the Group SLOPE model. The employed optimization algorithm is FISTA with backtracking Lipschitz search.

Value

A list with members:

x Solution (n-by-1 matrix)

status Convergence status: 1 if optimal, 2 if iteration limit reached

L Approximation of the Lipschitz constant (step size)

iter Iterations of the proximal gradient method

L.iter Total number of iterations spent in Lischitz search

References

D. Brzyski, A. Gossmann, W. Su, and M. Bogdan (2016) *Group SLOPE – adaptive selection of groups of predictors*, https://arxiv.org/abs/1610.04960

A. Gossmann, S. Cao, Y.-P. Wang (2015) *Identification of Significant Genetic Variants via SLOPE, and Its Extension to Group SLOPE*, http://dx.doi.org/10.1145/2808719.2808743

Examples

```
set.seed(1)
A   <- matrix(runif(100, 0, 1), 10, 10)
grp <- c(0, 0, 1, 1, 2, 2, 2, 2, 2, 3)
wt  <- c(2, 2, 2, 2, 5, 5, 5, 5, 5, 1)
x   <- c(0, 0, 5, 1, 0, 0, 0, 1, 0, 3)
y   <- A %*% x
lam <- 0.1 * (10:7)
result <- proximalGradientSolverGroupSLOPE(y=y, A=A, group=grp, wt=wt, lambda=lam, verbose=FALSE)
result$x
#           [,1]
#  [1,] 0.000000
#  [2,] 0.000000
#  [3,] 3.856005
#  [4,] 2.080736
#  [5,] 0.000000
#  [6,] 0.000000
#  [7,] 0.000000
#  [8,] 0.000000
#  [9,] 0.000000
# [10,] 3.512833
```

`sigma` *Extract (estimated) noise level*

Description

Extract the noise level of the `grpSLOPE` model.

Usage

```
## S3 method for class 'grpSLOPE'
sigma(object, ...)
```

Arguments

object	A grpSLOPE object
...	Potentially further arguments passed to and from methods

Details

This basically obtains object$sigma. For R (>= 3.3.0) sigma is an S3 method with the default method coming from the stats package.

Examples

```
set.seed(1)
A   <- matrix(rnorm(100^2), 100, 100)
grp <- rep(rep(1:20), each = 5)
b   <- c(rep(1, 20), rep(0, 80))
y   <- A %*% b + rnorm(10)
# model with unknown noise level
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1)
sigma(result)
# [1] 0.6505558
# model with known noise level
result <- grpSLOPE(X = A, y = y, group = grp, fdr = 0.1, sigma = 1)
sigma(result)
# [1] 1
```

Index

coef.grpSLOPE, 2

getGroupID, 3, *8*
grpSLOPE, 4

lambdaGroupSLOPE, *4*, *5*, 6

predict.grpSLOPE, 7
prox_sorted_L1, *4*, *8*, *9*
proxGroupSortedL1, 8
proximalGradientSolverGroupSLOPE, *4*, 9

sigma, 10