Package 'sglssnal'

September 11, 2026

Title Sparse-Group Lasso via Semismooth Newton Augmented Lagrangian

Version 0.1.0

Description Implements the sparse-group lasso method of Zhang et al. (2020) <doi:10.1007/s10107-018-1329-6>. Unlike many widely available methods based on first-order descent, this method uses second-order information to solve the dual optimization problem via a semismooth Newton method.

License MIT + file LICENSE

URL https://github.com/roobnloo/sglssnal

BugReports https://github.com/roobnloo/sglssnal/issues

Depends R (>= 4.4.0)

Encoding UTF-8

LazyData true

Imports Matrix, Rcpp, RSpectra, methods, utils

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Language en-US

VignetteBuilder knitr

NeedsCompilation yes

Author Robin Liu [aut, cre],
Yangjing Zhang [ctb] (Author of the original MATLAB SSNAL implementation this package ports)

Maintainer Robin Liu <robin28liu@gmail.com>

Repository CRAN

Date/Publication 2026-09-11 15:30:02 UTC

Contents

`coef.sglssnal` *Extract estimated coefficients from a* `sglssnal` *object*

Description

Extract estimated coefficients from a `sglssnal` object

Usage

```
## S3 method for class 'sglssnal'
coef(object, ...)
```

Arguments

`object`	Fitted object of class `sglssnal`.
`...`	Additional arguments passed to or from other methods.

Value

A numeric matrix of estimated coefficients.

Examples

```
set.seed(1)
n <- 50
p <- 20
A <- matrix(rnorm(n * p), n, p)
b <- rnorm(n)
group <- rep(1:4, each = 5)

fit <- sglssnal(A, b, group, lambda = 0.5, alpha = 0.5)
coef(fit)
```

`cv.sglssnal` *Cross-validation for sglssnal*

Description

Perform cross-validation for the sglssnal algorithm over a path of lambda values.

Usage

```
cv.sglssnal(
  A,
  b,
  group,
  alpha = 0.05,
  lambda = NULL,
  nlambda = 100,
  lambda_min_ratio = 1e-04,
  nfolds = 5,
  foldid = NULL,
  verbose = 1L,
  stoptol = 1e-06,
  stoptolcv = 1e-04,
  ...
)
```

Arguments

`A`	$n \times p$ design matrix.
`b`	n response vector.
`group`	Length-p group-membership vector: `group[j]` is the group id of column `j` of `A`. Group ids may be any atomic vector (integer, character, or factor) and need not be contiguous or sorted; groups are formed from `sort(unique(group))`.
`alpha`	Mixing parameter determining the relative weight of the ℓ_1 and group ℓ_2 penalty. Must be in $[0, 1]$. Default is `0.05`.
`lambda`	Vector of lambda parameters to tune. If `NULL`, the lambda path is automatically generated by `sglssnal` using `nlambda` and `lambda_min_ratio`. If supplied, `nlambda` and `lambda_min_ratio` are ignored.
`nlambda`	Number of lambda values to use when lambda is NULL.
`lambda_min_ratio`	Minimum ratio of the smallest to largest lambda when lambda is NULL.
`nfolds`	Number of folds for cross-validation. Ignored if `foldid` is provided.
`foldid`	Vector of integers specifying the fold for each observation. If `NULL`, a default assignment is used.

verbose	How much to print to the console: 0 is silent; 1 (default) shows a compact text progress bar tracking the lambda path; 2 prints full per-lambda SSNAL diagnostics (objective, feasibility, iteration counts, ...) instead of the progress bar; 3 additionally prints the inner semismooth-Newton subproblem's iteration trace.
stoptol	Tolerance for stopping criteria. Default is 1e-6.
stoptolcv	Tolerance for convergence in cross-validation folds. May be set to a smaller value than stoptol for faster convergence. Default is 1e-4.
...	Additional arguments passed to sglssnal().

Value

List of class cv.sglssnal, sglssnal, containing the following components:

- obj: Matrix containing primal and dual objective values for the solution.
- x: Matrix of primal variables, with each column corresponding to a lambda.
- x0: Vector of intercept values for each lambda.
- y: Matrix of y dual variables, with each column corresponding to a lambda.
- z: Matrix of z dual variables, with each column corresponding to a lambda.
- info: List containing information about the optimization process.
- cv_info: List containing the lambdas, cross-validation errors, and the index of the optimal lambda.

Examples

```
set.seed(1)
n <- 50
p <- 20
A <- matrix(rnorm(n * p), n, p)
bstar <- c(rnorm(5), rep(0, p - 5))
b <- as.numeric(A %*% bstar + rnorm(n, sd = 0.1))
group <- rep(1:4, each = 5)

cvfit <- cv.sglssnal(A, b, group, nlambda = 10, nfolds = 3)
cvfit$cv_info$cv_lambda_id
```

---

predict.sglssnal *Predict method for* sglssnal *objects*

Description

Predict method for sglssnal objects

Usage

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

Arguments

`object`	Fitted object of class `sglssnal`.
`newdata`	A matrix of new data.
`...`	Additional arguments passed to or from other methods.

Value

A numeric matrix of predictions, with columns corresponding to lambda values.

Examples

```
set.seed(1)
n <- 50
p <- 20
A <- matrix(rnorm(n * p), n, p)
b <- rnorm(n)
group <- rep(1:4, each = 5)

fit <- sglssnal(A, b, group, lambda = 0.5, alpha = 0.5)
predict(fit, A)
```

`riboflavin` *Riboflavin production and gene expression, grouped by GO Slim term*

Description

A real, pre-processed dataset of riboflavin (vitamin B2) production by engineered *Bacillus subtilis* strains, arranged for direct use with `sglssnal()`/`cv.sglssnal()`. Predictors are gene expression levels; genes are grouped by their "generic GO Slim" Biological Process term (36 groups).

Usage

```
riboflavin
```

Format

A list with components:

A 71 x 1199 numeric matrix of log gene-expression levels, one row per strain variant, one column per gene. Column names are the original Affymetrix probe IDs (e.g. `"AADK_at"`); row names are the original microarray scan identifiers.

b Length-71 numeric vector of log-transformed riboflavin production rate.

group Length-1199 character vector giving the GO term ID (e.g. `"GO:0006766"`) for each column of `A`, suitable for the `group` argument of `sglssnal()`. Group sizes range from 284 genes (transmembrane transport) down to singletons. 13% of these genes have GO annotations spanning more than one Slim term; since `group` requires a strict partition, each such gene was assigned to its most-frequently annotated Slim term.

Source

Gene expression and production-rate data: Bühlmann, P., Kalisch, M. and Meier, L. (2014). High-dimensional statistics with a view towards applications in biology. *Annual Review of Statistics and its Application*, 1, 255-278. Data kindly provided by DSM (Switzerland); originally distributed in the `hdi` package. Gene-to-GO-term annotations: UniProt-GOA *B. subtilis* 168 proteome annotation file, https://ftp.ebi.ac.uk/pub/databases/GO/goa/proteomes/ (GO release 2026-07-26). GO Slim term set: `goslim_generic`, https://current.geneontology.org/ontology/subsets/goslim_generic.obo. Gene Ontology Consortium data and data products are licensed under CC BY 4.0; see https://geneontology.org/docs/go-citation-policy/.

Examples

```
fit <- sglssnal(riboflavin$A, riboflavin$b, riboflavin$group, lambda = 0.05)
coef(fit)
```

`sglssnal` *Run Sparse-Group Lasso via Semismooth Newton Augmented Lagrangian*

Description

Fits a sparse-group lasso model using second-order information to solve the dual problem. The penalty function is given by

$$\Phi(x) = \alpha\lambda||x||_1 + (1-\alpha)\lambda\sum_{i=1}^{g} w_i||x_{G_i}||_2$$

where G_i is the i-th group, w_i its penalty factor, and $\alpha \in [0,1]$ mixes the ℓ_1 and group ℓ_2 penalties. The primal problem is given by

$$\min_{x\in\mathbb{R}^p} \ \frac{1}{2}||Ax-b||_2^2 + \Phi(x)$$

while the dual problem has the form

$$\begin{aligned} \max_{y\in\mathbb{R}^n, z\in\mathbb{R}^p} \ & -\langle b, y\rangle - \tfrac{1}{2}||y||_2^2 - \Phi^*(z) \\ \text{s.t. } & A^\top y + z = 0 \end{aligned}$$

where $\Phi^*(z)$ denotes the Fenchel conjugate of Φ. The algorithm is based on the work of Zhang et al. (2020).

Usage

```
sglssnal(
  A,
  b,
  group,
  lambda = NULL,
```

```
  nlambda = 100,
  lambda_min_ratio = 1e-04,
  alpha = 0.05,
  pfgroup = sqrt(as.numeric(table(group))),
  intercept = TRUE,
  standardize = TRUE,
  stoptol = 1e-06,
  stopopt = 2L,
  verbose = 1L,
  maxit = 5000L,
  Lip = NULL,
  y0 = NULL,
  z0 = NULL,
  x0 = NULL
)
```

Arguments

A	$n \times p$ design matrix.
b	n response vector.
group	Length-p group-membership vector: `group[j]` is the group id of column `j` of `A`. Group ids may be any atomic vector (integer, character, or factor) and need not be contiguous or sorted; groups are formed from `sort(unique(group))`.
lambda	Vector of penalty parameters or a single value. If `NULL`, a path is automatically generated from the largest lambda at which the all-zero solution is optimal down to that value times `lambda_min_ratio`. The path is fit using warm starts from larger to smaller lambda values.
nlambda	Number of lambda values to use when `lambda` is `NULL`. Default is 100.
lambda_min_ratio	Minimum ratio of the smallest to largest lambda when `lambda` is `NULL`. Default is 1e-4.
alpha	Determines the relative weight of the ℓ_1 and group ℓ_2 penalty. Must be in $[0, 1]$. Default is `0.05`.
pfgroup	Penalty factor for each group in the group lasso, ordered by `sort(unique(group))`. Default is `sqrt(table(group))`, the standard group-lasso convention (Yuan & Lin 2006).
intercept	Centers mean function at 0 through $y - \bar{y}$. Default is `TRUE`.
standardize	Whether to standardize the columns of A to have unit norm. Default is `TRUE`.
stoptol	Tolerance for stopping criteria. Default is `1e-6`.
stopopt	Stopping criteria. 1: relative duality gap and feasibility, 2: KKT conditions, 3: dual feasibility and relative duality gap, 4: dual feasibility and absolute duality gap. Default is `2L`.
verbose	How much to print to the console: `0` is silent; `1` (default) shows a compact text progress bar tracking the lambda path; `2` prints full per-lambda SSNAL diagnostics (objective, feasibility, iteration counts, ...) instead of the progress bar; `3` additionally prints the inner semismooth-Newton subproblem's iteration trace.

maxit	Maximum number of iterations. Default is 5000L.
Lip	Lipschitz constant for the step size. Automatically computed as the maximum eigenvalue of AA' if NULL.
y0	optional initialization vector for dual variable y
z0	optional initialization vector for dual variable z
x0	optional initialization vector for primal variable x

Value

List of class sglssnal containing the following components:

- obj: Matrix containing primal and dual objective values for each lambda.
- x: Matrix of primal variables, with each column corresponding to a lambda.
- x0: Vector of intercept values for each lambda.
- y: Matrix of y dual variables, with each column corresponding to a lambda.
- z: Matrix of z dual variables, with each column corresponding to a lambda.
- info: List containing information about the optimization process for each lambda.
- lambda: Vector of lambda values used.

References

Zhang, Y., Zhang, N., Sun, D., & Toh, K. C. (2020). *An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems.* Mathematical Programming, 179, 223-263. doi:10.1007/s1010701813296.

Examples

```
set.seed(1)
n <- 50
p <- 20
A <- matrix(rnorm(n * p), n, p)
bstar <- c(rnorm(5), rep(0, p - 5))
b <- as.numeric(A %*% bstar + rnorm(n, sd = 0.1))
group <- rep(1:4, each = 5)

fit <- sglssnal(A, b, group, lambda = 0.5, alpha = 0.5)
coef(fit)
```

Index