Package ‘sgs’

September 18, 2024

Title Sparse-Group SLOPE: Adaptive Bi-Level Selection with FDR Control

Version 0.3.0

Date 2024-09-18

Maintainer Fabio Feser <ff120@ic.ac.uk>

Description Implementation of Sparse-group SLOPE (SGS) (Feser and Evangelou (2023) <doi:10.48550/arXiv.2305.09467>) models. Linear and logistic regression models are supported, both of which can be fit using k-fold cross-validation. Dense and sparse input matrices are supported. In addition, a general adaptive three operator splitting (ATOS) implementation is provided. Group SLOPE (gSLOPE) (Brzyski et al. (2019) <doi:10.1080/01621459.2017.1411269>) and group-based OSCAR models (Feser and Evangelou (2024) <doi:10.48550/arXiv.2405.15357>) are also implemented. All models are available with strong screening rules (Feser and Evangelou (2024) <doi:10.48550/arXiv.2405.15357>) for computational speed-up.

Imports Matrix, MASS, caret, grDevices, graphics, methods, stats, faux, SLOPE, Rlab, Rcpp (>= 1.0.10)

LinkingTo Rcpp, RcppArmadillo

Suggests SGL, gglasso, glmnet, testthat, knitr, grpSLOPE, rmarkdown

RoxygenNote 7.3.1

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/ff1201/sgs

BugReports https://github.com/ff1201/sgs/issues

VignetteBuilder knitr

NeedsCompilation yes

Author Fabio Feser [aut, cre] (<https://orcid.org/0009-0007-3088-9727>)

Repository CRAN

Date/Publication 2024-09-18 21:20:02 UTC

Contents

`arma_mv` *Matrix Product in RcppArmadillo.*

Description

Matrix Product in RcppArmadillo.

Usage

```
arma_mv(m, v)
```

Arguments

`m`	numeric matrix
`v`	numeric vector

Value

matrix product of m and v

`arma_sparse` *Matrix Product in RcppArmadillo.*

Description

Matrix Product in RcppArmadillo.

Usage

```
arma_sparse(m, v)
```

Arguments

`m`	numeric sparse matrix
`v`	numeric vector

Value

matrix product of m and v

`as_sgs` *Fits the adaptively scaled SGS model (AS-SGS).*

Description

Fits an SGS model using the noise estimation procedure, termed adaptively scaled SGS (Algorithm 2 from Feser and Evangelou (2023)). This adaptively estimates λ and then fits the model using the estimated value. It is an alternative approach to cross-validation (`fit_sgs_cv()`). The approach is only compatible with the SGS penalties.

Usage

```
as_sgs(
  X,
  y,
  groups,
  type = "linear",
  pen_method = 2,
  alpha = 0.95,
  vFDR = 0.1,
  gFDR = 0.1,
  standardise = "l2",
  intercept = TRUE,
  verbose = FALSE
)
```

Arguments

X — Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class "sparseMatrix" from the Matrix package).

y — Output vector of dimension n. For type="linear" should be continuous and for type="logistic" should be a binary variable.

groups — A grouping structure for the input data. Should take the form of a vector of group indices.

type — The type of regression to perform. Supported values are: "linear" and "logistic".

pen_method — The type of penalty sequences to use.

- "1" uses the vMean and gMean SGS sequences.
- "2" uses the vMax and gMax SGS sequences.

alpha — The value of α, which defines the convex balance between SLOPE and gSLOPE. Must be between 0 and 1.

vFDR — Defines the desired variable false discovery rate (FDR) level, which determines the shape of the variable penalties. Must be between 0 and 1.

gFDR — Defines the desired group false discovery rate (FDR) level, which determines the shape of the group penalties. Must be between 0 and 1.

standardise — Type of standardisation to perform on X:

- "l2" standardises the input data to have ℓ_2 norms of one.
- "l1" standardises the input data to have ℓ_1 norms of one.
- "sd" standardises the input data to have standard deviation of one.
- "none" no standardisation applied.

intercept — Logical flag for whether to fit an intercept.

verbose — Logical flag for whether to print fitting information.

Value

An object of type "sgs" containing model fit information (see fit_sgs()).

References

Feser, F., Evangelou, M. (2023). *Sparse-group SLOPE: adaptive bi-level selection with FDR-control*, https://arxiv.org/abs/2305.09467

See Also

scaled_sgs()

Other model-selection: fit_goscar_cv(), fit_gslope_cv(), fit_sgo_cv(), fit_sgs_cv(), scaled_sgs()

Other SGS-methods: coef.sgs(), fit_sgo(), fit_sgo_cv(), fit_sgs(), fit_sgs_cv(), plot.sgs(), predict.sgs(), print.sgs(), scaled_sgs()

atos *Adaptive three operator splitting (ATOS).*

Description

Function for fitting adaptive three operator splitting (ATOS) with general convex penalties. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
atos(
  X,
  y,
  type = "linear",
  prox_1,
  prox_2,
  pen_prox_1 = 0.5,
  pen_prox_2 = 0.5,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  prox_1_opts = NULL,
  prox_2_opts = NULL,
  standardise = "l2",
  intercept = TRUE,
  x0 = NULL,
  u = NULL,
  verbose = FALSE
)
```

Arguments

`X`	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class `"sparseMatrix"` from the `Matrix` package)
`y`	Output vector of dimension n. For `type="linear"` needs to be continuous and for `type="logistic"` needs to be a binary variable.
`type`	The type of regression to perform. Supported values are: `"linear"` and `"logistic"`.
`prox_1`	The proximal operator for the first function, $h(x)$.
`prox_2`	The proximal operator for the second function, $g(x)$.
`pen_prox_1`	The penalty for the first proximal operator. For the lasso, this would be the sparsity parameter, λ. If operator does not include a penalty, set to 1.
`pen_prox_2`	The penalty for the second proximal operator.
`max_iter`	Maximum number of ATOS iterations to perform.
`backtracking`	The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

max_iter_backtracking
: Maximum number of backtracking line search iterations to perform per global iteration.

tol
: Convergence tolerance for the stopping criteria.

prox_1_opts
: Optional argument for first proximal operator. For the group lasso, this would be the group IDs. Note: this must be inserted as a list.

prox_2_opts
: Optional argument for second proximal operator.

standardise
: Type of standardisation to perform on X:
  - "l2" standardises the input data to have ℓ_2 norms of one.
  - "l1" standardises the input data to have ℓ_1 norms of one.
  - "sd" standardises the input data to have standard deviation of one.
  - "none" no standardisation applied.

intercept
: Logical flag for whether to fit an intercept.

x0
: Optional initial vector for x_0.

u
: Optional initial vector for u.

verbose
: Logical flag for whether to print fitting information.

Details

atos() solves convex minimization problems of the form

$$f(x) + g(x) + h(x),$$

where f is convex and differentiable with L_f-Lipschitz gradient, and g and h are both convex. The algorithm is not symmetrical, but usually the difference between variations are only small numerical values, which are filtered out. However, both variations should be checked regardless, by looking at x and u. An example for the sparse-group lasso (SGL) is given.

Value

An object of class "atos" containing:

beta
: The fitted values from the regression. Taken to be the more stable fit between x and u, which is usually the former.

x
: The solution to the original problem (see Pedregosa and Gidel (2018)).

u
: The solution to the dual problem (see Pedregosa and Gidel (2018)).

z
: The updated values from applying the first proximal operator (see Pedregosa and Gidel (2018)).

type
: Indicates which type of regression was performed.

success
: Logical flag indicating whether ATOS converged, according to tol.

num_it
: Number of iterations performed. If convergence is not reached, this will be max_iter.

certificate
: Final value of convergence criteria.

intercept
: Logical flag indicating whether an intercept was fit.

References

Pedregosa, F., Gidel, G. (2018). *Adaptive Three Operator Splitting*, https://proceedings.mlr.press/v80/pedregosa18a.html

---

coef.sgs — *Extracts coefficients for one of the following object types:* "sgs", "sgs_cv", "gslope", "gslope_cv".

Description

Print the coefficients using model fitted with one of the following functions: fit_sgs(), fit_sgs_cv(), fit_gslope(), fit_gslope_cv(), fit_sgo(), fit_sgo_cv(), fit_goscar(), fit_goscar_cv(). The predictions are calculated for each "lambda" value in the path.

Usage

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

Arguments

object	Object of one of the following classes: "sgs", "sgs_cv", "gslope", "gslope_cv".
...	further arguments passed to stats function.

Value

The fitted coefficients

See Also

fit_sgs(), fit_sgs_cv(), fit_gslope(), fit_gslope_cv()

Other SGS-methods: as_sgs(), fit_sgo(), fit_sgo_cv(), fit_sgs(), fit_sgs_cv(), plot.sgs(), predict.sgs(), print.sgs(), scaled_sgs()

Other gSLOPE-methods: fit_goscar(), fit_goscar_cv(), fit_gslope(), fit_gslope_cv(), plot.sgs(), predict.sgs(), print.sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run SGS
model = fit_sgs(X = data$X, y = data$y, groups = groups, type="linear", lambda = 1, alpha=0.95,
vFDR=0.1, gFDR=0.1, standardise = "l2", intercept = TRUE, verbose=FALSE)
# use predict function
model_coef = coef(model)
```

fit_goscar — *Fit a gOSCAR model.*

Description

Group OSCAR (gOSCAR) main fitting function. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_goscar(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  screen = TRUE,
  verbose = FALSE,
  w_weights = NULL
)
```

Arguments

`X`	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class `"sparseMatrix"` from the `Matrix` package).
`y`	Output vector of dimension n. For `type="linear"` should be continuous and for `type="logistic"` should be a binary variable.
`groups`	A grouping structure for the input data. Should take the form of a vector of group indices.
`type`	The type of regression to perform. Supported values are: `"linear"` and `"logistic"`.
`lambda`	The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models: • `"path"` computes a path of regularisation parameters of length `"path_length"`. The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by `min_frac`. • User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned `"lambda"` value will be the original unscaled value(s).

path_length
: The number of λ values to fit the model for. If "lambda" is user-specified, this is ignored.

min_frac
: Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "min_frac" of the first λ value.

max_iter
: Maximum number of ATOS iterations to perform.

backtracking
: The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

max_iter_backtracking
: Maximum number of backtracking line search iterations to perform per global iteration.

tol
: Convergence tolerance for the stopping criteria.

standardise
: Type of standardisation to perform on X:
  - "l2" standardises the input data to have ℓ_2 norms of one. When using this "lambda" is scaled internally by $1/\sqrt{n}$.
  - "l1" standardises the input data to have ℓ_1 norms of one. When using this "lambda" is scaled internally by $1/n$.
  - "sd" standardises the input data to have standard deviation of one.
  - "none" no standardisation applied.

intercept
: Logical flag for whether to fit an intercept.

screen
: Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.

verbose
: Logical flag for whether to print fitting information.

w_weights
: Optional vector for the group penalty weights. Overrides the OSCAR penalties when specified. When entering custom weights, these are multiplied internally by λ. To void this behaviour, set $\lambda = 1$.

Details

fit_goscar() fits a gOSCAR model (Feser and Evangelou (2024)) using adaptive three operator splitting (ATOS). gOSCAR uses the same model set-up as for gSLOPE, but with different weights (see Bao et. al. (2020) and Feser and Evangelou (2024)). The penalties are given by (for a group g with m groups):

$$w_g = \sigma_1 + \sigma_3(m - g),$$

where

$$\sigma_1 = d_i \|X^\intercal y\|_\infty,\ \sigma_3 = \sigma_1/m.$$

Value

A list containing:

beta
: The fitted values from the regression. Taken to be the more stable fit between x and z, which is usually the former. A filter is applied to remove very small values, where ATOS has not been able to shrink exactly to zero. Check this against x and z.

group_effects	The group values from the regression. Taken by applying the ℓ_2 norm within each group on beta.
selected_var	A list containing the indicies of the active/selected variables for each "lambda" value.
selected_grp	A list containing the indicies of the active/selected groups for each "lambda" value.
pen_gslope	Vector of the group penalty sequence.
lambda	Value(s) of λ used to fit the model.
type	Indicates which type of regression was performed.
standardise	Type of standardisation used.
intercept	Logical flag indicating whether an intercept was fit.
num_it	Number of iterations performed. If convergence is not reached, this will be max_iter.
success	Logical flag indicating whether ATOS converged, according to tol.
certificate	Final value of convergence criteria.
x	The solution to the original problem (see Pedregosa and Gidel (2018)).
u	The solution to the dual problem (see Pedregosa and Gidel (2018)).
z	The updated values from applying the first proximal operator (see Pedregosa and Gidel (2018)).
screen_set	List of groups that were kept after screening step for each "lambda" value. (corresponds to $\mathcal{S}$ in Feser and Evangelou (2024)).
epsilon_set	List of groups that were used for fitting after screening for each "lambda" value. (corresponds to $\mathcal{E}$ in Feser and Evangelou (2024)).
kkt_violations	List of groups that violated the KKT conditions each "lambda" value. (corresponds to $\mathcal{K}$ in Feser and Evangelou (2024)).
screen	Logical flag indicating whether screening was applied.

References

Bao, R., Gu B., Huang, H. (2020). *Fast OSCAR and OWL Regression via Safe Screening Rules*, https://proceedings.mlr.press/v119/bao20b

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://proceedings.mlr.press/v80/pedregosa18a.html

Pedregosa, F., Gidel, G. (2018). *Adaptive Three Operator Splitting*, https://proceedings.mlr.press/v80/pedregosa18a.html

See Also

Other gSLOPE-methods: coef.sgs(), fit_goscar_cv(), fit_gslope(), fit_gslope_cv(), plot.sgs(), predict.sgs(), print.sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run gOSCAR
model = fit_goscar(X = data$X, y = data$y, groups = groups, type="linear", path_length = 5,
standardise = "l2", intercept = TRUE, verbose=FALSE)
```

`fit_goscar_cv` *Fit a gOSCAR model using k-fold cross-validation.*

Description

Function to fit a pathwise solution of group OSCAR (gOSCAR) models using k-fold cross-validation. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_goscar_cv(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  nfolds = 10,
  backtracking = 0.7,
  max_iter = 5000,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  error_criteria = "mse",
  screen = TRUE,
  verbose = FALSE,
  w_weights = NULL
)
```

Arguments

X	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class "sparseMatrix" from the Matrix package).
y	Output vector of dimension n. For type="linear" should be continuous and for type="logistic" should be a binary variable.

groups
: A grouping structure for the input data. Should take the form of a vector of group indices.

type
: The type of regression to perform. Supported values are: "linear" and "logistic".

lambda
: The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:
  - "path" computes a path of regularisation parameters of length "path_length". The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by "min_frac".
  - User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned "lambda" value will be the original unscaled value(s).

path_length
: The number of λ values to fit the model for. If "lambda" is user-specified, this is ignored.

min_frac
: Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "min_frac" of the first λ value.

nfolds
: The number of folds to use in cross-validation.

backtracking
: The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

max_iter
: Maximum number of ATOS iterations to perform.

max_iter_backtracking
: Maximum number of backtracking line search iterations to perform per global iteration.

tol
: Convergence tolerance for the stopping criteria.

standardise
: Type of standardisation to perform on X:
  - "l2" standardises the input data to have ℓ_2 norms of one.
  - "l1" standardises the input data to have ℓ_1 norms of one.
  - "sd" standardises the input data to have standard deviation of one.
  - "none" no standardisation applied.

intercept
: Logical flag for whether to fit an intercept.

error_criteria
: The criteria used to discriminate between models along the path. Supported values are: "mse" (mean squared error) and "mae" (mean absolute error).

screen
: Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.

verbose
: Logical flag for whether to print fitting information.

w_weights
: Optional vector for the group penalty weights. Overrides the OSCAR penalties when specified. When entering custom weights, these are multiplied internally by λ. To void this behaviour, set $\lambda = 1$.

Details

Fits gOSCAR models under a pathwise solution using adaptive three operator splitting (ATOS), picking the 1se model as optimum. Warm starts are implemented.

Value

A list containing:

`errors`	A table containing fitting information about the models on the path.
`all_models`	Fitting information for all models fit on the path, which is a `"gslope"` object type.
`fit`	The 1se chosen model, which is a `"gslope"` object type.
`best_lambda`	The value of λ which generated the chosen model.
`best_lambda_id`	The path index for the chosen model.

References

Bao, R., Gu B., Huang, H. (2020). *Fast OSCAR and OWL Regression via Safe Screening Rules*, https://proceedings.mlr.press/v119/bao20b

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://proceedings.mlr.press/v80/pedregosa18a.html

See Also

`fit_goscar()`

Other gSLOPE-methods: `coef.sgs()`, `fit_goscar()`, `fit_gslope()`, `fit_gslope_cv()`, `plot.sgs()`, `predict.sgs()`, `print.sgs()`

Other model-selection: `as_sgs()`, `fit_gslope_cv()`, `fit_sgo_cv()`, `fit_sgs_cv()`, `scaled_sgs()`

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run gOSCAR with cross-validation
cv_model = fit_goscar_cv(X = data$X, y = data$y, groups=groups, type = "linear", path_length = 5,
nfolds=5, min_frac = 0.05, standardise="l2",intercept=TRUE,verbose=TRUE)
```

`fit_gslope` *Fit a gSLOPE model.*

Description

Group SLOPE (gSLOPE) main fitting function. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_gslope(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  gFDR = 0.1,
  pen_method = 1,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  screen = TRUE,
  verbose = FALSE,
  w_weights = NULL
)
```

Arguments

`X` Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class `"sparseMatrix"` from the `Matrix` package).

`y` Output vector of dimension n. For `type="linear"` should be continuous and for `type="logistic"` should be a binary variable.

`groups` A grouping structure for the input data. Should take the form of a vector of group indices.

`type` The type of regression to perform. Supported values are: `"linear"` and `"logistic"`.

`lambda` The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:

- `"path"` computes a path of regularisation parameters of length `"path_length"`. The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by `"min_frac"`.
- User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned `"lambda"` value will be the original unscaled value(s).

`path_length` The number of λ values to fit the model for. If `"lambda"` is user-specified, this is ignored.

`min_frac` Smallest value of λ as a fraction of the maximum value. That is, the final λ will be `"min_frac"` of the first λ value.

`gFDR` Defines the desired group false discovery rate (FDR) level, which determines the shape of the group penalties. Must be between 0 and 1.

`pen_method` The type of penalty sequences to use (see Brzyski et al. (2019)):

- `"1"` uses the gMean gSLOPE sequence.
- `"2"` uses the gMax gSLOPE sequence.

`max_iter` Maximum number of ATOS iterations to perform.

`backtracking` The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

`max_iter_backtracking`
Maximum number of backtracking line search iterations to perform per global iteration.

`tol` Convergence tolerance for the stopping criteria.

`standardise` Type of standardisation to perform on `X`:

- `"l2"` standardises the input data to have ℓ_2 norms of one. When using this `"lambda"` is scaled internally by $1/\sqrt{n}$.
- `"l1"` standardises the input data to have ℓ_1 norms of one. When using this `"lambda"` is scaled internally by $1/n$.
- `"sd"` standardises the input data to have standard deviation of one.
- `"none"` no standardisation applied.

`intercept` Logical flag for whether to fit an intercept.

`screen` Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.

`verbose` Logical flag for whether to print fitting information.

`w_weights` Optional vector for the group penalty weights. Overrides the penalties from `pen_method` if specified. When entering custom weights, these are multiplied internally by λ. To void this behaviour, set $\lambda = 1$.

Details

`fit_gslope()` fits a gSLOPE model (Brzyski et al. (2019)) using adaptive three operator splitting (ATOS). gSLOPE is a sparse-group method, so that it selects both variables and groups. Unlike group selection approaches, not every variable within a group is set as active. It solves the convex optimisation problem given by

$$\frac{1}{2n} f(b; y, \mathbf{X}) + \lambda \sum_{g=1}^{m} w_g \sqrt{p_g} \|b^{(g)}\|_2,$$

where the penalty sequences are sorted and $f(\cdot)$ is the loss function. In the case of the linear model, the loss function is given by the mean-squared error loss:

$$f(b; y, \mathbf{X}) = \|y - \mathbf{X}b\|_2^2 .$$

In the logistic model, the loss function is given by

$$f(b; y, \mathbf{X}) = -1/n \log(\mathcal{L}(b; y, \mathbf{X})).$$

where the log-likelihood is given by

$$\mathcal{L}(b; y, \mathbf{X}) = \sum_{i=1}^{n} \left\{ y_i b^\intercal x_i - \log(1 + \exp(b^\intercal x_i)) \right\} .$$

The penalty parameters in gSLOPE are sorted so that the largest group effects are matched with the largest penalties, to reduce the group FDR. The gMean sequence (pen_method=1) is given by

$$w_i^{\text{mean}} = \overline{F}_{\chi_{p_j}}^{-1}(1 - q_g i/m),\ i = 1, \dots, m, \text{where } \overline{F}_{\chi_{p_j}}(x) := \frac{1}{m}\sum_{j=1}^{m} F_{\chi_{p_j}}(\sqrt{p_j}x),$$

where $F_{\chi_{p_j}}$ is the cumulative distribution function of a χ distribution with p_j degrees of freedom. The gMax sequence (pen_method=2) is given by

$$w_i^{\max} = \max_{j=1,\dots,m}\left\{\frac{1}{\sqrt{p_j}} F_{\chi_{p_j}}^{-1}\left(1 - \frac{q_g i}{m}\right)\right\},$$

where $F_{\chi_{p_j}}$ is the cumulative distribution function of a χ distribution with p_j degrees of freedom.

Value

A list containing:

beta	The fitted values from the regression. Taken to be the more stable fit between x and z, which is usually the former. A filter is applied to remove very small values, where ATOS has not been able to shrink exactly to zero. Check this against x and z.
group_effects	The group values from the regression. Taken by applying the ℓ_2 norm within each group on beta.
selected_var	A list containing the indicies of the active/selected variables for each "lambda" value.
selected_grp	A list containing the indicies of the active/selected groups for each "lambda" value.
pen_gslope	Vector of the group penalty sequence.
lambda	Value(s) of λ used to fit the model.
type	Indicates which type of regression was performed.
standardise	Type of standardisation used.
intercept	Logical flag indicating whether an intercept was fit.
num_it	Number of iterations performed. If convergence is not reached, this will be max_iter.
success	Logical flag indicating whether ATOS converged, according to tol.
certificate	Final value of convergence criteria.
x	The solution to the original problem (see Pedregosa and Gidel (2018)).
u	The solution to the dual problem (see Pedregosa and Gidel (2018)).
z	The updated values from applying the first proximal operator (see Pedregosa and Gidel (2018)).
screen_set	List of groups that were kept after screening step for each "lambda" value. (corresponds to $\mathcal{S}$ in Feser and Evangelou (2024)).
epsilon_set	List of groups that were used for fitting after screening for each "lambda" value. (corresponds to $\mathcal{E}$ in Feser and Evangelou (2024)).
kkt_violations	List of groups that violated the KKT conditions each "lambda" value. (corresponds to $\mathcal{K}$ in Feser and Evangelou (2024)).
screen	Logical flag indicating whether screening was applied.

References

Brzyski, D., Gossmann, A., Su, W., Bodgan, M. (2019). *Group SLOPE – Adaptive Selection of Groups of Predictors*, https://www.tandfonline.com/doi/full/10.1080/01621459.2017.1411269

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://proceedings.mlr.press/v80/pedregosa18a.html

Pedregosa, F., Gidel, G. (2018). *Adaptive Three Operator Splitting*, https://proceedings.mlr.press/v80/pedregosa18a.html

See Also

Other gSLOPE-methods: coef.sgs(), fit_goscar(), fit_goscar_cv(), fit_gslope_cv(), plot.sgs(), predict.sgs(), print.sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run gSLOPE
model = fit_gslope(X = data$X, y = data$y, groups = groups, type="linear", path_length = 5,
gFDR=0.1, standardise = "l2", intercept = TRUE, verbose=FALSE)
```

---

fit_gslope_cv *Fit a gSLOPE model using k-fold cross-validation.*

Description

Function to fit a pathwise solution of group SLOPE (gSLOPE) models using k-fold cross-validation. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_gslope_cv(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  nfolds = 10,
  gFDR = 0.1,
  pen_method = 1,
  backtracking = 0.7,
  max_iter = 5000,
```

```
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  error_criteria = "mse",
  screen = TRUE,
  verbose = FALSE,
  w_weights = NULL
)
```

Arguments

`X` Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class "sparseMatrix" from the Matrix package).

`y` Output vector of dimension n. For type="linear" should be continuous and for type="logistic" should be a binary variable.

`groups` A grouping structure for the input data. Should take the form of a vector of group indices.

`type` The type of regression to perform. Supported values are: "linear" and "logistic".

`lambda` The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:

- "path" computes a path of regularisation parameters of length "path_length". The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by "min_frac".
- User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned "lambda" value will be the original unscaled value(s).

`path_length` The number of λ values to fit the model for. If "lambda" is user-specified, this is ignored.

`min_frac` Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "min_frac" of the first λ value.

`nfolds` The number of folds to use in cross-validation.

`gFDR` Defines the desired group false discovery rate (FDR) level, which determines the shape of the penalties. Must be between 0 and 1.

`pen_method` The type of penalty sequences to use (see Brzyski et al. (2019)):

- "1" uses the gMean gSLOPE sequence.
- "2" uses the gMax gSLOPE sequence.

`backtracking` The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

`max_iter` Maximum number of ATOS iterations to perform.

`max_iter_backtracking`
Maximum number of backtracking line search iterations to perform per global iteration.

`tol` Convergence tolerance for the stopping criteria.

`standardise` Type of standardisation to perform on X:

- "l2" standardises the input data to have ℓ_2 norms of one.
- "l1" standardises the input data to have ℓ_1 norms of one.
- "sd" standardises the input data to have standard deviation of one.
- "none" no standardisation applied.

`intercept`	Logical flag for whether to fit an intercept.
`error_criteria`	The criteria used to discriminate between models along the path. Supported values are: "mse" (mean squared error) and "mae" (mean absolute error).
`screen`	Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.
`verbose`	Logical flag for whether to print fitting information.
`w_weights`	Optional vector for the group penalty weights. Overrides the penalties from `pen_method` if specified. When entering custom weights, these are multiplied internally by λ. To void this behaviour, set $\lambda = 1$.

Details

Fits gSLOPE models under a pathwise solution using adaptive three operator splitting (ATOS), picking the 1se model as optimum. Warm starts are implemented.

Value

A list containing:

`errors`	A table containing fitting information about the models on the path.
`all_models`	Fitting information for all models fit on the path, which is a "gslope" object type.
`fit`	The 1se chosen model, which is a "gslope" object type.
`best_lambda`	The value of λ which generated the chosen model.
`best_lambda_id`	The path index for the chosen model.

References

Brzyski, D., Gossmann, A., Su, W., Bodgan, M. (2019). *Group SLOPE – Adaptive Selection of Groups of Predictors*, https://www.tandfonline.com/doi/full/10.1080/01621459.2017.1411269

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://proceedings.mlr.press/v80/pedregosa18a.html

See Also

`fit_gslope()`

Other gSLOPE-methods: `coef.sgs()`, `fit_goscar()`, `fit_goscar_cv()`, `fit_gslope()`, `plot.sgs()`, `predict.sgs()`, `print.sgs()`

Other model-selection: `as_sgs()`, `fit_goscar_cv()`, `fit_sgo_cv()`, `fit_sgs_cv()`, `scaled_sgs()`

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run gSLOPE with cross-validation
cv_model = fit_gslope_cv(X = data$X, y = data$y, groups=groups, type = "linear", path_length = 5,
nfolds=5, gFDR = 0.1, min_frac = 0.05, standardise="l2",intercept=TRUE,verbose=TRUE)
```

---

fit_sgo *Fit an SGO model.*

Description

Sparse-group OSCAR (SGO) main fitting function. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_sgo(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  alpha = 0.95,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  screen = TRUE,
  verbose = FALSE,
  w_weights = NULL,
  v_weights = NULL
)
```

Arguments

X	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class "sparseMatrix" from the Matrix package).
y	Output vector of dimension n. For type="linear" should be continuous and for type="logistic" should be a binary variable.

`groups` A grouping structure for the input data. Should take the form of a vector of group indices.

`type` The type of regression to perform. Supported values are: "`linear`" and "`logistic`".

`lambda` The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:

- "`path`" computes a path of regularisation parameters of length "`path_length`". The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by "`min_frac`".
- User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned "`lambda`" value will be the original unscaled value(s).

`path_length` The number of λ values to fit the model for. If "`lambda`" is user-specified, this is ignored.

`min_frac` Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "`min_frac`" of the first λ value.

`alpha` The value of α, which defines the convex balance between OSCAR and gOSCAR. Must be between 0 and 1. Recommended value is 0.95.

`max_iter` Maximum number of ATOS iterations to perform.

`backtracking` The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

`max_iter_backtracking`
Maximum number of backtracking line search iterations to perform per global iteration.

`tol` Convergence tolerance for the stopping criteria.

`standardise` Type of standardisation to perform on `X`:

- "`l2`" standardises the input data to have ℓ_2 norms of one. When using this "`lambda`" is scaled internally by $1/\sqrt{n}$.
- "`l1`" standardises the input data to have ℓ_1 norms of one. When using this "`lambda`" is scaled internally by $1/n$.
- "`sd`" standardises the input data to have standard deviation of one.
- "`noBaone`" no standardisation applied.

`intercept` Logical flag for whether to fit an intercept.

`screen` Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.

`verbose` Logical flag for whether to print fitting information.

`w_weights` Optional vector for the group penalty weights. Overrides the OSCAR penalties when specified. When entering custom weights, these are multiplied internally by λ and $1-\alpha$. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$.

`v_weights` Optional vector for the variable penalty weights. Overrides the OSCAR penalties when specified. When entering custom weights, these are multiplied internally by λ and α. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$.

Details

`fit_sgo()` fits an SGO model (Feser and Evangelou (2024)) using adaptive three operator splitting (ATOS). SGO uses the same model set-up as for SGS, but with different weights (see Bao et. al. (2020) and Feser and Evangelou (2024)). The penalties are given by (for a group g and variable i, with p variables and m groups):

$$v_i = \sigma_1 + \sigma_2(p-i),\ w_g = \sigma_1 + \sigma_3(m-g),$$

where

$$\sigma_1 = d_i\|X^\intercal y\|_\infty,\ \sigma_2 = \sigma_1/p,\ \sigma_3 = \sigma_1/m,\ d_i = i \times \exp(-2).$$

Value

A list containing:

`beta`	The fitted values from the regression. Taken to be the more stable fit between `x` and `z`, which is usually the former. A filter is applied to remove very small values, where ATOS has not been able to shrink exactly to zero. Check this against `x` and `z`.
`x`	The solution to the original problem (see Pedregosa and Gidel (2018)).
`u`	The solution to the dual problem (see Pedregosa and Gidel (2018)).
`z`	The updated values from applying the first proximal operator (see Pedregosa and Gidel (2018)).
`type`	Indicates which type of regression was performed.
`pen_slope`	Vector of the variable penalty sequence.
`pen_gslope`	Vector of the group penalty sequence.
`lambda`	Value(s) of λ used to fit the model.
`success`	Logical flag indicating whether ATOS converged, according to `tol`.
`num_it`	Number of iterations performed. If convergence is not reached, this will be `max_iter`.
`certificate`	Final value of convergence criteria.
`intercept`	Logical flag indicating whether an intercept was fit.

References

Bao, R., Gu B., Huang, H. (2020). *Fast OSCAR and OWL Regression via Safe Screening Rules*, https://proceedings.mlr.press/v119/bao20b

Feser, F., Evangelou, M. (2023). *Sparse-group SLOPE: adaptive bi-level selection with FDR-control*, https://arxiv.org/abs/2305.09467

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://arxiv.org/abs/2405.15357

Pedregosa, F., Gidel, G. (2018). *Adaptive Three Operator Splitting*, https://proceedings.mlr.press/v80/pedregosa18a.html

See Also

Other SGS-methods: as_sgs(), coef.sgs(), fit_sgo_cv(), fit_sgs(), fit_sgs_cv(), plot.sgs(), predict.sgs(), print.sgs(), scaled_sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run SGO
model = fit_sgo(X = data$X, y = data$y, groups = groups, type="linear", path_length = 5,
alpha=0.95, standardise = "l2", intercept = TRUE, verbose=FALSE)
```

---

fit_sgo_cv *Fit an SGO model using k-fold cross-validation.*

Description

Function to fit a pathwise solution of sparse-group SLOPE (SGO) models using k-fold cross-validation. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_sgo_cv(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  alpha = 0.95,
  nfolds = 10,
  backtracking = 0.7,
  max_iter = 5000,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  error_criteria = "mse",
  screen = TRUE,
  verbose = FALSE,
  v_weights = NULL,
  w_weights = NULL
)
```

Arguments

X
: Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class "sparseMatrix" from the Matrix package).

y
: Output vector of dimension n. For type="linear" should be continuous and for type="logistic" should be a binary variable.

groups
: A grouping structure for the input data. Should take the form of a vector of group indices.

type
: The type of regression to perform. Supported values are: "linear" and "logistic".

lambda
: The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:
  - "path" computes a path of regularisation parameters of length "path_length". The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by "min_frac".
  - User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned "lambda" value will be the original unscaled value(s).

path_length
: The number of λ values to fit the model for. If "lambda" is user-specified, this is ignored.

min_frac
: Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "min_frac" of the first λ value.

alpha
: The value of α, which defines the convex balance between OSCAR and gOSCAR. Must be between 0 and 1. Recommended value is 0.95.

nfolds
: The number of folds to use in cross-validation.

backtracking
: The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

max_iter
: Maximum number of ATOS iterations to perform.

max_iter_backtracking
: Maximum number of backtracking line search iterations to perform per global iteration.

tol
: Convergence tolerance for the stopping criteria.

standardise
: Type of standardisation to perform on X:
  - "l2" standardises the input data to have ℓ_2 norms of one.
  - "l1" standardises the input data to have ℓ_1 norms of one.
  - "sd" standardises the input data to have standard deviation of one.
  - "none" no standardisation applied.

intercept
: Logical flag for whether to fit an intercept.

error_criteria
: The criteria used to discriminate between models along the path. Supported values are: "mse" (mean squared error) and "mae" (mean absolute error).

screen
: Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.

verbose
: Logical flag for whether to print fitting information.

v_weights	Optional vector for the variable penalty weights. Overrides the OSCAR penalties when specified. When entering custom weights, these are multiplied internally by λ and α. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$.
w_weights	Optional vector for the group penalty weights. Overrides the OSCAR penalties when specified. When entering custom weights, these are multiplied internally by λ and $1 - \alpha$. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$.

Details

Fits SGO models under a pathwise solution using adaptive three operator splitting (ATOS), picking the 1se model as optimum. Warm starts are implemented.

Value

A list containing:

all_models	A list of all the models fitted along the path.
fit	The 1se chosen model, which is a "sgs" object type.
best_lambda	The value of λ which generated the chosen model.
best_lambda_id	The path index for the chosen model.
errors	A table containing fitting information about the models on the path.
type	Indicates which type of regression was performed.

References

Bao, R., Gu B., Huang, H. (2020). *Fast OSCAR and OWL Regression via Safe Screening Rules*, https://proceedings.mlr.press/v119/bao20b

Feser, F., Evangelou, M. (2023). *Sparse-group SLOPE: adaptive bi-level selection with FDR-control*, https://arxiv.org/abs/2305.09467

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://arxiv.org/abs/2405.15357

See Also

fit_sgo()

Other model-selection: as_sgs(), fit_goscar_cv(), fit_gslope_cv(), fit_sgs_cv(), scaled_sgs()

Other SGS-methods: as_sgs(), coef.sgs(), fit_sgo(), fit_sgs(), fit_sgs_cv(), plot.sgs(), predict.sgs(), print.sgs(), scaled_sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run SGO with cross-validation
cv_model = fit_sgo_cv(X = data$X, y = data$y, groups=groups, type = "linear",
path_length = 5, nfolds=5, alpha = 0.95, min_frac = 0.05,
standardise="l2",intercept=TRUE,verbose=TRUE)
```

fit_sgs *Fit an SGS model.*

Description

Sparse-group SLOPE (SGS) main fitting function. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_sgs(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  alpha = 0.95,
  vFDR = 0.1,
  gFDR = 0.1,
  pen_method = 1,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  screen = TRUE,
  verbose = FALSE,
  w_weights = NULL,
  v_weights = NULL
)
```

Arguments

`X`	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class `"sparseMatrix"` from the `Matrix` package).
`y`	Output vector of dimension n. For `type="linear"` should be continuous and for `type="logistic"` should be a binary variable.
`groups`	A grouping structure for the input data. Should take the form of a vector of group indices.
`type`	The type of regression to perform. Supported values are: `"linear"` and `"logistic"`.
`lambda`	The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:

- "path" computes a path of regularisation parameters of length "path_length". The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by "min_frac".
- User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned "lambda" value will be the original unscaled value(s).

path_length The number of λ values to fit the model for. If "lambda" is user-specified, this is ignored.

min_frac Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "min_frac" of the first λ value.

alpha The value of α, which defines the convex balance between SLOPE and gSLOPE. Must be between 0 and 1. Recommended value is 0.95.

vFDR Defines the desired variable false discovery rate (FDR) level, which determines the shape of the variable penalties. Must be between 0 and 1.

gFDR Defines the desired group false discovery rate (FDR) level, which determines the shape of the group penalties. Must be between 0 and 1.

pen_method The type of penalty sequences to use (see Feser and Evangelou (2023)):

- "1" uses the vMean SGS and gMean gSLOPE sequences.
- "2" uses the vMax SGS and gMean gSLOPE sequences.
- "3" uses the BH SLOPE and gMean gSLOPE sequences, also known as SGS Original.

max_iter Maximum number of ATOS iterations to perform.

backtracking The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).

max_iter_backtracking
Maximum number of backtracking line search iterations to perform per global iteration.

tol Convergence tolerance for the stopping criteria.

standardise Type of standardisation to perform on X:

- "l2" standardises the input data to have ℓ_2 norms of one. When using this "lambda" is scaled internally by $1/\sqrt{n}$.
- "l1" standardises the input data to have ℓ_1 norms of one. When using this "lambda" is scaled internally by $1/n$.
- "sd" standardises the input data to have standard deviation of one.
- "none" no standardisation applied.

intercept Logical flag for whether to fit an intercept.

screen Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.

verbose Logical flag for whether to print fitting information.

w_weights Optional vector for the group penalty weights. Overrides the penalties from pen_method if specified. When entering custom weights, these are multiplied internally by λ and $1-\alpha$. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$.

v_weights Optional vector for the variable penalty weights. Overrides the penalties from pen_method if specified. When entering custom weights, these are multiplied internally by λ and α. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$.

Details

`fit_sgs()` fits an SGS model (Feser and Evangelou (2023)) using adaptive three operator splitting (ATOS). SGS is a sparse-group method, so that it selects both variables and groups. Unlike group selection approaches, not every variable within a group is set as active. It solves the convex optimisation problem given by

$$\frac{1}{2n} f(b; y, \mathbf{X}) + \lambda\alpha \sum_{i=1}^{p} v_i |b|_{(i)} + \lambda(1-\alpha) \sum_{g=1}^{m} w_g \sqrt{p_g} \|b^{(g)}\|_2,$$

where the penalty sequences are sorted and $f(\cdot)$ is the loss function. In the case of the linear model, the loss function is given by the mean-squared error loss:

$$f(b; y, \mathbf{X}) = \|y - \mathbf{X}b\|_2^2 .$$

In the logistic model, the loss function is given by

$$f(b; y, \mathbf{X}) = -1/n \log(\mathcal{L}(b; y, \mathbf{X})).$$

where the log-likelihood is given by

$$\mathcal{L}(b; y, \mathbf{X}) = \sum_{i=1}^{n} \left\{ y_i b^\intercal x_i - \log(1 + \exp(b^\intercal x_i)) \right\} .$$

SGS can be seen to be a convex combination of SLOPE and gSLOPE, balanced through `alpha`, such that it reduces to SLOPE for `alpha = 0` and to gSLOPE for `alpha = 1`. The penalty parameters in SGS are sorted so that the largest coefficients are matched with the largest penalties, to reduce the FDR. For the group penalties, see `fit_gslope()`. For the variable penalties, the vMean SGS sequence (`pen_method=1`) (Feser and Evangelou (2023)) is given by

$$v_i^{\text{mean}} = \overline{F}_{\mathcal{N}}^{-1} \left(1 - \frac{q_v i}{2p} \right), \text{ where } \overline{F}_{\mathcal{N}}(x) := \frac{1}{m} \sum_{j=1}^{m} F_{\mathcal{N}} \left(\alpha x + \frac{1}{3}(1-\alpha) a_j w_j \right), \ i = 1, \dots, p,$$

where $F_{\mathcal{N}}$ is the cumulative distribution functions of a standard Gaussian distribution. The vMax SGS sequence (`pen_method=2`) (Feser and Evangelou (2023)) is given by

$$v_i^{\text{max}} = \max_{j=1,\dots,m} \left\{ \frac{1}{\alpha} F_{\mathcal{N}}^{-1} \left(1 - \frac{q_v i}{2p} \right) - \frac{1}{3\alpha}(1-\alpha) a_j w_j \right\},$$

The BH SLOPE sequence (`pen_method=3`) (Bogdan et. al. (2015)) is given by

$$v_i = z(1 - i q_v / 2p),$$

where z is the quantile function of a standard normal distribution.

Value

A list containing:

`beta` The fitted values from the regression. Taken to be the more stable fit between `x` and `z`, which is usually the former. A filter is applied to remove very small values, where ATOS has not been able to shrink exactly to zero. Check this against `x` and `z`.

x	The solution to the original problem (see Pedregosa and Gidel (2018)).
u	The solution to the dual problem (see Pedregosa and Gidel (2018)).
z	The updated values from applying the first proximal operator (see Pedregosa and Gidel (2018)).
type	Indicates which type of regression was performed.
pen_slope	Vector of the variable penalty sequence.
pen_gslope	Vector of the group penalty sequence.
lambda	Value(s) of λ used to fit the model.
success	Logical flag indicating whether ATOS converged, according to `tol`.
num_it	Number of iterations performed. If convergence is not reached, this will be `max_iter`.
certificate	Final value of convergence criteria.
intercept	Logical flag indicating whether an intercept was fit.

References

Bogdan, M., van den Berg, E., Sabatti, C., Candes, E. (2015). *SLOPE - Adaptive variable selection via convex optimization*, https://projecteuclid.org/journals/annals-of-applied-statistics/volume-9/issue-3/SLOPEAdaptive-variable-selection-via-convex-optimization/10.1214/15-AOAS842.full

Feser, F., Evangelou, M. (2023). *Sparse-group SLOPE: adaptive bi-level selection with FDR-control*, https://arxiv.org/abs/2305.09467

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://arxiv.org/abs/2405.15357

Pedregosa, F., Gidel, G. (2018). *Adaptive Three Operator Splitting*, https://proceedings.mlr.press/v80/pedregosa18a.html

See Also

Other SGS-methods: `as_sgs()`, `coef.sgs()`, `fit_sgo()`, `fit_sgo_cv()`, `fit_sgs_cv()`, `plot.sgs()`, `predict.sgs()`, `print.sgs()`, `scaled_sgs()`

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run SGS
model = fit_sgs(X = data$X, y = data$y, groups = groups, type="linear", path_length = 5,
alpha=0.95, vFDR=0.1, gFDR=0.1, standardise = "l2", intercept = TRUE, verbose=FALSE)
```

fit_sgs_cv *Fit an SGS model using k-fold cross-validation.*

Description

Function to fit a pathwise solution of sparse-group SLOPE (SGS) models using k-fold cross-validation. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
fit_sgs_cv(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  min_frac = 0.05,
  alpha = 0.95,
  vFDR = 0.1,
  gFDR = 0.1,
  pen_method = 1,
  nfolds = 10,
  backtracking = 0.7,
  max_iter = 5000,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  error_criteria = "mse",
  screen = TRUE,
  verbose = FALSE,
  v_weights = NULL,
  w_weights = NULL
)
```

Arguments

`X`	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class `"sparseMatrix"` from the `Matrix` package).
`y`	Output vector of dimension n. For `type="linear"` should be continuous and for `type="logistic"` should be a binary variable.
`groups`	A grouping structure for the input data. Should take the form of a vector of group indices.
`type`	The type of regression to perform. Supported values are: `"linear"` and `"logistic"`.

- `lambda` — The regularisation parameter. Defines the level of sparsity in the model. A higher value leads to sparser models:
  - "path" computes a path of regularisation parameters of length "path_length". The path will begin just above the value at which the first predictor enters the model and will terminate at the value determined by "min_frac".
  - User-specified single value or sequence. Internal scaling is applied based on the type of standardisation. The returned "lambda" value will be the original unscaled value(s).
- `path_length` — The number of λ values to fit the model for. If "lambda" is user-specified, this is ignored.
- `min_frac` — Smallest value of λ as a fraction of the maximum value. That is, the final λ will be "min_frac" of the first λ value.
- `alpha` — The value of α, which defines the convex balance between SLOPE and gSLOPE. Must be between 0 and 1. Recommended value is 0.95.
- `vFDR` — Defines the desired variable false discovery rate (FDR) level, which determines the shape of the variable penalties. Must be between 0 and 1.
- `gFDR` — Defines the desired group false discovery rate (FDR) level, which determines the shape of the group penalties. Must be between 0 and 1.
- `pen_method` — The type of penalty sequences to use (see Feser and Evangelou (2023)):
  - "1" uses the vMean SGS and gMean gSLOPE sequences.
  - "2" uses the vMax SGS and gMean gSLOPE sequences.
  - "3" uses the BH SLOPE and gMean gSLOPE sequences, also known as SGS Original.
- `nfolds` — The number of folds to use in cross-validation.
- `backtracking` — The backtracking parameter, τ, as defined in Pedregosa and Gidel (2018).
- `max_iter` — Maximum number of ATOS iterations to perform.
- `max_iter_backtracking` — Maximum number of backtracking line search iterations to perform per global iteration.
- `tol` — Convergence tolerance for the stopping criteria.
- `standardise` — Type of standardisation to perform on `X`:
  - "l2" standardises the input data to have ℓ_2 norms of one.
  - "l1" standardises the input data to have ℓ_1 norms of one.
  - "sd" standardises the input data to have standard deviation of one.
  - "none" no standardisation applied.
- `intercept` — Logical flag for whether to fit an intercept.
- `error_criteria` — The criteria used to discriminate between models along the path. Supported values are: "mse" (mean squared error) and "mae" (mean absolute error).
- `screen` — Logical flag for whether to apply screening rules (see Feser and Evangelou (2024)). Screening discards irrelevant groups before fitting, greatly improving speed.
- `verbose` — Logical flag for whether to print fitting information.

`v_weights` Optional vector for the variable penalty weights. Overrides the penalties from pen_method if specified. When entering custom weights, these are multiplied internally by λ and α. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$

`w_weights` Optional vector for the group penalty weights. Overrides the penalties from pen_method if specified. When entering custom weights, these are multiplied internally by λ and $1 - \alpha$. To void this behaviour, set $\lambda = 2$ and $\alpha = 0.5$

Details

Fits SGS models under a pathwise solution using adaptive three operator splitting (ATOS), picking the 1se model as optimum. Warm starts are implemented.

Value

A list containing:

`all_models`	A list of all the models fitted along the path.
`fit`	The 1se chosen model, which is a `"sgs"` object type.
`best_lambda`	The value of λ which generated the chosen model.
`best_lambda_id`	The path index for the chosen model.
`errors`	A table containing fitting information about the models on the path.
`type`	Indicates which type of regression was performed.

References

Feser, F., Evangelou, M. (2023). *Sparse-group SLOPE: adaptive bi-level selection with FDR-control*, https://arxiv.org/abs/2305.09467

Feser, F., Evangelou, M. (2024). *Strong screening rules for group-based SLOPE models*, https://arxiv.org/abs/2405.15357

See Also

`fit_sgs()`

Other model-selection: `as_sgs()`, `fit_goscar_cv()`, `fit_gslope_cv()`, `fit_sgo_cv()`, `scaled_sgs()`

Other SGS-methods: `as_sgs()`, `coef.sgs()`, `fit_sgo()`, `fit_sgo_cv()`, `fit_sgs()`, `plot.sgs()`, `predict.sgs()`, `print.sgs()`, `scaled_sgs()`

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run SGS with cross-validation
cv_model = fit_sgs_cv(X = data$X, y = data$y, groups=groups, type = "linear",
path_length = 5, nfolds=5, alpha = 0.95, vFDR = 0.1, gFDR = 0.1, min_frac = 0.05,
standardise="l2",intercept=TRUE,verbose=TRUE)
```

`gen_pens` *Generate penalty sequences for SGS.*

Description

Generates variable and group penalties for SGS.

Usage

```
gen_pens(gFDR, vFDR, pen_method, groups, alpha)
```

Arguments

`gFDR`	Defines the desired group false discovery rate (FDR) level, which determines the shape of the group penalties.
`vFDR`	Defines the desired variable false discovery rate (FDR) level, which determines the shape of the variable penalties.
`pen_method`	The type of penalty sequences to use (see Feser and Evangelou (2023)): • `"1"` uses the vMean SGS and gMean gSLOPE sequences. • `"2"` uses the vMax SGS and gMean gSLOPE sequences. • `"3"` uses the BH SLOPE and gMean gSLOPE sequences, also known as SGS Original. • `"4"` uses the gMax gSLOPE sequence. For a gSLOPE model only.
`groups`	A grouping structure for the input data. Should take the form of a vector of group indices.
`alpha`	The value of α, defines the convex balance between SLOPE and gSLOPE.

Details

The vMean and vMax SGS sequences are variable sequences derived specifically to give variable false discovery rate (FDR) control for SGS under orthogonal designs (see Feser and Evangelou (2023)). The BH SLOPE sequence is derived in Bodgan et. al. (2015) and has links to the Benjamini-Hochberg critical values. The sequence provides variable FDR-control for SLOPE under orthogonal designs. The gMean gSLOPE sequence is derived in Brzyski et. al. (2015) and provides group FDR-control for gSLOPE under orthogonal designs.

Value

A list containing:

`pen_slope_org`	A vector of the variable penalty sequence.
`pen_gslope_org`	A vector of the group penalty sequence.

References

Bogdan, M., Van den Berg, E., Sabatti, C., Su, W., Candes, E. (2015). *SLOPE — Adaptive variable selection via convex optimization*, https://projecteuclid.org/journals/annals-of-applied-statistics/volume-9/issue-3/SLOPEAdaptive-variable-selection-via-convex-optimization/10.1214/15-AOAS842.full

Brzyski, D., Gossmann, A., Su, W., Bodgan, M. (2019). *Group SLOPE – Adaptive Selection of Groups of Predictors*, https://www.tandfonline.com/doi/full/10.1080/01621459.2017.1411269

Feser, F., Evangelou, M. (2023). *Sparse-group SLOPE: adaptive bi-level selection with FDR-control*, https://arxiv.org/abs/2305.09467

Examples

```
# specify a grouping structure
groups = c(rep(1:20, each=3),
          rep(21:40, each=4),
          rep(41:60, each=5),
          rep(61:80, each=6),
          rep(81:100, each=7))
# generate sequences
sequences = gen_pens(gFDR=0.1, vFDR=0.1, pen_method=1, groups=groups, alpha=0.5)
```

---

gen_toy_data *Generate toy data.*

Description

Generates different types of datasets, which can then be fitted using sparse-group SLOPE.

Usage

```
gen_toy_data(
  p,
  n,
  rho = 0,
  seed_id = 2,
  grouped = TRUE,
  groups,
  noise_level = 1,
  group_sparsity = 0.1,
  var_sparsity = 0.5,
  orthogonal = FALSE,
  data_mean = 0,
  data_sd = 1,
  signal_mean = 0,
  signal_sd = sqrt(10)
)
```

Arguments

p	The number of input variables.
n	The number of observations.
rho	Correlation coefficient. Must be in range $[0, 1]$.
seed_id	Seed to be used to generate the data matrix X.
grouped	A logical flag indicating whether grouped data is required.
groups	If grouped=TRUE, the grouping structure is required. Each input variable should have a group id.
noise_level	Defines the level of noise (σ) to be used in generating the response vector y.
group_sparsity	Defines the level of group sparsity. Must be in the range $[0, 1]$.
var_sparsity	Defines the level of variable sparsity. Must be in the range $[0, 1]$. If grouped=TRUE, this defines the level of sparsity within each group, not globally.
orthogonal	Logical flag as to whether the input matrix should be orthogonal.
data_mean	Defines the mean of input predictors.
data_sd	Defines the standard deviation of the signal (β).
signal_mean	Defines the mean of the signal (β).
signal_sd	Defines the standard deviation of the signal (β).

Details

The data is generated under a Gaussian linear model. The generated data can be grouped and sparsity can be provided at both a group and/or variable level.

Value

A list containing:

y	The response vector.
X	The input matrix.
true_beta	The true values of β used to generate the response.
true_grp_id	Indices of which groups are non-zero in true_beta.

Examples

```
# specify a grouping structure
groups = c(rep(1:20, each=3),
          rep(21:40, each=4),
          rep(41:60, each=5),
          rep(61:80, each=6),
          rep(81:100, each=7))
# generate data
data =  gen_toy_data(p=500, n=400, groups = groups, seed_id=3)
```

plot.sgs — *Plot models of the following object types:* "sgs", "sgs_cv", "gslope", "gslope_cv".

Description

Plots the pathwise solution of a cross-validation fit, from a call to one of the following: fit_sgs(), fit_sgs_cv(), fit_gslope(), fit_gslope_cv(), fit_sgo(), fit_sgo_cv(), fit_goscar(), fit_goscar_cv().

Usage

```
## S3 method for class 'sgs'
plot(x, how_many = 10, ...)
```

Arguments

x	Object of one of the following classes: "sgs", "sgs_cv", "gslope", "gslope_cv".
how_many	Defines how many predictors to plot. Plots the predictors in decreasing order of largest absolute value.
...	further arguments passed to base function.

Value

A list containing:

response	The predicted response. In the logistic case, this represents the predicted class probabilities.
class	The predicted class assignments. Only returned if type = "logistic" in the model object.

See Also

fit_sgs(), fit_sgs_cv(), fit_gslope(), fit_gslope_cv(), fit_sgo(), fit_sgo_cv(), fit_goscar(), fit_goscar_cv()

Other SGS-methods: as_sgs(), coef.sgs(), fit_sgo(), fit_sgo_cv(), fit_sgs(), fit_sgs_cv(), predict.sgs(), print.sgs(), scaled_sgs()

Other gSLOPE-methods: coef.sgs(), fit_goscar(), fit_goscar_cv(), fit_gslope(), fit_gslope_cv(), predict.sgs(), print.sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,2,2,3)
# generate data
data =  gen_toy_data(p=5, n=4, groups = groups, seed_id=3,signal_mean=20,group_sparsity=1)
# run SGS
```

```
model = fit_sgs(X = data$X, y = data$y, groups=groups, type = "linear",
path_length = 20, alpha = 0.95, vFDR = 0.1, gFDR = 0.1,
min_frac = 0.05, standardise="l2",intercept=TRUE,verbose=FALSE)
plot(model, how_many = 10)
```

`predict.sgs` *Predict using one of the following object types:* `"sgs"`, `"sgs_cv"`, `"gslope"`, `"gslope_cv"`.

Description

Performs prediction from one of the following fits: `fit_sgs()`, `fit_sgs_cv()`, `fit_gslope()`, `fit_gslope_cv()`, `fit_sgo()`, `fit_sgo_cv()`, `fit_goscar()`, `fit_goscar_cv()`. The predictions are calculated for each `"lambda"` value in the path.

Usage

```
## S3 method for class 'sgs'
predict(object, x, ...)
```

Arguments

`object`	Object of one of the following classes: `"sgs"`, `"sgs_cv"`, `"gslope"`, `"gslope_cv"`.
`x`	Input data to use for prediction.
`...`	further arguments passed to stats function.

Value

A list containing:

`response`	The predicted response. In the logistic case, this represents the predicted class probabilities.
`class`	The predicted class assignments. Only returned if type = "logistic" in the `"sgs"` object.

See Also

`fit_sgs()`, `fit_sgs_cv()`, `fit_gslope()`, `fit_gslope_cv()`, `fit_sgo()`, `fit_sgo_cv()`, `fit_goscar()`, `fit_goscar_cv()`

Other SGS-methods: `as_sgs()`, `coef.sgs()`, `fit_sgo()`, `fit_sgo_cv()`, `fit_sgs()`, `fit_sgs_cv()`, `plot.sgs()`, `print.sgs()`, `scaled_sgs()`

Other gSLOPE-methods: `coef.sgs()`, `fit_goscar()`, `fit_goscar_cv()`, `fit_gslope()`, `fit_gslope_cv()`, `plot.sgs()`, `print.sgs()`

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data =  gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run SGS
model = fit_sgs(X = data$X, y = data$y, groups = groups, type="linear", lambda = 1, alpha=0.95,
vFDR=0.1, gFDR=0.1, standardise = "l2", intercept = TRUE, verbose=FALSE)
# use predict function
model_predictions = predict(model, x = data$X)
```

---

print.sgs — *Prints information for one of the following object types:* "sgs", "sgs_cv", "gslope", "gslope_cv".

Description

Prints out useful metric from a model fit.

Usage

```
## S3 method for class 'sgs'
print(x, ...)
```

Arguments

x	Object of one of the following classes: "sgs", "sgs_cv", "gslope", "gslope_cv".
...	further arguments passed to base function.

Value

A summary of the model fit(s).

See Also

fit_sgs(), fit_sgs_cv(), fit_gslope(), fit_gslope_cv(), fit_sgo(), fit_sgo_cv(), fit_goscar(), fit_goscar_cv()

Other SGS-methods: as_sgs(), coef.sgs(), fit_sgo(), fit_sgo_cv(), fit_sgs(), fit_sgs_cv(), plot.sgs(), predict.sgs(), scaled_sgs()

Other gSLOPE-methods: coef.sgs(), fit_goscar(), fit_goscar_cv(), fit_gslope(), fit_gslope_cv(), plot.sgs(), predict.sgs()

Examples

```
# specify a grouping structure
groups = c(rep(1:20, each=3),
           rep(21:40, each=4),
           rep(41:60, each=5),
           rep(61:80, each=6),
           rep(81:100, each=7))
# generate data
data =  gen_toy_data(p=500, n=400, groups = groups, seed_id=3)
# run SGS
model = fit_sgs(X = data$X, y = data$y, groups = groups, type="linear", lambda = 1, alpha=0.95,
vFDR=0.1, gFDR=0.1, standardise = "l2", intercept = TRUE, verbose=FALSE)
# print model
print(model)
```

---

scaled_sgs   *Fits a scaled SGS model.*

Description

Fits an SGS model using the noise estimation procedure (Algorithm 5 from Bogdan et. al. (2015)). This estimates λ and then fits the model using the estimated value. It is an alternative approach to cross-validation (fit_sgs_cv()).

Usage

```
scaled_sgs(
  X,
  y,
  groups,
  type = "linear",
  pen_method = 1,
  alpha = 0.95,
  vFDR = 0.1,
  gFDR = 0.1,
  standardise = "l2",
  intercept = TRUE,
  verbose = FALSE
)
```

Arguments

X	Input matrix of dimensions $n \times p$. Can be a sparse matrix (using class "sparseMatrix" from the Matrix package).
y	Output vector of dimension n. For type="linear" should be continuous and for type="logistic" should be a binary variable.

groups	A grouping structure for the input data. Should take the form of a vector of group indices.
type	The type of regression to perform. Supported values are: "linear" and "logistic".
pen_method	The type of penalty sequences to use. • "1" uses the vMean SGS and gMean gSLOPE sequences. • "2" uses the vMax SGS and gMean gSLOPE sequences. • "1" uses the BH SLOPE and gMean gSLOPE sequences, also known as SGS Original.
alpha	The value of α, which defines the convex balance between SLOPE and gSLOPE. Must be between 0 and 1.
vFDR	Defines the desired variable false discovery rate (FDR) level, which determines the shape of the variable penalties. Must be between 0 and 1.
gFDR	Defines the desired group false discovery rate (FDR) level, which determines the shape of the group penalties. Must be between 0 and 1.
standardise	Type of standardisation to perform on X: • "l2" standardises the input data to have ℓ_2 norms of one. • "l1" standardises the input data to have ℓ_1 norms of one. • "sd" standardises the input data to have standard deviation of one. • "none" no standardisation applied.
intercept	Logical flag for whether to fit an intercept.
verbose	Logical flag for whether to print fitting information.

Value

An object of type "sgs" containing model fit information (see fit_sgs()).

References

Bogdan, M., Van den Berg, E., Sabatti, C., Su, W., Candes, E. (2015). *SLOPE — Adaptive variable selection via convex optimization*, https://projecteuclid.org/journals/annals-of-applied-statistics/volume-9/issue-3/SLOPEAdaptive-variable-selection-via-convex-optimization/10.1214/15-AOAS842.full

See Also

as_sgs()

Other model-selection: as_sgs(), fit_goscar_cv(), fit_gslope_cv(), fit_sgo_cv(), fit_sgs_cv()

Other SGS-methods: as_sgs(), coef.sgs(), fit_sgo(), fit_sgo_cv(), fit_sgs(), fit_sgs_cv(), plot.sgs(), predict.sgs(), print.sgs()

Examples

```
# specify a grouping structure
groups = c(1,1,2,2,3)
# generate data
data =  gen_toy_data(p=5, n=4, groups = groups, seed_id=3,
signal_mean=20,group_sparsity=1,var_sparsity=1)
# run noise estimation
model = scaled_sgs(X=data$X, y=data$y, groups=groups, pen_method=1)
```

Index