Package ‘dfr’

November 28, 2024

Title Dual Feature Reduction for SGL

Version 0.1.2

Date 2024-11-28

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

Description Implementation of the Dual Feature Reduction (DFR) approach for the Sparse Group Lasso (SGL) and the Adaptive Sparse Group Lasso (aSGL) (Feser and Evangelou (2024) <doi:10.48550/arXiv.2405.17094>). The DFR approach is a feature reduction approach that applies strong screening to reduce the feature space before optimisation, leading to speed-up improvements for fitting SGL (Simon et al. (2013) <doi:10.1080/10618600.2012.681250>) and aSGL (Mendez-Civieta et al. (2020) <doi:10.1007/s11634-020-00413-8> and Poignard (2020) <doi:10.1007/s10463-018-0692-7>) models. DFR is implemented using the Adaptive Three Operator Splitting (ATOS) (Pedregosa and Gidel (2018) <doi:10.48550/arXiv.1804.02339>) algorithm, with linear and logistic SGL models supported, both of which can be fit using k-fold cross-validation. Dense and sparse input matrices are supported.

Imports sgs, caret, MASS, methods, stats, grDevices, graphics, Matrix

Suggests SGL, gglasso, glmnet, testthat

RoxygenNote 7.3.1

License GPL (>= 3)

Encoding UTF-8

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

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

NeedsCompilation no

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

Repository CRAN

Date/Publication 2024-11-28 12:50:08 UTC

Contents

`dfr_adap_sgl` *Fit a DFR-aSGL model.*

Description

Adaptive Sparse-group lasso (aSGL) with DFR main fitting function. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
dfr_adap_sgl(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  alpha = 0.95,
  gamma_1 = 0.1,
  gamma_2 = 0.1,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  path_length = 20,
  min_frac = 0.05,
  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).
- `alpha` — The value of α, which defines the convex balance between the lasso and group lasso. Must be between 0 and 1. Recommended value is 0.95.
- `gamma_1` — Hyperparameter which determines the shape of the variable penalties.
- `gamma_2` — Hyperparameter which determines the shape of the group penalties.
- `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.
- `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.
- `screen` — Logical flag for whether to apply the DFR screening rules (see Feser and Evangelou (2024)).
- `verbose` — Logical flag for whether to print fitting information.

`v_weights` Optional vector for the variable penalty weights. Overrides the adaptive SGL penalties 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 adaptive SGL penalties 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

`dfr_adap_sgl()` fits a DFR-aSGL model (Feser and Evangelou (2024)) using Adaptive Three Operator Splitting (ATOS) (Pedregosa and Gidel (2018)). It solves the convex optimisation problem given by (Poignard (2020) and Mendez-Civieta et al. (2020))

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

where $f(\cdot)$ is the loss function, p_g are the group sizes, and (v, w) are adaptive weights. 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 adaptive weights are chosen as, for a group g and variable i (Mendez-Civieta et al. (2020))

$$v_i = \frac{1}{|q_{1i}|^{\gamma_1}}, \ w_g = \frac{1}{\|q_1^{(g)}\|_2^{\gamma_2}},$$

DFR uses the dual norm (the ϵ-norm) and the KKT conditions to discard features at λ_k that would have been inactive at λ_{k+1}. It applies two layers of screening, so that it first screens out any groups that satisfy

$$\|\nabla_g f(\hat{\beta}(\lambda_k))\|_{\epsilon'_g} \leq \gamma_g (2\lambda_{k+1} - \lambda_k)$$

and then screens out any variables that satisfy

$$|\nabla_i f(\hat{\beta}(\lambda_k))| \leq \alpha v_i (2\lambda_{k+1} - \lambda_k)$$

leading to effective input dimensionality reduction. See Feser and Evangelou (2024) for full details.

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. Index 1 corresponds to the first column in X.

selected_grp
: A list containing the indicies of the active/selected groups for each "lambda" value. Index 1 corresponds to the first group in the groups vector.

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_var
: List of variables that were kept after screening step for each "lambda" value. (see Feser and Evangelou (2024)).

screen_set_grp
: List of groups that were kept after screening step for each "lambda" value. (see Feser and Evangelou (2024)).

epsilon_set_var
: List of variables that were used for fitting after screening for each "lambda" value. (see Feser and Evangelou (2024)).

epsilon_set_grp
: List of groups that were used for fitting after screening for each "lambda" value. (see Feser and Evangelou (2024)).

kkt_violations_var
: List of variables that violated the KKT conditions each "lambda" value. (see Feser and Evangelou (2024)).

kkt_violations_grp
: List of groups that violated the KKT conditions each "lambda" value. (see Feser and Evangelou (2024)).

v_weights
: Vector of the variable penalty sequence.

w_weights
: Vector of the group penalty sequence.

screen
: Logical flag indicating whether screening was performed.

type
: Indicates which type of regression was performed.

intercept
: Logical flag indicating whether an intercept was fit.

lambda
: Value(s) of λ used to fit the model.

References

Feser, F., Evangelou, M. (2024). *Dual feature reduction for the sparse-group lasso and its adaptive variant*, https://arxiv.org/abs/2405.17094

Mendez-Civieta, A., Carmen Aguilera-Morillo, M., Lillo, R. (2020). *Adaptive sparse group LASSO in quantile regression*, https://link.springer.com/article/10.1007/s11634-020-00413-8

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

Poignard, B. (2020). *Asymptotic theory of the adaptive Sparse Group Lasso*, https://link.springer.com/article/10.1007/s10463-018-0692-7

See Also

Other SGL-methods: `dfr_adap_sgl.cv()`, `dfr_sgl()`, `dfr_sgl.cv()`, `plot.sgl()`, `predict.sgl()`, `print.sgl()`

Examples

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

`dfr_adap_sgl.cv` *Fit a DFR-aSGL model using k-fold cross-validation.*

Description

Function to fit a pathwise solution of the adaptive sparse-group lasso (aSGL) applied with DFR using k-fold cross-validation. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
dfr_adap_sgl.cv(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  nfolds = 10,
  alpha = 0.95,
  gamma_1 = 0.1,
```

```
  gamma_2 = 0.1,
  backtracking = 0.7,
  max_iter = 5000,
  max_iter_backtracking = 100,
  tol = 1e-05,
  min_frac = 0.05,
  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.

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

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

`gamma_1` Hyperparameter which determines the shape of the variable penalties.

`gamma_2` Hyperparameter which determines the shape of the group penalties.

`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.
`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.
`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 the DFR screening rules (see Feser and Evangelou (2024)).
`verbose`	Logical flag for whether to print fitting information.
`v_weights`	Optional vector for the variable penalty weights. Overrides the adaptive SGL penalties 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 adaptive SGL penalties 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 DFR-aSGL models under a pathwise solution using Adaptive Three Operator Splitting (ATOS) (Pedregosa and Gidel (2018)), 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 `"sgl"` 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. (2024). *Dual feature reduction for the sparse-group lasso and its adaptive variant*, https://arxiv.org/abs/2405.17094

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

See Also

dfr_adap_sgl()

Other SGL-methods: dfr_adap_sgl(), dfr_sgl(), dfr_sgl.cv(), plot.sgl(), predict.sgl(), print.sgl()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data = sgs::gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run DFR-SGL with cross-validation
cv_model = dfr_adap_sgl.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)
```

---

dfr_sgl *Fit a DFR-SGL model.*

Description

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

Usage

```
dfr_sgl(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  alpha = 0.95,
  max_iter = 5000,
  backtracking = 0.7,
  max_iter_backtracking = 100,
  tol = 1e-05,
  standardise = "l2",
  intercept = TRUE,
  path_length = 20,
  min_frac = 0.05,
  screen = 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".
- `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).
- `alpha` The value of α, which defines the convex balance between the lasso and group lasso. 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.
  - "none" no standardisation applied.
- `intercept` Logical flag for whether to fit an intercept.
- `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.
- `screen` Logical flag for whether to apply the DFR screening rules (see Feser and Evangelou (2024)).
- `verbose` Logical flag for whether to print fitting information.

Details

`dfr_sgl()` fits a DFR-SGL model (Feser and Evangelou (2024)) using Adaptive Three Operator Splitting (ATOS) (Pedregosa and Gidel (2018)). It solves the convex optimisation problem given by (Simon et al. (2013))

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

where $f(\cdot)$ is the loss function and p_g are the group sizes. 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

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

SGL can be seen to be a convex combination of the lasso and group lasso, balanced through `alpha`, such that it reduces to the lasso for `alpha = 0` and to the group lasso for `alpha = 1`. By applying both the lasso and group lasso norms, SGL shrinks inactive groups to zero, as well as inactive variables in active groups. DFR uses the dual norm (the ϵ-norm) and the KKT conditions to discard features at λ_k that would have been inactive at λ_{k+1}. It applies two layers of screening, so that it first screens out any groups that satisfy

$$\|\nabla_g f(\hat{\beta}(\lambda_k))\|_{\epsilon_g} \leq \tau_g(2\lambda_{k+1}-\lambda_k)$$

and then screens out any variables that satisfy

$$|\nabla_i f(\hat{\beta}(\lambda_k))| \leq \alpha(2\lambda_{k+1}-\lambda_k)$$

leading to effective input dimensionality reduction. See Feser and Evangelou (2024) for full details.

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. Index 1 corresponds to the first column in X.

selected_grp — A list containing the indicies of the active/selected groups for each "lambda" value. Index 1 corresponds to the first group in the groups vector.

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_var — List of variables that were kept after screening step for each "lambda" value. (see Feser and Evangelou (2024)).

screen_set_grp — List of groups that were kept after screening step for each "lambda" value. (see Feser and Evangelou (2024)).

epsilon_set_var — List of variables that were used for fitting after screening for each "lambda" value. (see Feser and Evangelou (2024)).

epsilon_set_grp — List of groups that were used for fitting after screening for each "lambda" value. (see Feser and Evangelou (2024)).

kkt_violations_var — List of variables that violated the KKT conditions each "lambda" value. (see Feser and Evangelou (2024)).

kkt_violations_grp — List of groups that violated the KKT conditions each "lambda" value. (see Feser and Evangelou (2024)).

screen — Logical flag indicating whether screening was performed.

type — Indicates which type of regression was performed.

intercept — Logical flag indicating whether an intercept was fit.

lambda — Value(s) of λ used to fit the model.

References

Feser, F., Evangelou, M. (2024). *Dual feature reduction for the sparse-group lasso and its adaptive variant*, https://arxiv.org/abs/2405.17094

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

Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2013). *A Sparse-Group Lasso*, https://www.tandfonline.com/doi/abs/10.1080/10618600.2012.681250

See Also

Other SGL-methods: dfr_adap_sgl(), dfr_adap_sgl.cv(), dfr_sgl.cv(), plot.sgl(), predict.sgl(), print.sgl()

Examples

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

---

dfr_sgl.cv *Fit a DFR-SGL model using k-fold cross-validation.*

Description

Function to fit a pathwise solution of the sparse-group lasso (SGL) applied with DFR using k-fold cross-validation. Supports both linear and logistic regression, both with dense and sparse matrix implementations.

Usage

```
dfr_sgl.cv(
  X,
  y,
  groups,
  type = "linear",
  lambda = "path",
  path_length = 20,
  nfolds = 10,
  alpha = 0.95,
  backtracking = 0.7,
  max_iter = 5000,
  max_iter_backtracking = 100,
  tol = 1e-05,
  min_frac = 0.05,
  standardise = "l2",
  intercept = TRUE,
  error_criteria = "mse",
  screen = 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"`.
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.
nfolds	The number of folds to use in cross-validation.
alpha	The value of α, which defines the convex balance between the lasso and group lasso. Must be between 0 and 1. Recommended value is 0.95.
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.
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.
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 the DFR screening rules (see Feser and Evangelou (2024)).
verbose	Logical flag for whether to print fitting information.

Details

Fits DFR-SGL models under a pathwise solution using Adaptive Three Operator Splitting (ATOS) (Pedregosa and Gidel (2018)), 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 "sgl" 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. (2024). *Dual feature reduction for the sparse-group lasso and its adaptive variant*, https://arxiv.org/abs/2405.17094

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

See Also

dfr_sgl()

Other SGL-methods: dfr_adap_sgl(), dfr_adap_sgl.cv(), dfr_sgl(), plot.sgl(), predict.sgl(), print.sgl()

Examples

```
# specify a grouping structure
groups = c(1,1,1,2,2,3,3,3,4,4)
# generate data
data = sgs::gen_toy_data(p=10, n=5, groups = groups, seed_id=3,group_sparsity=1)
# run DFR-SGL with cross-validation
cv_model = dfr_sgl.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)
```

---

plot.sgl — *Plot models of the following object types:* "sgl", "sgl_cv".

Description

Plots the pathwise solution of a cross-validation fit, from a call to one of the following: dfr_sgl(), dfr_sgl.cv(), dfr_adap_sgl(), dfr_adap_sgl.cv().

Usage

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

Arguments

`x`	Object of one of the following classes: `"sgl"`, `"sgl_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

`dfr_sgl()`, `dfr_sgl.cv()`, `dfr_adap_sgl()`, `dfr_adap_sgl.cv()`

Other SGL-methods: `dfr_adap_sgl()`, `dfr_adap_sgl.cv()`, `dfr_sgl()`, `dfr_sgl.cv()`, `predict.sgl()`, `print.sgl()`

Examples

```
# specify a grouping structure
groups = c(1,1,2,2,3)
# generate data
data = sgs::gen_toy_data(p=5, n=4, groups = groups, seed_id=3,signal_mean=20,group_sparsity=1)
# run DFR-SGL
model = dfr_sgl(X = data$X, y = data$y, groups=groups, type = "linear",
path_length = 20, alpha = 0.95,
min_frac = 0.05, standardise="l2",intercept=TRUE,verbose=FALSE)
plot(model, how_many = 10)
```

`predict.sgl` *Predict using one of the following object types:* `"sgl"`, `"sgl_cv"`.

Description

Performs prediction from one of the following fits: `dfr_sgl()`, `dfr_sgl.cv()`, `dfr_adap_sgl()`, `dfr_adap_sgl.cv()`. The predictions are calculated for each `"lambda"` value in the path.

Usage

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

Arguments

`object`	Object of one of the following classes: `"sgl"`, `"sgl_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 `"sgl"` or `"sgl_cv"` object.

See Also

`dfr_sgl()`, `dfr_sgl.cv()`, `dfr_adap_sgl()`, `dfr_adap_sgl.cv()`

Other SGL-methods: `dfr_adap_sgl()`, `dfr_adap_sgl.cv()`, `dfr_sgl()`, `dfr_sgl.cv()`, `plot.sgl()`, `print.sgl()`

Examples

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

`print.sgl` *Prints information for one of the following object types: `"sgl"`, `"sgl_cv"`.*

Description

Prints out useful metric from a model fit.

Usage

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

Arguments

x	Object of one of the following classes: "sgl", "sgl_cv".
...	further arguments passed to base function.

Value

A summary of the model fit(s).

See Also

dfr_sgl(), dfr_sgl.cv(), dfr_adap_sgl(), dfr_adap_sgl.cv()

Other SGL-methods: dfr_adap_sgl(), dfr_adap_sgl.cv(), dfr_sgl(), dfr_sgl.cv(), plot.sgl(), predict.sgl()

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 = sgs::gen_toy_data(p=500, n=400, groups = groups, seed_id=3)
# run DFR-SGL
model = dfr_sgl(X = data$X, y = data$y, groups = groups, type="linear", lambda = 1, alpha=0.95,
standardise = "l2", intercept = TRUE, verbose=FALSE)
# print model
print(model)
```

Index