Package ‘SSGL’

January 20, 2025

Type Package

Title Spike-and-Slab Group Lasso for Group-Regularized Generalized Linear Models

Version 1.0

Date 2023-06-25

Author Ray Bai

Maintainer Ray Bai <raybaistat@gmail.com>

Description Fits group-regularized generalized linear models (GLMs) using the spike-and-slab group lasso (SSGL) prior introduced by Bai et al. (2022) <doi:10.1080/01621459.2020.1765784> and extended to GLMs by Bai (2023) <arXiv:2007.07021>. This package supports fitting the SSGL model for the following GLMs with group sparsity: Gaussian linear regression, binary logistic regression, Poisson regression, negative binomial regression, and gamma regression. Stand-alone functions for group-regularized negative binomial regression and group-regularized gamma regression are also available, with the option of employing the group lasso penalty of Yuan and Lin (2006) <doi:10.1111/j.1467-9868.2005.00532.x>, the group minimax concave penalty (MCP) of Breheny and Huang <doi:10.1007/s11222-013-9424-2>, or the group smoothly clipped absolute deviation (SCAD) penalty of Breheny and Huang (2015) <doi:10.1007/s11222-013-9424-2>.

License GPL-3

Depends R (>= 3.6.0)

Imports stats, MASS, pracma, grpreg

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-06-27 16:40:02 UTC

Contents

cv_gamma_grpreg *Cross-validation for Group-Regularized Gamma Regression*

Description

This function implements K-fold cross-validation for group-regularized gamma regression with a known shape parameter ν and the log link. The cross-validation error (CVE) and cross-validation standard error (CVSE) are computed using the deviance for gamma regression.

For a description of group-regularized gamma regression, see the description for the gamma_grpreg function. Our implementation is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than $\frac{K-1}{K} \times$ sample size, where K is the number of folds.

Note that the gamma_grpreg function also returns the generalized information criterion (GIC) of Fan and Tang (2013) for each regularization parameter in lambda, and the GIC can also be used for model selection instead of cross-validation.

Usage

```
cv_gamma_grpreg(Y, X, groups, gamma_shape=1, penalty=c("gLASSO","gSCAD","gMCP"),
                n_folds=10, group_weights, taper, n_lambda=100, lambda,
                max_iter=10000, tol=1e-4)
```

Arguments

Y	$n \times 1$ vector of strictly positive, continuous responses for training data.
X	$n \times p$ design matrix for training data, where the jth column corresponds to the jth overall feature.
groups	p-dimensional vector of group labels. The jth entry in groups should contain either the group number *or* the factor level name that the feature in the jth column of X belongs to. groups must be either a vector of integers or factors.
gamma_shape	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for the responses. Default is gamma_shape=1.
penalty	group regularization method to use on the groups of regression coefficients. The options are "gLASSO", "gSCAD", "gMCP". To implement cross-validation for gamma regression with the SSGL penalty, use the cv_SSGL function.
n_folds	number of folds K to use in K-fold cross-validation. Default is n_folds=10.
group_weights	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.
taper	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is taper=4 for group SCAD and taper=3 for group MCP. Ignored if "gLASSO" is specified as the penalty.

`n_lambda` number of regularization parameters L. Default is `n_lambda=100`.

`lambda` grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

`max_iter` maximum number of iterations in the algorithm. Default is `max_iter=10000`.

`tol` convergence threshold for algorithm. Default is `tol=1e-4`.

Value

The function returns a list containing the following components:

`lambda` $L \times 1$ vector of regularization parameters `lambda` used to fit the model. `lambda` is displayed in descending order.

`cve` $L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in `cve` corresponds to the kth regularization parameter in `lambda`.

`cvse` $L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in `cvse` corresponds to the kth regularization parameter in `lambda`.

`lambda_min` The value in `lambda` that minimizes mean cross-validation error `cve`.

`min_index` The index of `lambda_min` in `lambda`.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Fan, Y. and Tang, C. Y. (2013). "Tuning parameter selection in high-dimensional penalized likelihood." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**:531-552.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Yuan, M. and Lin, Y. (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**:49-67.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*11), nrow=100)
n = dim(X)[1]
groups = c(1,1,1,2,2,2,3,3,4,5,5)
beta_true = c(-1,1,1,0,0,0,0,0,0,1.5,-1.5)

## Generate responses from gamma regression with known shape parameter 1
eta = crossprod(t(X), beta_true)
shape = 1
Y = rgamma(n, rate=shape/exp(eta), shape=shape)

## 10-fold cross-validation for group-regularized gamma regression
## with the group LASSO penalty
gamma_cv = cv_gamma_grpreg(Y, X, groups, penalty="gLASSO")
```

```
## Plot cross-validation curve
plot(gamma_cv$lambda, gamma_cv$cve, type="l", xlab="lambda", ylab="CVE")
## lambda which minimizes mean CVE
gamma_cv$lambda_min
## index of lambda_min in lambda
gamma_cv$min_index
```

---

cv_nb_grpreg *Cross-validation for Group-Regularized Negative Binomial Regression*

Description

This function implements K-fold cross-validation for group-regularized negative binomial regression with a known size parameter α and the log link. The cross-validation error (CVE) and cross-validation standard error (CVSE) are computed using the deviance for negative binomial regression.

For a description of group-regularized negative binomial regression, see the description for the `nb_grpreg` function. Our implementation is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than $\frac{K-1}{K} \times$ sample size, where K is the number of folds.

Note that the `nb_grpreg` function also returns the generalized information criterion (GIC) of Fan and Tang (2013) for each regularization parameter in `lambda`, and the GIC can also be used for model selection instead of cross-validation.

Usage

```
cv_nb_grpreg(Y, X, groups, nb_size=1, penalty=c("gLASSO","gSCAD","gMCP"),
             n_folds=10, group_weights, taper, n_lambda=100, lambda,
             max_iter=10000, tol=1e-4)
```

Arguments

`Y` $n \times 1$ vector of strictly nonnegative integer responses for training data.

`X` $n \times p$ design matrix for training data, where the jth column corresponds to the jth overall feature.

`groups` p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the factor level name that the feature in the jth column of `X` belongs to. `groups` must be either a vector of integers or factors.

`nb_size` known size parameter α in $NB(\alpha, \mu_i)$ distribution for the responses. Default is `nb_size=1`.

`penalty` group regularization method to use on the groups of regression coefficients. The options are `"gLASSO"`, `"gSCAD"`, `"gMCP"`. To implement cross-validation for gamma regression with the SSGL penalty, use the `cv_SSGL` function.

`n_folds` number of folds K to use in K-fold cross-validation. Default is `n_folds=10`.

`group_weights` group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.

`taper` tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is `taper=4` for group SCAD and `taper=3` for group MCP. Ignored if `"gLASSO"` is specified as the penalty.

`n_lambda` number of regularization parameters L. Default is `n_lambda=100`.

`lambda` grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

`max_iter` maximum number of iterations in the algorithm. Default is `max_iter=10000`.

`tol` convergence threshold for algorithm. Default is `tol=1e-4`.

Value

The function returns a list containing the following components:

`lambda` $L \times 1$ vector of regularization parameters `lambda` used to fit the model. `lambda` is displayed in descending order.

`cve` $L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in `cve` corresponds to the kth regularization parameter in `lambda`.

`cvse` $L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in `cvse` corresponds to the kth regularization parameter in `lambda`.

`lambda_min` The value in `lambda` that minimizes mean cross-validation error `cve`.

`min_index` The index of `lambda_min` in `lambda`.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Fan, Y. and Tang, C. Y. (2013). "Tuning parameter selection in high dimensional penalized likelihood." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**:531-552.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Yuan, M. and Lin, Y. (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**:49-67.

Examples

```
## Generate data
set.seed(1234)
X = matrix(runif(100*14), nrow=100)
n = dim(X)[1]
groups = c(1,1,1,2,2,2,2,3,3,4,5,5,6,6)
beta_true = c(-1,1,1,0,0,0,0,-1,1,0,0,0,-1.5,1.5)

## Generate count responses from negative binomial regression
```

```
eta = crossprod(t(X), beta_true)
Y = rnbinom(n, size=1, mu=exp(eta))

## 10-fold cross-validation for group-regularized negative binomial
## regression with the group MCP penalty
nb_cv = cv_nb_grpreg(Y, X, groups, penalty="gMCP")

## Plot cross-validation curve
plot(nb_cv$lambda, nb_cv$cve, type="l", xlab="lambda", ylab="CVE")
## lambda which minimizes mean CVE
nb_cv$lambda_min
## index of lambda_min in lambda
nb_cv$min_index
```

cv_SSGL *Cross-Validation for Spike-and-Slab Group Lasso in Group-Regularized Generalized Linear Models (GLMs)*

Description

This function implements K-fold cross-validation for group-regularized GLMs with the spike-and-slab group lasso (SSGL) penalty of Bai et al. (2022) and Bai (2023). The identity link function is used for Gaussian regression, the logit link is used for binomial regression, and the log link is used for Poisson, negative binomial, and gamma regression.

Although one can choose `lambda0` from cross-validation with this function, it can be very time-consuming to do so if the number of groups G and/or the number of total covariantes p is moderate to large. It is *strongly* recommended that the user simply run the `SSGL` function on the training dataset and select the final model according to the `lambda0` that minimizes the generalized information criterion (GIC). See description of the `SSGL` function for more details.

Usage

```
cv_SSGL(Y, X, groups,
        family=c("gaussian","binomial","poisson","negativebinomial","gamma"),
        nb_size=1, gamma_shape=1, group_weights, n_folds=5, n_lambda0=25,
        lambda0, lambda1=1, a=1, b=dim(X)[2],
        max_iter=100, tol=1e-6, print_fold=TRUE)
```

Arguments

`Y` $n \times 1$ vector of responses for training data.

`X` $n \times p$ design matrix for training data, where the jth column corresponds to the jth overall feature.

`groups` p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the factor level name that the feature in the jth column of `X` belongs to. `groups` must be either a vector of integers or factors.

`family` exponential dispersion family of the response variables. Allows for `"gaussian"`, `"binomial"`, `"poisson"`, `"negativebinomial"`, and `"gamma"`. Note that for `"negativebinomial"`, the size parameter must be specified in advance, while for `"gamma"`, the shape parameter must be specified in advance.

`nb_size` known size parameter α in $NB(\alpha, \mu_i)$ distribution for the responses if the user specifies `family="negativebinomial"`. Default is `nb_size=1`. Ignored if `family` is not `"negativebinomial"`.

`gamma_shape` known shape parameter ν in $G(\mu_i, \nu)$ distribution for the responses if the user specifies `family="gamma"`. Default is `gamma_shape=1`. Ignored if `family` is not `"gamma"`.

`group_weights` group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.

`n_folds` number of folds K to use in K-fold cross-validation. Default is `n_folds=5`.

`n_lambda0` number of spike hyperparameters L. Default is `n_lambda0=25`.

`lambda0` grid of L spike hyperparameters λ_0. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

`lambda1` slab hyperparameter λ_1 in the SSGL prior. Default is `lambda1=1`.

`a` shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is `a=1`.

`b` shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is `b=dim(X)[2]`.

`max_iter` maximum number of iterations in the algorithm. Default is `max_iter=100`.

`tol` convergence threshold for algorithm. Default is `tol=1e-6`.

`print_fold` Boolean variable for whether or not to print the current fold in the algorithm. Default is `print_fold=TRUE`.

Value

The function returns a list containing the following components:

`lambda0` $L \times 1$ vector of spike hyperparameters `lambda0` used to fit the model. `lambda0` is displayed in descending order.

`cve` $L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in `cve` corresponds to the kth spike hyperparameter parameter in `lambda0`.

`cvse` $L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in `cvse` corresponds to the kth spike hyperparameter parameter in `lambda0`.

`lambda0_min` The value in `lambda0` that minimizes mean cross-validation error `cve`.

`min_index` The index of `lambda0_min` in `lambda0`.

References

Bai, R. (2023). "Bayesian group regularization in generalized linear models with a continuous spike-and-slab prior." *arXiv pre-print arXiv:2007.07021*.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2022). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, **117**:184-197.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(50*6), nrow=50)
n = dim(X)[1]
groups = c(1,1,1,2,2,2)
beta_true = c(-2,1,1.5,0,0,0)

## Generate responses from Gaussian distribution
Y = crossprod(t(X), beta_true) + rnorm(n)

## K-fold cross-validation
## NOTE: If you do not specify lambda0, the function will automatically choose a suitable grid.
ssgl_mods = cv_SSGL(Y, X, groups, family="gaussian", lambda0=seq(from=16,to=4,by=-4))

## Plot cross-validation curve
plot(ssgl_mods$lambda0, ssgl_mods$cve, type="l", xlab="lambda0", ylab="CVE")
## lambda which minimizes mean CVE
ssgl_mods$lambda0_min
ssgl_mods$min_index


## Example with Poisson regression

## Generate count responses
eta = crossprod(t(X), beta_true)
Y = rpois(n,exp(eta))

## K-fold cross-validation
## NOTE: If you do not specify lambda0, the program will automatically choose a suitable grid.
ssgl_poisson_mods = cv_SSGL(Y, X, groups, family="poisson", lambda0=seq(from=20,to=2,by=-4))

## Plot cross-validation curve
plot(ssgl_poisson_mods$lambda0, ssgl_poisson_mods$cve, type="l", xlab="lambda0", ylab="CVE")
## lambda which minimizes mean CVE
ssgl_poisson_mods$lambda0_min
ssgl_poisson_mods$min_index
```

gamma_grpreg *Group-regularized Gamma Regression*

Description

This function implements group-regularized gamma regression with a known shape parameter ν and the log link. In gamma regression, we assume that $y_i \sim Gamma(\mu_i, \nu)$, where

$$f(y_i|\mu_i, \nu) = \frac{1}{\Gamma(\nu)}(\frac{\nu}{\mu_i})^\nu \exp(-\frac{\nu}{\mu_i} y_i) y_i^{\nu-1}, y > 0.$$

Then $E(y_i) = \mu_i$, and we relate μ_i to a set of p covariates x_i through the log link,

$$\log(\mu_i) = \beta_0 + x_i^T \beta, i = 1, ..., n$$

If the covariates in each x_i are grouped according to known groups $g = 1, ..., G$, then this function can estimate some of the G groups of coefficients as all zero, depending on the amount of regularization. Our implementation for regularized gamma regression is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than sample size.

In addition, this function has the option of returning the generalized information criterion (GIC) of Fan and Tang (2013) for each regularization parameter in the grid `lambda`. The GIC can be used for model selection and serves as a useful alternative to cross-validation.

Usage

```
gamma_grpreg(Y, X, groups, X_test, gamma_shape=1,
             penalty=c("gLASSO","gSCAD","gMCP"),
             group_weights, taper, n_lambda=100, lambda,
             max_iter=10000, tol=1e-4, return_GIC=TRUE)
```

Arguments

`Y` $n \times 1$ vector of strictly positive, continuous responses for training data.

`X` $n \times p$ design matrix for training data, where the jth column corresponds to the jth overall feature.

`groups` p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the factor level name that the feature in the jth column of `X` belongs to. `groups` must be either a vector of integers or factors.

`X_test` $n_{test} \times p$ design matrix for test data to calculate predictions. `X_test` must have the *same* number of columns as `X`, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X_test=X` in order to calculate *in-sample* predictions.

`gamma_shape` known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for the responses. Default is `gamma_shape=1`.

`penalty` group regularization method to use on the groups of regression coefficients. The options are `"gLASSO"`, `"gSCAD"`, `"gMCP"`. To implement gamma regression with the SSGL penalty, use the `SSGL` function.

`group_weights` group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.

`taper` tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is `taper=4` for group SCAD and `taper=3` for group MCP. Ignored if `"gLASSO"` is specified as the penalty.

`n_lambda` number of regularization parameters L. Default is `n_lambda=100`.

`lambda` grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

`max_iter` maximum number of iterations in the algorithm. Default is `max_iter=10000`.

`tol` convergence threshold for algorithm. Default is `tol=1e-4`.

`return_GIC` Boolean variable for whether or not to return the GIC. Default is `return_GIC=TRUE`.

Value

The function returns a list containing the following components:

`lambda` $L \times 1$ vector of regularization parameters `lambda` used to fit the model. `lambda` is displayed in descending order.

`beta` $p \times L$ matrix of estimated regression coefficients. The kth column in `beta` corresponds to the kth regularization parameter in `lambda`.

`beta0` $L \times 1$ vector of estimated intercepts. The kth entry in `beta0` corresponds to the kth regularization parameter in `lambda`.

`classifications`
$G \times L$ matrix of classifications, where G is the number of groups. An entry of "1" indicates that the group was classified as nonzero, and an entry of "0" indicates that the group was classified as zero. The kth column of `classifications` corresponds to the kth regularization parameter in `lambda`.

`Y_pred` $n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in `X_test` (or training data `X` if no argument was specified for `X_test`). The kth column in `Y_pred` corresponds to the predictions for the kth regularization parameter in `lambda`.

`GIC` $L \times 1$ vector of GIC values. The kth entry of `GIC` corresponds to the kth entry in our `lambda` grid. This is not returned if `return_GIC=FALSE`.

`lambda_min` The value in `lambda` that minimizes `GIC`. This is not returned if `return_GIC=FALSE`.

`min_index` The index of `lambda_min` in `lambda`. This is not returned if `return_GIC=FALSE`.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Fan, Y. and Tang, C. Y. (2013). "Tuning parameter selection in high dimensional penalized likelihood." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**:531-552.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Yuan, M. and Lin, Y. (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**:49-67.

Examples

```
## Generate data
set.seed(1234)
X = matrix(runif(100*11), nrow=100)
n = dim(X)[1]
groups = c(1,1,1,2,2,2,3,3,4,5,5)
beta_true = c(-1,1,1,0,0,0,0,0,1.5,-1.5)

## Generate responses from gamma regression with known shape parameter 1
eta = crossprod(t(X), beta_true)
shape = 1
Y = rgamma(n, rate=shape/exp(eta), shape=shape)

## Generate test data
n_test = 50
X_test = matrix(runif(n_test*11), nrow=n_test)

## Fit gamma regression models with the group SCAD penalty
gamma_mod = gamma_grpreg(Y, X, groups, X_test, penalty=”gSCAD”)

## Tuning parameters used to fit models
gamma_mod$lambda

## Predicted n_test-dimensional vectors mu=E(Y_test) based on test data, X_test.
## The kth column of 'Y_pred' corresponds to the kth entry in 'lambda.'
gamma_mod$Y_pred

## Classifications of the 5 groups. The kth column of 'classifications'
# corresponds to the kth entry in 'lambda.'
gamma_mod$classifications

## Plot lambda vs. GIC
plot(gamma_mod$lambda, gamma_mod$GIC, type='l')

## Model selection with the lambda that minimizes GIC
gamma_mod$lambda_min
gamma_mod$min_index
gamma_mod$classifications[, gamma_mod$min_index]
gamma_mod$beta[, gamma_mod$min_index]
```

nb_grpreg *Group-regularized Negative Binomial Regression*

Description

This function implements group-regularized negative binomial regression with a known size parameter α and the log link. In negative binomial regression, we assume that $y_i \sim NB(\alpha, \mu_i)$, where

$$f(y_i|\alpha,\mu_i) = \frac{\Gamma(y_i+\alpha)}{y_i!\Gamma(\alpha)}(\frac{\mu_i}{\mu_i+\alpha})^{y_i}(\frac{\alpha}{\mu_i+\alpha})^{\alpha}, y_i = 0,1,2,...$$

Then $E(y_i) = \mu_i$, and we relate μ_i to a set of p covariates x_i through the log link,

$$\log(\mu_i) = \beta_0 + x_i^T\beta, i = 1,...,n$$

If the covariates in each x_i are grouped according to known groups $g = 1,...,G$, then this function can estimate some of the G groups of coefficients as all zero, depending on the amount of regularization. Our implementation for regularized negative binomial regression is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than sample size.

In addition, this function has the option of returning the generalized information criterion (GIC) of Fan and Tang (2013) for each regularization parameter in the grid `lambda`. The GIC can be used for model selection and serves as a useful alternative to cross-validation.

Usage

```
nb_grpreg(Y, X, groups, X_test, nb_size=1, penalty=c("gLASSO","gSCAD","gMCP"),
          group_weights, taper, n_lambda=100, lambda,
          max_iter=10000, tol=1e-4, return_GIC=TRUE)
```

Arguments

`Y` $n \times 1$ vector of strictly nonnegative integer responses for training data.

`X` $n \times p$ design matrix for training data, where the jth column corresponds to the jth overall feature.

`groups` p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the factor level name that the feature in the jth column of `X` belongs to. `groups` must be either a vector of integers or factors.

`X_test` $n_{test} \times p$ design matrix for test data to calculate predictions. `X_test` must have the *same* number of columns as `X`, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X_test=X` in order to calculate *in-sample* predictions.

`nb_size` known size parameter α in $NB(\alpha,\mu_i)$ distribution for the responses. Default is `nb_size=1`.

`penalty` group regularization method to use on the groups of regression coefficients. The options are `"gLASSO"`, `"gSCAD"`, `"gMCP"`. To implement gamma regression with the SSGL penalty, use the `SSGL` function.

`group_weights` group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.

`taper` tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is `taper=4` for group SCAD and `taper=3` for group MCP. Ignored if `"gLASSO"` is specified as the penalty.

`n_lambda` number of regularization parameters L. Default is `n_lambda=100`.

`lambda` grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

`max_iter` maximum number of iterations in the algorithm. Default is `max_iter=10000`.

`tol` convergence threshold for algorithm. Default is `tol=1e-4`.

`return_GIC` Boolean variable for whether or not to return the GIC. Default is `return_GIC=TRUE`.

Value

The function returns a list containing the following components:

`lambda` $L \times 1$ vector of regularization parameters `lambda` used to fit the model. `lambda` is displayed in descending order.

`beta` $p \times L$ matrix of estimated regression coefficients. The kth column in `beta` corresponds to the kth regularization parameter in `lambda`.

`beta0` $L \times 1$ vector of estimated intercepts. The kth entry in `beta0` corresponds to the kth regularization parameter in `lambda`.

`classifications` $G \times L$ matrix of classifications, where G is the number of groups. An entry of "1" indicates that the group was classified as nonzero, and an entry of "0" indicates that the group was classified as zero. The kth column of `classifications` corresponds to the kth regularization parameter in `lambda`.

`Y_pred` $n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in `X_test` (or training data `X` if no argument was specified for `X_test`). The kth column in `Y_pred` corresponds to the predictions for the kth regularization parameter in `lambda`.

`GIC` $L \times 1$ vector of GIC values. The kth entry of `GIC` corresponds to the kth entry in our `lambda` grid. This is not returned if `return_GIC=FALSE`.

`lambda_min` The value in `lambda` that minimizes `GIC`. This is not returned if `return_GIC=FALSE`.

`min_index` The index of `lambda_min` in `lambda`. This is not returned if `return_GIC=FALSE`.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Fan, Y. and Tang, C. Y. (2013). "Tuning parameter selection in high dimensional penalized likelihood." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**:531-552.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Yuan, M. and Lin, Y. (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**:49-67.

Examples

```
## Generate data
set.seed(1234)
X = matrix(runif(100*15), nrow=100)
n = dim(X)[1]
groups = c(”A”,”A”,”A”,”A”,”B”,”B”,”B”,”B”,”C”,”C”,”D”,”D”,”E”,”E”,”E”)
groups = as.factor(groups)
beta_true = c(-1.5,1.5,-1.5,1.5,0,0,0,0,0,0,2,-2,0,0,0)

## Generate count responses from negative binomial regression
eta = crossprod(t(X), beta_true)
Y = rnbinom(n,size=1, mu=exp(eta))

## Generate test data
n_test = 50
X_test = matrix(runif(n_test*15), nrow=n_test)

## Fit negative binomial regression models with the group MCP penalty
nb_mod = nb_grpreg(Y, X, groups, X_test, penalty=”gMCP”)

## Tuning parameters used to fit models
nb_mod$lambda

# Predicted n_test-dimensional vectors mu=E(Y_test) based on test data, X_test.
# The kth column of 'Y_pred' corresponds to the kth entry in 'lambda.'
nb_mod$Y_pred

# Classifications of the 8 groups. The kth column of 'classifications'
# corresponds to the kth entry in lambda.
nb_mod$classifications

## Plot lambda vs. GIC
plot(nb_mod$lambda, nb_mod$GIC, type='l')

## Model selection with the lambda that minimizes GIC
nb_mod$lambda_min
nb_mod$min_index
nb_mod$classifications[, nb_mod$min_index]
nb_mod$beta[, nb_mod$min_index]
```

SSGL *Spike-and-Slab Group Lasso for Group-Regularized Generalized Linear Models (GLMs)*

Description

This is a function to implement group-regularized GLMs with the spike-and-slab group lasso (SSGL) penalty of Bai et al. (2022) and Bai (2023). The identity link function is used for Gaussian regression, the logit link is used for binomial regression, and the log link is used for Poisson, negative binomial, and gamma regression. If the covariates in each x_i are grouped according to known groups $g = 1, ..., G$, then this function can estimate some of the G groups of coefficients as all zero, depending on the amount of regularization.

In addition, this function has the option of returning the generalized information criterion (GIC) of Fan and Tang (2013) for each regularization parameter in the grid `lambda0`. The GIC can be used for model selection and serves as a useful alternative to cross-validation. The formula for the GIC and a given λ_0 is

$$DIC(\lambda_0) = \frac{1}{n} Deviance_{\lambda_0} + a_n \times \nu),$$

where $Deviance_{\lambda_0}$ is the deviance computed with the estimate of `beta` based on spike hyperparameter λ_0, ν_0 is the number of nonzero elements in the estimated `beta`, and a_n is a sequence that diverges at a suitable rate relative to n. As recommended by Fan and Tang (2013), we set $a_n = \{\log(\log(n))\} \log(p)$.

Usage

```
SSGL(Y, X, groups,
     family=c("gaussian","binomial","poisson","negativebinomial","gamma"),
     X_test, nb_size=1, gamma_shape=1, group_weights, n_lambda0=25,
     lambda0, lambda1=1, a=1, b=dim(X)[2],
     max_iter=100, tol = 1e-6, return_GIC=TRUE, print_lambda0=TRUE)
```

Arguments

`Y` $n \times 1$ vector of responses for training data.

`X` $n \times p$ design matrix for training data, where the jth column of `X` corresponds to the jth overall covariate.

`groups` p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the factor level name that the feature in the jth column of `X` belongs to. `groups` must be either a vector of integers or factors.

`family` exponential dispersion family of the response variables. Allows for `"gaussian"`, `"binomial"`, `"poisson"`, `"negativebinomial"`, and `"gamma"`. Note that for `"negativebinomial"`, the size parameter must be specified in advance, while for `"gamma"`, the shape parameter must be specified in advance.

`X_test` $n_{test} \times p$ design matrix for test data to calculate predictions. `X_test` must have the *same* number of columns as X, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X_test=X` in order to calculate *in-sample* predictions.

`nb_size` known size parameter α in $NB(\alpha, \mu_i)$ distribution for the responses if the user specifies `family="negativebinomial"`. Default is `nb_size=1`. Ignored if `family` is not `"gamma"`.

`gamma_shape` known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for the responses if the user specifies `family="gamma"`. Default is `gamma_shape=1`.

`group_weights` group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.

n_lambda0 number of spike hyperparameters L. Default is n_lambda0=25.

lambda0 grid of L spike hyperparameters λ_0. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

lambda1 slab hyperparameter λ_1 in the SSGL prior. Default is lambda1=1.

a shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is a=1.

b shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is b=dim(X)[2].

max_iter maximum number of iterations in the algorithm. Default is max_iter=100.

tol convergence threshold for algorithm. Default is tol=1e-6.

return_GIC Boolean variable for whether or not to return the GIC. Default is return_GIC=TRUE.

print_lambda0 Boolean variable for whether or not to print the current value in lambda0. Default is print_lambda0=TRUE.

Value

The function returns a list containing the following components:

lambda0 $L \times 1$ vector of spike hyperpameters lambda0 used to fit the model. lambda0 is displayed in descending order.

beta $p \times L$ matrix of estimated regression coefficients. The kth column in beta corresponds to the kth spike hyperparameter in lambda0.

beta0 $L \times 1$ vector of estimated intercepts. The kth entry in beta0 corresponds to the kth spike hyperparameter in lambda0.

classifications

$G \times L$ matrix of classifications, where G is the number of groups. An entry of "1" indicates that the group was classified as nonzero, and an entry of "0" indicates that the group was classified as zero. The kth column of classifications corresponds to the kth spike hyperparameter in lambda0.

Y_pred $n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in X_test (or training data X if no argument was specified for X_test). The kth column in Y_pred corresponds to the predictions for the kth spike hyperparameter in lambda0.

GIC $L \times 1$ vector of GIC values. The kth entry of GIC corresponds to the kth entry in our lambda0 grid. This is not returned if return_GIC=FALSE.

lambda0_min The value in lambda0 that minimizes GIC. This is not returned if return_GIC=FALSE.

min_index The index of lambda0_min in lambda0. This is not returned if return_GIC=FALSE.

References

Bai, R. (2023). "Bayesian group regularization in generalized linear models with a continuous spike-and-slab prior." *arXiv pre-print arXiv:2007.07021.*

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2022). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, **117**:184-197.

Fan, Y. and Tang, C. Y. (2013). "Tuning parameter selection in high dimensional penalized likelihood." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**:531-552.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*10), nrow=100)
n = dim(X)[1]
groups = c("A","A","A","B","B","B","C","C","D","D")
groups = as.factor(groups)
beta_true = c(-2.5,1.5,1.5,0,0,0,2,-2,0,0)

## Generate responses from Gaussian distribution
Y = crossprod(t(X), beta_true) + rnorm(n)

## Generate test data
n_test = 50
X_test = matrix(runif(n_test*10), nrow=n_test)

## Fit SSGL model with 10 spike hyperparameters
## NOTE: If you do not specify lambda0, the program will automatically choose a suitable grid.
SSGL_mod = SSGL(Y, X, groups, family="gaussian", X_test, lambda0=seq(from=50,to=5,by=-5))

## Regression coefficient estimates
SSGL_mod$beta

## Predicted n_test-dimensional vectors mu=E(Y.test) based on test data, X_test.
## The kth column of 'Y_pred' corresponds to the kth entry in 'lambda.'
SSGL_mod$Y_pred

## Classifications of the 8 groups. The kth column of 'classifications'
## corresponds to the kth entry in 'lambda.'
SSGL_mod$classifications

## Plot lambda vs. GIC
plot(SSGL_mod$lambda0, SSGL_mod$GIC, type='l')

## Model selection with the lambda that minimizes GIC
SSGL_mod$lambda0_min
SSGL_mod$min_index
SSGL_mod$classifications[, SSGL_mod$min_index]
SSGL_mod$beta[, SSGL_mod$min_index]



## Example with binary logistic regression

set.seed(12345)
```

```
X = matrix(runif(100*8), nrow=100)
n = dim(X)[1]
groups = c(”A”,”A”,”A”,”B”,”B”,”B”,”C”,”C”)
groups = as.factor(groups)
beta_true = c(-2.5,1.5,1.5,0,0,0,2,-2)

## Generate binary responses
eta = crossprod(t(X), beta_true)
Y = rbinom(n, size=1, prob=1/(1+exp(-eta)))

## Generate test data
n_test = 50
X_test = matrix(runif(n_test*8), nrow=n_test)

## Fit SSGL logistic regression model with 10 spike hyperparameters
## NOTE: If you do not specify lambda0, the program will automatically choose a suitable grid.
SSGL_logistic_mod = SSGL(Y, X, groups, family=”binomial”, X_test, lambda0=seq(from=10,to=1,by=-1.5))

## Regression coefficient estimates
SSGL_logistic_mod$beta

## Predicted n_test-dimensional vectors mu=E(Y_test) based on test data, X_test.
## The kth column of 'Y_pred' corresponds to the kth entry in 'lambda.'
SSGL_logistic_mod$Y_pred

## Classifications of the 8 groups. The kth column of 'classifications'
## corresponds to the kth entry in 'lambda.'
SSGL_logistic_mod$classifications

## Plot lambda vs. GIC
plot(SSGL_logistic_mod$lambda0, SSGL_logistic_mod$GIC, type='l')

## Model selection with the lambda that minimizes GIC
SSGL_logistic_mod$lambda0_min
SSGL_logistic_mod$min_index
SSGL_logistic_mod$classifications[, SSGL_logistic_mod$min_index]
SSGL_logistic_mod$beta[, SSGL_logistic_mod$min_index]
```

Index