Package ‘gesso’

October 13, 2022

Type Package
Title Hierarchical GxE Interactions in a Regularized Regression Model
Version 1.0.2
Date 2021-11-28
Author Natalia Zemlianskaia
Maintainer Natalia Zemlianskaia <natasha.zemlianskaia@gmail.com>
Description The method focuses on a single environmental exposure and induces a main-effect-before-interaction hierarchical structure for the joint selection of interaction terms in a regularized regression model. For details see Zemlianskaia et al. (2021) <arxiv:2103.13510>.
License MIT + file LICENSE
Imports Rcpp (>= 1.0.3), Matrix, bigmemory, methods
Depends dplyr, R (>= 3.5)
Suggests glmnet, testthat, knitr, rmarkdown, ggplot2
LinkingTo Rcpp, RcppEigen, RcppThread, BH, bigmemory
VignetteBuilder knitr
NeedsCompilation yes
Repository CRAN
Date/Publication 2021-11-30 07:30:02 UTC

R topics documented:

gesso-package ......................................................... 2
data.gen .......................................................... 2
gesso.coef ..................................................... 4
gesso.coefnum ............................................... 5
gesso.cv ....................................................... 6
gesso.fit ....................................................... 7
gesso.predict ............................................... 9
selection.metrics .............................................. 10

Index 12
Hierarchical GxE Interactions in a Regularized Regression Model

Description

The method focuses on a single environmental exposure and induces a main-effect-before-interaction hierarchical structure for the joint selection of interaction terms in a regularized regression model. For details see Zemlianskaia et al. (2021) <arxiv:2103.13510>.

Author(s)

Natalia Zemlianskaia

Maintainer: Natalia Zemlianskaia <natasha.zemlianskaia@gmail.com>

References


---

Data Generation

Description

Generates genotypes data matrix G (sample_size by p), vector of environmental measurements E, and an outcome vector Y of size sample_size. Simulates training, validation, and test datasets.

Usage

data.gen(sample_size = 100, p = 20, n_g_non_zero = 15, n_gxe_non_zero = 10, family = "gaussian", mode = "strong hierarchical", normalize = FALSE, normalize_response = FALSE, seed = 1, pG = 0.2, pE = 0.3, n_confounders = NULL)

Arguments

- **sample_size**: sample size of the data
- **p**: total number of main effects
- **n_g_non_zero**: number of non-zero main effects to generate
- **n_gxe_non_zero**: number of non-zero interaction effects to generate
- **family**: "gaussian" for continuous outcome Y and "binomial" for binary 0/1 outcome
mode  
either "strong_hierarchical", "hierarchical", or "anti_hierarchical". In the strong
hierarchical mode the hierarchical structure is maintained (beta_g = 0 then
beta_gxe = 0) and also |beta_g| >= |beta_gxe|. In the hierarchical mode the hier-
ararchical structure is maintained, but |beta_G| < |beta_gxe|. In the anti_hierarchical
mode the hierarchical structure is violated (beta_g = 0 then beta_gxe != 0).

normalize  
TRUE to normalize matrix G and vector E

normalize_response  
TRUE to normalize vector Y

pG  
genotypes prevalence, value from 0 to 1

pE  
environment prevalence, value from 0 to 1

seed  
random seed

n_confounders  
number of confounders to generate, either NULL or >1

Value

A list of simulated datasets and generating coefficients

G_train, G_valid, G_test  
generated genotypes matrices

E_train, E_valid, E_test  
generated vectors of environmental values

Y_train, Y_valid, Y_test  
generated outcome vectors

C_train, C_valid, C_test  
generated confounders matrices

GxE_train, GxE_valid, GxE_test  
generated GxE matrix

Beta_G  
main effect coefficients vector

Beta_GxE  
interaction coefficients vector

beta_0  
intercept coefficient value

beta_E  
environment coefficient value

Beta_C  
confounders coefficient values

index_beta_non_zero, index_beta_gxe_non_zero, index_beta_zero, index_beta_gxe_zero  
inner data generation variables

n_g_non_zero  
number of non-zero main effects generated

n_gxe_non_zero  
number of non-zero interactions generated

n_total_non_zero  
total number of non-zero variables

SNR_g  
signal-to-noise ratio for the main effects

SNR_gxe  
signal-to-noise ratio for the interactions

family, p, sample_size, mode, seed  
input simulation parameters
gesso.coef

Get model coefficients

Description
A function to obtain coefficients from the model fit object corresponding to the desired pair of tuning parameters $\lambda = (\lambda_1, \lambda_2)$.

Usage

```r
gesso.coef(fit, lambda)
```

Arguments

- `fit`: model fit object obtained either by using function `gesso.fit` or `gesso.cv`
- `lambda`: a pair of tuning parameters organized in a tibble (ex: `lambda = tibble(lambda_1=grid[1], lambda_2=grid[1])`)

Value

A list of model coefficients corresponding to lambda values of tuning parameters

- `beta_0`: estimated intercept value
- `beta_e`: estimated environmental coefficient value
- `beta_g`: a vector of estimated main effect coefficients
- `beta_c`: a vector of estimated confounders coefficients
- `beta_gxe`: a vector of estimated interaction coefficients

Examples

```r
data = data.gen()
model = gesso.cv(data$G_train, data$E_train, data$Y_train, grid_size=20, parallel=TRUE, nfolds=3)
gxe_coefficients = gesso.coef(model$fit, model$lambda_min)$beta_gxe
g_coefficients = gesso.coef(model$fit, model$lambda_min)$beta_g
```
gesso.coefnum

Get model coefficients with specified number of non-zero interactions

Description
A function to obtain coefficients with target_b_gxe_non_zero specified to control the desired sparsity of interactions in the model.

Usage

\[
gesso.coefnum(cv\_model, target\_b\_gxe\_non\_zero, \text{less\_than} = \text{TRUE})
\]

Arguments

- `cv_model`: cross-validated model fit object obtained by using function `gesso.cv`
- `target_b_gxe_non_zero`: number of non-zero interactions we want to include in the model
- `less_than`: TRUE if we want to control a number of at most non-zero interactions, FALSE if we want to control a number of at least non-zero interactions

Value
A list of model coefficients corresponding to the best model that contains at most or at least target_b_gxe_non_zero non-zero interaction terms.

The target model is selected based on the averaged cross-validation (cv) results: for each pair of parameters \(\lambda=(\lambda_1, \lambda_2)\) in the grid and each cv fold we obtain a number of non-zero estimated interaction terms, then average cv results by \(\lambda\) and choose the tuning parameters corresponding to the minimum average cv loss that have at most or at least target_b_gxe_non_zero non-zero interaction terms. Returned coefficients are obtained by fitting the model on the full data with the selected tuning parameters.

Note that the number of estimated non-zero interactions will only approximately reflect the numbers obtained on cv datasets.

- `beta_0`: estimated intercept value
- `beta_e`: estimated environmental coefficient value
- `beta_g`: a vector of estimated main effect coefficients
- `beta_gxe`: a vector of estimated interaction coefficients
- `beta_c`: a vector of estimated confounders coefficients

Examples

```r
data = data.gen()
model = gesso.cv(data$G_train, data$E_train, data$Y_train)
model_coefficients = gesso.coefnum(model, 5)
gxe_coefficients = model_coefficients$beta_gxe; sum(gxe_coefficients!=0)
```
gesso.cv  

Cross-Validation

Description
Perform nfolds-fold cross-validation to tune hyperparameters $\lambda_1$ and $\lambda_2$ for the gesso model.

Usage
```
gesso.cv(G, E, Y, C = NULL, normalize = TRUE, normalize_response = FALSE, grid = NULL,
    grid_size = 20, grid_min_ratio = NULL, alpha = NULL, family = "gaussian",
    type_measure = "loss", fold_ids = NULL, nfolds = 4,
    parallel = TRUE, seed = 42, tolerance = 1e-3, max_iterations = 5000,
    min_working_set_size = 100, verbose = TRUE)
```

Arguments
- **G**: matrix of main effects of size $n \times p$, variables organized by columns
- **E**: vector of environmental measurements
- **Y**: outcome vector. Set family="gaussian" for the continuous outcome and family="binomial" for the binary outcome with 0/1 levels
- **C**: matrix of confounders of size $n \times m$, variables organized by columns
- **normalize**: TRUE to normalize matrix G and vector E
- **normalize_response**: TRUE to normalize vector Y (for family="gaussian")
- **grid**: grid sequence for tuning hyperparameters, we use the same grid for $\lambda_1$ and $\lambda_2$
- **grid_size**: specify grid_size to generate grid automatically. Grid is generated by calculating max_lambda from the data (smallest lambda such that all the coefficients are zero). min_lambda is calculated as a product of max_lambda and grid_min_ratio. The program then generates grid_size values equidistant on the log10 scale from min_lambda to max_lambda
- **grid_min_ratio**: parameter to determine min_lambda (smallest value for the grid of lambdas), default is 0.1 for $p > n$, 0.01 otherwise
- **alpha**: if NULL independent 2D grid is used for ($\lambda_1$, $\lambda_2$), else 1D grid is used where $\lambda_2 = \alpha \times \lambda_1$, i.e. ($\lambda_1$, $\alpha \times \lambda_1$)
- **family**: "gaussian" for continuous outcome and "binomial" for binary
- **type_measure**: loss to use for cross-validation. Specify type_measure="loss" for negative log likelihood or type_measure="auc" for AUC (for family="binomial" only)
- **fold_ids**: option to input custom folds assignments
- **tolerance**: tolerance for the dual gap convergence criterion
- **max_iterations**: maximum number of iterations
min_working_set_size
minimum size of the working set

nfolds
number of cross-validation splits

parallel
TRUE to enable parallel cross-validation

seed
set random seed to control random folds assignments

verbose
TRUE to print messages

Value
A list of objects

cv_result
table with cross-validation results: averaged across folds loss and the number of non-zero coefficients for each value of \((\lambda_1, \lambda_2)\) path. Could be used for custom parameters tuning (ex: select \((\lambda_1, \lambda_2)\) with a certain number of non-zero main effects and/or a certain number of interactions).

- mean_loss averaged across folds loss value, vector of size \(\lambda_1 \times \lambda_2\)
- mean_beta_g_nonzero averaged across folds number of non-zero main effects, vector of size \(\lambda_1 \times \lambda_2\)
- mean_beta_gxe_nonzero averaged across folds number of non-zero interactions, vector of size \(\lambda_1 \times \lambda_2\)
- \(\lambda_1\) \(\lambda_2\) pass, decreasing
- \(\lambda_2\) \(\lambda_2\) pass, oscillating

lambda_min
table of optimal \((\lambda_1, \lambda_2)\) values, tuning parameter values that give minimum cross-validation loss (mean_loss)

fit
list, return of the function gesso.fit on the full data

ggrid
vector of values used for hyperparameters tuning

full_cv_result inner variables

Examples

data = data.gen()
tune_model = gesso.cv(data$G_train, data$E_train, data$Y_train,
grid_size=20, parallel=TRUE, nfolds=3)
gxe_coefficients = gesso.coef(tune_model$fit, tune_model$lambda_min)$beta_gxe
g_coefficients = gesso.coef(tune_model$fit, tune_model$lambda_min)$beta_g

Description
Fits gesso model over the two dimensional grid of hyperparameters \(\lambda_1\) and \(\lambda_2\), returns estimated coefficients for each pair of hyperparameters.
Usage

gesso.fit(G, E, Y, C = NULL, normalize = TRUE, normalize_response = FALSE, grid = NULL, grid_size = 20, grid_min_ratio = NULL, alpha = NULL, family = "gaussian", weights = NULL, tolerance = 1e-3, max_iterations = 5000, min_working_set_size = 100, verbose = FALSE)

Arguments

G matrix of main effects of size n x p, variables organized by columns
E vector of environmental measurements
Y outcome vector. Set family="gaussian" for the continuous outcome and family="binomial" for the binary outcome with 0/1 levels
C matrix of confounders of size n x m, variables organized by columns
normalize TRUE to normalize matrix G and vector E
normalize_response TRUE to normalize vector Y
grid grid sequence for tuning hyperparameters, we use the same grid for lambda_1 and lambda_2
grid_size specify grid_size to generate grid automatically. Grid is generated by calculating max_lambda from the data (smallest lambda such that all the coefficients are zero). min_lambda is calculated as a product of max_lambda and grid_min_ratio. The program then generates grid_size values equidistant on the log10 scale from min_lambda to max_lambda
grid_min_ratio parameter to determine min_lambda (smallest value for the grid of lambdas), default is 0.1 for p > n, 0.01 otherwise
alpha if NULL independent 2D grid is used for (lambda_1, lambda_2), else 1D grid is used where lambda_2 = alpha * lambda_1, i.e. (lambda_1, alpha * lambda_1)
family "gaussian" for continuous outcome and "binomial" for binary
tolerance tolerance for the dual gap convergence criterion
max_iterations maximum number of iterations
min_working_set_size minimum size of the working set
weights inner fitting parameter
verbose TRUE to print messages

Value

A list of estimated coefficients and other model fit metrics for each pair of hyperparameters (lambda_1, lambda_2)

beta_0 vector of estimated intercept values of size lambda_1*lambda_2
beta_e vector of estimated environment coefficients of size lambda_1*lambda_2
gesso.predict

Predict new outcome vector

description

Predict new outcome vector based on the new data and estimated model coefficients.

Usage

```r
  gesso.predict(beta_0, beta_e, beta_g, beta_gxe, new_G, new_E, 
                 beta_c==NULL, new_C=NULL, family = "gaussian")
```
Arguments

- \( \beta_0 \): estimated intercept value
- \( \beta_e \): estimated environmental coefficient value
- \( \beta_g \): a vector of estimated main effect coefficients
- \( \beta_{gxe} \): a vector of estimated interaction coefficients
- \( \text{new}_G \): matrix of main effects, variables organized by columns
- \( \text{new}_E \): vector of environmental measurements
- \( \beta_c \): a vector of estimated confounders coefficients
- \( \text{new}_C \): matrix of confounders, variables organized by columns
- \( \text{family} \): set \( \text{family} = \text{"gaussian"} \) for the continuous outcome and \( \text{family} = \text{"binomial"} \) for the binary outcome with 0/1 levels

Value

- Returns a vector of predicted values

Examples

```r
data = data.gen()
tune_model = gesso.cv(data$G_train, data$E_train, data$Y_train)
coefficients = gesso.coef(tune_model$fit, tune_model$lambda_min)
beta_0 = coefficients$beta_0; beta_e = coefficients$beta_e
beta_g = coefficients$beta_g; beta_gxe = coefficients$beta_gxe
new_G = data$G_test; new_E = data$E_test
new_Y = gesso.predict(beta_0, beta_e, beta_g, beta_gxe, new_G, new_E)
cor(new_Y, data$Y_test)^2
```

---

**selection.metrics**

Selection metrics

Description

Calculates principal selection metrics for the binary zero/non-zero classification problem (sensitivity, specificity, precision, auc).

Usage

```r
selection.metrics(true_b_g, true_b_gxe, estimated_b_g, estimated_b_gxe)
```

Arguments

- \( \text{true}_b_g \): vector of true main effect coefficients
- \( \text{true}_b_gxe \): vector of true interaction coefficients
- \( \text{estimated}_b_g \): vector of estimated main effect coefficients
- \( \text{estimated}_b_gxe \): vector of estimated interaction coefficients
**selection.metrics**

A list of principal selection metrics:

- **b_g_non_zero**: number of non-zero main effects
- **b_gxe_non_zero**: number of non-zero interactions
- **mse_b_g**: mean squared error for estimation of main effects effect sizes
- **mse_b_gxe**: mean squared error for estimation of interactions effect sizes
- **sensitivity_g**: recall of the non-zero main effects
- **specificity_g**: recall of the zero main effects
- **precision_g**: precision with respect to non-zero main effects
- **sensitivity_gxe**: recall of the non-zero interactions
- **specificity_gxe**: recall of the zero interactions
- **precision_gxe**: precision with respect to non-zero interactions
- **auc_g**: area under the curve for zero/non-zero binary classification problem for main effects
- **auc_gxe**: area under the curve for zero/non-zero binary classification problem for interactions

**Examples**

```r
data = data.gen()
model = gesso.cv(data$G_train, data$E_train, data$Y_train)
gxe_coefficients = gesso.coef(model$fit, model$lambda_min)$beta_gxe
g_coefficients = gesso.coef(model$fit, model$lambda_min)$beta_g
selection.metrics(data$Beta_G, data$Beta_GxE, g_coefficients, gxe_coefficients)
```
Index

* package
  gesso-package, 2

data.gen, 2

gesso (gesso-package), 2
gesso-package, 2
gesso.coef, 4
gesso.coefnum, 5
gesso.cv, 6
gesso.fit, 7
gesso.predict, 9

selection.metrics, 10