Package 'seagull'

January 19, 2020

Type Package

Title Lasso, Group Lasso, and Sparse-Group Lasso for Mixed Models

Version 1.0.3

Date 2020-01-14

Maintainer Jan Klosa <klosa@fbn-dummerstorf.de>

Description Proximal gradient descent solver for the operators lasso, group lasso, and sparse-group lasso. The implementation involves backtracking line search and warm starts. Input data needs to be clustered/grouped for the (sparse-)group lasso before calling these algorithms.

License GPL (>= 2)

URL https://github.com/jklosa/seagull

BugReports https://github.com/jklosa/seagull/issues

Depends R (>= 3.5.0)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.3)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.0.2

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Jan Klosa [aut, cre],
Noah Simon [ths],
Dörte Wittenburg [ths]

Repository CRAN

Date/Publication 2020-01-19 16:20:02 UTC

R topics documented:

`genotypes` *Simulated genotypes*

Description

A genotype matrix that contains information from single nucleotide polymorphisms (SNPs). Dimensions are 1000 rows, 466 columns. The data was simulated using the software AlphaSim of which an R-package is available. Each row corresponds to a single individual. 1000 individuals were simulated, where 10 half sib families were created. Each family consists of 100 half sibs. The half sibs share a common Sire. 466 SNPs are available, distributed over 2 chromosomes to an equal amount, i.e., the first 233 SNPs are located on chromosome 1, the remaining SNPs are on the second chromosome. The complementary homozygote genotypes are coded as 0 and 2, respectively. The heterozygote genotype as 1.

Usage

```
genotypes
```

Format

A matrix with 1000 rows and 466 columns. One row per individual. One column per SNP.

Author(s)

Jan Klosa <klosa@fbn-dummerstorf.de>

groups *Group indices for explanatory variables*

Description

A vector of integers which assigns each variable (genotype marker) to a particular group. The clustering was performed via the R package BALD. This package uses linkage disequilibrium as a measure of proximity. In total, 98 groups are available. Group sizes vary from 1 to 23. The median of group sizes is equal to 3. For more details about the distribution of group sizes, please check out the example below.

Usage

```
groups
```

Format

A vector consisting of 466 integer values.

Author(s)

Jan Klosa <klosa@fbn-dummerstorf.de>

See Also

plot

Examples

```
data(seagull_data)

## Create a vector with group sizes:
sizes <- rep(as.integer(NA), max(groups))
for (i in 1:98) {
  sizes[i] = max(which(groups==i)) - min(which(groups==i)) + 1
}

## Plot the sorted group sizes:
plot(x = seq(1, max(groups), 1), y = sort(sizes))
```

lambda_max *Maximal λ*

Description

An effective grid search for the lasso variants is based on starting with a maximal value for the penalty parameter λ. This is due to ability of the lasso variants to select features. The idea is to determine a value of λ such that any further increase of this value simply results in a solution with no selection (apart from fixed effects).

Usage

```
lambda_max_group_lasso(
  VECTOR_Y,
  VECTOR_GROUPS,
  VECTOR_WEIGHTS_FEATURES,
  VECTOR_BETA,
  MATRIX_X
)

lambda_max_lasso(VECTOR_Y, VECTOR_WEIGHTS_FEATURES, VECTOR_BETA, MATRIX_X)

lambda_max_sparse_group_lasso(
  ALPHA,
  VECTOR_Y,
  VECTOR_GROUPS,
  VECTOR_WEIGHTS_FEATURES,
  VECTOR_BETA,
  MATRIX_X
)
```

Arguments

VECTOR_Y	numeric vector of observations.
VECTOR_GROUPS	integer vector specifying which effect (fixed and random) belongs to which group.
VECTOR_WEIGHTS_FEATURES	numeric vector of weights for the vectors of fixed and random effects $[b^T, u^T]^T$.
VECTOR_BETA	numeric vector of features. At the end of this function, the random effects are initialized with zero, but the fixed effects are initialized via a least squares procedure.
MATRIX_X	numeric design matrix relating y to fixed and random effects $[XZ]$.
ALPHA	mixing parameter of the penalty terms. Satisfies: $0 < \alpha < 1$. The penalty term looks as follows: $$\alpha * "lassopenalty" + (1 - \alpha) * "grouplassopenalty".$$

Details

The value is calculated under the following prerequisites: The algorithm shall converge after a single iteration and the solution shall be equal to the initial solution. Additionally, the converged solution shall be zero for all random effects (which corresponds to "no selection".) The estimates for fixed effects shall simply remain unchanged after one iteration. Due to the explicit formulas of the proximal gradient descent algorithm, this naturally leads to a set of values for λ which guarantee to meet all the mentioned requirements. The lower bound of this set is then λ_{max}.

Particularly for the sparse-group lasso, the calculation involves to find the positive root of a non-trivial polynomial of second degree. In order to solve this, an additional bisection algorithm is implemented. (See: `seagull_bisection`.)

Value

the maximum value for the penalty parameter λ.

`lasso_variants` *Lasso, group lasso, and sparse-group lasso*

Description

Fit a mixed model with lasso, group lasso, or sparse-group lasso via proximal gradient descent. As this is an iterative algorithm, the step size for each iteration is determined via backtracking line search. A grid search for the regularization parameter λ is performed using warm starts. The mixed model has the form:

$$y = Xb + Zu + residual.$$

The penalty of the sparse-group lasso (without additional weights for features) is then:

$$\alpha\lambda||u||_1 + (1-\alpha)\lambda \sum_l \omega_l^G ||u^{(l)}||_2.$$

If $\alpha = 1$, this leads to the lasso. If $\alpha = 0$, this leads to the group lasso.

Usage

```
seagull_group_lasso(
  VECTOR_Yc,
  MATRIX_Xc,
  VECTOR_WEIGHTS_FEATURESc,
  VECTOR_GROUPS,
  VECTOR_BETAc,
  VECTOR_INDEX_PERMUTATION,
  EPSILON_CONVERGENCE,
  ITERATION_MAX,
  GAMMA,
  LAMBDA_MAX,
  PROPORTION_XI,
```

```
  NUMBER_INTERVALS,
  NUMBER_FIXED_EFFECTS,
  TRACE_PROGRESS
)

seagull_lasso(
  VECTOR_Yc,
  MATRIX_Xc,
  VECTOR_WEIGHTS_FEATURESc,
  VECTOR_BETAc,
  EPSILON_CONVERGENCE,
  ITERATION_MAX,
  GAMMA,
  LAMBDA_MAX,
  PROPORTION_XI,
  NUMBER_INTERVALS,
  NUMBER_FIXED_EFFECTS,
  TRACE_PROGRESS
)

seagull_sparse_group_lasso(
  VECTOR_Yc,
  MATRIX_Xc,
  VECTOR_WEIGHTS_FEATURESc,
  VECTOR_GROUPS,
  VECTOR_BETAc,
  VECTOR_INDEX_PERMUTATION,
  ALPHA,
  EPSILON_CONVERGENCE,
  ITERATION_MAX,
  GAMMA,
  LAMBDA_MAX,
  PROPORTION_XI,
  NUMBER_INTERVALS,
  NUMBER_FIXED_EFFECTS,
  TRACE_PROGRESS
)
```

Arguments

VECTOR_Yc	numeric vector of observations.
MATRIX_Xc	numeric design matrix relating y to fixed and random effects $[XZ]$. The columns may be permuted corresponding to their group assignments.
VECTOR_WEIGHTS_FEATURESc	numeric vector of weights for the vectors of fixed and random effects $[b^T, u^T]^T$. The entries may be permuted corresponding to their group assignments.
VECTOR_GROUPS	integer vector specifying which effect (fixed and random) belongs to which group.

VECTOR_BETAc numeric vector whose partitions will be returned (partition 1: estimates of fixed effects, partition 2: predictions of random effects). During the computation the entries may be in permuted order. But they will be returned according to the order of the user's input.

VECTOR_INDEX_PERMUTATION
integer vector that contains information about the original order of the user's input.

EPSILON_CONVERGENCE
value for relative accuracy of the solution to stop the algorithm for the current value of λ. The algorithm stops after iteration m, if:

$$||sol^{(m)} - sol^{(m-1)}||_\infty < \epsilon_c * ||sol1(m-1)||_2.$$

ITERATION_MAX maximum number of iterations for each value of the penalty parameter λ. Determines the end of the calculation if the algorithm didn't converge according to EPSILON_CONVERGENCE before.

GAMMA multiplicative parameter to decrease the step size during backtracking line search. Has to satisfy: $0 < \gamma < 1$.

LAMBDA_MAX maximum value for the penalty parameter. This is the start value for the grid search of the penalty parameter λ.

PROPORTION_XI multiplicative parameter to determine the minimum value of λ for the grid search, i.e. $\lambda_{min} = \xi * \lambda_{max}$. Has to satisfy: $0 < \xi \leq 1$. If xi=1, only a single solution for $\lambda = \lambda_{max}$ is calculated.

NUMBER_INTERVALS
number of lambdas for the grid search between λ_{max} and $\xi * \lambda_{max}$. Loops are performed on a logarithmic grid.

NUMBER_FIXED_EFFECTS
non-negative integer to determine the number of fixed effects present in the mixed model.

TRACE_PROGRESS if TRUE, a message will occur on the screen after each finished loop of the λ grid. This is particularly useful for larger data sets.

ALPHA mixing parameter of the penalty terms. Satisfies: $0 < \alpha < 1$. The penalty term looks as follows:

$$\alpha * "lassopenalty" + (1 - \alpha) * "grouplassopenalty".$$

Value

A list of estimates and parameters relevant for the computation:

fixed_effects estimates for the fixed effects b, if present in the model. Each row corresponds to a particular value of λ.

random_effects predictions for the random effects u. Each row corresponds to a particular value of λ.

lambda all values for λ which were used during the grid search.

iterations a sequence of actual iterations for each value of λ. If an occurring number is equal to max_iter, then the algorithm most likely did not converge to rel_acc during the corresponding run of the grid search.

The following parameters are also returned. But primarily for the purpose of comparison and repetition: `alpha` = `ALPHA` (only for the sparse-group lasso), `max_iter` = `ITERATION_MAX`, `gamma_bls` = `GAMMA`, `xi` = `PROPORTION_XI`, and `loops_lambda` = `NUMBER_INTERVALS`.

`phenotypes` *Simulated phenotypes*

Description

A matrix consisting of 1000 rows and 3 columns. Each row corresponds to a different individual. Each column corresponds to a different trait. The different traits were simulated to be uncorrelated to one another. The trait in the first, second, and third column have a heritability equal to 0.1, 0.3, and 0.5, respectively.

Usage

```
phenotypes
```

Format

A matrix with of 1000 rows and 3 columns. One row per individual. One column per trait.

Author(s)

Jan Klosa <klosa@fbn-dummerstorf.de>

`seagull` *Mixed model fitting with lasso, group lasso, or sparse-group lasso regularization*

Description

Fit a mixed model with lasso, group lasso, or sparse-group lasso via proximal gradient descent. As this is an iterative algorithm, the step size for each iteration is determined via backtracking line search. A grid search for the regularization parameter λ is performed using warm starts. Depending on the input parameter `alpha` this function subsequently calls one of the three implemented `lasso_variants`. None of the input variables will be centered or standardized throughout any of the calculations of this package.

Usage

```
seagull(y, X, Z, weights_u, groups, alpha, rel_acc, max_lambda, xi,
        loops_lambda, max_iter, gamma_bls, trace_progress)
```

Arguments

`y`	numeric vector of observations.
`X`	(optional) numeric design matrix relating y to fixed effects b.
`Z`	numeric design matrix relating y to random effects u.
`weights_u`	(optional) numeric vector of weights for the vector of random effects u. If no weights are passed by the user, all weights for u are set to `1`.
`groups`	(not required for the lasso) integer vector specifying which effect belongs to which group. If the lasso is called, this vector is without use and may remain empty. For group and sparse-group lasso, at least each random effect variable needs to be assigned to one group. If in this case also fixed effect variables are present in the model, but without assigned group values, then the same value will be assigned to all fixed effect variables automatically. Float or double input values will be truncated.
`alpha`	mixing parameter for the sparse-group lasso. Has to satisfy: $0 \leq \alpha \leq 1$. The penalty term looks as follows: $$\alpha *' "lassopenalty" + (1-\alpha) * "grouplassopenalty".$$ If `alpha=1`, the lasso is called. If `alpha=0`, the group lasso is called. Default value is `0.9`.
`rel_acc`	(optional) relative accuracy of the solution to stop the algorithm for the current value of λ. The algorithm stops after iteration m, if: $$\|\|sol^{(m)} - sol^{(m-1)}\|\|_\infty < rel_acc * \|\|sol1(m-1)\|\|_2.$$ Default value is `0.0001`.
`max_lambda`	(optional) maximum value for the penalty parameter. Default option is an integrated algorithm to calculate this value. This is the start value for the grid search of the penalty parameter λ. (More details about the integrated algorithms are available here: `lambda_max`.)
`xi`	(optional) multiplicative parameter to determine the minimum value of λ for the grid search, i.e. $\lambda_{min} = \xi * \lambda_{max}$. Has to satisfy: $0 < \xi \leq 1$. If `xi=1`, only a single solution for $\lambda = \lambda_{max}$ is calculated. Default value is `0.01`.
`loops_lambda`	(optional) number of lambdas for the grid search between λ_{max} and $\xi * \lambda_{max}$. Loops are performed on a logarithmic grid. Float or double input values will be truncated. Default value is `50`.
`max_iter`	(optional) maximum number of iterations for each value of the penalty parameter λ. Determines the end of the calculation if the algorithm didn't converge according to `rel_acc` before. Float or double input values will be truncated. Default value is `1000`.
`gamma_bls`	(optional) multiplicative parameter to decrease the step size during backtracking line search. Has to satisfy: $0 < \gamma_{bls} < 1$. Default value is `0.8`.
`trace_progress`	(optional) if `TRUE`, a message will occur on the screen after each finished loop of the λ grid. This is particularly useful for larger data sets.

Details

The underlying mixed model has the form:

$$y = Xb + Zu + residual,$$

where b is the vector of fixed effects and u is the vector of random effects. The penalty of the sparse-group lasso (without additional weights for features) is then:

$$\alpha\lambda||u||_1 + (1-\alpha)\lambda \sum_l \omega_l^G ||u^{(l)}||_2.$$

The variable ω_l^G is a weight for group l. If $\alpha = 1$, this leads to the lasso. If $\alpha = 0$, this leads to the group lasso.

The above penalty can be enhanced by introducing additional weights ω^F for features in the lasso penalty:

$$\alpha\lambda \sum_j |\omega_j^F u_j| + (1-\alpha)\lambda \sum_l \omega_l^G ||u^{(l)}||_2.$$

The group weights ω_l^G are implemented as follows: $\omega_l^G = \sum_{j,j\in G} \sqrt{\omega_j^F}$. So, in the case that the weights for features in a particular group are all equal to one, this term collapses to the square root of the group size.

In addition, the above penalty can be formally rewritten by including the fixed effects and assigning weights equal to zero to all these features. This is how the algorithms are implemented. Consequently, if a weight for any random effect is set to zero by the user, the function drops an error and the calculation will not start.

Value

A list of estimates and parameters relevant for the computation:

fixed_effects estimates for the fixed effects b, if present in the model. Each row corresponds to a particular value of λ.

random_effects predictions for the random effects u. Each row corresponds to a particular value of λ.

lambda all values for λ which were used during the grid search.

iterations a sequence of actual iterations for each value of λ. If an occurring number is equal to `max_iter`, then the algorithm most likely did not converge to `rel_acc` during the corresponding run of the grid search.

The following parameters are also returned. But primarily for the purpose of comparison and repetition: `alpha` (only for the sparse-group lasso), `max_iter`, `gamma_bls`, `xi`, and `loops_lambda`.

Author(s)

Jan Klosa

See Also

`plot`

Examples

```
set.seed(62)
n <- 50  ## observations
p <- 8   ## variables

## Create a deisng matrix X for fixed effects to model the intercept:
X <- matrix(1, nrow = n, ncol = 1)

## Create a design matrix Z for random effects:
Z <- matrix(rnorm(p * n, mean = 0, sd = 1), nrow = n)

## Intercept b, random effect vector u, and response y:
b <- 0.2
u <- c(0, 1.5, 0, 0.5, 0, 0, -2, 1)
y <- X%*%b + Z%*%u + rnorm(n, mean = 0, sd = 1)

## Create a vector of three groups corresponding to vector u:
group_indices <- c(1L, 2L, 2L, 2L, 1L, 1L, 3L, 1L)

## Calculate the solution:
fit_l   <- seagull(y = y, X = X, Z = Z, alpha = 1.0)
fit_gl  <- seagull(y = y, X = X, Z = Z, alpha = 0.0, groups = group_indices)
fit_sgl <- seagull(y = y, X = X, Z = Z, groups = group_indices)

## Combine the estimates for fixed and random effects:
estimates_l   <- cbind(fit_l$fixed_effects, fit_l$random_effects)
estimates_gl  <- cbind(fit_gl$fixed_effects, fit_gl$random_effects)
estimates_sgl <- cbind(fit_sgl$fixed_effects, fit_sgl$random_effects)
true_effects  <- c(b, u)

## Calculate mean squared errors along the solution paths:
MSE_l   <- rep(as.numeric(NA), fit_l$loops_lambda)
MSE_gl  <- rep(as.numeric(NA), fit_l$loops_lambda)
MSE_sgl <- rep(as.numeric(NA), fit_l$loops_lambda)

for (i in 1:fit_l$loops_lambda) {
  MSE_l[i] <- t(estimates_l[i,] - true_effects)%*%(estimates_l[i,] - true_effects)/(p+1)
 MSE_gl[i] <- t(estimates_gl[i,] - true_effects)%*%(estimates_gl[i,] - true_effects)/(p+1)
 MSE_sgl[i] <- t(estimates_sgl[i,] - true_effects)%*%(estimates_sgl[i,] - true_effects)/(p+1)
}

## Plot a fraction of the results of the MSEs:
plot(x = seq(1, fit_l$loops_lambda, 1)[25:50], MSE_l[25:50], type = "l", lwd = 2)
points(x = seq(1, fit_l$loops_lambda, 1)[25:50], MSE_gl[25:50], type = "l", lwd = 2, col = "blue")
points(x = seq(1, fit_l$loops_lambda, 1)[25:50], MSE_sgl[25:50], type = "l", lwd = 2, col = "red")



## A larger example with simulated genetic data:
data(seagull_data)

fit_l1 <- seagull(y = phenotypes[,1], Z = genotypes, alpha = 1.0)
```

```
fit_l2 <- seagull(y = phenotypes[,2], Z = genotypes, alpha = 1.0)
fit_l3 <- seagull(y = phenotypes[,3], Z = genotypes, alpha = 1.0,
          trace_progress = T)

fit_gl1 <- seagull(y = phenotypes[,1], Z = genotypes, alpha = 0.0,
           groups = groups)
fit_gl2 <- seagull(y = phenotypes[,2], Z = genotypes, alpha = 0.0,
           groups = groups)
fit_gl3 <- seagull(y = phenotypes[,3], Z = genotypes, alpha = 0.0,
           groups = groups, trace_progress = T)

fit_sgl1 <- seagull(y = phenotypes[,1], Z = genotypes, groups = groups)
fit_sgl2 <- seagull(y = phenotypes[,2], Z = genotypes, groups = groups)
fit_sgl3 <- seagull(y = phenotypes[,3], Z = genotypes, groups = groups,
            trace_progress = T)
```

`seagull_bisection` *Internal bisection algorithm*

Description

This algorithm finds the smallest positive root of a polynomial of second degree in λ. Bisection is an implicit algorithm, i.e., it calls itself until a certain precision is reached.

Usage

```
seagull_bisection(
  ROWS,
  ALPHA,
  LEFT_BORDER,
  RIGHT_BORDER,
  GROUP_WEIGHT,
  VECTOR_WEIGHTS,
  VECTOR_IN
)
```

Arguments

ROWS	the length of the input vectors.
ALPHA	mixing parameter of the penalty terms. Satisfies: $0 < \alpha < 1$. The penalty term looks as follows: $$\alpha * "lassopenalty" + (1 - \alpha) * "grouplassopenalty".$$
LEFT_BORDER	value of the left border of the current interval that for sure harbors a root.
RIGHT_BORDER	value of the right border of the current interval that for sure harbors a root.

GROUP_WEIGHT	a multiplicative scalar which is part of the polynomial.
VECTOR_WEIGHTS	an input vector of multiplicative scalars which are part of the polynomial. This vector is a subset of the vector of weights for features.
VECTOR_IN	another input vector which is required to compute the value of the polynomial.

Details

The polynomial has the following form:

$$\sum_j (|vector_j| - \alpha weight_j \lambda)_+^2 - (1-\alpha)^2 weight^2 \lambda^2.$$

The polynomial is non-trivial, because summands are part of the sum if and only if the terms are non-negative.

Value

If a certain precision (TOLERANCE) is reached, this algorithm returns the center point of the current interval, in which the root is located. Otherwise, the function calls itself using half of the initial interval, in which the root is surely located.

---

seagull_data *A simulated data set to get quickly started*

Description

This data set contains a genotype matrix, phenotype vectors for three traits (stored in a matrix with three columns) and a vector of groups. The data resembles a sample of a dairy cattle population. The sample includes 1000 individuals and 466 genotypes. For the simulation, a mixed model without fixed effects was used.

Format

A data set which is based on 1000 individuals and 466 explanatory variables.

genotypes a genotype matrix that contains information from single nucleotide polymorphisms (SNPs). Dimensions are 1000 rows, 466 columns. The data was simulated using the software AlphaSim of which an R-package is available. Each row corresponds to a single individual. 1000 individuals were simulated, where 10 half sib families were created. Each family consists of 100 half sibs. The half sibs share a common Sire. 466 SNPs are available, distributed over 2 chromosomes to an equal amount, i.e., the first 233 SNPs are located on chromosome 1, the remaining SNPs are on the second chromosome. The complementary homozygote genotypes are coded as 0 and 2, respectively. The heterozygote genotype as 1.

groups a vector of integers which assigns each variable (genotype marker) to a particular group. The clustering was performed via the R package BALD. This package uses linkage disequilibrium as a measure of proximity. In total, 98 groups are available. Group sizes vary from 1 to 23. The median of group sizes is equal to 3. For more details about the distribution of the group sizes, please check out the example on this page: groups.

phenotypes a matrix consisting of 1000 rows and 3 columns. Each row corresponds to a different individual. Each column corresponds to a different trait. The different traits were simulated to be uncorrelated to one another. The trait in the first, second, and third column have a heritability equal to 0.1, 0.3, and 0.5, respectively.

Author(s)

Jan Klosa <klosa@fbn-dummerstorf.de>

Index