# Package 'prioritylasso'

## April 10, 2023

**Type** Package  
**Title** Analyzing Multiple Omics Data with an Offset Approach  
**Version** 0.3.1  
**Author** Simon Klau, Roman Hornung, Alina Bauer, Jonas Hagenberg  
**Maintainer** Roman Hornung <hornung@ibe.med.uni-muenchen.de>  
**Description** Fits successive Lasso models for several blocks of (omics) data with different priorities and takes the predicted values as an offset for the next block. Also offers options to deal with block-wise missingness in multi-omics data.  
**Depends** R (>= 3.5.0)  
**License** GPL-2  
**LazyData** TRUE  
**Imports** survival, glmnet, utils, checkmate  
**RoxygenNote** 7.2.3  
**Encoding** UTF-8  
**Suggests** testthat, knitr, rmarkdown, pROC, bookdown, ipflasso  
**VignetteBuilder** knitr  
**NeedsCompilation** no  
**Repository** CRAN  
**Date/Publication** 2023-04-10 21:00:02 UTC

### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Documentation Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate_offsets</td>
<td>2</td>
</tr>
<tr>
<td>coef.prioritylasso</td>
<td>3</td>
</tr>
<tr>
<td>compare_boolean</td>
<td>3</td>
</tr>
<tr>
<td>cvm_prioritylasso</td>
<td>4</td>
</tr>
<tr>
<td>missing.control</td>
<td>6</td>
</tr>
<tr>
<td>pl_data</td>
<td>7</td>
</tr>
<tr>
<td>predict.prioritylasso</td>
<td>8</td>
</tr>
<tr>
<td>prioritylasso</td>
<td>10</td>
</tr>
</tbody>
</table>

**Index** 15
**calculate_offsets**  
*Calculates the offsets for the current block*

**Description**

Calculates the offsets for the current block

**Usage**

```r
calculate_offsets(
  current_missings,
  current_observations,
  mcontrol,
  current_block,
  pred,
  liste,
  X,
  blocks,
  current_intercept
)
```

**Arguments**

- `current_missings`  
  index vector (indices) of current missing observations
- `current_observations`  
  index vector (indices) of current used observations
- `mcontrol`  
  control for missing data handling
- `current_block`  
  index of current block
- `pred`  
  predictions of current block
- `liste`  
  list with offsets
- `X`  
  original data
- `blocks`  
  information which variable belongs to which block
- `current_intercept`  
  the intercept estimated for the current block

**Value**

list with offsets, used imputation model and the blocks used for the imputation model (if applicable)
coef.prioritylasso

Extract coefficients from a prioritylasso object

Usage

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

Arguments

object       model of type prioritylasso
...          additional arguments, currently not used

Value

List with the coefficients and the intercepts

cmpare_boolean

Compare the rows of a matrix with a pattern

Usage

compare_boolean(object, pattern)

Arguments

object       matrix
pattern      pattern which is compared against the rows of the matrix

Value

logical vector if the pattern matches the rows
cvm_prioritylasso

prioritylasso with several block specifications

Description

Runs prioritylasso for a list of block specifications and gives the best results in terms of cv error.

Usage

cvm_prioritylasso(
  X,
  Y,
  weights,
  family,
  type.measure,
  blocks.list,
  max.coef.list = NULL,
  block1.penalization = TRUE,
  lambda.type = "lambda.min",
  standardize = TRUE,
  nfolds = 10,
  foldid,
  cvoffset = FALSE,
  cvoffsetnfolds = 10,
  ...
)

Arguments

X  a (nxp) matrix of predictors with observations in rows and predictors in columns.
Y  n-vector giving the value of the response (either continuous, numeric-binary 0/1, or Surv object).
weights  observation weights. Default is 1 for each observation.
family  should be "gaussian" for continuous Y, "binomial" for binary Y, "cox" for Y of type Surv.
type.measure  accuracy/error measure computed in cross-validation. It should be "class" (classification error) or "auc" (area under the ROC curve) if family="binomial", "mse" (mean squared error) if family="gaussian" and "deviance" if family="cox" which uses the partial-likelihood.
blocks.list  list of the format list(list(bp1=...,bp2=...), list(bp1=,...,bp2=...), ...). For the specification of the entries, see prioritylasso.
max.coef.list  list of max.coef vectors. The first entries are omitted if block1.penalization = FALSE. Default is NULL.
block1.penalization  whether the first block should be penalized. Default is TRUE.
lambda.type specifies the value of lambda used for the predictions. lambda.min gives lambda with minimum cross-validated errors. lambda.1se gives the largest value of lambda such that the error is within 1 standard error of the minimum. Note that lambda.1se can only be chosen without restrictions of max.coef.

standardize logical, whether the predictors should be standardized or not. Default is TRUE.

nfolds the number of CV procedure folds.

foldid an optional vector of values between 1 and nfold identifying what fold each observation is in.

cvoffset logical, whether CV should be used to estimate the offsets. Default is FALSE.

cvoffsetnfolds the number of folds in the CV procedure that is performed to estimate the offsets. Default is 10. Only relevant if cvoffset=TRUE.

... other arguments that can be passed to the function prioritylasso.

Value

object of class cvm_prioritylasso with the following elements. If these elements are lists, they contain the results for each penalized block of the best result.

lambda.ind list with indices of lambda for lambda.type.

lambda.type type of lambda which is used for the predictions.

lambda.min list with values of lambda for lambda.type.

min.cvm list with the mean cross-validated errors for lambda.type.

nzero list with numbers of non-zero coefficients for lambda.type.

glmnet.fit list of fitted glmnet objects.

name a text string indicating type of measure.

block1unpen if block1.penalization = FALSE, the results of either the fitted glm or coxph object.

best.blocks character vector with the indices of the best block specification.

best.blocks.indices list with the indices of the best block specification ordered by best to worst.

best.max.coef vector with the number of maximal coefficients corresponding to best.blocks.

best.model complete prioritylasso model of the best solution.

coefficients coefficients according to the results obtained with best.blocks.

call the function call.

Note

The function description and the first example are based on the R package ipflasso.

Author(s)

Simon Klau
Maintainer: Roman Hornung (<hornung@ibe.med.uni-muenchen.de>)
References


See Also

pl_data, prioritylasso, cvr2.ipflasso

Examples

cvm_prioritylasso(X = matrix(rnorm(50*500),50,500), Y = rnorm(50), family = "gaussian", type.measure = "mse", lambda.type = "lambda.min", nfolds = 5, blocks.list = list(list(bp1=1:75, bp2=76:200, bp3=201:500), list(bp1=1:75, bp2=201:500, bp3=76:200)))

## Not run:
cvm_prioritylasso(X = pl_data[,1:1028], Y = pl_data[,1029], family = "binomial", type.measure = "auc", standardize = FALSE, block1.penalization = FALSE, blocks.list = list(list(1:4, 5:9, 10:28, 29:1028), list(1:4, 5:9, 29:1028, 10:28)), max.coef.list = list(c(Inf, Inf, Inf, 10), c(Inf, Inf, 10, Inf)))

## End(Not run)

missing.control

missing.control

Construct control structures for handling of missing data for prioritylasso

Usage

missing.control(
  handle.missingdata = c("none", "ignore", "impute.offset"),
  offset.firstblock = c("zero", "intercept"),
  impute.offset.cases = c("complete.cases", "available.cases"),
  nfolds.imputation = 10,
  lambda.imputation = c("lambda.min", "lambda.1se"),
  perc.comp.cases.warning = 0.3,
  threshold.available.cases = 30,
  select.available.cases = c("maximise.blocks", "max")
)
Arguments

handle.missingdata
how blockwise missing data should be treated. Default is none which does noth-
ing. ignore ignores the observations with missing data for the current block,
impute.offset imputes the offset for the missing values.

offset.firstblock
determines if the offset of the first block for missing observations is zero or the
intercept of the observed values for handle.missingdata = ignore

impute.offset.cases
which cases/observations should be used for the imputation model to impute
missing offsets. Supported are complete cases (additional constraint is that every
observation can only contain one missing block) and all available observations
which have an overlap with the current block.

nfold.imputation
nfolds for the glmnet of the imputation model

lambda.imputation
which lambda-value should be used for predicting the imputed offsets in cv.glmnet

perc.comp.cases.warning
percentage of complete cases when a warning is issued of too few cases for the
imputation model

threshold.available.cases
if the number of available cases for impute.offset.cases = available.cases
is below this threshold, prioritylasso tries to reduce the number of blocks
taken into account for the imputation model to increase the number of observa-
tions used for the imputation model.

select.available.cases
determines how the blocks which are used for the imputation model are selected
when impute.offset.cases = available.cases. max selects the blocks that
maximise the number of observations, maximise.blocks tries to include as
many blocks as possible, starting with the blocks with the highest priority

Value

list with control parameters

---

pl_data

Simulated AML data with binary outcome

---

Description

A data set containing the binary outcome and 1028 predictor variables of 400 artificial AML pa-
tients.

Usage

pl_data
predict.prioritylasso

Format

A data frame with 400 rows and 1029 variables:

- **pl_out**: \( \text{pl_data[,1029]} \) binary outcome representing refractory status.
- **b1**: \( \text{pl_data[,1:4]} \) 4 binary variables representing variables with a known influence on the outcome.
- **b2**: \( \text{pl_data[,5:9]} \) 5 continuous variables representing clinical variables.
- **b3**: \( \text{pl_data[,10:28]} \) 19 binary variables representing mutations.
- **b4**: \( \text{pl_data[,29:1028]} \) 1000 continuous variables representing gene expression data.

Details

We generated the data in the following way: We took the empirical correlation of 1028 variables related to 315 AML patients. This correlation served as a correlation matrix when generating 1028 multivariate normally distributed variables with the R function `rmvnorm`. Because we didn’t have a positive definite matrix, we took the nearest positive definite matrix according to the function `nearPD`. The variables that should be binary were dichotomized, so that their marginal probabilities corresponded to the marginal probabilities they were based on. The coefficients were defined by

- \( \beta_b1 \leftarrow c(0.8, 0.8, 0.6, 0.6) \)
- \( \beta_b2 \leftarrow c(\text{rep}(0.5, 3), \text{rep}(0, 2)) \)
- \( \beta_b3 \leftarrow c(\text{rep}(0.4, 4), \text{rep}(0, 15)) \)
- \( \beta_b4 \leftarrow c(\text{rep}(0.5, 5), \text{rep}(0.3, 5), \text{rep}(0, 990)) \).

We included them in the vector \( \beta \leftarrow c(\beta_b1, \beta_b2, \beta_b3, \beta_b4) \) and calculated the probability through

\[
pi = \frac{\exp(\beta \cdot x)}{1 + \exp(\beta \cdot x)}
\]

where \( x \) denotes our data matrix with 1028 predictor variables. Finally we got the outcome through \( \text{pl_out} \leftarrow \text{rbinom}(400, \text{size} = 1, p = \pi) \).

---

predict.prioritylasso  Predictions from prioritylasso

Description

Makes predictions for a `prioritylasso` object. It can be chosen between linear predictors or fitted values.
predict.prioritylasso

Usage

## S3 method for class 'prioritylasso'
predict(
  object,  # S3 method for class 'prioritylasso'
  newdata = NULL,  # An object of class prioritylasso.
  type = c("link", "response"),  # (nnew x p) matrix or data frame with new values.
  handle.missingtestdata = c("none", "omit.prediction", "set.zero", "impute.block"),  # Specifies the type of predictions. link gives the linear predictors for all types of response and response gives the fitted values.
  include.allintercepts = FALSE,  # Specifies how to deal with missing data in the test data; possibilities are none, omit.prediction, set.zero and impute.block
  use.blocks = "all",  # should the intercepts from all blocks included in the prediction? If FALSE, only the intercept from the first block is included (default in the past).
  ...  # determines which blocks are used for the prediction, the default is all. Otherwise one can specify the number of blocks which are used in a vector

Arguments

object  # Further arguments passed to or from other methods.
newdata

Arguments

type

Arguments

handle.missingtestdata

Arguments

include.allintercepts

Arguments

use.blocks

Arguments

Details

handle.missingtestdata specifies how to deal with missing data. The default none cannot handle missing data, omit.prediction does not make a prediction for observations with missing values and return NA. set.zero ignores the missing data for the calculation of the prediction (the missing value is set to zero). impute.block uses an imputation model to impute the offset of a missing block. This only works if the prioritylasso object was fitted with handle.missingdata = "impute.offset". If impute.offset.cases = "complete.cases" was used, then every observation can have only one missing block. For observations with more than one missing block, NA is returned. If impute.offset.cases = "available.cases" was used, the missingness pattern in the test data has to be the same as in the train data. For observations with an unknown missingness pattern, NA is returned.

Value

Predictions that depend on type.

Author(s)

Simon Klau
See Also

pl_data, prioritylasso

Examples

pl_bin <- prioritylasso(X = matrix(rnorm(50*190),50,190), Y = rbinom(50,1,0.5),
                        family = "binomial", type.measure = "auc",
                        blocks = list(block1=1:13, block2=14:80, block3=81:190),
                        block1.penalization = TRUE, lambda.type = "lambda.min",
                        standardize = FALSE, nfolds = 3)

newdata_bin <- matrix(rnorm(10*190),10,190)

predict(object = pl_bin, newdata = newdata_bin, type = "response")

prioritylasso

Patient outcome prediction based on multi-omics data taking practitioners' preferences into account

Description

Fits successive Lasso models for several ordered blocks of (omics) data and takes the predicted values as an offset for the next block.

Usage

prioritylasso(
  x,
  y,
  weights,
  family = c("gaussian", "binomial", "cox"),
  type.measure,
  blocks,
  max.coef = NULL,
  block1.penalization = TRUE,
  lambda.type = "lambda.min",
  standardize = TRUE,
  nfolds = 10,
  foldid,
  cvoffset = FALSE,
  cvoffsetnfolds = 10,
  mcontrol = missing.control(),
  scale.y = FALSE,
  return.x = TRUE,
  ...
)
prioritylasso

Arguments

**X**
- a (nxp) matrix of predictors with observations in rows and predictors in columns.

**Y**
- n-vector giving the value of the response (either continuous, numeric-binary 0/1, or `Surv` object).

**weights**
- observation weights. Default is 1 for each observation.

**family**
- should be "gaussian" for continuous Y, "binomial" for binary Y, "cox" for Y of type `Surv`.

**type.measure**
- accuracy/error measure computed in cross-validation. It should be "class" (classification error) or "auc" (area under the ROC curve) if `family="binomial"`, "mse" (mean squared error) if `family="gaussian"` and "deviance" if `family="cox"` which uses the partial-likelihood.

**blocks**
- list of the format `list(bp1=...,bp2=...)`, where the dots should be replaced by the indices of the predictors included in this block. The blocks should form a partition of 1:p.

**max.coef**
- vector with integer values which specify the number of maximal coefficients for each block. The first entry is omitted if `block1.penalization = FALSE`. Default is NULL.

**block1.penalization**
- whether the first block should be penalized. Default is TRUE.

**lambda.type**
- specifies the value of lambda used for the predictions. `lambda.min` gives lambda with minimum cross-validated errors. `lambda.1se` gives the largest value of lambda such that the error is within 1 standard error of the minimum. Note that `lambda.1se` can only be chosen without restrictions of `max.coef`.

**standardize**
- logical, whether the predictors should be standardized or not. Default is TRUE.

**nfolds**
- the number of CV procedure folds.

**foldid**
- an optional vector of values between 1 and nfold identifying what fold each observation is in.

**cvoffset**
- logical, whether CV should be used to estimate the offsets. Default is FALSE.

**cvoffsetnfolds**
- the number of folds in the CV procedure that is performed to estimate the offsets. Default is 10. Only relevant if `cvoffset=TRUE`.

**mcontrol**
- controls how to deal with blockwise missing data. For details see below or `missing.control`.

**scale.y**
- determines if y gets scaled before passed to glmnet. Can only be used for `family = 'gaussian'`.

**return.x**
- logical, determines if the input data should be returned by `prioritylasso`. Default is TRUE.

**...**
- other arguments that can be passed to the function `cv.glmnet`.

Details

For `block1.penalization = TRUE`, the function fits a Lasso model for each block. First, a standard Lasso for the first entry of `blocks` (block of priority 1) is fitted. The predictions are then taken as an offset in the Lasso fit of the block of priority 2, etc. For `block1.penalization = FALSE`,
the function fits a model without penalty to the block of priority 1 (recommended as a block with clinical predictors where \( p < n \)). This is either a generalized linear model for family "gaussian" or "binomial", or a Cox model. The predicted values are then taken as an offset in the following Lasso fit of the block with priority 2, etc.

The first entry of \( \text{blocks} \) contains the indices of variables of the block with priority 1 (first block included in the model). Assume that \( \text{blocks} = \text{list}(1:100, 101:200, 201:300) \) then the block with priority 1 consists of the first 100 variables of the data matrix. Analogously, the block with priority 2 consists of the variables 101 to 200 and the block with priority 3 of the variables 201 to 300.

\[ \text{standardize} = \text{TRUE} \] leads to a standardisation of the covariables (\( X \)) in \text{glmnet} which is recommend by \text{glmnet}. In case of an unpenalized first block, the covariables for the first block are not standardized. Please note that the returned coefficients are rescaled to the original scale of the covariables as provided in \( X \). Therefore, new data in \text{predict.prioritylasso} should be on the same scale as \( X \).

To use the method with blockwise missing data, one can set \( \text{handle.missingdata} = \text{ignore} \). Then, to calculate the coefficients for a given block only the observations with values for this blocks are used. For the observations with missing values, the result from the previous block is used as the offset for the next block. Crossvalidated offsets are not supported with \( \text{handle.missingdata} = \text{ignore} \). Please note that dealing with single missing values is not supported. Normally, every observation gets a unique foldid which stays the same across all blocks for the call to \text{cv.glmnet}. However when \( \text{handle.missingdata} \neq \text{none} \), the foldid is set new for every block.

Value

object of class \text{prioritylasso} with the following elements. If these elements are lists, they contain the results for each penalized block.

- \text{lambda.ind} list with indices of lambda for \text{lambda.type}.
- \text{lambda.type} type of lambda which is used for the predictions.
- \text{lambda.min} list with values of lambda for \text{lambda.type}.
- \text{min.cvm} list with the mean cross-validated errors for \text{lambda.type}.
- \text{nzero} list with numbers of non-zero coefficients for \text{lambda.type}.
- \text{glmnet.fit} list of fitted \text{glmnet} objects.
- \text{name} a text string indicating type of measure.
- \text{block1unpen} if \text{block1.penalization} = \text{FALSE}, the results of either the fitted \text{glm} or \text{coxph} object corresponding to \text{best.blocks}.
- \text{coefficients} vector of estimated coefficients. If \text{block1.penalization} = \text{FALSE} and \text{family} = \text{gaussian} or \text{binomial}, the first entry contains an intercept.
- \text{call} the function call.
- \text{X} the original data used for the calculation or \text{NA} if \text{return.x} = \text{FALSE}
- \text{missing.data} list with logical entries for every block which observation is missing (\text{TRUE} means missing)
- \text{imputation.models} if \text{handle.missingdata} = "impute.offsets", it contains the used imputation models
blocks_used_for_imputation if handle.missingdata = "impute.offsets", it contains the blocks which were used for the imputation model for every block

y.scale.param if scale.y = TRUE, then it contains the mean and sd used for scaling.
bounds.list with the description which variables belong to which block

mcontrol the missing control settings used

family the family of the fitted data
dim.x the dimension of the used training data

Note
The function description and the first example are based on the R package ipflasso. The second example is inspired by the example of cv.glmnet from the glmnet package.

Author(s)
Simon Klau, Roman Hornung, Alina Bauer
Maintainer: Roman Hornung (<hornung@ibe.med.uni-muenchen.de>)

References

See Also
pl_data, cvm_prioritylasso, cvr.ipflasso, cvr2.ipflasso, missing.control

Examples

# gaussian
prioritylasso(X = matrix(rnorm(50*500),50,500), Y = rnorm(50), family = "gaussian",
  type.measure = "mse", blocks = list(bp1=1:75, bp2=76:200, bp3=201:500),
  max.coef = c(Inf,8,5), block1.penalization = TRUE,
  lambda.type = "lambda.min", standardize = TRUE, nfolds = 5, cvoffset = FALSE)

## Not run:
# cox
# simulation of survival data:
# n <- 50;p <- 300
# nzc <- trunc(p/10)
x <- matrix(rnorm(n*p), n, p)
beta <- rnorm(nzc)
fx <- x[, seq(nzc)]%*%beta/3
hx <- exp(fx)
# survival times:
ty <- rexp(n,hx)
# censoring indicator:
tcens <- rbinom(n = n,prob = .3,size = 1)
library(survival)
y <- Surv(ty, 1-tcens)
blocks <- list(bp1=1:20, bp2=21:200, bp3=201:300)
# run prioritylasso:
prioritylasso(x, y, family = "cox", type.measure = "deviance", blocks = blocks,
    block1.penalization = TRUE, lambda.type = "lambda.min", standardize = TRUE,
    nfolds = 5)

# binomial
# using pl_data:
prioritylasso(X = pl_data[,1:1028], Y = pl_data[,1029], family = "binomial", type.measure = "auc",
    blocks = list(bp1=1:4, bp2=5:9, bp3=10:28, bp4=29:1028), standardize = FALSE)
## End(Not run)
Index

* datasets
  - pl_data, 7

calculate_offsets, 2
coef.prioritylasso, 3
cv.glmnet, 13
cvm_prioritylasso, 4, 13
cvr.ipflasso, 13
cvr2.ipflasso, 6, 13

missing.control, 6, 11, 13

nearPD, 8

pl_data, 6, 7, 10, 13
predict.prioritylasso, 8
prioritylasso, 4, 6, 10, 10

rmvnorm, 8