Package ‘prioritylasso’

December 17, 2019

Type Package
Title Analyzing Multiple Omics Data with an Offset Approach
Version 0.2.3
Author Simon Klau, Roman Hornung, Alina Bauer
Maintainer Simon Klau <simonklau@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.
Depends R (>= 2.10.0)
License GPL-2
LazyData TRUE
Imports survival, glmnet, utils
RoxygenNote 7.0.2
Suggests testthat, knitr, rmarkdown, pROC
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2019-12-17 17:30:04 UTC

R topics documented:
cvm_prioritylasso ................................................. 2
pl_data .................................................................. 4
predict.prioritylasso ............................................. 5
prioritylasso ......................................................... 6

Index 9
cvm_prioritylasso

Description

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

Usage

```r
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 or data frame 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` The 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 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 cv.glmnet.
Value

object of class `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` ob-
  ject.
- `best.blocks` character vector with the indices of the best block specification.
- `best.max.coef` vector with the number of maximal coefficients corresponding to `best.blocks`.
- `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: Simon Klau (<simonklau@ibe.med.uni-muenchen.de>)

References

hierarchical approach to the prediction of clinical outcome using multi-omics data. BMC Bioinfor-
matics 19, 322

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),
Simulated AML data with binary outcome

Description
A data set containing the binary outcome and 1028 predictor variables of 400 artificial AML patients.

Usage
pl_data

Format
A data frame with 400 rows and 1029 variables:

pl_out: (pl_data[,1029]) binary outcome representing refractory status.
b1: (pl_data[,1:4]) 4 binary variables representing variables with a known influence on the outcome.
b2: (pl_data[,5:9]) 5 continuous variables representing clinical variables.
b3: (pl_data[,10:28]) 19 binary variables representing mutations.
b4: (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 <- c(0.8, 0.8, 0.6, 0.6)`
- `beta_b2 <- c(rep(0.5, 3), rep(0, 2))`
- `beta_b3 <- c(rep(0.4, 4), rep(0, 15))`
- `beta_b4 <- c(rep(0.5, 5), rep(0.3, 5), rep(0, 990))`.

We included them in the vector `beta <- c(beta_b1, beta_b2, beta_b3, beta_b4)` and calculated the probability through

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

where x denotes our data matrix with 1028 predictor variables. Finally we got the outcome through `pl_out <- rbinom(400, 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.

Usage

## S3 method for class 'prioritylasso'
predict(object, newdata, type = c("link", "response"), ...)  

Arguments

object     An object of class prioritylasso.
newdata    (n x p) matrix or data frame with new values.
type       Specifies the type of predictions. link gives the linear predictors for all types of response and response gives the fitted values.
...        Further arguments passed to or from other methods.

Value

Predictions that depend on type.

Author(s)

Simon Klau

See Also

pl_data, prioritylasso

Examples

pl_bin <- prioritylasso(X = matrix(rnorm(50*500),50,500), Y = rbinom(50,1,0.5), family = "binomial", type.measure = "auc", blocks = list(block1=1:13,block2=14:200, block3=201:500), block1.penalization = TRUE, lambda.type = "lambda.min", standardize = FALSE, nfolds = 5)

newdata_bin <- matrix(rnorm(30*500),30,500)

predict(object = pl_bin, newdata = newdata_bin, type = "response")
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, type.measure, blocks,
max.coef = 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 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.

... 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 blocks contains the indices of variables of the block with priority 1 (first block included in the model). Assume that blocks = 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.

Value

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

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 corresponding to best.blocks.

coefficients vector of estimated coefficients. If block1.penalization = FALSE and family = gaussian or binomial, the first entry contains an intercept.

call the function call.

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: Simon Klau (<simonklau@ibe.med.uni-muenchen.de>)

References

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

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

*Topic datasets
  pl_data, 4

  cv.glmnet, 7
cvm_prioritylasso, 2, 8
cvr.ipflasso, 8
cvr2.ipflasso, 3, 8

  nearPD, 4

  pl_data, 3, 4, 5, 8
  predict.prioritylasso, 5
  prioritylasso, 2, 3, 5, 6

  rmvnorm, 4