

Package ‘priorityelasticnet’

January 13, 2025

Type Package

Title Comprehensive Analysis of Multi-Omics Data Using an Offset-Based Method

Version 0.1.0

Description

Priority-ElasticNet extends the Priority-LASSO method (Klau et al. (2018) <[doi:10.1186/s12859-018-2344-6](https://doi.org/10.1186/s12859-018-2344-6)>) by incorporating the ElasticNet penalty, allowing for both L1 and L2 regularization. This approach fits successive ElasticNet models for several blocks of (omics) data with different priorities, using the predicted values from each block as an offset for the subsequent block. It also offers robust options to handle block-wise missingness in multi-omics data, improving the flexibility and applicability of the model in the presence of incomplete datasets.

License GPL-3

Depends R (>= 3.5.0)

Imports survival, glmnet, utils, checkmate, shiny, tidyr, dplyr, caret, pROC, PRROC, plotrix, ggplot2, magrittr, tibble, broom, cvms

Suggests ipflasso, rlang, knitr, rmarkdown, glmSparseNet

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Laila Qadir Musib [aut, cre],
Eunice Carrasquinha [aut],
Helena Mouriño [aut]

Maintainer Laila Qadir Musib <statleila98@gmail.com>

Repository CRAN

Date/Publication 2025-01-13 17:30:13 UTC

Contents

calculate_offsets	2
coef.priorityelasticnet	3
compare_boolean	3
cvm_priorityelasticnet	4
missing.control	6
Pen_Data	7
predict.priorityelasticnet	16
priorityelasticnet	18
weightedThreshold	21

Index	22
--------------	-----------

calculate_offsets	<i>Calculates the offsets for the current block</i>
-------------------	---

Description

Calculates the offsets for the current block

Usage

```
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.priorityelasticnet
```

Extract coefficients from a priorityelasticnet object

Description

Extract coefficients from a priorityelasticnet object

Usage

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

Arguments

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

Value

List with the coefficients and the intercepts

```
compare_boolean
```

Compare the rows of a matrix with a pattern

Description

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_priorityelasticnet

priorityelasticnet with several block specifications

Description

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

Usage

```
cvm_priorityelasticnet(
  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,
  alpha = 1,
  ...
)
```

Arguments

X	A numeric matrix of predictors.
Y	A response vector. For family = "multinomial", Y should be a factor with more than two levels.
weights	Optional observation weights. Default is NULL.
family	A character string specifying the model type. Options are "gaussian", "binomial", "cox", and "multinomial". Default is "gaussian".
type.measure	Loss function for cross-validation. Options are "mse", "deviance", "class", "auc". Default depends on the family.
blocks.list	list of the format <code>list(list(bp1=..., bp2=...), list(bp1=..., bp2=...), ...)</code> . For the specification of the entries, see priorityelasticnet .

max.coef.list	list of max.coef vectors. The first entries are omitted if block1.penalization = FALSE. Default is NULL.
block1.penalization	Logical. If FALSE, the first block will not be penalized. Default is TRUE.
lambda.type	Type of lambda to select. Options are "lambda.min" or "lambda.1se". Default is "lambda.min".
standardize	Logical flag for variable standardization, prior to fitting the model. Default is TRUE.
nfolds	Number of folds for cross-validation. Default is 10.
foldid	Optional vector of values between 1 and nfolds identifying what fold each observation is in. Default is NULL.
cvoffset	Logical. If TRUE, a cross-validated offset is used. Default is FALSE.
cvoffsetnfolds	Number of folds for cross-validation of the offset. Default is 10.
alpha	Elastic net mixing parameter. The elastic net penalty is defined as $(1 - \alpha)/2 \ \beta\ _2^2 + \alpha \ \beta\ _1$ Defaults to 1 (lasso penalty).
...	other arguments that can be passed to the function priorityelasticnet.

Value

object of class `cvm_priorityelasticnet` 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 `priorityelasticnet` 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`.

missing.control	<i>Construct control structures for handling of missing data for priorityelasticnet</i>
-----------------	---

Description

Construct control structures for handling of missing data for priorityelasticnet

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 nothing, 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.

`nfolds.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, priorityelasticnet tries to reduce the number of blocks taken into account for the imputation model to increase the number of observations 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

Pen_Data

Simulated Patient Data for Binary Classification

Description

This dataset contains simulated data for a binary classification problem, representing patient data with clinical, proteomics, and RNA variables. The data is organized into three blocks of variables: clinical variables, proteomics variables, and RNA variables. The outcome is a binary variable generated based on a logistic function.

Usage

`Pen_Data`

Format

A data frame with 406 rows and 325 columns:

Clinical_Var1 Numeric variable representing age.

Clinical_Var2 Binary variable representing gender (0 = male, 1 = female).

Clinical_Var3 Categorical variable representing race (values 0, 1, 2, or 3).

Clinical_Var4 Binary variable representing ethnicity (0 or 1).

Clinical_Var5 Binary variable representing radiation therapy status (0 or 1).

Proteomic_Var1 Continuous variable representing a proteomic measurement.

Proteomic_Var2 Continuous variable representing a proteomic measurement.

Proteomic_Var3 Continuous variable representing a proteomic measurement.

Proteomic_Var4 Continuous variable representing a proteomic measurement.

Proteomic_Var5 Continuous variable representing a proteomic measurement.

Proteomic_Var6 Continuous variable representing a proteomic measurement.

Proteomic_Var7 Continuous variable representing a proteomic measurement.

Proteomic_Var8 Continuous variable representing a proteomic measurement.

Proteomic_Var9 Continuous variable representing a proteomic measurement.

Proteomic_Var10 Continuous variable representing a proteomic measurement.

- RNA_Var133** Continuous variable representing an RNA measurement.
- RNA_Var134** Continuous variable representing an RNA measurement.
- RNA_Var135** Continuous variable representing an RNA measurement.
- RNA_Var136** Continuous variable representing an RNA measurement.
- RNA_Var137** Continuous variable representing an RNA measurement.
- RNA_Var138** Continuous variable representing an RNA measurement.
- RNA_Var139** Continuous variable representing an RNA measurement.
- RNA_Var140** Continuous variable representing an RNA measurement.
- RNA_Var141** Continuous variable representing an RNA measurement.
- RNA_Var142** Continuous variable representing an RNA measurement.
- RNA_Var143** Continuous variable representing an RNA measurement.
- RNA_Var144** Continuous variable representing an RNA measurement.
- RNA_Var145** Continuous variable representing an RNA measurement.
- Pen_out** Binary outcome variable generated using a logistic function applied to a linear predictor based on the combined variables.

predict.priorityelasticnet

Predictions from priorityelasticnet

Description

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

Usage

```
## S3 method for class 'priorityelasticnet'
predict(
  object,
  newdata = NULL,
  type = c("link", "response"),
  handle.missingtestdata = c("none", "omit.prediction", "set.zero", "impute.block"),
  include.allintercepts = FALSE,
  use.blocks = "all",
  alpha = 1,
  ...
)
```


Arguments

<code>object</code>	An object of class <code>priorityelasticnet</code> .
<code>newdata</code>	(<code>nnew</code> x <code>p</code>) matrix or data frame with new values.
<code>type</code>	Specifies the type of predictions. <code>link</code> gives the linear predictors for all types of response and <code>response</code> gives the fitted values.
<code>handle.missingtestdata</code>	Specifies how to deal with missing data in the test data; possibilities are <code>none</code> , <code>omit.prediction</code> , <code>set.zero</code> and <code>impute.block</code>
<code>include.allintercepts</code>	should the intercepts from all blocks included in the prediction? If <code>FALSE</code> , only the intercept from the first block is included (default in the past).
<code>use.blocks</code>	determines which blocks are used for the prediction, the default is <code>all</code> . Otherwise one can specify the number of blocks which are used in a vector
<code>alpha</code>	Elastic net mixing parameter used in the model fitting.
<code>...</code>	Further arguments passed to or from other methods.

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 `priorityelasticnet` 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`.

Examples

```
pl_bin <- priorityelasticnet(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, alpha = 1)

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

predict(object = pl_bin, newdata = newdata_bin, type = "response", alpha = 1)
```

priorityelasticnet *Priority Elastic Net for High-Dimensional Data*

Description

This function performs penalized regression analysis using the elastic net method, tailored for high-dimensional data with a known group structure. It also includes an optional feature to launch a Shiny application for model evaluation with weighted threshold optimization.

Usage

```
priorityelasticnet(
  X,
  Y,
  weights = NULL,
  family = c("gaussian", "binomial", "cox", "multinomial"),
  alpha = 0.5,
  type.measure,
  blocks,
  max.coef = NULL,
  block1.penalization = TRUE,
  lambda.type = "lambda.min",
  standardize = TRUE,
  nfolds = 10,
  foldid = NULL,
  cvoffset = FALSE,
  cvoffsetnfolds = 10,
  mcontrol = missing.control(),
  scale.y = FALSE,
  return.x = TRUE,
  adaptive = FALSE,
  initial_global_weight = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

X	A numeric matrix of predictors.
Y	A response vector. For family = "multinomial", Y should be a factor with more than two levels.
weights	Optional observation weights. Default is NULL.
family	A character string specifying the model type. Options are "gaussian", "binomial", "cox", and "multinomial". Default is "gaussian".
alpha	The elastic net mixing parameter, with $0 \leq \alpha \leq 1$. The penalty is defined as $(1 - \alpha)/2 \ \beta\ _2^2 + \alpha \ \beta\ _1$. Default is 1.

<code>type.measure</code>	Loss function for cross-validation. Options are "mse", "deviance", "class", "auc". Default depends on the family.
<code>blocks</code>	A list where each element is a vector of indices indicating the predictors in that block.
<code>max.coef</code>	A numeric vector specifying the maximum number of non-zero coefficients allowed in each block. Default is NULL, meaning no limit.
<code>block1.penalization</code>	Logical. If FALSE, the first block will not be penalized. Default is TRUE.
<code>lambda.type</code>	Type of lambda to select. Options are "lambda.min" or "lambda.1se". Default is "lambda.min".
<code>standardize</code>	Logical flag for variable standardization, prior to fitting the model. Default is TRUE.
<code>nfolds</code>	Number of folds for cross-validation. Default is 10.
<code>foldid</code>	Optional vector of values between 1 and <code>nfolds</code> identifying what fold each observation is in. Default is NULL.
<code>cvoffset</code>	Logical. If TRUE, a cross-validated offset is used. Default is FALSE.
<code>cvoffsetnfolds</code>	Number of folds for cross-validation of the offset. Default is 10.
<code>mcontrol</code>	Control parameters for handling missing data. Default is <code>missing.control()</code> .
<code>scale.y</code>	Logical. If TRUE, the response variable Y is scaled. Default is FALSE.
<code>return.x</code>	Logical. If TRUE, the function returns the input matrix X. Default is TRUE.
<code>adaptive</code>	Logical. If TRUE, the adaptive elastic net is used, where penalties are adjusted based on the importance of the coefficients from an initial model fit. Default is FALSE.
<code>initial_global_weight</code>	Logical. If TRUE (the default), global initial weights will be calculated based on all predictors. If FALSE, initial weights will be calculated separately for each block.
<code>verbose</code>	Logical. If TRUE prints detailed logs of the process. Default is FALSE.
<code>...</code>	Additional arguments to be passed to <code>cv.glmnet</code> .

Value

A list with the following components:

<code>lambda.ind</code>	Indices of the selected lambda values.
<code>lambda.type</code>	Type of lambda used.
<code>lambda.min</code>	Selected lambda values.
<code>min.cvm</code>	Cross-validated mean squared error for each block.
<code>nzero</code>	Number of non-zero coefficients for each block.
<code>glmnet.fit</code>	Fitted <code>glmnet</code> objects for each block.
<code>name</code>	Name of the model.
<code>block1unpen</code>	Fitted model for the unpenalized first block, if applicable.

coefficients	Coefficients of the fitted models.
call	The function call.
X	The input matrix X, if return.x is TRUE.
missing.data	Logical vector indicating missing data.
imputation.models	Imputation models used, if applicable.
blocks.used.for.imputation	Blocks used for imputation, if applicable.
missingness.pattern	Pattern of missing data, if applicable.
y.scale.param	Parameters for scaling Y, if applicable.
blocks	The input blocks.
mcontrol	Control parameters for handling missing data.
family	The model family.
dim.x	Dimensions of the input matrix X.

Note

Ensure that glmnet version $\geq 2.0.13$ is installed. The function does not support single missing values within a block.

Examples

```
# Simulation of multinomial data:
set.seed(123)
n <- 100
p <- 50
k <- 3
x <- matrix(rnorm(n * p), n, p)
y <- sample(1:k, n, replace = TRUE)
y <- factor(y)
blocks <- list(bp1 = 1:10, bp2 = 11:30, bp3 = 31:50)

# Run priorityelasticnet:
fit <- priorityelasticnet(x, y, family = "multinomial", alpha = 0.5,
  type.measure = "class", blocks = blocks,
  block1.penalization = TRUE, lambda.type = "lambda.min",
  standardize = TRUE, nolds = 5,
  adaptive = FALSE)

fit$coefficients
```

weightedThreshold	<i>A Shiny App for Model Evaluation and Weighted Threshold Optimization</i>
-------------------	---

Description

This function starts a Shiny application that enables users to interactively adjust the threshold for binary classification and view related metrics, the confusion matrix, ROC curve, and PR curve. The app also includes a feature for calculating the optimal threshold using a weighted version of Youden's J-statistic.

Usage

```
weightedThreshold(object, ...)
```

Arguments

object	A result from priorityelasticnet function with binomial model family.
...	Additional arguments

Details

To calculate the optimal threshold, a weighted version of Youden's J-statistic (Youden, 1950) is used. The optimal cutoff is the threshold that maximizes the distance from the identity (diagonal) line. The function optimizes the metric $(w * \text{sensitivity} + (1 - w) * \text{specificity})$, where 'w' is the weight parameter adjusted using the second slider. After selecting the desired value on the optimal threshold slider, the user must press the "Set" button to update the threshold slider with the calculated optimal value. Metrics will then be automatically recalculated based on the user's selection. This function adapted from 'Monahov, A. (2021). Model Evaluation with Weighted Threshold Optimization (and the "mewto" R package). Available at SSRN 3805911.'

Value

No return value. This function is used for side effects only, specifically to launch a Shiny application for model evaluation with weighted threshold optimization. The Shiny app provides an interactive interface to visualize model performance metrics and optimize thresholds for classification models based on user-defined criteria.

Index

* datasets

Pen_Data, [7](#)

calculate_offsets, [2](#)

coef.priorityelasticnet, [3](#)

compare_boolean, [3](#)

cvm.priorityelasticnet, [4](#)

missing.control, [6](#)

Pen_Data, [7](#)

predict.priorityelasticnet, [16](#)

priorityelasticnet, [4](#), [18](#)

weightedThreshold, [21](#)