Package 'ipflasso'

October 13, 2022

Type Package

Title Integrative Lasso with Penalty Factors

Version 1.1

Date 2019-12-10

Author Anne-Laure Boulesteix, Mathias Fuchs, Gerhard Schulze

Maintainer Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Description The core of the package is cvr2.ipflasso(), an extension of glmnet to be used when the (large) set of available predictors is partitioned into several modalities which potentially differ with respect to their information content in terms of prediction. For example, in biomedical applications patient outcome such as survival time or response to therapy may have to be predicted based on, say, mRNA data, miRNA data, methylation data, CNV data, clinical data, etc. The clinical predictors are on average often much more important for outcome prediction than the mRNA data. The ipflasso method takes this problem into account by using different penalty parameters for predictors from different modalities. The ratio between the different penalty parameters can be chosen from a set of optional candidates by cross-validation or alternatively generated from the input data.

Depends glmnet, survival

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-10 15:10:02 UTC

R topics documented:

cvr.adaptive.ipflasso *Cross-validated integrative lasso with adaptive penalty factors*

Description

Runs cvr.ipflasso applying different data based penalty factors to predictors from different blocks.

Usage

```
cvr.adaptive.ipflasso(X, Y, family, type.measure, standardize = TRUE,
                                  alpha, type.step1, blocks, nfolds, ncv)
```

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).

`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. If not specified, type.measure is "class" (classification error) if `family="binomial"`, "mse" (mean squared error) if `family="gaussian"` and partial likelihood if `family="cox"`. If `family="binomial"`, one may specify `type.measure="auc"` (area under the ROC curve).

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

`alpha` the elastic net mixing parameter for step 1: `alpha=1` yields the L1 penalty (Lasso), `alpha=0` yields the L2 penalty (Ridge).

`type.step1` whether the models of step 1 should be run on the whole data set `X` (`type.step1="comb"`) or separately for each block (`type.step1="sep"`).

`blocks` a list of length M of the format `list(block1=...,block2=...,` 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.

`nfolds` the number of folds of the CV procedure.

`ncv` the number of repetitions of the CV. Not to be confused with `nfolds`. For example, if one repeats 50 times 5-fold-CV (i.e. considers 50 random partitions into 5 folds in turn and averages the results), `nfolds` equals 5 and `ncv` equals 50.

Details

The penalty factors are the inverse arithmetic means of the absolute model coefficients per block, generated in a first step of the function. The user can choose to determine these coefficients by running a Lasso model (`alpha=1`) or a Ridge model (`alpha=0`) either on the whole data set (`type.step1="comb"`) or seperately for each block (`type.step1="sep"`). If `type.step1` is ommited, it will be set to `"sep"` for Lasso and to `"comb"` for Ridge. If a Lasso model in step 1 returns any zero coefficient mean, the corresponding block will be excluded from the input date set `X` and step 2 will be run with the remaining blocks. If all model coefficient means are zero, step 2 will not be performed.

Value

A list with the following arguments:

`coeff` the matrix of coefficients with predictors corresponding to rows and lambda values corresponding to columns. The first row contains the intercept of the models (for all families other than `"cox"`).

In the special case of separate step 1 Lasso models and all coefficient means equal to zero, the intercept is the average of the separate model intercepts per block.

`ind.bestlambda` the index of the best lambda according to CV.

`lambda` the lambda sequence.

In the special case of separate step 1 Lasso models and all coefficient means equal to zero, it is the lambda sequence with the highest lambda value among the lambda sequences of all blocks.

`cvm` the CV estimate of the measure specified by `type.measure` for each candidate lambda value.

In the special case of separate step 1 Lasso models and all coefficient means equal to zero, cmv is the average of the separate model cvms per block.

`nzero` the number of non-zero coefficients in the selected model.

In the special case of separate step 1 Lasso models and all coefficient means equal to zero, nzero is the sum of the non-zero coefficients of the separate models per block.

`family` see arguments.

`means.step1` the arithmetic means of the absolute model coefficients per block, returned by the first step of the function.

`exc` the exclusion vector containing the indices of the block(s) to be excluded from `X`.

Author(s)

Gerhard Schulze (g-schulze@t-online.de)

References

Schulze, Gerhard (2017): Clinical Outcome Prediction Based on Multi-Omics Data: Extension of IPF-LASSO. Masterarbeit, Ludwig-Maximilians-Universitaet Muenchen (Department of Statistics: Technical Reports) https://doi.org/10.5282/ubm/epub.59092

Examples

```
# load ipflasso library
library(ipflasso)

# generate dummy data
X<-matrix(rnorm(50*200),50,200)
Y<-rbinom(50,1,0.5)
```

```
cvr.adaptive.ipflasso(X=X,Y=Y,family="binomial",type.measure="class",standardize=FALSE,
                      alpha = 1,blocks=list(block1=1:50,block2=51:200),nfolds=5,ncv=10)
```

cvr.glmnet *Repeating cv.glmnet*

Description

the same as cv.glmnet but with several ncv repetitions of CV: cross-validation repeated ncv times (i.e. for ncv different random partitions)

Usage

```
cvr.glmnet(X, Y, family, standardize=TRUE,alpha=1, nfolds, ncv, type.measure,...)
```

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)

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

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

`alpha` the elastic net mixing parameter: `alpha`=1 yields the L1 penalty (lasso), `alpha`=0 yields the L2 penalty. Default is `alpha`=1 (lasso).

`nfolds` the number of folds of CV procedure.

`ncv` the number of repetitions of CV. Not to be confused with `nfolds`. For example, if one repeats 50 times 5-fold-CV (i.e. considers 50 random partitions into 5 folds in turn and averages the results), nfolds equals 5 and ncv equals 50.

`type.measure` The accuracy/error measure computed in cross-validation. If not specified, type.measure is "class" (classification error) if `family="binomial"`, "mse" (mean squared error) if `family="gaussian"` and partial likelihood if `family="cox"`. If `family="binomial"`, one may specify `type.measure="auc"` (area under the ROC curve).

`...` Other arguments to be passed to the function `cv.glmnet`.

Value

A list with the following arguments:

`coeff` the matrix of coefficients with predictors corresponding to rows and lambda values corresponding to columns. The first rows contains the intercept of the model (for all families other than `"cox"`).

`lambda` the lambda sequence

`cvm` the CV estimate of the measure specified by `type.measure` for each candidate lambda value, averaged over the ncv runs of `cv.glmnet`

Author(s)

Anne-Laure Boulesteix (https://www.en.ibe.med.uni-muenchen.de/mitarbeiter/professoren/boulesteix/index.html)

References

Boulesteix AL, De Bin R, Jiang X, Fuchs M, 2017. IPF-lasso: integrative L1-penalized regression with penalty factors for prediction based on multi-omics data. Comput Math Methods Med 2017:7691937.

Examples

```
# load ipflasso library
library(ipflasso)

# generate dummy data
X<-matrix(rnorm(50*200),50,200)
Y<-rbinom(50,1,0.5)

cvr.glmnet(X=X,Y=Y,family="binomial",standardize=FALSE,nfolds=5,ncv=10,type.measure="class")
```

`cvr.ipflasso` *Cross-validated integrative lasso with fixed penalty factors*

Description

Runs cvr.glmnet giving different penalty factors to predictors from different blocks.

Usage

```
cvr.ipflasso(X, Y, family, type.measure, standardize=TRUE, alpha=1, blocks, pf, nfolds,
  ncv)
```

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)

`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. If not specified, type.measure is "class" (classification error) if `family="binomial"`, "mse" (mean squared error) if `family="gaussian"` and partial likelihood if `family="cox"`. If `family="binomial"`, one may specify `type.measure="auc"` (area under the ROC curve).

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

`alpha` the elastic net mixing parameter: `alpha`=1 yields the L1 penalty (lasso), `alpha`=0 yields the L2 penalty. Default is `alpha`=1 (lasso).

`blocks` a list of length M the format `list(block1=...,block2=...,` 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.

`pf` a vector of length equal to the number of blocks M. Each entry contains the penalty factor to be applied to the predictors of the corresponding block. Example: if `pf`=c(1,2), the penalty applied to the predictors of the 2nd block is twice as large as the penalty applied to the predictors of the first block.

`nfolds` the number of folds of CV procedure.

`ncv` the number of repetitions of CV. Not to be confused with `nfolds`. For example, if one repeats 50 times 5-fold-CV (i.e. considers 50 random partitions into 5 folds in turn and averages the results), `nfolds` equals 5 and `ncv` equals 50.

Value

A list with the following arguments:

`coeff` the matrix of coefficients with predictors corresponding to rows and lambda values corresponding to columns. The first rows contains the intercept of the model (for all families other than `"cox"`).

`ind.bestlambda` the index of the best lambda according to CV.

`lambda` the lambda sequence.

`cvm` the CV estimate of the measure specified by `type.measure` for each candidate lambda value.

`nzero` the number of non-zero coefficients in the selected model.

`family` See arguments.

Author(s)

Anne-Laure Boulesteix (`https://www.en.ibe.med.uni-muenchen.de/mitarbeiter/professoren/boulesteix/index.html`)

References

Boulesteix AL, De Bin R, Jiang X, Fuchs M, 2017. IPF-lasso: integrative L1-penalized regression with penalty factors for prediction based on multi-omics data. Comput Math Methods Med 2017:7691937.

Examples

```
# load ipflasso library
library(ipflasso)

# generate dummy data
X<-matrix(rnorm(50*200),50,200)
Y<-rbinom(50,1,0.5)
```

```
cvr.ipflasso(X=X,Y=Y,family="binomial",standardize=FALSE,
          blocks=list(block1=1:50,block2=51:200),
          pf=c(1,2),nfolds=5,ncv=10,type.measure="class")
```

---

cvr2.ipflasso *Cross-validated integrative lasso with cross-validated penalty factors*

Description

Runs cvr.glmnet giving different penalty factors to predictors from different blocks and chooses the penalty factors by cross-validation from the list `pflist` of candidates.

Usage

```
cvr2.ipflasso(X, Y, family, type.measure, standardize=TRUE,
              alpha=1, blocks, pflist, nfolds, ncv,
              nzeromax = +Inf, plot=FALSE)
```

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)

`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. If not specified, type.measure is "class" (classification error) if `family="binomial"`, "mse" (mean squared error) if `family="gaussian"` and partial likelihood if `family="cox"`. If `family="binomial"`, one may specify `type.measure="auc"` (area under the ROC curve).

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

`alpha` the elastic net mixing parameter: `alpha`=1 yields the L1 penalty (lasso), `alpha`=0 yields the L2 penalty. Default is `alpha`=1 (lasso).

`blocks` a list of length M the format `list(block1=...,block2=...,` 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.

`pflist` a list of candidate penalty factors (see the argument `pf` of the function `cvr.ipflasso`) of the format weightslist=list(c(1,1),c(1,2),c(2,1),...).

`nfolds` the number of folds of CV procedure.

`ncv` the number of repetitions of CV. Not to be confused with `nfolds`. For example, if one repeats 50 times 5-fold-CV (i.e. considers 50 random partitions into 5 folds in turn and averages the results), `nfolds` equals 5 and `ncv` equals 50.

`nzeromax` the maximal number of predictors allowed in the final model. Default is +Inf, i.e. the best model is selected based on CV without restriction.

`plot` If `plot=TRUE`, the function outputs plots of CV errors and number of included predictors for each block.

Value

A list with the following arguments:

`coeff` the matrix of coefficients obtained with the best combination of penalty factors, with covariates corresponding to rows and lambda values corresponding to columns. The first row contains the intercept of the model.

`ind.bestlambda` the index of the best lambda as selected by CV for the best combination of penalty factors.

`bestlambda` the best lambda as selected by CV for the best combination of penalty factors.

`ind.bestpf` the index of the best penalty factor selected by CV from the list of candidates `pflist`.

`cvm` the CV error for each candidate lambda value, averaged over the ncv runs of `cv.glmnet`.

`a` a list of length `length(pflist)` containing the outputs of the function `cvr.ipflasso` for all candidate penalty factors from `pflist`.

`family` See arguments.

Author(s)

Anne-Laure Boulesteix (https://www.en.ibe.med.uni-muenchen.de/mitarbeiter/professoren/boulesteix/index.html)

References

Boulesteix AL, De Bin R, Jiang X, Fuchs M, 2017. IPF-lasso: integrative L1-penalized regression with penalty factors for prediction based on multi-omics data. Comput Math Methods Med 2017:7691937.

Examples

```
# load ipflasso library
library(ipflasso)

# generate dummy data
X<-matrix(rnorm(50*200),50,200)
Y<-rbinom(50,1,0.5)

cvr2.ipflasso(X=X,Y=Y,family="binomial",type.measure="class",standardize=FALSE,
              blocks=list(block1=1:50,block2=51:200),
              pflist=list(c(1,1),c(1,2),c(2,1)),nfolds=5,ncv=10)
```

`ipflasso.predict` *Using an IPF-lasso model for prediction of new observations*

Description

Derives predictions for new observations from a model fitted by the functions `cvr.ipflasso` or `cvr2.ipflasso`.

Usage

```
ipflasso.predict(object, Xtest)
```

Arguments

`object`	the output of either `cvr.ipflasso` (if the user chooses the penalty factor himself) or `cvr2.ipflasso` (if the user cross-validates the penalty factor).
`Xtest`	a ntest x p matrix containing the values of the predictors for the test data. It should have the same number of columns as the matrix `X` used to obtain the model `result`.

Value

A list with the following arguments:

`linpredtest`	a ntest-vector giving the value of the linear predictor for the test observations
`classtest`	a ntest-vector with values 0 or 1 giving the predicted class for the test observations (for binary Y).
`probabilitiestest`	a ntest-vector giving the predicted probability of Y=1 for the test observations (for binary Y).

Author(s)

Anne-Laure Boulesteix (`https://www.en.ibe.med.uni-muenchen.de/mitarbeiter/professoren/boulesteix/index.html`)

References

Boulesteix AL, De Bin R, Jiang X, Fuchs M, 2017. IPF-lasso: integrative L1-penalized regression with penalty factors for prediction based on multi-omics data. Comput Math Methods Med 2017:7691937.

Examples

```
# load ipflasso library
library(ipflasso)

# generate dummy data
X<-matrix(rnorm(50*200),50,200)
Xtest<-matrix(rnorm(20*200),20,200)
Y<-rbinom(50,1,0.5)

# fitting the IPF-lasso model
model1<-cvr.ipflasso(X=X,Y=Y,family="binomial",standardize=FALSE,
                    blocks=list(block1=1:50,block2=51:200),
                    pf=c(1,2),nfolds=5,ncv=10,type.measure="class")

# making predictions from Xtest
ipflasso.predict(object=model1,Xtest=Xtest)
```

my.auc *Area under the curve (AUC)*

Description

computes the area under the ROC curve (AUC) for the marker ’linpred’ and the binary status ’Y’.

Usage

```
my.auc(linpred, Y)
```

Arguments

linpred	n-vector giving the value of the marker.
Y	n-vector giving the binary status, coded as 0/1.

Value

the area under the curve

Index