Package ‘joinet’

August 9, 2021

Version 0.0.10

Title Multivariate Elastic Net Regression

Description


Depends R (>= 3.0.0)

Imports glmnet, palasso, cornet

Suggests knitr, rmarkdown, testthat, MASS

Enhances mice, earth, spls, MRCE, remMap, MultivariateRandomForest, SiER, mcen, GPM, RMTL, MTPS

VignetteBuilder knitr

License GPL-3

Language en-GB

RoxygenNote 7.1.1

URL https://github.com/rauschenberger/joinet

BugReports https://github.com/rauschenberger/joinet/issues

NeedsCompilation no

Author Armin Rauschenberger [aut, cre]

Maintainer Armin Rauschenberger <armin.rauschenberger@uni.lu>

Repository CRAN

Date/Publication 2021-08-09 07:40:02 UTC
joinet-package

R topics documented:

joinet-package ........................................... 2
coef.joinet .................................................. 3
cv.joinet ................................................... 4
joinet ........................................................ 6
predict.joinet .............................................. 8
weights.joinet ............................................. 9

Index

joinet-package  Multivariate Elastic Net Regression

Description

The R package joinet implements multivariate ridge and lasso regression using stacked generalisation. This multivariate regression typically outperforms univariate regression at predicting correlated outcomes. It provides predictive and interpretable models in high-dimensional settings.

Details

Use function joinet for model fitting. Type library(joinet) and then ?joinet or help("joinet") to open its help file.

See the vignette for further examples. Type vignette("joinet") or browseVignettes("joinet") to open the vignette.

References

<armin.rauschenberger@uni.lu>

Examples

## Not run:
#--- data simulation ---
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
Y <- replicate(n=q,expr=rnorm(n=n,mean=rowSums(X[,1:5])))
# n samples, p inputs, q outputs

#--- model fitting ---
object <- joinet(Y=Y,X=X)
# slot "base": univariate
# slot "meta": multivariate

#--- make predictions ---
```r
y_hat <- predict(object,newx=X)
# n x q matrix "base": univariate
# n x q matrix "meta": multivariate

#--- extract coefficients ---
coef <- coef(object)
# q vector "alpha": intercepts
# p x q matrix "beta": slopes

#--- model comparison ---
loss <- cv.joinet(Y=Y,X=X)
# cross-validated loss
# row "base": univariate
# row "meta": multivariate

## End(Not run)
```

### coef.joinet

**Extract Coefficients**

Extracts pooled coefficients. (The meta learners linearly combines the coefficients from the base learners.)

#### Description

Extracts pooled coefficients. (The meta learners linearly combines the coefficients from the base learners.)

#### Usage

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

#### Arguments

- `object`: joinet object
- `...`: further arguments (not applicable)

#### Value

This function returns the pooled coefficients. The slot `alpha` contains the intercepts in a vector of length `q`, and the slot `beta` contains the slopes in a matrix with `p` rows (inputs) and `q` columns.

#### Examples

```r
## Not run:
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
Y <- replicate(n=q,expr=rnorm(n=n,mean=rowSums(X[,1:5])))
```
cv.joinet <- joinet(Y=Y, X=X)
coef <- coef(object)
## End(Not run)

---

**cv.joinet**

#### Model comparison

**Description**

- Compares univariate and multivariate regression.

**Usage**

```r
cv.joinet(
  Y,  # outputs: numeric matrix with n rows (samples) and q columns (outputs)
  X,  # inputs: numeric matrix with n rows (samples) and p columns (inputs)
  family = "gaussian",  # distribution: vector of length 1 or q with entries "gaussian", "binomial" or "poisson"
  nfolds.ext = 5,  # number of external folds
  nfolds.int = 10,  # number of internal folds
  foldid.ext = NULL,  # external fold identifiers: vector of length n with entries between 1 and nfolds.ext; or NULL
  foldid.int = NULL,  # internal fold identifiers: vector of length n with entries between 1 and nfolds.int; or NULL
  type.measure = "deviance",  # loss function: vector of length 1 or q with entries "deviance", "class", "mse" or "mae" (see cv.glmnet)
  alpha.base = 1,
  alpha.meta = 1,
  compare = FALSE,
  mice = FALSE,
  cvpred = FALSE,
  times = FALSE,
  ...
)
```

**Arguments**

- `Y` outputs: numeric matrix with n rows (samples) and q columns (outputs)
- `X` inputs: numeric matrix with n rows (samples) and p columns (inputs)
- `family` distribution: vector of length 1 or q with entries "gaussian", "binomial" or "poisson"
- `nfolds.ext` number of external folds
- `nfolds.int` number of internal folds
- `foldid.ext` external fold identifiers: vector of length n with entries between 1 and nfolds.ext; or NULL
- `foldid.int` internal fold identifiers: vector of length n with entries between 1 and nfolds.int; or NULL
- `type.measure` loss function: vector of length 1 or q with entries "deviance", "class", "mse" or "mae" (see cv.glmnet)
cv.joinet

alpha.base elastic net mixing parameter for base learners: numeric between 0 (ridge) and 1 (lasso)
alpha.meta elastic net mixing parameter for meta learners: numeric between 0 (ridge) and 1 (lasso)
compare experimental arguments: character vector with entries "mnorm", "spls", "mrce", "sier", "mtps", "rmtl", "gpm" and others (requires packages spls, MRCE, SiER, MTPS, RMTL or GPM)
mice missing data imputation: logical (mice=TRUE requires package mice)
cvpred return cross-validated predictions: logical
times measure computation time: logical
... further arguments passed to glmnet and cv.glmnet

Value

This function returns a matrix with q columns, including the cross-validated loss from the univariate models (base), the multivariate models (meta), and the intercept-only models (none).

Examples

```r
## Not run:
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
Y <- replicate(n=q,expr=rnorm(n=n,mean=rowSums(X[,1:5])))
cv.joinet(Y=Y,X=X)
## End(Not run)

## Not run:
# correlated features
n <- 50; p <- 100; q <- 3
mu <- rep(0,times=p)
Sigma <- 0.90*abs(col(diag(p))-row(diag(p)))
X <- MASS::mvrnorm(n=n,mu=mu,Sigma=Sigma)
mu <- rowSums(X[,sample(seq_len(p),size=5)])
Y <- replicate(n=q,expr=rnorm(n=n,mean=mu))
#Y <- t(MASS::mvrnorm(n=q,mu=mu,Sigma=diag(n)))
cv.joinet(Y=Y,X=X)
## End(Not run)

## Not run:
# other distributions
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
eta <- rowSums(X[,1:5])
Y <- replicate(n=q,expr=rbinom(n=n,size=1,prob=1/(1+exp(-eta))))
cv.joinet(Y=Y,X=X,family="binomial")
Y <- replicate(n=q,expr=rpois(n=n,lambda=exp(scale(eta))))
cv.joinet(Y=Y,X=X,family="poisson")
## End(Not run)
```
### Not run:
```
# uncorrelated outcomes
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
y <- rnorm(n=n,mean=rowSums(X[,1:5]))
Y <- cbind(y,matrix(rnorm(n*(q-1)),nrow=n,ncol=q-1))
cv.joinet(Y=Y,X=X)
```
### Not run:
```
# sparse and dense models
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
Y <- replicate(n=q,expr=rnorm(n=n,mean=rowSums(X[,1:5])))
set.seed(1) # fix folds
cv.joinet(Y=Y,X=X,alpha.base=1) # lasso
set.seed(1)
cv.joinet(Y=Y,X=X,alpha.base=0) # ridge
```

---

**joinet**  
*Multivariate Elastic Net Regression*

---

**Description**

Implements multivariate elastic net regression.

**Usage**

```
joinet(
  Y,
  X,
  family = "gaussian",
  nfolds = 10,
  foldid = NULL,
  type.measure = "deviance",
  alpha.base = 1,
  alpha.meta = 1,
  weight = NULL,
  sign = NULL,
  ...
)
```

**Arguments**

- **Y**  
  outputs: numeric matrix with *n* rows (samples) and *q* columns (outputs)

- **X**  
  inputs: numeric matrix with *n* rows (samples) and *p* columns (inputs)
family: vector of length 1 or q with entries "gaussian", "binomial" or "poisson"
nfolds: number of folds
foldid: fold identifiers: vector of length n with entries between 1 and nfolds; or NULL (balance)
type.measure: loss function: vector of length 1 or q with entries "deviance", "class", "mse" or "mae" (see cv.glmnet)
alpha.base: elastic net mixing parameter for base learners: numeric between 0 (ridge) and 1 (lasso)
alpha.meta: elastic net mixing parameter for meta learners: numeric between 0 (ridge) and 1 (lasso)
weight: input-output relations: matrix with p rows (inputs) and q columns (outputs) with entries 0 (exclude) and 1 (include), or NULL (see details)
sign: output-output relations: matrix with q rows ("meta-inputs") and q columns (outputs), with entries −1 (negative), 0 (none), 1 (positive) and NA (any), or NULL (see details)

... further arguments passed to glmnet

Details

**input-output relations**: In this matrix with p rows and q columns, the entry in the jth row and the kth column indicates whether the jth input may be used for modelling the kth output (where 0 means "exclude" and 1 means "include"). By default (sign=NULL), all entries are set to 1.

**output-output relations**: In this matrix with q rows and q columns, the entry in the lth row and the kth column indicates how the lth output may be used for modelling the kth output (where −1 means negative effect, 0 means no effect, 1 means positive effect, and NA means any effect).

There are three short-cuts for filling up this matrix: (1) sign=1 sets all entries to 1 (non-negativity constraints). This is useful if all pairs of outcomes are assumed to be positively correlated (potentially after changing the sign of some outcomes). (2) code=NA sets all diagonal entries to 1 and all off-diagonal entries to NA (no constraints). (3) sign=NULL uses Spearman correlation to determine the entries, with −1 for significant negative, 0 for insignificant, 1 for significant positive correlations.

**elastic net**: alpha.base controls input-output effects, alpha.meta controls output-output effects; lasso renders sparse models (alpha= 1), ridge renders dense models (alpha= 0)

Value

This function returns an object of class joinet. Available methods include predict, coef, and weights. The slots base and meta each contain q cv.glmnet-like objects.

References

predict.joinet

Make Predictions

Description

Predicts outcome from features with stacked model.

Usage

## S3 method for class 'joinet'
predict(object, newx, type = "response", ...)

Arguments

object      joinet object
newx        covariates: numeric matrix with \(n\) rows (samples) and \(p\) columns (variables)
type        character "link" or "response"
...          further arguments (not applicable)

Value

This function returns predictions from base and meta learners. The slots `base` and `meta` each contain a matrix with \(n\) rows (samples) and \(q\) columns (variables).
weights.joinet

**Examples**

```r
## Not run:
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
Y <- replicate(n=q,expr=rnorm(n=n,mean=rowSums(X[,1:5])))
Y[,1] <- 1*(Y[,1]>median(Y[,1]))
object <- joinet(Y=Y,X=X,family=c("binomial","gaussian","gaussian"))
predict(object,newx=X)
## End(Not run)
```

**Description**

Extracts coefficients from the meta learner, i.e. the weights for the base learners.

**Usage**

```r
## S3 method for class 'joinet'
weights(object, ...)
```

**Arguments**

- `object` : `joinet` object
- `...` : further arguments (not applicable)

**Value**

This function returns a matrix with \(1 + q\) rows and \(q\) columns. The first row contains the intercepts, and the other rows contain the slopes, which are the effects of the outcomes in the row on the outcomes in the column.

**Examples**

```r
## Not run:
n <- 50; p <- 100; q <- 3
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
Y <- replicate(n=q,expr=rnorm(n=n,mean=rowSums(X[,1:5])))
object <- joinet(Y=Y,X=X)
weights(object)
## End(Not run)
```
Index

* documentation
  joinet-package, 2

cof, 7
coef.joinet, 3
cv.glmnet, 4, 5, 7
cv.joinet, 4, 8
glmnet, 5, 7
joinet, 2, 3, 6, 8, 9
joinet-package, 2

predict, 7
predict.joinet, 8

weights, 7
weights.joinet, 9