Package ‘gamlss.lasso’

May 21, 2021

Description Interface for extra high-dimensional smooth functions for Generalized Additive Models for Location Scale and Shape (GAMLSS) including (adaptive) lasso, ridge, elastic net and least angle regression.

Title Extra Lasso-Type Additive Terms for GAMLSS

LazyLoad yes

Version 1.0-1

Date 2021-05-01

Depends R (>= 2.15.0), gamlss (>= 2.4.0), glmnet, lars, Matrix

Suggests lattice

Maintainer Florian Ziel <florian.ziel@uni-due.de>

License GPL-2 | GPL-3

URL https://www.gamlss.com/

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-21 09:10:02 UTC

Author Florian Ziel [aut, cre],
  Peru Muniain [aut],
  Mikis Stasinopoulos [ctb]

R topics documented:

  gamlss.lasso-package .................................................. 2
  gamlss.gnet ............................................................. 3
  gamlss.lrs ............................................................... 5
  gnet ................................................................. 6
  lrs ................................................................. 10

Index 13
Extra Lasso-Type Additive Terms for GAMLSS

Description

Interface for extra high-dimensional smooth functions for Generalized Additive Models for Location Scale and Shape (GAMLSS) including (adaptive) lasso, ridge, elastic net and least angle regression.

Details

The DESCRIPTION file:

Package: gamlss.lasso
Description: Interface for extra high-dimensional smooth functions for Generalized Additive Models for Location Scale and Shape (GAMLSS) including (adaptive) lasso, ridge, elastic net and least angle regression.
Title: Extra Lasso-Type Additive Terms for GAMLSS
LazyLoad: yes
Version: 1.0-1
Date: 2021-05-01
Depends: R (>= 2.15.0), gamlss (>= 2.4.0), glmnet, lars, Matrix
Suggests: lattice
Authors@R: c(person("Florian", "Ziel", role = c("aut", "cre"), email = "florian.ziel@uni-due.de"), person("Peru", "Muniain", role = "aut"), person("Mikis", "Stasinopoulos", role = "ctb"))
Maintainer: Florian Ziel <florian.ziel@uni-due.de>
License: GPL-2 | GPL-3
URL: https://www.gamlss.com/
NeedsCompilation: no
Packaged: 2021-04-01 06:51:36 UTC; florian
Repository: CRAN
Date/Publication: 2021-04-01 06:55:55 UTC
Author: Florian Ziel [aut, cre], Peru Muniaín [aut], Mikis Stasinopoulos [ctb]

Index of help topics:

gamlss.gnet Support for Function gnet()
gamlss.lasso-package Extra Lasso-Type Additive Terms for GAMLSS
gamlss.lrs Support for Function lrs()
gnet (Adaptive) elastic net in GAMLSS
lrs Least angle regression and lasso in GAMLSS

Author(s)

NA

Maintainer: Florian Ziel <florian.ziel@uni-due.de>
References


(see also https://www.gamlss.com/).


See Also
gamlss, gamlss.family, gamlss.add

Examples

# Constructing the data
library(gamlss.lasso)
set.seed(123)
n<- 500
d<- 50
X<- matrix(rnorm(n*d), n,d)
BETA<- cbind( "mu"=rbinom(d,1,.1), "sigma"= rbinom(d,1,.1)*.3)
ysd<- exp(1 + tcrossprod( BETA[,2],X))
data<- cbind(y=as.numeric(rnorm(n, sd=ysd))+t(tcrossprod( BETA[,1],X)), as.data.frame(X))

# Estimating the model with gnet default setting
mod <- gamlss(y~gnet(x.vars=names(data)[-1] ),
sigma.fo=gnet(x.vars=names(data)[-1]), data=data, family=NO,
i.control = glim.control(cyc=1, bf.cyc=1))

# Estimated parameters are available at
rbind(true=BETA[,1],estimate=tail(getSmo(mod, "mu") ,1)[[1]]$beta )## beta for mu
rbind(true=BETA[,2],estimate=tail(getSmo(mod, "sigma") ,1)[[1]]$beta )## beta for sigma
Description

This is support for the smoother function gnet() an interface for Tibshirani’s glmnet() function. It is not intended to be called directly by users.

Usage

gamlss.gnet(x, y, w, xeval = NULL, ...)

Arguments

x the explanatory variables
y iterative y variable
w iterative weights
xeval if xeval=TRUE then prediction is used
... for extra arguments

Value

No return value, called for GAMLSS gnet procedure.

Author(s)

Florian Ziel, Peru Muniaín and Mikis Stasinopoulos

References


(see also https://www.gamlss.com/).


See Also

gnet
Support for Function lrs()

Description

This is support for the smoother function lrs() an interface for Brad Efron and Trevor Hastie for lars() function. It is not intended to be called directly by users.

Usage

gamlss.lrs(x, y, w, xeval = NULL, ...)

Arguments

- **x**: the explanatory variables
- **y**: iterative y variable
- **w**: iterative weights
- **xeval**: if xeval=TRUE then prediction is used
- **...**: for extra arguments

Value

No return value, called for GAMLSS lrs procedure.

Author(s)

Florian Ziel, Peru Muniain and Mikis Stasinopoulos

References


(see also https://www.gamlss.com/).


See Also

lrs
(Adaptive) elastic net in GAMLSS

Description

This function allows estimating the different components of a GAMLSS model (location, shape, scale parameters) using the (adaptive) elastic net (with adaptive lasso as default special case) estimation method via glmnet. This method is appropriate for models with many variables.

Usage

```r
library(GAMLSS)

# Example of usage
# Data
X <- matrix(rnorm(1000), nrow = 100)
# Use gnet function
fit <- gnet(X = X, lambda = NULL, method = 'IC', type = 'agg', ICpen = 'AIC', k.se = 0, sparse = FALSE, control = gnet.control(...))
```

Arguments

- `X` The data frame containing the explanatory variables.
- `x.vars` Indicates the name of the variables that must be included as explanatory variables from data the data object of GAMLSS. The explanatory variables must be included by `X` or by `x.vars`.
- `lambda` The provided lambda grid. By default `NULL`.
- `method` The method used to calculate the optimal lambda. If `method`="IC" information criteria are used, the penalization for the information criterion is selected in `ICpen`. If `method`="CV" cross validation resp. sampling is used, the penalization for the cross-validation is selected in `CVp`.
- `type` The way to select the optimal lambda across the subsample fits. If `type`="sel" the optimal lambda is computed by selection. If `method="agg"` the optimal lambda is computed by aggregation.
- `ICpen` The penalization for the information criteria. If `ICpen="AIC"` or `ICpen=2` the optimal lambda is computed by Akaike Information Criterion. If `ICpen="BIC"` the optimal lambda is computed by Bayesian Information Criterion. If `ICpen="HQC"` the optimal lambda is computed by Hannan-Quinn Information Criterion.
- `CVp` The penalization for the cross-validation, establishes the power of the error term. By default is equal to 2, i.e. squared error.
- `k.se` This parameter establishes how many times the standard deviation is summed to the mean to select the optimal lambda. By default is equal to 0.
adaptive

This parameter specifies if adaptive lasso shall be used, the default is 1. If NULL then standard lasso is used, otherwise adaptive lasso with penalty weights \((\text{abs}(\beta)+\epsilon)^{(-\text{adaptive})}\) where \(\beta\) is chosen from an initial standard lasso estimate and \(\epsilon\) is specified by the next parameter. Note, estimating standard lasso requires about half of computation time, but adaptive lasso has smaller bias and satisfies the oracle property.

epsilon

This parameter specifies the adaptive lasso penalty weights. The default is \(1/\sqrt{\text{dim}(X)[1]}\).

subsets

The subsets for cross-validation, information criteria or bootstrapping, by default 5 random folds are selected.

sparse

If sparse converts input matrix for glmnet into a sparse Matrix, may reduces computation time for sparse designs.

control

List of further input parameters for glmnet, e.g. alpha for elastic net parameters.

... for extra arguments

family

Either a character string representing one of the built-in families, or else a glm() family object.

offset

A vector of length no bs that is included in the linear predictor (a no bs x no c matrix for the "multinomial" family). Useful for the "poisson" family (e.g. log of exposure time), or for refining a model by starting at a current fit. Default is NULL. If supplied, then values must also be supplied to the predict function.

alpha

The elastic net mixing parameter, with \(0 \leq \alpha \leq 1\). The penalty is defined as

\[
(1 - \alpha)/2||\beta||_2^2 + \alpha||\beta||_1.
\]

alpha=1 is the lasso penalty, and alpha=0 the ridge penalty. Default is lasso.

nlambda

Size of the tuning parameter grid, default is 100. It is irrelevant if lambda is explicitly specified.

lambda.min.ratio

Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default is 0.001. A very small value of lambda.min.ratio will lead to a saturated fit in the no bs < no vars case. This is undefined for "binomial" and "multinomial" models, and glmnet will exit gracefully when the percentage deviance explained is almost 1. It is irrelevant if lambda is explicitly specified.

standardize

Logical flag for X or x.vars variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is standardize=TRUE and it is highly recommended.

intercept

Should intercept(s) be fitted (default=TRUE) or set to zero (FALSE).

thresh

Convergence threshold for coordinate descent. Each inner coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than thresh times the null deviance. Defaults value is 1E-7.

dfmax

Limit the maximum number of variables in the model. Useful for very large no vars, if a partial path is desired.

pmax

Limit the maximum number of variables ever to be nonzero.
exclude Indices of variables to be excluded from the model. Default is none. Equivalent to an infinite penalty factor (next item).

penalty.factor Separate penalty factors can be applied to each coefficient. This is a number that multiplies lambda to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables (and implicitly infinity for variables listed in exclude). Note: the penalty factors are internally rescaled to sum to nvars, and the lambda sequence will reflect this change.

lower.limits Vector of lower limits for each coefficient; default -Inf. Each of these must be non-positive. Can be presented as a single value (which will then be replicated), else a vector of length nvars.

upper.limits Vector of upper limits for each coefficient; default Inf. See lower.limits.

maxit Maximum number of passes over the data for all lambda values; default is $10^5$.

type.gaussian Two algorithm types are supported for (only) family="gaussian". The default when nvar<500 is type.gaussian="covariance", and saves all inner-products ever computed. This can be much faster than type.gaussian="naive", which loops through nobs every time an inner-product is computed. The latter can be far more efficient for nvar >> nobs situations, or when nvar > 500.

type.logistic If "Newton" then the exact hessian is used (default), while "modified.Newton" uses an upper-bound on the hessian, and can be faster.

Details
The estimation of the lambda is carried out by BIC by default. If the objective is to predict the model must be defined by x.vars. Different types of subsets must be constructed if bootstrapping and aggregation are applied, as in this case observations might be repeated.

Value
This function returns a smooth object of the GAMLSS model. It contains the estimated parameters and related characteristics for the glmnet component in the GAMLSS model we are estimating.

Author(s)
Florian Ziel, Peru Muniain and Mikis Stasinopoulos

References


Examples

# Constructing the data
library(gamlss.lasso)
set.seed(123)
n<- 500
d<- 50
X<- matrix(rnorm(n*d), n,d)
BETA<- cbind( "mu"=rbinom(d,1,.1), "sigma"= rbinom(d,1,.1)*.3)
ysd<- exp(1 + tcrossprod( BETA[,2],X))
data<- cbind(y=as.numeric(rnorm(n, sd=ysd))+t(tcrossprod( BETA[,1],X)), as.data.frame(X))

# Estimating the model using default setting: adaptive lasso with BIC tuning
mod <- gamlss(y~gnet(x.vars=names(data)[-1]),
              sigma.fo=~gnet(x.vars=names(data)[-1]), data=data,
              family=NO, i.control = glim.control(cyc=1, bf.cyc=1))

# Estimating the model with standard lasso (BIC tuning)
mod.lasso <- gamlss(y~gnet(x.vars=names(data)[-1], adaptive=NULL),
                     sigma.fo=~gnet(x.vars=names(data)[-1], adaptive=NULL), data=data,
                     family=NO, i.control = glim.control(cyc=1, bf.cyc=1))

# Estimated parameters are available at
rbind(true=BETA[,1],alasso=tail(getSmo(mod, "mu") ,1)[[1]]$beta,
      lasso=tail(getSmo(mod.lasso, "mu") ,1)[[1]]$beta) ##beta for mu
rbind(true=BETA[,2],alasso=tail(getSmo(mod, "sigma") ,1)[[1]]$beta,
      lasso=tail(getSmo(mod.lasso, "sigma") ,1)[[1]]$beta)##beta for sigma

# Estimating with other setting
nfolds<- 6
n<- dim(data)[1]
# folds for cross-validation and bootstrap
CVfolds<- lapply(as.data.frame(t(sapply(sample(rep_len(1:nfolds,length=n),replace=FALSE)
                                          ,"!", 1:nfolds)))), which)
BOOTfolds<- lapply(as.data.frame(matrix(sample(1:n, nfolds*n, replace=TRUE), n)),sort)

#Bootstrap + Aggregation = Bagging:
mod1 <- gamlss(y~gnet(x.vars=names(data)[-1], method="CV", type="agg", subsets=BOOTfolds),
               sigma.fo=gnet(x.vars=names(data)[-1]), data=data, family=NO,
               i.control = glim.control(cyc=1, bf.cyc=1))


lrs

Least angle regression and lasso in GAMLSS

Description

This function allows estimating the different components of a GAMLSS model (mean, sd. dev., skewness and kurtosis) using the elastic net (with lasso as default special case) estimation method via glmnet. This method is appropriate for models with many variables.

Usage

lrs(X = NULL, x.vars = NULL, lambda = NULL, method = c("IC","CV"), type = c("agg","sel"), ICpen = c("BIC", "HQC", "AIC"), CVp = 2, k.se = 0, subsets = NULL, lars.type= "lasso", use.gram = TRUE, eps = .Machine$double.eps, max.steps = NULL, ...)

Arguments

X
The data frame containing the explanatory variables.

x.vars
Indicates the name of the variables that must be included as explanatory variables from data the data object of GAMLSS. The explanatory variables must be included by X or by x.vars.

lambda
The provided lambda grid. By default NULL.

method
The method used to calculate the optimal lambda. If method="IC" information criteria are used, the penalization for the information criterion is selected in ICpen. If method="CV" cross validation resp. sampling is used, the penalization for the cross-validation is selected in CVp.

type
The way to select the optimal lambda across the subsample fits. If type="sel" the optimal lambda is computed by selection. If method="agg" the optimal lambda is computed by aggregation.
ICpen: The penalization for the information criteria. If ICpen="AIC" or ICpen=2 the optimal lambda is computed by Akaike Information Criterion. If ICpen="BIC" the optimal lambda is computed by Bayesian Information Criterion. If ICpen="HQC" the optimal lambda is computed by Hannan-Quinn Information Criterion.

CVp: The penalization for the cross-validation, establishes the power of the error term. By default is equal to 2, i.e. squared error.

k.se: This parameter establishes how many times the standard deviation is summed to the mean to select the optimal lambda. By default is equal to 0.

subsets: The subsets for cross-validation, information criteria or bootstrapping, by default 5 random fold are selected.

lars.type: As in lars, lars type, e.g. "lasso", "lar" (least angle regression), "forward.stagewise" or "stepwise".

use.gram: States if Gramian should be precomputed, default TRUE - recommended as gamlss will call lars often during the estimation.

eps: As in lars, a small constant.

max.steps: As in lars, number of updating steps (for "lars" method equal to number of variables, for "lasso" it can be smaller), default NULL.

... for extra arguments

Details

The estimation of the lambda is carried out by BIC by default. If the objective is to predict the model must be defined by x.vars. Different types of subsets must be constructed if bootstrapping and aggregation are applied, as in this case observations might be repeated.

Value

This function returns a smooth object of the GAMLSS model. It contains the estimated parameters and related characteristics for the lars component in the GAMLSS model we are estimating.

Author(s)

Florian Ziel, Peru Muniain and Mikis Stasinopoulos

References


Examples

# Constructing the data
library(gamlss.lasso)
set.seed(123)
n <- 500
d <- 50
X <- matrix(rnorm(n*d), n,d)
BETA <- cbind( "mu"=rbinom(d,1,.1), "sigma"= rbinom(d,1,.1)*.3)
ysd <- exp(1 + tcrossprod( BETA[,2],X))
data <- cbind(y=as.numeric(rnorm(n,sd=ysd)) + t(tcrossprod( BETA[,1],X)),as.data.frame(X))

# Estimating the model with lrs default setting
mod <- gamlss(y~lrs(x.vars=names(data)[-1] ),
              sigma.fo=~lrs(x.vars=names(data)[-1]), data=data, family=NO,
             i.control = glim.control(cyc=1, bf.cyc=1))

# Estimated parameters are available at
rbind(true=BETA[,1],estimate=tail(getSmo(mod, "mu") ,1)[[1]]$beta )## beta for mu
rbind(true=BETA[,2],estimate=tail(getSmo(mod, "sigma") ,1)[[1]]$beta )## beta for sigma
Index

* package
  gamlss.lasso-package, 2

* regression
  gamlss.gnet, 3
  gamlss.lrs, 5

gamlss, 3
gamlss.add, 3
gamlss.family, 3
gamlss.gnet, 3
gamlss.lasso (gamlss.lasso-package), 2
gamlss.lasso-package, 2
gamlss.lrs, 5
gnet, 4, 6

lrs, 5, 10