Package ‘betaboost’

July 7, 2018

Type Package
Title Boosting Beta Regression
Version 1.0.1
Date 2018-07-07
Author Andreas Mayr, Benjamin Hofner, Leonie Weinhold, Matthias Schmid
Maintainer Andreas Mayr <mayr@uni-bonn.de>
Description Implements boosting beta regression for potentially high-dimensional data (Mayr et al., 2018 <doi:10.1093/ije/dyy093>). The 'betaboost' package uses the same parametrization as 'betareg' (Cribari-Neto and Zeileis, 2010 <doi:10.18637/jss.v034.i02>) to make results directly comparable. The underlying algorithms are implemented via the R add-on packages 'mboost' (Hofner et al., 2014 <doi:10.1007/s00180-012-0382-5>) and 'gamboostLSS' (Mayr et al., 2012 <doi:10.1111/j.1467-9876.2011.01033.x>).
Depends mboost, gamboostLSS
Suggests gamlss.dist, knitr, rmarkdown, betareg
VignetteBuilder knitr
LazyLoad yes
LazyData yes
License GPL-3
URL For source code, development versions and issue tracker see https://github.com/boost-R/betaboost
NeedsCompilation no
Repository CRAN
Date/Publication 2018-07-07 20:30:03 UTC

R topics documented:

betaboost .......................................................... 2
BetaReg ............................................................ 4
confint.betaboost ................................................. 6
betaboost

Function for boosting beta regression

Description

Wrapper function to use mode-based boosting via mboost or gamboostLSS to fit beta regression.

Usage

betaboost(formula = NULL, phi.formula = NULL, data = list(), sl = 0.01,
       iterations = 100, form.type = c("classic", "betaboost"),
       start.mu = NULL, start.phi = NULL,
       stabilization = c("none", "MAD", "L2"),
       y = NULL, x = NULL, mat.parameter = c("mean", "both"),
       mat.effect = c("linear", "smooth"), ...)

Arguments

formula description of the model to be fit for location parameter (mu).
phi.formula description of the model to be fit for precision parameter (phi).
data a data frame containing the variables.
iterations number of boosting iterations to be used.
sl step-length, default is 0.01
form.type formula type: either gamboost (y ~ bols(x1) + bbs(x2)) using the mboost interface for specifying base-learners, or classic (y ~ x1 + s(x2)).
start.mu offset value for mu, must be > 0 and < 1; will be estimated from the outcome if none is specified (default).
start.phi offset value for phi, must be > 0; will be estimated from the outcome if none is specified (default).
stabilization governs if the negative gradient should be standardized in each boosting step. It can be either "none", "MAD" or "L2". Only applicable when besides mu also phi is modeled (extended beta regression).
y response vector when no formula is specified.
x matrix of explanatory variables when no formula is specified, per default they are included as linear effects: can be changed to smooth via mat.effects.
betaboost

mat.effect controls what type of effect the entries in matrix x have on the response. It can be either linear or smooth, while linear is the default. Only applicable if no formula is provided, but y and x.

mat.parameter controls for which parameters the entries in matrix x are included. It can be either mean (classical beta regression) or both (extended beta regression), while mean is the default. Only applicable if no formula is provided, but y and x.

Details

A wrapper function to fit beta regression via different boosting functions.

Value

A boosting object.

References


See Also

The original function **gamboostLSS** and **gamboost** from the model-based boosting framework.

Examples

```r
#--------- data example
data(QoLdata)

## Model for mu
b1 <- betaboost(formula = QoL ~ arm + pain, data = QoLdata,
                 iterations = 500)

# Coefficients
coef(b1, off2int = TRUE)

# Phi
nuisance(b1)

## Model for mu and phi
b2 <- betaboost(formula = QoL ~ arm + pain, data = QoLdata,
                 iterations = 1000,
                 phi.formula = QoL ~ arm + pain)
```
# Coefficients
coef(b2, off2int = TRUE)

#-------- simple simulated example

require(gamlss.dist)
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
x4 <- rnorm(100)
y <- rBE(n = 100, mu = plogis(x1 + x2),
        sigma = plogis(x3 + x4))
data <- data.frame(y, x1, x2, x3, x4)
data <- data[!data$y %in% c(0, 1), ]

# 'classic' beta regression
b3 <- betaboost(formula = y ~ x1 + x2, data = data,
                 iterations = 120)
coef(b3)

# beta regression including modeled precision parameter
b4 <- betaboost(formula = y ~ x1 + x2,
                phi.formula = y ~ x3 + x4,
                data = data, iterations = 120)

# with smooth effects for x1 and x3
b5 <- betaboost(formula = y ~ s(x1) + x2,
                phi.formula = y ~ s(x3) + x4, form.type = "classic",
                data = data, iterations = 120)

# using matrix interface
b6 <- betaboost(y = data$y, x = data[, 2:5], iterations = 200,
                mat.parameter = "both")

---

**BetaReg**

**BetaReg family for boosting beta regression**

**Description**

BetaReg implements a mboost family object to boost beta regression.

**Usage**

*BetaReg(mu = NULL, phirange = c(.001, 1000))*
Arguments

mu starting value for location parameter.
phirange range for the optimization of scale parameter \(\phi\).

Details

BetaReg implements 'classical' beta regression for model-based boosting. Location parameter \(\mu\) is modeled by additive predictor, scale parameter \(\phi\) is simultaneously optimized as a scalar and treated as nuisance.

Author(s)

Andreas Mayr <mayr@uni-bonn.de>

References


Examples

```r
require(gamlss.dist)
# simple simulated example
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
x4 <- rnorm(100)
y <- rBE(n = 100, mu = plogis(x1 + x2),
          sigma = plogis(x3 + x4))
data <- data.frame(y, x1, x2, x3, x4)
data <- data[data$y %in% c(0, 1),]

# 'classic' beta regression
b1 <- betaboost(formula = y ~ x1 + x2, data = data,
                  iterations = 120)
coef(b1)

# compare to mboost
b2 <- glmboost(y ~ x1 + x2, data = data, family = BetaReg())
coef(b2)

# different values due to different defaults for step length and mstop

# same model with mboost
b3 <- glmboost(y ~ x1 + x2, data = data, family = BetaReg(),
```
confint.betaboost

control = boost_control(mstop = 120, nu = 0.01)
coef(b3)
coef(b1)

confint.betaboost  Pointwise Bootstrap Confidence Intervals

Description
Compute pointwise bootstrap confidence intervals

Usage
## S3 method for class 'betaboost'
confint(object, ...)

Arguments

object  a fitted model object of class betaboost for which the confidence intervals should be computed.

...  additional arguments. See confint.mboost for further details.

Details
Use a nested bootstrap approach to compute pointwise confidence intervals for the predicted partial functions or regression parameters. The approach is further described in Hofner et al. (2016).
Note that confidence intervals are currently only provided for beta regression models with constant precision parameter (i.e., phi cannot be modeled as a function of covariates).

Value
An object of class glmboost.ci or mboost.ci with special print and/or plot functions.

Author(s)
Benjamin Hofner <benjamin.hofner@pei.de>

References

See Also
confint.mboost
Description

A data frame with 6 quality of life measures for 60 patients, originally published in the QoL.R package: Analysis of Health-Related Quality of Life in oncology. For more details, see the CRAN archive, the corresponding GitHub page, or the references below.

Usage

data(dataqol2)

Format

- id  patient identification number
- time  visit number for quality of life assessment
- date  date of quality of life measure
- QoL  score of global quality of life on a 0-100 scale in order that a high score reflects a high quality of life level
- pain  score of pain on a 0-100 sale in order that a high score reflects a high level of pain
- arm  treatment arm equal to 0 or 1
- death  date of death. Missing if the patient is not died

Author(s)

Amelie Anota aanota@chu-besancon.fr

References


make_mboostform  
Building mboost formulas

Description

Transforms 'classic' formula objects ($y \sim x1 + s(x2)$) to mboost formulas $y \sim bols(x1) + bbs(x2)$.

Usage

```r
make_mboostform(formula, data = NULL)
add_bolsform(formula, data = NULL)
```

Arguments

- **formula**: formula object describing a model.
- **data**: data set, only necessary in case of "-." formulas

Value

formula

Examples

```r
make_mboostform(y ~ x1 + s(x2))
```

---

predict.betaboost  
Predictions for betaboost models

Description

Make predictions for betaboost models

Usage

```r
# S3 method for class 'betaboost'
predict(object, newdata = NULL,
        type = c("link", "response", "class"), which = NULL,
        aggregate = c("sum", "cumsum", "none"), ...)
```
**Arguments**

- **object**: a fitted model object of class `betaboost` for which the predictions should be made.
- **newdata**: optional; A data frame in which to look for variables with which to predict or with which to plot the marginal prediction intervals.
- **type**: the type of prediction required. The default is on the scale of the predictors; the alternative "response" is on the scale of the response variable. Thus for a binomial model the default predictions are on the log-odds scale (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "class" option returns predicted classes.
- **which**: a subset of base-learners to take into account when computing predictions or coefficients. If which is given (as an integer vector or characters corresponding to base-learners), a list or matrix is returned. In `plot_pi` the argument which must be specified and it must be given as a character string containing the name of the variable.
- **aggregate**: a character specifying how to aggregate predictions or coefficients of single base-learners. The default returns the prediction or coefficient for the final number of boosting iterations. "cumsum" returns a matrix with the predictions for all iterations simultaneously (in columns). "none" returns a list with matrices where the jth columns of the respective matrix contains the predictions of the base-learner of the jth boosting iteration (and zero if the base-learner is not selected in this iteration).
- **...**: additional arguments. Currently, only parameter is supported. See `predict.mboostLSS` for further details.

**Details**

The `predict` function can be used for predictions for the distribution parameters depending on new observations.

**Author(s)**

Benjamin Hofner <benjamin.hofner@pei.de>

**See Also**

`predict.mboost` and `predict.mboostLSS`

**Examples**

```r
## load data
data(QoLdata)

## define test data
test <- QoLdata[1:10,]
train <- QoLdata[11:nrow(QoLdata),]
```
## QoLdata

**Examplary Quality of Life data**

### Description
A data frame with quality of life measures for 57 patients, originally published in the QoLR package: Analysis of Health-Related Quality of Life in oncology. For more details, see the CRAN archive, the corresponding GitHub pare, or the references below.

### Usage
```r
data(QoLdata)
```

### Format
- **id**: patient identification number
- **time**: visit number for quality of life assessment, in this case all measurements are from the first time-point (hence, all are set to 0)
- **date**: date of quality of life measure
- **QoL**: score of global quality of life on a 0-1 scale in order that a high score reflects a high quality of life level
- **pain**: score of pain on a 0-100 scale in order that a high score reflects a high level of pain
- **arm**: treatment arm equal to 0 or 1
- **death**: date of death. Missing if the patient is not died

### Author(s)
Amelie Anota aanota@chu-besancon.fr
References


See Also

Original data set dataqol.

Examples

```r
# was constructed from dataqol
data(dataqol)
data(QoLdata)
## take one time-point
dataqol <- dataqol2[dataqol2$time == 0,]
## remove missings
dataqol <- dataqol[complete.cases(dataqol[,c("QoL", "arm", "pain")]),]
## rescale outcome to [0,1]
dataqol$QoL <- dataqol$QoL/100

identical(dataqol, QoLdata)
```

---

R2.betaboost Computing pseudo R^2 for betaboost models.

Description

Computes different pseudo R^2 for betaboost models

Usage

R2.betaboost(model, data, newdata = NULL)

Arguments

model A boosting model object for beta regression.
data Underlying data frame
newdata test-data (optional), if omitted R^2 is computed on data (training-data)
References


Examples

```r
# simple simulated example
require(gamlss.dist)
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
x4 <- rnorm(100)
y <- rBE(n = 100, mu = plogis(x1 + x2),
         sigma = plogis(x3 + x4))
data <- data.frame(y, x1, x2, x3, x4)
data <- data[!data$y%in%c(0,1),]
rm(x1, x2, x3, x4, y)

b1 <- betaboost(formula = y ~ x1 + x2,
                 phi.formula = y ~ x3 + x4,
                 data = data, form.type = "classic",
                 iterations = 120)
R2.betaboost(b1, data = data)
```
Index

*Topic **methods**
  confint.betaboost, 6
  predict.betaboost, 8

add_bolsform(make_mboostform), 8

betaboost, 2
BetaReg, 4

confint (confint.betaboost), 6
confint.betaboost, 6
confint.mboost, 6

dataqol2, 7, 11

gamboost, 3
gamboostLSS, 7

LH.betaboost (R2.betaboost), 11

make_mboostform, 8

predict (predict.betaboost), 8
predict.betaboost, 8
predict.mboost, 9
predict.mboostLSS, 9

QoLdata, 10

R2.betaboost, 11