Package 'relgam'

October 14, 2022

Type Package

Title Reluctant Generalized Additive Models

Version 1.0

Author Kenneth Tay, Robert Tibshirani

Maintainer Kenneth Tay <kjytay@stanford.edu>

Description A method for fitting the entire regularization path of the reluctant generalized additive model (RGAM) for linear regression, logistic, Poisson and Cox regression models. See Tay, J. K., and Tibshirani, R., (2019) <arXiv:1912.01808> for details.

URL https://arxiv.org/abs/1912.01808

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports glmnet, foreach

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-13 16:10:02 UTC

R topics documented:

cv.rgam *Cross-validation for reluctant generalized additive model (rgam)*

Description

Does k-fold cross-validation for rgam.

Usage

```
cv.rgam(x, y, lambda = NULL, family = c("gaussian", "binomial",
  "poisson", "cox"), offset = NULL, init_nz, gamma, nfolds = 10,
  foldid = NULL, keep = FALSE, parallel = FALSE, verbose = TRUE,
  ...)
```

Arguments

x	Input matrix, of dimension nobs x nvars; each row is an observation vector.
y	Response y as in rgam.
lambda	A user-supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence; supplying a value of lambda overrides this.
family	Response type. Either "gaussian" (default) for linear regression, "binomial" for logistic regression, "poisson" for Poisson regression or "cox" for Cox regression.
offset	Offset vector as in rgam.
init_nz	A vector specifying which features we must include when computing the non-linear features. Default is to construct non-linear features for all given features.
gamma	Scale factor for non-linear features (vs. original features), to be between 0 and 1. Default is 0.8 if init_nz = c(), 0.6 otherwise.
nfolds	Number of folds for CV (default is 10). Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds = 4.
foldid	An optional vector of values between 1 and nfolds identifying what fold each observation is in. If supplied, nfolds can be missing.
keep	If keep = TRUE, a prevalidated array is returned containing fitted values for each observation at each value of lambda. This means these fits are computed with this observation and the rest of its fold omitted. Default is FALSE.

`parallel` If TRUE, use parallel foreach to fit each fold. Must register parallel before hand, such as doMC or others. Note that this also passes `parallel = TRUE` to the `rgam()` call within each fold. Default is FALSE.

`verbose` Print information as model is being fit? Default is `TRUE`.

`...` Other arguments that can be passed to `rgam`.

Details

The function runs `rgam` nfolds+1 times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The error is accumulated, and the average error and standard deviation over the folds is computed.

Note that `cv.rgam` only does cross-validation for lambda but not for the degrees of freedom hyperparameter.

Value

An object of class `"cv.rgam"`.

`glmfit` A fitted `rgam` object for the full data.

`lambda` The values of `lambda` used in the fits.

`nzero_feat` The number of non-zero features for the model `glmfit`.

`nzero_lin` The number of non-zero linear components for the model `glmfit`.

`nzero_nonlin` The number of non-zero non-linear components for the model `glmfit`.

`fit.preval` If `keep=TRUE`, this is the array of prevalidated fits.

`cvm` The mean cross-validated error: a vector of length `length(lambda)`.

`cvse` Estimate of standard error of `cvm`.

`cvlo` Lower curve = `cvm - cvsd`.

`cvup` Upper curve = `cvm + cvsd`.

`lambda.min` The value of `lambda` that gives minimum `cvm`.

`lambda.1se` The largest value of `lambda` such that the CV error is within one standard error of the minimum.

`foldid` If `keep=TRUE`, the fold assignments used.

`name` Name of error measurement used for CV.

`call` The call that produced this object.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)

cvfit <- cv.rgam(x, y)
```

```
# specify number of folds
cvfit <- cv.rgam(x, y, nfolds = 5)
```

getf *Get RGAM model component for one feature*

Description

Returns the additive component of the RGAM model for a given feature at given data points, i.e. f_j(X_j).

Usage

```
getf(object, x, j, index)
```

Arguments

`object` Fitted `rgam` object.

`x` Data for which we want the additive component. If `x` is a matrix, it assumed that `X_j` is the `j`th column of this matrix. If `x` is a vector, it is assumed to be `X_j` itself.

`j` The index of the original feature whose additive component we want.

`index` Index of lambda value for which plotting is desired. Default is the last lambda value in `object$lambda`.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)

fit <- rgam(x, y)

# get the additive component for the feature 6, x as matrix
f6 <- getf(fit, x, 6)  # last value of lambda
plot(x[, 6], f6)
f6 <- getf(fit, x, 6, index = 20)  # f1 at 20th value of lambda
plot(x[, 6], f6)

# get the additive component for the feature 6, x as vector
new_x6 <- seq(-1, 1, length.out = 30)
new_f6 <- getf(fit, new_x6, 6)  # last value of lambda
plot(new_x6, new_f6)
```

makef *Make non-linear features*

Description

Internal function for making non-linear features.

Usage

```
makef(x, r, df, tol = 0.01, removeLin = T)
```

Arguments

`x`	Input matrix, of dimension `nobs` x `nvars`; each row is an observation vector.
`r`	Vector of residuals.
`df`	Degrees of freedom for the fit.
`tol`	A tolerance for same-ness or uniqueness of the `x` values. To be passed to the `smooth.spline()` function. Default is `0.01`.
`removeLin`	If `TRUE` (default), removes the linear component from the newly created non-linear features.

Value

A list:

`f`	Non-linear features associated with the features in `x`.
`spline_fit`	A list of the spline fits of the residual against each feature. Useful for creating the non-linear features for new data.
`lin_comp_fit`	If `removeLin = TRUE`, a list of coefficients for simple linear regression of non-linear feature on original feature. Useful for creating the non-linear features for new data.

myroc *Compute ROC and other performance measures for binomial model*

Description

Given a vector of true outcomes and a vector of predictions, returns a list containing performance measures.

Usage

```
myroc(ytest, rit, N = 100)
```

Arguments

`ytest`	True test outcome: vector of 0s and 1s.
`rit`	Predictions for the true outcome. Should be vector of continuous variables between 0 and 1.
`N`	Number of breakpoints where we evaluate the performance measures. Default is 100.

Details

We currently evaluate the performance measures at 100 quantiles of the predicted values; this can be adjusted via the `N` option.

Value

A list of performance measures and intermediate computations.

`sens`	Vector of sensitivity values.
`spec`	Vector of specificity values.
`ppv`	Vector of PPV values.
`npv`	Vector of NPV values
`area`	Area under ROC curve (AUC).
`se`	Standard error for AUC.
`cutp`	Cut points at which the performance measures were computed.
`cutp.max`	Cut point which maximizes (sens + spec) / 2.

`plot.cv.rgam` *Plot the cross-validation curve produced by "cv.rgam" object*

Description

Plots the cross-validation curve produced by a `cv.rgam` object, along with upper and lower standard deviation curves, as a function of the `lambda` values used. The plot also shows the number of nonzero features picked for each value of lambda.

Usage

```
## S3 method for class 'cv.rgam'
plot(x, sign.lambda = 1, ...)
```

Arguments

`x`	Fitted "`cv.rgam`" object.
`sign.lambda`	Either plot against `log(lambda)` (default) or `-log(lambda)` (if `sign.lambda = -1`).
`...`	Other graphical parameters to plot.

Details

A plot is produced and nothing is returned.

See Also

`rgam` and `cv.rgam`.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)

cvfit <- cv.rgam(x, y)
plot(cvfit)
```

`plot.rgam` *Make a plot of rgam model fit*

Description

Produces plots of the estimated functions for specified variables at a given value of lambda.

Usage

```
## S3 method for class 'rgam'
plot(x, newx, index, which = NULL, rugplot = TRUE,
  grid_length = 100, names, ...)
```

Arguments

`x` Fitted `rgam` object.

`newx` Matrix of values of each predictor at which to plot.

`index` Index of lambda value for which plotting is desired. Default is the last lambda value in `x$lambda`.

`which` Which features to plot. Default is the first 4 or `nvars` variables, whichever is smaller.

`rugplot` If `TRUE` (default), adds a rugplot showing the values of x at the bottom of each fitted function plot.

`grid_length` The number of points to evaluate the estimated function at. Default is 100.

`names` Vector of variable names of features in `which`. By default, name of the `j`th variable is `xj`.

`...` Optional graphical parameters to plot.

Details

A plot of the specified fitted functions is produced. Nothing is returned.

Examples

```
set.seed(1)
n <- 100; p <- 12
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 3), rep(0, 9)), ncol = 1)
y <- x %*% beta + x[, 4]^2 + rnorm(n)
fit <- rgam(x, y)

# default: print functions for first 4 variables
opar <- par(mfrow = c(2, 2))
plot(fit, newx = x, index = 20)
par(opar)

# print for variables 5 to 8
opar <- par(mfrow = c(2, 2))
plot(fit, newx = x, index = 20, which = 5:8)
par(opar)
```

---

predict.cv.rgam *Make predictions from a "cv.rgam" object*

Description

This function returns the predictions for a new data matrix from a cross-validated `rgam` model by using the stored "`glmfit`" object and the optimal value chosen for `lambda`.

Usage

```
## S3 method for class 'cv.rgam'
predict(object, xnew, s = c("lambda.1se",
  "lambda.min"), ...)
```

Arguments

`object`	Fitted "`cv.rgam`" object.
`xnew`	Matrix of new values for `x` at which predictions are to be made.
`s`	Value of the penalty parameter `lambda` at which predictions are required. Default is the value `s="lambda.1se"` stored in the CV `fit`. Alternatively, `s="lambda.min"` can be used. If `s` is numeric, it is taken as the value(s) of lambda to be used.
`...`	Other arguments to be passed to `predict.rgam()`).

Details

This function makes it easier to use the results of cross-validation to make a prediction.

Value

Predictions which the cross-validated model makes for `xnew` at the optimal value of `lambda`. Note that the default is the "lambda.1se" for lambda, to make this function consistent with `cv.glmnet` in the `glmnet` package.

The output depends on the `...` argument which is passed on to the predict method for `rgam` objects.

See Also

`cv.rgam` and `predict.rgam`.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)
cvfit <- cv.rgam(x, y)

# predictions at the lambda.1se value
predict(cvfit, xnew = x[1:5, ])

# predictions at the lambda.min value
predict(cvfit, xnew = x[1:5, ], s = "lambda.min")

# predictions at specific lambda value
predict(cvfit, xnew = x[1:5, ], s = 0.1)

# probability predictions for binomial family
bin_y <- ifelse(y > 0, 1, 0)
cvfit2 <- cv.rgam(x, bin_y, family = "binomial")
predict(cvfit2, xnew = x[1:5, ], type = "response", s = "lambda.min")
```

`predict.rgam` *Make predictions from a "rgam" object*

Description

This function returns the predictions from a "`rgam`" object for a new data matrix.

Usage

```
## S3 method for class 'rgam'
predict(object, xnew, ...)
```

Arguments

object	Fitted "rgam" object.
xnew	Matrix of new values for x at which predictions are to be made.
...	Any other arguments to be passed to predict.glmnet().

Value

Predictions of which the model object makes at xnew. The type of predictions depends on whether a type argument is passed. By default it givs the linear predictors for the regression model.

If an offset is used in the fit, then one must be supplied via the newoffset option.

See Also

rgam.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)
fit <- rgam(x, y)

# predict for full lambda path
predict(fit, xnew = x[1:5, ])

# predict for specific lambda values
predict(fit, xnew = x[1:5, ], s = 0.1)

# predictions for binomial family
bin_y <- ifelse(y > 0, 1, 0)
fit2 <- rgam(x, bin_y, family = "binomial")
# linear predictors
predict(fit2, xnew = x[1:5, ], s = 0.05)
# probabilities
predict(fit2, xnew = x[1:5, ], type = "response", s = 0.05)
```

---

print.cv.rgam *Print a cross-validated rgam object*

Description

Print a summary of the results of cross-validation for a RGAM model.

Usage

```
## S3 method for class 'cv.rgam'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	Fitted `rgam` object.
`digits`	Significant digits in printout.
`...`	Additional print arguments.

Details

The call that produced the object x is printed, followed by some information on the performance for `lambda.min` and `lambda.1se`.

See Also

`cv.rgam`, `print.rgam`.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)
cvfit <- cv.rgam(x, y)
print(cvfit)
```

`print.rgam` *Print a rgam object*

Description

Print a summary of the rgam path at each step along the path.

Usage

```
## S3 method for class 'rgam'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	Fitted `rgam` object.
`digits`	Significant digits in printout.
`...`	Additional print arguments.

Details

The call that produced the object x is printed, followed by a five-column matrix with columns NonZero, Lin, NonLin, columns say how many nonzero, linear and nonlinear terms there are. the percent deviance explained (relative to the null deviance).

Value

The matrix above is silently returned.

See Also

rgam.

Examples

```
set.seed(1)
n <- 100; p <- 12
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 3), rep(0, 9)), ncol = 1)
y <- x %*% beta + x[, 4]^2 + rnorm(n)
fit <- rgam(x, y)
print(fit)
```

---

rgam *Fit reluctant generalized additive model*

Description

Fits a reluctant generalized additive model (RGAM) for an entire regularization path indexed by the parameter lambda. Fits linear, logistic, Poisson and Cox regression models. RGAM is a three-step algorithm: Step 1 fits the lasso and computes residuals, Step 2 constructs the non-linear features, and Step 3 fits a lasso of the response on both the linear and non-linear features.

Usage

```
rgam(x, y, lambda = NULL, lambda.min.ratio = ifelse(nrow(x) < ncol(x),
  0.01, 1e-04), standardize = TRUE, family = c("gaussian", "binomial",
  "poisson", "cox"), offset = NULL, init_nz, removeLin = TRUE,
  nfolds = 5, foldid = NULL, df = 4, gamma, tol = 0.01,
  parallel = FALSE, verbose = TRUE)
```

Arguments

x	Input matrix, of dimension nobs x nvars; each row is an observation vector.

`y` Response variable. Quantitative for `family = "gaussian"` or `family = "poisson"` (non-negative counts). For `family="binomial"`, should be a numeric vector consisting of 0s and 1s. For `family="cox"`, `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right-censored.

`lambda` A user-supplied `lambda` sequence. Typical usage is to have the program compute its own `lambda` sequence; supplying a value of lambda overrides this.

`lambda.min.ratio`
Smallest value for lambda as a fraction of the largest lambda value. The default depends on the sample size nobs relative to the number of variables nvars. If nobs > nvars, the default is 0.0001, close to zero. If nobs < nvars, the default is 0.01.

`standardize` If `TRUE` (default), the columns of the input matrix are standardized before the algorithm is run. See details section for more information.

`family` Response type. Either `"gaussian"` (default) for linear regression, `"binomial"` for logistic regression, `"poisson"` for Poisson regression or `"cox"` for Cox regression.

`offset` A vector of length nobs. 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.

`init_nz` A vector specifying which features we must include when computing the non-linear features. Default is to construct non-linear features for all given features.

`removeLin` When constructing the non-linear features, do we remove the linear component from them? Default is `TRUE`.

`nfolds` Number of folds for CV in Step 1 (default is 5). Although `nfolds` can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is `nfolds = 3`.

`foldid` An optional vector of values between 1 and `nfolds` identifying what fold each observation is in. If supplied, `nfolds` can be missing.

`df` Degrees of freedom for the non-linear fit in Step 2. Default is 4.

`gamma` Scale factor for non-linear features (vs. original features), to be between 0 and 1. Default is 0.8 if `init_nz = c()`, 0.6 otherwise.

`tol` Parameter to be passed to `smooth.spline`: a tolerance for same-ness or uniqueness of the `x` values. Default is 0.01. See `smooth.spline` documentation for more details.

`parallel` If TRUE, the `cv.glmnet()` call in Step 1 is parallelized. Must register parallel before hand, such as doMC or others. Default is FALSE.

`verbose` If `TRUE` (default), model-fitting is tracked with a progress bar.

Details

If there are variables which the user definitely wants to compute non-linear versions for in Step 2 of the algorithm, they can be specified as a vector for the `init_nz` option. The algorithm will compute non-linear versions for these features as well as the features suggested by Step 1 of the algorithm.

If `standardize = TRUE`, the standard deviation of the linear and non-linear features would be 1 and gamma respectively. If `standardize = FALSE`, linear features will remain on their original scale while non-linear features would have standard deviation gamma times the mean standard deviation of the linear features.

For `family="gaussian"`, `rgam` standardizes `y` to have unit variance (using `1/n` rather than `1/(n-1)` formula).

Value

An object of class `"rgam"`.

`full_glmfit` The glmnet object resulting from Step 3: fitting a `glmnet` model for the response against the linear & non-linear features.

`spline_fit` List of spline fits for residual against each response. Needed for predicting on new data.

`lin_comp_fit` If `removeLin = TRUE`, a list of coefficients for simple linear regression of non-linear feature on original feature. Needed for predicting on new data.

`init_nz` Column indices for the features which we allow to have non-linear relationship with the response.

`step1_nz` Indices of features which CV in Step 1 chose.

`removeLin` Did we remove the linear components when constructing the non-linear features? Needed for predicting on new data.

`mxf` Means of the features (both linear and non-linear).

`sxf` Scale factors of the features (both linear and non-linear).

`feat` Column indices of the non-zero features for each value of `lambda`.

`linfeat` Column indices of the non-zero linear components for each value of `lambda`.

`nonlinfeat` Column indices of the non-zero non-linear components for each value of `lambda`.

`nzero_feat` The number of non-zero features for each value of `lambda`.

`nzero_lin` The number of non-zero linear components for each value of `lambda`.

`nzero_nonlin` The number of non-zero non-linear components for each value of `lambda`.

`lambda` The actual sequence of `lambda` values used.

`p` The number of features in the original data matrix.

`family` Response type.

`call` The call that produced this object.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)

fit <- rgam(x, y)
```

```
# construct non-linear features for only those selected by Step 1
fit <- rgam(x, y, init_nz = c())

# specify scale factor gamma and degrees of freedom
fit <- rgam(x, y, gamma = 1, df = 6)

# binomial family
bin_y <- ifelse(y > 0, 1, 0)
fit2 <- rgam(x, bin_y, family = "binomial")

# Poisson family
poi_y <- rpois(n, exp(x %*% beta))
fit3 <- rgam(x, poi_y, family = "poisson")
# Poisson with offset
offset <- rnorm(n)
fit3 <- rgam(x, poi_y, family = "poisson", offset = offset)
```

summary.rgam *rgam summary routine*

Description

Makes a two-panel plot of the rgam object showing coefficient paths.

Usage

```
## S3 method for class 'rgam'
summary(object, label = FALSE, index = NULL,
  which = NULL, ...)
```

Arguments

object	Fitted rgam object.
label	If TRUE, annotate the plot with variable labels. Default is FALSE.
index	The indices of the lambda hyperparameter which we want the plot for. The default is to plot for the entire lambda path.
which	Which values to plot. Default is all variables.
...	Additional arguments to summary.

Details

A two panel plot is produced, that summarizes the linear components and the nonlinear components, as a function of lambda. For the linear components, it is the coefficient for each variable. For the nonlinear components, it is the coefficient of the non-linear variable. Nothing is returned.

Examples

```
set.seed(1)
n <- 100; p <- 20
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(c(rep(2, 5), rep(0, 15)), ncol = 1)
y <- x %*% beta + rnorm(n)

fit <- rgam(x, y)
opar <- par(mfrow = c(1, 2))
summary(fit)
par(opar)

# with labels, just variables 1 to 5
opar <- par(mfrow = c(1, 2))
summary(fit, label = TRUE, which = 1:5)
par(opar)

# as above, but just the first 30 values of lambda
opar <- par(mfrow = c(1, 2))
summary(fit, label = TRUE, which = 1:5, index = 1:30)
par(opar)
```

Index