Package 'BranchGLM'

August 29, 2022

Type Package

Title Efficient Branch and Bound Variable Selection for GLMs using 'RcppArmadillo'

Version 1.3.1

Date 2022-8-28

Maintainer Jacob Seedorff <jwseedorff@uiowa.edu>

URL https://github.com/JacobSeedorff21/BranchGLM

BugReports https://github.com/JacobSeedorff21/BranchGLM/issues

Description Performs efficient and scalable glm best subset selection using a novel implementation of a branch and bound algorithm. To speed up the model fitting process, a range of optimization methods are implemented in 'RcppArmadillo'. Parallel computation is available using 'OpenMP'.

License Apache License (>= 2)

Depends R (>= 3.3.0)

Imports Rcpp (>= 1.0.7), methods

LinkingTo Rcpp, RcppArmadillo, BH

RoxygenNote 7.2.0

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Jacob Seedorff [aut, cre]

Repository CRAN

Date/Publication 2022-08-29 07:30:13 UTC

R topics documented:

`BranchGLM` *Fits GLMs*

Description

Fits generalized linear models via RcppArmadillo. Also has the ability to fit the models with parallelization via OpenMP.

Usage

```
BranchGLM(
  formula,
  data,
  family,
  link,
  offset = NULL,
  method = "Fisher",
  grads = 10,
  parallel = FALSE,
  nthreads = 8,
  tol = 1e-06,
  maxit = NULL,
  init = NULL,
  contrasts = NULL,
  keepData = TRUE,
  keepY = TRUE
)
```

```
BranchGLM.fit(
  x,
  y,
  family,
  link,
  offset = NULL,
  method = "Fisher",
  grads = 10,
  parallel = FALSE,
  nthreads = 8,
  init = NULL,
  maxit = NULL,
  tol = 1e-06
)
```

Arguments

formula	a formula for the model.
data	a dataframe that contains the response and predictor variables.
family	distribution used to model the data, one of "gaussian", "gamma", "binomial", or "poisson".
link	link used to link mean structure to linear predictors. One of "identity", "logit", "probit", "cloglog", "sqrt", "inverse", or "log".
offset	offset vector, by default the zero vector is used.
method	one of "Fisher", "BFGS", or "LBFGS". BFGS and L-BFGS are quasi-newton methods which are typically faster than Fisher's scoring when there are many covariates (at least 50).
grads	number of gradients used to approximate inverse information with, only for `method = "LBFGS"`.
parallel	whether or not to make use of parallelization via OpenMP.
nthreads	number of threads used with OpenMP, only used if `parallel = TRUE`.
tol	tolerance used to determine model convergence.
maxit	maximum number of iterations performed. The default for Fisher's scoring is 50 and for the other methods the default is 200.
init	initial values for the betas, if not specified then they are automatically selected.
contrasts	see `contrasts.arg` of `model.matrix.default`.
keepData	Whether or not to store a copy of data and design matrix, the default is TRUE. If this is FALSE, then the results from this cannot be used inside of `VariableSelection`.
keepY	Whether or not to store a copy of y, the default is TRUE. If this is FALSE, then the binomial GLM helper functions may not work and this cannot be used inside of `VariableSelection`.
x	design matrix used for the fit, must be numeric.
y	outcome vector, must be numeric.

Details

Can use BFGS, L-BFGS, or Fisher's scoring to fit the GLM. BFGS and L-BFGS are typically faster than Fisher's scoring when there are at least 50 covariates and Fisher's scoring is typically best when there are fewer than 50 covariates. This function does not currently support the use of weights. In the special case of gaussian regression with identity link the `method` argument is ignored and the normal equations are solved directly.

The models are fit in C++ by using Rcpp and RcppArmadillo. In order to help convergence, each of the methods makes use of a backtracking line-search using the strong Wolfe conditions to find an adequate step size. There are also two conditions used to control convergence, the first is whether there is a sufficient decrease in the negative log-likelihood, and the other is whether the norm of the score is sufficiently small. The `tol` argument controls both of these criteria. If the algorithm fails to converge, then `iterations` will be -1.

All observations with any missing values are removed before model fitting.

The dispersion parameter for gamma regression is estimated via maximum likelihood, very similar to the `gamma.dispersion` function from the MASS package.

`BranchGLM.fit` can be faster than calling `BranchGLM` if the x matrix and y vector are already available, but doesn't return as much information. The object returned by `BranchGLM.fit` is not of class `BranchGLM`, so all of the methods for `BranchGLM` objects such as `predict` or `VariableSelection` cannot be used.

Value

`BranchGLM` returns a `BranchGLM` object which is a list with the following components

`coefficients`	a matrix with the coefficients estimates, SEs, wald test statistics, and p-values
`iterations`	number of iterations it took the algorithm to converge, if the algorithm failed to converge then this is -1
`dispersion`	the value of the dispersion parameter
`logLik`	the log-likelihood of the fitted model
`resdev`	the residual deviance of the fitted model
`AIC`	the AIC of the fitted model
`preds`	predictions from the fitted model
`linpreds`	linear predictors from the fitted model
`formula`	formula used to fit the model
`method`	iterative method used to fit the model
`y`	y vector used in the model, not included if `keepY = FALSE`
`x`	design matrix used to fit the model, not included if `keepData = FALSE`
`offset`	offset vector in the model, not included if `keepData = FALSE`
`data`	original dataframe supplied to the function, not included if `keepData = FALSE`
`numobs`	number of observations in the design matrix
`names`	names of the variables
`yname`	name of y variable

parallel	whether parallelization was employed to speed up model fitting process
missing	number of missing values removed from the original dataset
link	link function used to model the data
family	family used to model the data
ylevel	the levels of y, only included for binomial glms
xlev	the levels of the factors in the dataset
terms	the terms object used

BranchGLM.fit returns a list with the following components

coefficients	a matrix with the coefficients estimates, SEs, wald test statistics, and p-values
iterations	number of iterations it took the algorithm to converge, if the algorithm failed to converge then this is -1
dispersion	the value of the dispersion parameter
logLik	the log-likelihood of the fitted model
resdev	the residual deviance of the fitted model
AIC	the AIC of the fitted model
preds	predictions from the fitted model
linpreds	linear predictors from the fitted model

Examples

```
Data <- iris
### Using BranchGLM
BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")
### Using BranchGLM.fit
x <- model.matrix(Sepal.Length ~ ., data = Data)
y <- Data$Sepal.Length
BranchGLM.fit(x, y, family = "gaussian", link = "identity")
```

Cindex *Cindex/AUC*

Description

Calculates the c-index/AUC.

Usage

```
Cindex(object, ...)

AUC(object, ...)

## S3 method for class 'numeric'
Cindex(object, y, ...)

## S3 method for class 'BranchGLM'
Cindex(object, ...)

## S3 method for class 'BranchGLMROC'
Cindex(object, ...)
```

Arguments

object	a BranchGLM object, a BranchGLMROC object, or a numeric vector.
...	further arguments passed to other methods.
y	Observed values, can be a numeric vector of 0s and 1s, a two-level factor vector, or a logical vector.

Details

Uses trapezoidal rule to calculate AUC when given a BranchGLMROC object and uses Mann-Whitney U to calculate it otherwise. The trapezoidal rule method is less accurate, so the two methods may give different results.

Value

A number corresponding to the c-index/AUC.

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
Cindex(Fit)
AUC(Fit)
```

---

coef.BranchGLM *Extract Coefficients*

Description

Extract Coefficients

Usage

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

Arguments

object	a BranchGLM object.
...	further arguments passed to or from other methods.

Value

A named vector with the corresponding coefficient estimates.

---

logLik.BranchGLM *Extract Log-Likelihood*

Description

Extract Log-Likelihood

Usage

```
## S3 method for class 'BranchGLM'
logLik(object, ...)
```

Arguments

object	a BranchGLM object.
...	further arguments passed to or from other methods.

Value

An object of class logLik which is a number corresponding to the log-likelihood with the following attributes: "df" (degrees of freedom) and "nobs" (number of observations).

`MultipleROCCurves` *Plotting Multiple ROC Curves*

Description

Plotting Multiple ROC Curves

Usage

```
MultipleROCCurves(
  ...,
  legendpos = "bottomright",
  title = "ROC Curves",
  colors = NULL,
  names = NULL,
  lty = 1,
  lwd = 1
)
```

Arguments

`...`	any number of `BranchGLMROC` objects.
`legendpos`	a keyword to describe where to place the legend, such as "bottomright". The default is "bottomright"
`title`	title for the plot.
`colors`	vector of colors to be used on the ROC curves.
`names`	vector of names used to create a legend for the ROC curves.
`lty`	vector of linetypes used to create the ROC curves or a single linetype to be used for all ROC curves.
`lwd`	vector of linewidths used to create the ROC curves or a single linewidth to be used for all ROC curves.

Value

No return value, called to create the plot.

Examples

```
Data <- ToothGrowth

### Logistic ROC
LogisticFit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
LogisticROC <- ROC(LogisticFit)

### Probit ROC
ProbitFit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "probit")
```

```
ProbitROC <- ROC(ProbitFit)

### Cloglog ROC
CloglogFit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "cloglog")
CloglogROC <- ROC(CloglogFit)

### Plotting ROC curves

MultipleROCCurves(LogisticROC, ProbitROC, CloglogROC,
                  names = c("Logistic ROC", "Probit ROC", "Cloglog ROC"))
```

`plot.BranchGLMROC` *Plotting ROC Curve*

Description

This plots a ROC curve.

Usage

```
## S3 method for class 'BranchGLMROC'
plot(x, ...)
```

Arguments

`x`	a `BranchGLMROC` object.
`...`	arguments passed to generic plot function.

Value

No return value, called to create the plot.

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
MyROC <- ROC(Fit)
plot(MyROC)
```

predict.BranchGLM *Predict Method for BranchGLM Objects*

Description

Gets predictions from a BranchGLM object.

Usage

```
## S3 method for class 'BranchGLM'
predict(object, newdata = NULL, type = "response", ...)
```

Arguments

object	a BranchGLM object.
newdata	a dataframe, if not specified the data the model was fit on is used.
type	one of "linpreds" or "response", if not specified "response" is used.
...	further arguments passed to or from other methods.

Details

linpreds corresponds to the linear predictors and response is on the scale of the response variable. Offset variables are ignored for predictions on new data.

Value

A numeric vector of predictions.

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")
predict(Fit)
### Example with new data
predict(Fit, newdata = iris[1:20,])
```

predict.BranchGLMVS *Predict Method for BranchGLMVS Objects*

Description

Gets predictions from the best model found in the BranchGLMVS object.

Usage

```
## S3 method for class 'BranchGLMVS'
predict(object, newdata = NULL, type = "response", ...)
```

Arguments

object	a BranchGLMVS object.
newdata	a dataframe, if not specified the data the model was fit on is used.
type	one of "linpreds" or "response", if not specified "response" is used.
...	further arguments passed to predict.BranchGLM.

Details

linpreds corresponds to the linear predictors and response is on the scale of the response variable. Offset variables are ignored for predictions on new data.

Value

A numeric vector of predictions.

Examples

```
Data <- iris
VS <- VariableSelection(Sepal.Length ~ ., data = Data, family = "gaussian",
                      link = "identity", type = "branch and bound", showprogress = FALSE)
predict(VS)
### Example with new data
predict(VS, newdata = iris[1:20,])
```

print.BranchGLM *Print Method for BranchGLM*

Description

Print Method for BranchGLM

Usage

```
## S3 method for class 'BranchGLM'
print(x, coefdigits = 4, digits = 0, ...)
```

Arguments

x	a BranchGLM object.
coefdigits	number of digits to display for coefficients table.
digits	number of digits to display for information after table.
...	further arguments passed to or from other methods.

Value

The supplied BranchGLM object.

print.BranchGLMROC *Print Method for BranchGLMROC*

Description

Print Method for BranchGLMROC

Usage

```
## S3 method for class 'BranchGLMROC'
print(x, ...)
```

Arguments

x	a BranchGLMROC object.
...	further arguments passed to other methods.

Value

The supplied BranchGLMROC object.

print.BranchGLMTable *Print Method for BranchGLMTable*

Description

Print Method for BranchGLMTable

Usage

```
## S3 method for class 'BranchGLMTable'
print(x, digits = 4, ...)
```

Arguments

x	a BranchGLMTable object.
digits	number of digits to display.
...	further arguments passed to other methods.

Value

The supplied BranchGLMTable object.

print.BranchGLMVS *Print Method for BranchGLMVS*

Description

Print Method for BranchGLMVS

Usage

```
## S3 method for class 'BranchGLMVS'
print(x, coefdigits = 4, digits = 0, ...)
```

Arguments

x	a BranchGLMVS object.
coefdigits	number of digits to display for coefficients table.
digits	number of digits to display for information not in the table.
...	further arguments passed to other methods.

Value

The supplied BranchGLMVS object.

ROC *ROC Curve*

Description

Creates an ROC curve.

Usage

```
ROC(object, ...)

## S3 method for class 'numeric'
ROC(object, y, ...)

## S3 method for class 'BranchGLM'
ROC(object, ...)
```

Arguments

object	a BranchGLM object or a numeric vector.
...	further arguments passed to other methods.
y	observed values, can be a numeric vector of 0s and 1s, a two-level factor vector, or a logical vector.

Value

A BranchGLMROC object which can be plotted with plot(). The AUC can also be calculated using AUC().

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
MyROC <- ROC(Fit)
plot(MyROC)
```

Table *Confusion Matrix*

Description

Creates confusion matrix and calculates related measures.

Usage

```
Table(object, ...)

## S3 method for class 'numeric'
Table(object, y, cutoff = 0.5, ...)

## S3 method for class 'BranchGLM'
Table(object, cutoff = 0.5, ...)
```

Arguments

object	a BranchGLM object or a numeric vector.
...	further arguments passed to other methods.
y	observed values, can be a numeric vector of 0s and 1s, a two-level factor vector, or a logical vector.
cutoff	cutoff for predicted values, the default is 0.5.

Value

A BranchGLMTable object which is a list with the following components

table	a matrix corresponding to the confusion matrix
accuracy	a number corresponding to the accuracy
sensitivity	a number corresponding to the sensitivity
specificity	a number corresponding to the specificity
PPV	a number corresponding to the positive predictive value
levels	a vector corresponding to the levels of the response variable

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
Table(Fit)
```

VariableSelection *Variable Selection for GLMs*

Description

Performs forward selection, backward elimination, and branch and bound variable selection for generalized linear models.

Usage

```
VariableSelection(object, ...)

## S3 method for class 'formula'
VariableSelection(
  object,
  data,
  family,
  link,
  offset = NULL,
  method = "Fisher",
  type = "forward",
  metric = "AIC",
  keep = NULL,
  maxsize = NULL,
  grads = 10,
  parallel = FALSE,
  nthreads = 8,
  tol = 1e-06,
  maxit = NULL,
  contrasts = NULL,
  showprogress = TRUE,
  ...
)

## S3 method for class 'BranchGLM'
VariableSelection(
  object,
  type = "forward",
  metric = "AIC",
  keep = NULL,
  maxsize = NULL,
  method = "Fisher",
  grads = 10,
  parallel = FALSE,
  nthreads = 8,
  tol = 1e-06,
  maxit = NULL,
  showprogress = TRUE,
  ...
)
```

Arguments

object	a formula or a BranchGLM object.
...	further arguments passed to other methods.
data	a dataframe with the response and predictor variables.

family	distribution used to model the data, one of "gaussian", "gamma", "binomial", or "poisson".
link	link used to link mean structure to linear predictors. One of "identity", "logit", "probit", "cloglog", "sqrt", "inverse", or "log".
offset	offset vector, by default the zero vector is used.
method	one of "Fisher", "BFGS", or "LBFGS". Fisher's scoring is recommended for forward selection and branch and bound selection since they will typically fit many models with a small number of covariates.
type	one of "forward", "backward", "branch and bound", "backward branch and bound", or "switch branch and bound" to indicate which type of variable selection to perform.
metric	metric used to choose model, the default is "AIC", but "BIC" is also available.
keep	vector of names to denote variables that must be in the model.
maxsize	maximum number of variables to consider in a single model, the default is the total number of variables. This number adds onto any variables specified in keep. This argument only works for `method = "forward"` and `method = "branch and bound"`.
grads	number of gradients used to approximate inverse information with, only for `method = "LBFGS"`.
parallel	one of TRUE or FALSE to indicate if parallelization should be used
nthreads	number of threads used with OpenMP, only used if `parallel = TRUE`.
tol	tolerance used to determine model convergence.
maxit	maximum number of iterations performed. The default for Fisher's scoring is 50 and for the other methods the default is 200.
contrasts	see `contrasts.arg` of `model.matrix.default`.
showprogress	whether to show progress updates for branch and bound.

Details

The model in the formula or the formula from the fitted model is treated as the upper model. The variables specified in keep along with an intercept (if included in formula) is the lower model. When an intercept is included in the model formula it is kept in each model. Factor variables are either kept in their entirety or entirely removed.

The branch and bound method makes use of an efficient branch and bound algorithm to find the optimal model. This is will find the best model according to the metric and can be much faster than an exhaustive search and can be made even faster with parallel computation. The backward branch and bound method is very similar to the branch and bound method, except it tends to be faster when the best model contains most of the variables. The switch branch and bound method is a combination of the two methods and is typically very fast.

Fisher's scoring is recommended for branch and bound selection and forward selection. L-BFGS may be faster for backward elimination, especially when there are many variables.

All observations that have any missing values in the upper model are removed.

Value

A BranchGLMVS object which is a list with the following components

finalmodel	the final BranchGLM model selected
variables	a vector corresponding to the selected variables
numchecked	number of models fit
order	the order the variables were added to the model or removed from the model, this is not included for branch and bound selection
type	type of variable selection employed
keep	character vector of variables kept in each model, NULL if none specified
metric	metric used to select model
bestmetric	the best metric found in the search

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

### Doing branch and bound selection
VariableSelection(Fit, type = "branch and bound", metric = "BIC")

### Now doing it in parallel (although it isn't necessary for this dataset)
VariableSelection(Fit, type = "branch and bound", parallel = TRUE, metric = "BIC")

### Using a formula
VariableSelection(Sepal.Length ~ ., data = Data, family = "gaussian",
link = "identity", metric = "BIC", type = "branch and bound")

### Using the keep argument
VariableSelection(Fit, type = "branch and bound", keep = "Petal.Width", metric = "BIC")
```

Index