Package 'sirus'

January 20, 2020

Type Package

Title Stable and Interpretable RUle Set

Version 0.2.1

Date 2020-01-20

Author Clement Benard [aut, cre], Marvin N. Wright [ctb, cph]

Maintainer Clement Benard <clement.benard@safrangroup.com>

Description A regression and classification algorithm based on random forests, which takes the form of a short list of rules. SIRUS combines the simplicity of decision trees with the predictivity of random forests for problems with low order interactions. The core aggregation principle of random forests is kept, but instead of aggregating predictions, SIRUS selects the most frequent nodes of the forest to form a stable rule ensemble model. The algorithm is fully described in the following article: Benard C., Biau G., da Veiga S., Scornet E. (2019) <arXiv:1908.06852>. This R package is a fork from the project ranger (<https://github.com/imbs-hl/ranger>).

License GPL-3

Imports Rcpp (>= 0.11.2), Matrix, ROCR, ggplot2, glmnet

LinkingTo Rcpp

Depends R (>= 3.1)

Suggests survival, testthat

RoxygenNote 6.1.1

URL https://gitlab.com/drti/sirus

BugReports https://gitlab.com/drti/sirus/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-01-20 17:30:03 UTC

R topics documented:

`sirus.cv` *Estimation of p0.*

Description

Estimation by cross-validation of the hyperparameter p0 used to select rules in sirus.fit. For a robust estimation, it is recommanded to run multiple cross-validations.

Usage

```
sirus.cv(data, y, type = "auto", nfold = 10, ncv = NULL,
  num.rule.max = 25, q = 10, num.trees.step = 1000, alpha = 0.05,
  mtry = NULL, max.depth = 2, num.trees = NULL, num.threads = NULL,
  replace = TRUE, sample.fraction = NULL, verbose = TRUE,
  seed = NULL)
```

Arguments

`data`	Input dataframe, each row is an observation vector.
`y`	Numeric response variable. For classification, y takes only 0 and 1 values.
`type`	'reg' for regression, 'classif' for classification and 'auto' for automatic detection (classification if y takes only 0 and 1 values).
`nfold`	Number of folds in the cross-validation. Default is 10.
`ncv`	Number of repetitions of the cross-validation. Default is 10 for regression and 30 for classification.
`num.rule.max`	Maximum number of rules of SIRUS model in the cross-validation grid. Default is 25.
`q`	Number of quantiles used for node splitting in the forest construction. Default is 10.
`num.trees.step`	Number of trees grown between two evaluations of the stopping criterion. Ignored if num.trees is provided.
`alpha`	Parameter of the stopping criterion for the number of trees: stability has to reach 1 - alpha to stop the growing of the forest. Ignored if num.trees is provided.
`mtry`	Number of variables to possibly split at each node. Default is the number of variables divided by 3.

max.depth	Maximal tree depth. Default and strongly recommanded value is 2.
num.trees	Number of trees grown in the forest. If NULL (recommanded), the number of trees is automatically set using a stability stopping criterion.
num.threads	Number of threads used to grow the forest. Default is number of CPUs available.
replace	Boolean. If true (default), sample with replacement.
sample.fraction	Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement.
verbose	Boolean. If true, information messages are printed.
seed	Random seed. Default is NULL, which generates the seed from R. Set to 0 to ignore the R seed.

Value

Optimal value of p0 with the elements

p0.pred	Optimal p0 value to maximize model accuracy.
p0.stab	Optimal p0 value for a tradeoff between stability and accuracy.
error.grid.p0	Table with the full cross-validation results for a fine grid of p0: number of rules, stability, error.
type	'reg' for regression, 'classif' for classification.

Examples

```
## load sirus
require(sirus)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## run cv
cv.grid <- sirus.cv(data, y, nfold = 3, ncv = 2, num.trees = 100)
```

---

sirus.fit *Fit SIRUS.*

Description

Fit a SIRUS model for a given number of rules (10 by default) or a given p0. If the output y takes only 0 and 1 values a classification model is fit, otherwise a regression model is fit. The number of trees is tuned automatically with a stopping criterion based on stability. The hyperparameter p0 can be tuned using sirus.cv.

Usage

```
sirus.fit(data, y, type = "auto", num.rule = 10, p0 = NULL,
  num.rule.max = 25, q = 10, num.trees.step = 1000, alpha = 0.05,
  mtry = NULL, max.depth = 2, num.trees = NULL, num.threads = NULL,
  replace = TRUE, sample.fraction = ifelse(replace, 1, 0.632),
  verbose = TRUE, seed = NULL)
```

Arguments

data	Input dataframe, each row is an observation vector.
y	Numeric response variable. For classification, y takes only 0 and 1 values.
type	'reg' for regression, 'classif' for classification and 'auto' for automatic detection (classification if y takes only 0 and 1 values).
num.rule	Number of rules in SIRUS model. Default is 10. Ignored if a p0 value is provided. For regression, the effective number of rules can be smaller than num.rule because of the additional selection in the final linear aggregation of the rules.
p0	Selection threshold on the frequency of appearance of a path in the forest. Default is NULL and num.rule is used to select rules.
num.rule.max	Maximum number of rules in SIRUS model. Ignored if num.rule is provided.
q	Number of quantiles used for node splitting in the forest construction.
num.trees.step	Number of trees grown between two evaluations of the stopping criterion. Ignored if num.trees is provided.
alpha	Parameter of the stopping criterion for the number of trees: stability has to reach 1 - alpha to stop the growing of the forest. Ignored if num.trees is provided.
mtry	Number of variables to possibly split at each node. Default is the number of variables divided by 3.
max.depth	Maximal tree depth. Default and strongly recommanded value is 2.
num.trees	Number of trees grown in the forest. Default is NULL. If NULL (recommanded), the number of trees is automatically set using a stability based stopping criterion.
num.threads	Number of threads used to grow the forest. Default is number of CPUs available.
replace	Boolean. If true (default), sample with replacement.
sample.fraction	Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement.
verbose	Boolean. If true, information messages are printed.
seed	Random seed. Default is NULL, which generates the seed from R. Set to 0 to ignore the R seed.

Value

SIRUS model with elements

rules	List of rules in SIRUS model.

rules.out	List of rule outputs. rule.out: the output mean whether the rule is satisfied or not. supp.size: the number of points inside and outside the rule.
proba	Frequency of occurence of paths in the forest.
paths	List of selected paths.
mean	Mean output over the full training data. Default model output if no rule is selected.

Examples

```
## load sirus
require(sirus)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## fit sirus
sirus.m <- sirus.fit(data, y)
```

---

sirus.plot.cv *Plot SIRUS cross-validation.*

Description

Plot SIRUS cross-validation path: error and stability versus the number of rules when p0 varies.

Usage

```
sirus.plot.cv(sirus.cv.grid, p0.criterion = NULL, num.rule.max = 25)
```

Arguments

sirus.cv.grid	Cross-validation results returned by sirus.cv.
p0.criterion	Criterion to pick optimal p0 to display on plots: 'pred' to maximize accuracy and 'stab' for a tradeoff stability/accuracy. Default is 'stab' for regression and 'pred' for classification.
num.rule.max	Upper limit on the number of rules for the x-axis. Default is 25.

Value

Plots of cross-validation results.

error	plot of error vs number of rules (ggplot object).
stab	plot of stability vs number of rules (ggplot object).

Examples

```
## load sirus
require(sirus)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## run cv
cv.grid <- sirus.cv(data, y, nfold = 3, ncv = 2, num.trees = 100)

## plot cv result
plot.error <- sirus.plot.cv(cv.grid)$error
plot(plot.error)
```

`sirus.predict` *Predict*

Description

Predictions of a SIRUS model for new observations.

Usage

```
sirus.predict(sirus.m, data.test)
```

Arguments

`sirus.m`	A SIRUS model generated by sirus.fit.
`data.test`	Testing data (dataframe of new observations).

Value

Predictions. For classification, vector of the predicted probability of each new observation to be of class 1.

Examples

```
## load sirus
require(sirus)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL
```

```
#' ## fit sirus
sirus.m <- sirus.fit(data, y)

## predict
predictions <- sirus.predict(sirus.m, data)
```

`sirus.print` *Print SIRUS*

Description

Print the list of rules output by SIRUS.

Usage

```
sirus.print(sirus.m)
```

Arguments

`sirus.m`	A SIRUS model generated by sirus.fit.

Value

Formatted list of rules.

Examples

```
## load sirus
require(sirus)

## prepare data
data <- iris
y <- rep(0, nrow(data))
y[data$Species == 'setosa'] = 1
data$Species <- NULL

## fit sirus
sirus.m <- sirus.fit(data, y)

## print sirus model
sirus.print(sirus.m)
```

Index