Package 'sirus'

October 14, 2022

Type Package

Title Stable and Interpretable RUle Set

Version 0.3.3

Date 2022-06-10

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

Maintainer Clement Benard <`clement.benard5@gmail.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 a predictivity close to random forests. The core aggregation principle of random forests is kept, but instead of aggregating predictions, SIRUS aggregates the forest structure: the most frequent nodes of the forest are selected to form a stable rule ensemble model. The algorithm is fully described in the following articles: Benard C., Biau G., da Veiga S., Scornet E. (2021), Electron. J. Statist., 15:427-505 <DOI:10.1214/20-EJS1792> for classification, and Benard C., Biau G., da Veiga S., Scornet E. (2021), AISTATS, PMLR 130:937-945 <`http://proceedings.mlr.press/v130/benard21a`>, for regression. 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, RcppEigen

Depends R (>= 3.6)

Suggests survival, testthat, ranger

RoxygenNote 7.2.0

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

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-06-13 09:20:02 UTC

R topics documented:

`sirus.cv` *Estimate p0.*

Description

Estimate the optimal hyperparameter `p0` used to select rules in `sirus.fit` using cross-validation (Benard et al. 2021a, 2021b).

Usage

```
sirus.cv(
  data,
  y,
  type = "auto",
  nfold = 10,
  ncv = 10,
  num.rule.max = 25,
  q = 10,
  discrete.limit = 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. Each column is an input variable and is numeric or factor.
`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 a robust estimation of `p0`.
`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 and recommended value is 10.
`discrete.limit`	Maximum number of distinct values for a variable to be considered discrete. If higher, variable is continuous.
`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. Default value is 0.05.
`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 recommended value is 2.
`num.trees`	Number of trees grown in the forest. If NULL (recommended), 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.

Details

For a robust estimation of `p0`, it is recommended to run multiple cross-validations (typically `ncv` = 10). Two optimal values of `p0` are provided: `p0.pred` (Benard et al. 2021a) and `p0.stab` (Benard et al. 2021b), defined such that `p0.pred` minimizes the error, and `p0.stab` finds a tradeoff between error and stability. Error is 1-AUC for classification and the unexplained variance for regression. Stability is the average proportion of rules shared by two SIRUS models fit on two distinct folds of the cross-validation.

Value

Optimal value of `p0` with the elements

`p0.pred`	Optimal `p0` value to minimize model error (recommended for classification).
`p0.stab`	Optimal `p0` value for a tradeoff between error and stability (recommended for regression).

`error.grid.p0`	Table with the full cross-validation results for a fine grid of `p0`: number of rules, stability, and error. The last three columns of the table are the standard deviations of the metrics across the `ncv` repetitions of the cross-validation. See details for the definitions of the error and stability metrics.
`type`	'reg' for regression, 'classif' for classification.

References

- Benard, C., Biau, G., Da Veiga, S. & Scornet, E. (2021a). SIRUS: Stable and Interpretable RUle Set for Classification. Electronic Journal of Statistics, 15:427-505. doi:10.1214/20-EJS1792.
- Benard, C., Biau, G., Da Veiga, S. & Scornet, E. (2021b). Interpretable Random Forests via Rule Extraction. Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, PMLR 130:937-945. `http://proceedings.mlr.press/v130/benard21a`.

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 SIRUS for a given number of rules (10 by default) or a given `p0`.
SIRUS is a regression and classification algorithm, based on random forests (Breiman, 2001), that takes the form of a short list of rules. SIRUS combines the simplicity of rule algorithms or decision trees with an accuracy close to random forests. More importantly, the rule selection is stable with respect to data perturbation. SIRUS for classification is defined in (Benard et al. 2021a), and the extension to regression is provided in (Benard et al. 2021b).

Usage

```
sirus.fit(
  data,
  y,
  type = "auto",
```

```
  num.rule = 10,
  p0 = NULL,
  num.rule.max = 25,
  q = 10,
  discrete.limit = 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. Each column is an input variable and is numeric or factor.
`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 null coefficients in the final linear aggregation of the rules.
`p0`	Selection threshold on the frequency of appearance of a path in the forest to set the number of rules. Default is NULL and `num.rule` is used to select rules. `sirus.cv` provides the optimal `p0` by cross-validation.
`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. Default and recommended value is 10.
`discrete.limit`	Maximum number of distinct values for a variable to be considered discrete. If higher, variable is continuous.
`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. Default value is 0.05.
`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 recommended value is 2.
`num.trees`	Number of trees grown in the forest. Default is NULL. If NULL (recommended), 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.

Details

If the output `y` takes only 0 and 1 values, a classification model is fit, otherwise a regression model is fit. SIRUS algorithm proceeds the following steps:

1. Discretize data
2. Fit a random forest
3. Extract rules from tree nodes
4. Select the most frequent rules (which occur in at least a fraction p0 of the trees)
5. Filter rules to remove linear dependence between them
6. Aggregate the selected rules
    - Classification: rules are averaged
    - Regression: rules are linearly combined via a ridge regression (constrained to have all coefficients positive)

The hyperparameter `p0` can be tuned using `sirus.cv` to set the optimal number of rules.
The number of trees is automatically set with a stopping criterion based on stability: the forest growing is stopped when the number of trees is high enough to ensure that 95% of the rules in average are identical over two runs of SIRUS on the provided dataset.
Data is discretized depending on variable types: numerical variables are binned using `q`-quantiles, categorical variables are transformed in ordered variables as in `ranger` (standard method to handle categorical variables in trees), while discrete variables (numerical variables with less than `discrete.limit` distinct values) are left untouched. Notice that categorical variables with a high number of categories should be discarded or transformed, as SIRUS is likely to identify associated irrelevant rules.

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 (symbolic representation with quantile order for continuous variables).

`rule.weights`	Vector of positive or null coefficients assigned to each rule for the linear aggregation (1/number of rules for classification).
`rule.glm`	Fitted glmnet object for regression (linear rule aggregation with ridge penalty).
`type`	Type of SIRUS model: 'reg' for regression, 'classif' for classification.
`num.trees`	Number of trees used to build SIRUS.
`data.names`	Names of input variables.
`mean`	Mean output over the full training data. Default model output if no rule is selected.
`bins`	List of type and possible split values for all input variables.

References

- Benard, C., Biau, G., Da Veiga, S. & Scornet, E. (2021a). SIRUS: Stable and Interpretable RUle Set for Classification. Electronic Journal of Statistics, 15:427-505. doi:10.1214/20-EJS1792.
- Benard, C., Biau, G., Da Veiga, S. & Scornet, E. (2021b). Interpretable Random Forests via Rule Extraction. Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, PMLR 130:937-945. `http://proceedings.mlr.press/v130/benard21a`.
- Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.
- Wright, M. N. & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. J Stat Softw 77:1-17. doi:10.18637/jss.v077.i01.

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 path.*

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 the optimal `p0` displayed in the plots: if 'pred' then `p0.pred` is used for a minimal error, if 'stab' then `p0.stab` is used for a tradeoff error/stability. Default is 'pred' for classification and 'stab' for regression.
`num.rule.max`	Upper limit on the number of rules for the x-axis. Default is 25.

Details

Error is 1-AUC for classification and the unexplained variance for regression. Stability is the average proportion of rules shared by two SIRUS models fit on two distinct folds of the cross-validation.

Value

Plots of cross-validation results.

`error`	plot of error vs number of rules (ggplot2 object).
`stability`	plot of stability vs number of rules (ggplot2 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

Compute SIRUS predictions 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, digits = 3)
```

Arguments

`sirus.m`	A SIRUS model generated by `sirus.fit`.
`digits`	Number of significant digits for numerical values. Default value is 3.

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