Package: sbrl (via r-universe)

October 29, 2026

Type Package

Title Scalable Bayesian Rule Lists Model

Version 1.4

Date 2024-04-07

Author Hongyu Yang [aut, cre], Morris Chen [ctb], Cynthia Rudin [aut, ctb], Margo Seltzer [aut, ctb], The President and Fellows of Harvard College [cph]

Maintainer Hongyu Yang <edwardyhy1@gmail.com>

Description An efficient implementation of Scalable Bayesian Rule Lists Algorithm, a competitor algorithm for decision tree algorithms; see Hongyu Yang, Cynthia Rudin, Margo Seltzer (2017) <https://proceedings.mlr.press/v70/yang17h.html>. It builds from pre-mined association rules and have a logical structure identical to a decision list or one-sided decision tree. Fully optimized over rule lists, this algorithm strikes practical balance between accuracy, interpretability, and computational speed.

License GPL (>= 2)

Imports Rcpp (>= 0.12.4), arules, methods

RcppModules sbrl

LinkingTo Rcpp

NeedsCompilation yes

LazyData yes

SystemRequirements gmp (>= 4.2.0), gsl

Config/pak/sysreqs libgmp3-dev libgsl0-dev

Repository https://cran.r-universe.dev

Date/Publication 2024-04-09 02:33:00 UTC

RemoteUrl https://github.com/cran/sbrl

RemoteRef HEAD

RemoteSha 76a0e9f5ee7576667ed2813ed648d7bc413df729

Contents

sbrl-package 2
get_data_feature_mat 3
predict.sbrl 4
print.sbrl 5
sbrl 6
tictactoe 8

Index **10**

`sbrl-package` *SCALABLE BAYESIAN RULE LISTS*

Description

Fit a `sbrl` model. Learn from the data and create a decision rule list in the format of:

if (condition1) then positive probability = ...

else if (condition2) then positive probability = ...

else if (condition3) ...

...

else (default rule) then positive probability = ...

(See the examples below)

Details

This package contains three functions: `sbrl`, `print.sbrl`, `show.sbrl`, and `predict.sbrl`

Author(s)

Hongyu Yang, Cynthia Rudin, Margo Seltzer

References

Hongyu Yang, Morris Chen, Cynthia Rudin, Margo Seltzer (2016) *Scalable Bayesian Rule Lists.* Working paper on arXiv 2016.

Benjamin Letham, Cynthia Rudin, Tyler McCormick and David Madigan (2015) *Building Interpretable Classifiers with Rules using Bayesian Analysis.* Annals of Applied Statistics, 2015.

See Also

`sbrl`, `print.sbrl`, `show.sbrl` and `predict.sbrl`

Examples

```
# Let us use the titactoe dataset
data(tictactoe)
for (name in names(tictactoe)) {tictactoe[name] <- as.factor(tictactoe[,name])}

# Train on two-thirds of the data
b = round(2*nrow(tictactoe)/3, digit=0)
data_train <- tictactoe[1:b, ]
# Test on the remaining one third of the data
data_test <- tictactoe[(b+1):nrow(tictactoe), ]
# data_train, data_test are dataframes with factor columns
# The class column is "label"

# Run the sbrl algorithm on the training set
  sbrl_model <- sbrl(data_train, iters=20000, pos_sign="1",
   neg_sign="0", rule_minlen=1, rule_maxlen=3,
   minsupport_pos=0.10, minsupport_neg=0.10,
   lambda=10.0, eta=1.0, nchain=25)
  print(sbrl_model)

# Make predictions on the test set
  yhat <- predict(sbrl_model, data_test)
# yhat will be a list of predicted negative and positive probabilities for the test data.

#clean up
rm(list = ls())
gc()
```

---

get_data_feature_mat *GET BINARY MATRIX REPRESENTATION OF THE DATA-FEATURE RELAITONSHIP*

Description

Given some features in the form "feature1=x1", "feature2=x2"..., this function will generate a matrix representation of which data are captured by which features.

Usage

```
get_data_feature_mat(data, featurenames)
```

Arguments

`data`	a data.frame representing the observations.
`featurenames`	a character vector representing the features in the form: "feature1=x1", "feature2=x2"...

Value

a binary matrix of size #observations-by-#featurenames

Author(s)

Hongyu Yang, Morris Chen, Cynthia Rudin, Margo Seltzer

Examples

```
data(tictactoe)
featurenames <- c("c1=b", "c1=o", "c1=x")
get_data_feature_mat(tictactoe, featurenames)
#it will generate a binary matrix representing which observations are captured by which features.

#clean up
rm(list = ls())
gc()
```

---

predict.sbrl *PREDICT THE POSITIVE PROBABILITY FOR THE OBSERVATIONS*

Description

Returns a list of probabilities.

Usage

```
## S3 method for class 'sbrl'
predict(object, tdata, ...)
```

Arguments

object	sbrl model returned from the sbrl function.
tdata	test data
...	further arguments passed to or from other methods.

Value

return a list containing 2 lists of probablities for the rule list, corresponding to probability being 0 and 1 for each observation. The two probabilities for each rule add up to 1, P(y=0 | rule r) + p(y=1 | rule r) = 1

Examples

```
# Let us use the titactoe dataset
data(tictactoe)
for (name in names(tictactoe)) {tictactoe[name] <- as.factor(tictactoe[,name])}

# Train on two-thirds of the data
b = round(2*nrow(tictactoe)/3, digit=0)
data_train <- tictactoe[1:b, ]
```

```
# Test on the remaining one third of the data
data_test <- tictactoe[(b+1):nrow(tictactoe), ]
# data_train, data_test are dataframes with factor columns
# The class column is "label"

# Run the sbrl algorithm on the training set
  sbrl_model <- sbrl(data_train, iters=20000, pos_sign="1",
   neg_sign="0", rule_minlen=1, rule_maxlen=3,
   minsupport_pos=0.10, minsupport_neg=0.10,
   lambda=10.0, eta=1.0, nchain=25)
  print(sbrl_model)

# Make predictions on the test set
  yhat <- predict(sbrl_model, data_test)
# yhat will be a list of predicted negative and positive probabilities for the test data.

#clean up
rm(list = ls())
gc()
```

`print.sbrl` *INTERPRETABLE VERSION OF A SBRL MODEL*

Description

This function prints an sbrl object. It is a method for the generic function print of class "sbrl".

Usage

```
# S3 method for class 'sbrl'
# This complies with the form of the standard generic method print
## S3 method for class 'sbrl'
print(x, useS4=FALSE, ...)
## S3 method for class 'sbrl'
show(x, useS4=FALSE, ...)
```

Arguments

`x`	A sbrl model returned from `sbrl` function
`useS4`	An argument used to match showDefault function. Fixed as FALSE.
`...`	further arguments passed to or from other methods.

Details

This function is a method for the generic function print for class "sbrl". It can be invoked by calling print for an object of the appropriate class, or directly by calling print.sbrl regardless of the class of the object.

Value

No return value. This function prints out the logical structure of the sbrl model as a sequence of IF-THEN rules, identical to a decision list or one-sided decision tree.

Examples

```
# Let us use the titactoe dataset
data(tictactoe)
for (name in names(tictactoe)) {tictactoe[name] <- as.factor(tictactoe[,name])}

# Train on two-thirds of the data
b = round(2*nrow(tictactoe)/3, digit=0)
data_train <- tictactoe[1:b, ]
# Test on the remaining one third of the data
data_test <- tictactoe[(b+1):nrow(tictactoe), ]
# data_train, data_test are dataframes with factor columns
# The class column is "label"

# Run the sbrl algorithm on the training set
  sbrl_model <- sbrl(data_train, iters=20000, pos_sign="1",
   neg_sign="0", rule_minlen=1, rule_maxlen=3,
   minsupport_pos=0.10, minsupport_neg=0.10,
   lambda=10.0, eta=1.0, nchain=25)
  print(sbrl_model)

#clean up
rm(list = ls())
gc()
```

`sbrl` *fit the scalable bayesian rule lists model*

Description

Fit the scalable bayesian rule lists model with given data and parameters. It generates a model that is a probabilistic classifier that optimizes the posterior of a Bayesian hierarchical model over pre-mined association rules.

Usage

```
sbrl(tdata, iters=30000, pos_sign="1",
 neg_sign="0", rule_minlen=1, rule_maxlen=1,
 minsupport_pos=0.10, minsupport_neg=0.10,
 lambda=10.0, eta=1.0, alpha=c(1,1), nchain=10)
```

Arguments

tdata	a dataframe, with a "label" column specifying the correct labels for each observation.
iters	the number of iterations for each MCMC chain.
pos_sign	the sign for the positive labels in the "label" column.
neg_sign	the sign for the negative labels in the "label" column.
rule_minlen	the minimum number of cardinality for rules to be mined from the dataframe.
rule_maxlen	the maximum number of cardinality for rules to be mined from the dataframe.
minsupport_pos	a number between 0 and 1, for the minimum percentage support for the positive observations.
minsupport_neg	a number between 0 and 1, for the minimum percentage support for the negative observations.
lambda	a hyperparameter for the expected length of the rule list.
eta	a hyperparameter for the expected cardinality of the rules in the optimal rule list.
alpha	a prior pseudo-count for the positive and negative classes. fixed at 1's
nchain	an integer for the number of the chains that MCMC will be running.

Value

Return a list of :

rs	a ruleset which contains the rule indices and their positive probabilities for the best rule list by training sbrl with the given data and parameters.
rulenames	a list of all the rule names mined with arules.
featurenames	a list of all the feature names.
mat_feature_rule	a binary matrix representing which features are included in which rules.

Author(s)

Hongyu Yang, Morris Chen, Cynthia Rudin, Margo Seltzer

References

Hongyu Yang, Cynthia Rudin, Margo Seltzer (2017) *Scalable Bayesian Rule Lists.* Proceedings of the 34th International Conference on Machine Learning, PMLR 70:3921-3930, 2017.

Benjamin Letham, Cynthia Rudin, Tyler McCormick and David Madigan (2015) *Building Interpretable Classifiers with Rules using Bayesian Analysis.* Annals of Applied Statistics, 2015.

Examples

```
# Let us use the titactoe dataset
data(tictactoe)
for (name in names(tictactoe)) {tictactoe[name] <- as.factor(tictactoe[,name])}

# Train on two-thirds of the data
b = round(2*nrow(tictactoe)/3, digit=0)
data_train <- tictactoe[1:b, ]
# Test on the remaining one third of the data
data_test <- tictactoe[(b+1):nrow(tictactoe), ]
# data_train, data_test are dataframes with factor columns
# The class column is "label"

# Run the sbrl algorithm on the training set
  sbrl_model <- sbrl(data_train, iters=20000, pos_sign="1",
   neg_sign="0", rule_minlen=1, rule_maxlen=3,
   minsupport_pos=0.10, minsupport_neg=0.10,
   lambda=10.0, eta=1.0, nchain=25)
  print(sbrl_model)

# Make predictions on the test set
  yhat <- predict(sbrl_model, data_test)
# yhat will be a list of predicted negative and positive probabilities for the test data.

#clean up
rm(list = ls())
gc()
```

---

tictactoe *SHUFFLED TIC-TAC-TOE-ENDGAME DATASET*

Description

This is a shuffled version of the Tic-Tac-Toe Endgame Data Set on UCI Machine Learning Repository.

Usage

```
data("tictactoe")
```

Format

A data frame with 958 observations on the following 10 variables.

`c1` a factor with levels b, o, x

`c2` a factor with levels b, o, x

`c3` a factor with levels b, o, x

`c4` a factor with levels b, o, x

`c5` a factor with levels `b`, `o`, `x`

`c6` a factor with levels `b`, `o`, `x`

`c7` a factor with levels `b`, `o`, `x`

`c8` a factor with levels `b`, `o`, `x`

`c9` a factor with levels `b`, `o`, `x`

`label` an integer with values `0`, `1`

Details

This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row").

Source

https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame

Examples

```
data(tictactoe)
## maybe str(tictactoe) ; plot(tictactoe) ...

#clean up
rm(list = ls())
gc()
```

Index

∗ **package**
sbrl-package, 2

get_data_feature_mat, 3

predict *(*predict.sbrl*)*, 4
predict.sbrl, *2*, 4
print.sbrl, *2*, 5

sbrl, *2*, *4, 5*, 6
sbrl-package, 2
show.sbrl, *2*
show.sbrl *(*print.sbrl*)*, 5

tictactoe, 8