Package 'piRF'

October 14, 2022

Title Prediction Intervals for Random Forests

Version 0.1.0

Date 2020-04-17

Maintainer Chancellor Johnstone <chancellor.johnstone@gmail.com>

Description

Implements multiple state-of-the-art prediction interval methodologies for random forests. These include: quantile regression intervals, out-of-bag intervals, bag-of-observations intervals, one-step boosted random forest intervals, bias-corrected intervals, high-density intervals, and split-conformal intervals. The implementations include a combination of novel adjustments to the original random forest methodology and novel prediction interval methodologies. All of these methodologies can be utilized using solely this package, rather than a collection of separate packages. Currently, only regression trees are supported. Also capable of handling high dimensional data.
Roy, Marie-Helene and Larocque, Denis (2019) <doi:10.1177/0962280219829885>.
Ghosal, Indrayudh and Hooker, Giles (2018) <arXiv:1803.08000>.
Zhu, Lin and Lu, Jiaxin and Chen, Yihong (2019) <arXiv:1905.10101>.
Zhang, Haozhe and Zimmerman, Joshua and Nettleton, Dan and Nordman, Daniel J. (2019) <doi:10.1080/00031305.2019.1585288>.
Meinshausen, Nicolai (2006) <http://www.jmlr.org/papers/volume7/meinshausen06a/meinshausen06a.pdf>.
Romano, Yaniv and Patterson, Evan and Candes, Emmanuel (2019) <arXiv:1905.03222>.
Tung, Nguyen Thanh and Huang, Joshua Zhexue and Nguyen, Thuy Thi and Khan, Imran (2014) <doi:10.13140/2.1.2500.8002>.

License GPL-3

Encoding UTF-8

Depends R (>= 2.10)

Suggests testthat, devtools, foreach, doParallel, hdrcde, rfinterval, ranger

URL http://github.com/chancejohnstone/piRF

LazyData true

RoxygenNote 7.0.2

Imports Rdpack

RdMacros Rdpack

NeedsCompilation no

Author Chancellor Johnstone [cre, aut, cph],
Haozhe Zhang [aut, cph],
Martin Wright [ctb, cph],
Gregor DeCillia [ctb, cph]

Repository CRAN

Date/Publication 2020-05-12 09:50:02 UTC

R topics documented:

`airfoil` *airfoil self noise dataset*

Description

The NASA data set comprises different size NACA 0012 airfoils at various wind tunnel speeds and angles of attack. The span of the airfoil and the observer position were the same in all of the experiments.

Usage

```
data(airfoil)
```

Format

An object of class `data.frame` with 1503 rows and 6 columns.

Source

UCI Archive

References

T.F. Brooks, D.S. Pope, and A.M. Marcolini. Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.

K. Lau. A neural networks approach for aerofoil noise prediction. Master's thesis, Department of Aeronautics. Imperial College of Science, Technology and Medicine (London, United Kingdom), 2006.

Lopez, R. and Balsa-Canto, E. and Onate, E. Neural Networks for Variational Problems in Engineering. PhD Thesis, Technical University of Catalonia, 2008.

Examples

```
data(airfoil)
airfoil$pressure
```

`parse.formula` *Authors: Marvin N. Wright, Gregor DeCillia*

Description

taken from source code for ranger package; not exported with package

Usage

```
parse.formula(formula, data, env = parent.frame())
```

Arguments

`formula`	Object of class `formula` or `character` describing the model to fit.
`data`	Training data of class `data.frame`.
`env`	The environment in which the left hand side of `formula` is evaluated.

Details

Parse formula and return dataset containing selected columns. Interactions are supported for numerical columns only. An interaction column is the product of all interacting columns.

Value

Dataset including selected columns and interactions.

`rfint` *rfint()*

Description

Implements seven different random forest prediction interval methods.

Usage

```
rfint(
  formula = formula,
  train_data = NULL,
  test_data = NULL,
  method = "Zhang",
  alpha = 0.1,
  symmetry = TRUE,
  seed = NULL,
  m_try = 2,
  num_trees = 500,
  min_node_size = 5,
  num_threads = parallel::detectCores(),
  calibrate = FALSE,
  Roy_method = "quantile",
  featureBias = FALSE,
  predictionBias = TRUE,
  Tung_R = 5,
  Tung_num_trees = 75,
  variant = 1,
  Ghosal_num_stages = 2,
  prop = 0.618,
  concise = TRUE,
  interval_type = "two-sided"
)
```

Arguments

formula	Object of class formula or character describing the model to fit. Interaction terms supported only for numerical variables.
train_data	Training data of class data.frame.
test_data	Test data of class data.frame. Utilizes ranger::predict() to produce prediction intervals for test data.
method	Choose what method to generate RF prediction intervals. Options are method = c("Zhang", "quantile", "Romano", "Ghosal", "Roy", "Tung", "HDI"). Defaults to method = "Zhang".
alpha	Significance level for prediction intervals. Defaults to alpha = 0.1.
symmetry	True if constructing symmetric out-of-bag prediction intervals, False otherwise. Used only method = "Zhang". Defaults to symmetry = TRUE.
seed	Seed for random number generation. Currently not utilized.
m_try	Number of variables to randomly select from at each split.
num_trees	Number of trees used in the random forest.
min_node_size	Minimum number of observations before split at a node.
num_threads	The number of threads to use in parallel. Default is the current number of cores.

calibrate	If calibrate = TRUE, intervals are calibrated to achieve nominal coverage. Currently uses quantiles to calibrate. Only for method = "Roy".
Roy_method	Interval method for method = "Roy". Options are Roy_method = c("quantile", "HDI", "CHDI").
featureBias	Remove feature bias. Only for method = "Tung".
predictionBias	Remove prediction bias. Only for method = "Tung".
Tung_R	Number of repetitions used in bias removal. Only for method = "Tung".
Tung_num_trees	Number of trees used in bias removal. Only for method = "Tung".
variant	Choose which variant to use. Options are method = c("1", "2"). Only for method = "Ghosal".
Ghosal_num_stages	Number of total stages. Only for method = "Ghosal".
prop	Proportion of training data to sample for each tree. Only for method = "Ghosal".
concise	If concise = TRUE, only predictions output. Defaults to concise = FALSE.
interval_type	Type of prediction interval to generate. Options are method = c("two-sided", "lower", "upper"). Default is method = "two-sided".

Details

The seven methods implemented are cited in the References section. Additional information can be found within those references. Each of these methods are implemented by utilizing the ranger package. For method = "Zhang", prediction intervals are generated using out-of-bag residuals. method = "Romano" utilizes a split-conformal approach. method = "Roy" uses a bag-of-predictors approach. method = "Ghosal" performs boosting to reduce bias in the random forest, and estimates variance. The authors provide multiple variants to their methodology. method = "Tung" debiases feature selection and prediction. Prediction intervals are generated using quantile regression forests. method = "HDI" delivers prediction intervals through highest-density interval regression forests. method = "quantile" utilizes quantile regression forests.

Value

int	Default output. Includes prediction intervals for all methods in methods.
preds	Predictions for test data for all methods in methods. Output when concise = FALSE.

Author(s)

Chancellor Johnstone

Haozhe Zhang

References

Breiman L (2001). "Random forests." *Machine learning*, **45**(1), 5–32. https://link.springer.com/article/10.1023/A:1010933404324.

Ghosal I, Hooker G (2018). "Boosting random forests to reduce bias; one-step boosted forest and its variance estimate." *arXiv preprint*. https://arxiv.org/pdf/1803.08000.pdf.

Meinshausen N (2006). "Quantile regression forests." *Journal of Machine Learning Research*, **7**(Jun), 983–999. http://www.jmlr.org/papers/volume7/meinshausen06a/meinshausen06a.pdf.

Romano Y, Patterson E, Candes E (2019). "Conformalized quantile regression." *arXiv preprint*. https://arxiv.org/pdf/1905.03222v1.pdf.

Roy M, Larocque D (2019). "Prediction intervals with random forests." *Statistical methods in medical research*. https://doi.org/10.1177/0962280219829885.

Tung NT, Huang JZ, Nguyen TT, Khan I (2014). "Bias-corrected quantile regression forests for high-dimensional data." In *2014 International Conference on Machine Learning and Cybernetics*, volume 1, 1–6. IEEE. https://link.springer.com/article/10.1007/s10994-014-5452-1.

Zhang H, Zimmerman J, Nettleton D, Nordman DJ (2019). "Random forest prediction intervals." *The American Statistician*, 1–15. https://doi.org/10.1080/00031305.2019.1585288.

Zhu L, Lu J, Chen Y (2019). "HDI-Forest: Highest density interval regression forest." *arXiv preprint*. https://arxiv.org/pdf/1905.10101.pdf.

See Also

ranger

rfinterval

Examples

```
library(piRF)

#functions to get average length and average coverage of output
getPILength <- function(x){
#average PI length across each set of predictions
l <- x[,2] - x[,1]
avg_l <- mean(l)
return(avg_l)
}

getCoverage <- function(x, response){
  #output coverage for test data
  coverage <- sum((response >= x[,1]) * (response <= x[,2]))/length(response)
  return(coverage)
}

#import airfoil self noise dataset
data(airfoil)
method_vec <- c("quantile", "Zhang", "Tung", "Romano", "Roy", "HDI", "Ghosal")
#generate train and test data
ratio <- .975
nrow <- nrow(airfoil)
n <- floor(nrow*ratio)
samp <- sample(1:nrow, size = n)
train <- airfoil[samp,]
test <- airfoil[-samp,]
```

```
#generate prediction intervals
res <- rfint(pressure ~ . , train_data = train, test_data = test,
             method = method_vec,
             concise= FALSE,
             num_threads = 1)

#empirical coverage, and average prediction interval length for each method
coverage <- sapply(res$int, FUN = getCoverage, response = test$pressure)
coverage
length <- sapply(res$int, FUN = getPILength)
length

#get current mfrow setting
opar <- par(mfrow = c(2,2))

#plotting intervals and predictions
for(i in 1:7){
   col <- ((test$pressure >= res$int[[i]][,1]) *
   (test$pressure <= res$int[[i]][,2])-1)*(-1)+1
   plot(x = res$preds[[i]], y = test$pressure, pch = 20,
      col = "black", ylab = "true", xlab = "predicted", main = method_vec[i])
   abline(a = 0, b = 1)
   segments(x0 = res$int[[i]][,1], x1 = res$int[[i]][,2],
      y1 = test$pressure, y0 = test$pressure, lwd = 1, col = col)
}
par(opar)
```

Index