Package 'rjaf'

November 11, 2024

Title Regularized Joint Assignment Forest with Treatment Arm Clustering

Version 0.1.0

URL https://github.com/wustat/rjaf

BugReports https://github.com/wustat/rjaf/issues

Description Personalized assignment to one of many treatment arms via regularized and clustered joint assignment forests as described in Ladhania, Spiess, Ungar, and Wu (2023) <doi:10.48550/arXiv.2311.00577>. The algorithm pools information across treatment arms: it considers a regularized forest-based assignment algorithm based on greedy recursive partitioning that shrinks effect estimates across arms; and it incorporates a clustering scheme that combines treatment arms with consistently similar outcomes.

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports Rcpp, dplyr, tibble, magrittr, readr, randomForest, ranger, forcats, rlang (>= 1.1.0), tidyr, stringr, MASS

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

NeedsCompilation yes

Author Wenbo Wu [aut, cph] (<https://orcid.org/0000-0002-7642-9773>), Xinyi Zhang [aut, cre, cph] (<https://orcid.org/0009-0007-7306-491X>), Jann Spiess [aut, cph] (<https://orcid.org/0000-0002-4120-8241>), Rahul Ladhania [aut, cph] (<https://orcid.org/0000-0002-7902-7681>)

Maintainer Xinyi Zhang <zhang.xinyi@nyu.edu>

Repository CRAN

Date/Publication 2024-11-11 20:10:02 UTC

Contents

`Example_data` *Simulated randomized experiment data*

Description

A data set simulated based on the function `sim.data` in the Examples section of `rjaf.R`.

Usage

```
Example_data
```

Format

A data frame with 100 rows, 12 columns, and 5 treatment arms:

id Subject ID.

trt Character strings indicating treatment arms (0, 1, 2, 3, and 4) assigned to subjects, where arm 0 is considered to be the control.

X1 Covariate X1.

X2 Covariate X2.

X3 Covariate X3.

prob Probabilities of treatment assignment.

Y Observed outcomes.

Y0 Counterfactual outcomes when subjects are assigned to treatment 0.

Y1 Counterfactual outcomes when subjects are assigned to treatment 1.

Y2 Counterfactual outcomes when subjects are assigned to treatment 2.

Y3 Counterfactual outcomes when subjects are assigned to treatment 3.

Y4 Counterfactual outcomes when subjects are assigned to treatment 4.

`residualize` *Arbitrary residualization of outcomes*

Description

This function employs random forests and cross-validation to residualize outcomes following Wu and Gagnon-Bartsch (2018). That is, predicted outcomes resulting from random forests are subtracted from the original outcomes. Doing so helps in adjusting for small imbalanaces in baseline covariates and removing part of the variation in outcomes common across treatment arms

Usage

```
residualize(data, y, vars, nfold = 5, fun.rf = "ranger")
```

Arguments

`data`	input data used for training and estimation, where each row corresponds to an individual and columns contain information on treatments, covariates, probabilities of treatment assignment, and observed outcomes.
`y`	a character string denoting the column name of outcomes.
`vars`	a vector of character strings denoting the column names of covariates.
`nfold`	number of folds in cross-validation. The default value is 5.
`fun.rf`	a character string specifying which random forest package to use. Two options are `ranger` and `randomForest`, with the default being `ranger`.

Value

data for training and estimation with residualized outcomes.

References

Wu, Edward and Johann A Gagnon-Bartsch (2018). The LOOP Estimator: Adjusting for Covariates in Randomized Experiments. Evaluation Review, 42(4):458–488.

Examples

```
data(Example_data)
library(dplyr)
library(magrittr)
Example_trainest <- Example_data %>% slice_sample(n = floor(0.5 * nrow(Example_data)))
y <- "Y"
vars <- paste0("X", 1:3)
Example_resid <- residualize(Example_trainest, y, vars, nfold = 5, fun.rf = "ranger")
```

`rjaf` *Regularized Joint Assignment Forest with Treatment Arm Clustering*

Description

This algorithm trains a joint forest model to estimate the optimal treatment assignment by pooling information across treatment arms.

Usage

```
rjaf(
  data.trainest,
  data.validation,
  y,
  id,
  trt,
  vars,
  prob,
  ntrt = 5,
  nvar = 3,
  lambda1 = 0.5,
  lambda2 = 0.5,
  ipw = TRUE,
  nodesize = 5,
  ntree = 1000,
  prop.train = 0.5,
  eps = 0.1,
  resid = TRUE,
  clus.tree.growing = FALSE,
  clus.outcome.avg = FALSE,
  clus.max = 10,
  reg = TRUE,
  impute = TRUE,
  setseed = FALSE,
  seed = 1,
  nfold = 5
)
```

Arguments

`data.trainest`	input data used for training and estimation, where each row corresponds to an individual and columns contain information on treatments, covariates, probabilities of treatment assignment, and observed outcomes.
`data.validation`	input data used for validation with the same row and column information as in `data.trainest`.

y	a character string denoting the column name of outcomes.
id	a character string denoting the column name of individual IDs.
trt	a character string denoting the column name of treatments.
vars	a vector of character strings denoting the column names of covariates.
prob	a character string denoting the column name of probabilities of treatment assignment. If missing, a column named "prob" will be added to `data.trainest` and `data.validation` indicating simple random treatment assignment.
ntrt	number of treatments randomly sampled at each split. It should be at most equal to the number of unique treatments available. The default value is 5.
nvar	number of covariates randomly sampled at each split. It should be at most equal to the number of unique covariates available. The default value is 3.
lambda1	regularization parameter for shrinking arm-wise within-leaf average outcomes towards the overall within-leaf average outcome during recursive partitioning. The default value is 0.5.
lambda2	regularization parameter for shrinking arm-wise within-leaf average outcomes towards the overall within-leaf average outcome during outcome estimation. It is only valid when `reg` is `TRUE`. The default value is 0.5.
ipw	a logical indicator of inverse probability weighting when calculating leaf-wise weighted averages based on Wu and Gagnon-Bartsch (2018). The default value is `TRUE`.
nodesize	minimum number of observations in a terminal node. The default value is 5.
ntree	number of trees to grow in the forest. This should not be set to too small a number. The default value is 1000.
prop.train	proportion of data used for training in `data.trainest`. The default value is 0.5.
eps	threshold for minimal welfare gain in terms of the empirical standard deviation of the overall outcome `y`. The default value is 0.1.
resid	a logical indicator of arbitrary residualization. If `TRUE`, residualization is implemented to reduce the variance of the outcome. The default value is `TRUE`.
clus.tree.growing	a logical indicator of clustering for tree growing. The default value is `FALSE`.
clus.outcome.avg	a logical indicator of clustering for tree bagging. If `TRUE`, the average outcome is calculated across treatment clusters determined by the k-means. The default value is `FALSE`. This option is deprecated.
clus.max	the maximum number of clusters for k-means. It should be greater than 1 and at most equal to the number of unique treatments. The default value is 10.
reg	a logical indicator of regularization when calculating the arm-wise within-leaf average outcome.
impute	a logical indicator of imputation. If `TRUE`, the within-leaf average outcome is used to impute the arm-wise within-leaf average outcome when the arm has no observation. If `FALSE`, the within-leaf average outcome is set to zero when the arm has no observation. The default value is `TRUE`.

`setseed`	a logical indicator. If `TRUE`, a seed is set through the argument `seed` below and passed to the function `rjaf_cpp`. The default value is `FALSE`.
`seed`	an integer used as a random seed if `setseed=TRUE`. The default value is 1.
`nfold`	the number of folds used for cross-validation in outcome residualization and k-means clustering. The default value is 5.

Details

It first obtains an assignment forest by bagging trees as in Kallus (2017) with covariate and treatment arm randomization for each tree and estimating "honest" and regularized estimates of the treatment-specific counterfactual outcomes on the training sample following Wager and Athey (2018).

Like Bonhomme and Manresa (2015), it uses a clustering of treatment arms when constructing the assignment trees. It employs a k-means algorithm for clustering the K treatment arms into M treatment groups based on the K predictions for each of the n units in the training sample.

After clustering, it then repeats the assignment-forest algorithm on the full training data with M+1 (including control) "arms" (where data from the original arms are combined by groups) to obtain an ensemble of trees.

It obtains final regularized predictions and assignments, where it estimates regularized averages separately by the original treatment arms $k \in \{0, \ldots, K\}$ and obtain the corresponding assignment.

Value

If `clus.tree.growing` and `clus.outcome.avg` are `TRUE`, `rjaf` returns a list of two objects: a tibble named as `res` consisting of individual IDs, cluster identifiers, and predicted outcomes, and a data frame named as `clustering` consisting of cluster identifiers, probabilities of being assigned to the clusters, and treatment arms. Otherwise, `rjaf` simply returns a tibble of individual IDs, optimal treatment arms identified by the algorithm, treatment clusters if `clus.tree.growing` is `TRUE`, and predicted optimal outcomes (ending with `.rjaf`). If counterfactual outcomes are also present, they will be included in the tibble along with the column of predicted outcomes (ending with `.cf`).

References

Bonhomme, Stéphane and Elena Manresa (2015). Grouped Patterns of Heterogeneity in Panel Data. Econometrica, 83: 1147-1184.

Kallus, Nathan (2017). Recursive Partitioning for Personalization using Observational Data. In Precup, Doina and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 1789–1798. PMLR.

Wager, Stefan and Susan Athey (2018). Estimation and inference of heterogeneous treatment effects using random forests. Journal of the American Statistical Association, 113(523):1228–1242.

Wu, Edward and Johann A Gagnon-Bartsch (2018). The LOOP Estimator: Adjusting for Covariates in Randomized Experiments. Evaluation Review, 42(4):458–488.

Examples

```
library(dplyr)
library(MASS)
sim.data <- function(n, K, gamma, sigma, prob=rep(1,K+1)/(K+1)) {
   # K: number of treatment arms
  options(stringsAsFactors=FALSE)
  data <- left_join(data.frame(id=1:n,
                               trt=sample(0:K, n, replace=TRUE, prob),
                               mvrnorm(n, rep(0,3), diag(3))),
                    data.frame(trt=0:K, prob), by="trt")
  data <- mutate(data, tmp1=10+20*(X1>0)-20*(X2>0)-40*(X1>0&X2>0),
                 tmp2=gamma*(2*(X3>0)-1)/(K-1),
                 tmp3=-10*X1^2,
                 Y=tmp1+tmp2*(trt>0)*(2*trt-K-1)+tmp3*(trt==0)+rnorm(n,0,sigma))
  # Y: observed outcomes
  Y.cf <- data.frame(sapply(0:K, function(t) # counterfactual outcomes
    mutate(data, Y=tmp1+tmp2*(t>0)*(2*t-K-1)+tmp3*(t==0))$Y))
  names(Y.cf) <- paste0("Y",0:K)
  return(mutate(bind_cols(dplyr::select(data, -c(tmp1,tmp2,tmp3)), Y.cf),
                across(c(id, trt), as.character)))
}

n <- 200; K <- 3; gamma <- 10; sigma <- 10
Example_data <- sim.data(n, K, gamma, sigma)
Example_trainest <- Example_data %>% slice_sample(n = floor(0.5 * nrow(Example_data)))
Example_valid <- Example_data %>% filter(!id %in% Example_trainest$id)
id <- "id"; y <- "Y"; trt <- "trt"
vars <- paste0("X", 1:3)
forest.reg <- rjaf(Example_trainest, Example_valid, y, id, trt, vars, ntrt = 4, ntree = 100,
                   clus.tree.growing = FALSE)
```

Index