Package ‘BartMixVs’

May 5, 2022

Type Package
Title Variable Selection Using Bayesian Additive Regression Trees
Version 1.0.0
Date 2022-05-02
Author Chuji Luo [aut, cre],
       Michael J. Daniels [aut],
       Robert McCulloch [ctb],
       Rodney Sparapani [ctb]
Maintainer Chuji Luo <cjluo@ufl.edu>
Description Bayesian additive regression trees (BART) provides flexible non-parametric modeling of mixed-type predictors for continuous and binary responses. This package is built upon CRAN R package 'BART', version 2.7 (<https://github.com/cran/BART>). It implements the three proposed variable selection approaches in the paper: Luo, C and Daniels, M. J. (2021), "Variable Selection Using Bayesian Additive Regression Trees." <arXiv:2112.13998>, and other three existing BART-based variable selection approaches.
License GPL (>= 2)
Depends R (>= 2.10), nlme, nnet
Imports Rcpp (>= 1.0.6), loo, mvtnorm
LinkingTo Rcpp
RoxygenNote 7.1.1
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-05-05 05:50:02 UTC

R topics documented:

BartMixVs-package ................................................................. 2
abc.vs ............................................................................. 3
bartModelMatrix ................................................................. 7
Description

This package provides implementations of existing BART-based variable selection approaches.

Details

Bayesian additive regression trees (BART) provides flexible nonparametric modeling of mixed-type predictors for continuous and binary responses. This package is built upon CRAN R package BART, version 2.7 (https://github.com/cran/BART). It implements the three proposed variable selection approaches of the paper, Luo, C and Daniels, MJ (2022), “Variable Selection Using Bayesian Additive Regression Trees”, and other three existing BART-based variable selection approaches.

Author(s)

Chuji Luo <cjluo@ufl.edu>, Michael J. Daniels <daniels@ufl.edu>

Maintainer: Chuji Luo <cjluo@ufl.edu>

References


See Also
Optional links to other man pages

Examples

## Not run:
## Optional simple examples of the most important functions
## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)

---

abc.vs Variable selection with ABC Bayesian forest

Description

This function implements the variable selection approach proposed in Liu, Rockova and Wang (2021). Rockova and Pas (2020) introduce a spike-and-forest prior which wraps the BART prior with a spike-and-slab prior on the model space. Due to intractable marginal likelihood, Liu, Rockova and Wang (2021) propose an approximate Bayesian computation (ABC) sampling method based on data-splitting to help sample from the model space with higher ABC acceptance rate.

Usage

abc.vs(
  x,
  y,
  nabc = 1000,
  tolerance = 0.1,
  threshold = 0.25,
  beta.params = c(1, 1),
  beta.theta = NA,
  split.ratio = 0.5,
  probit = FALSE,
  true.idx = NULL,
  analysis = TRUE,
  sparse = FALSE,
  xinfo = matrix(0, 0, 0),
  numcut = 100L,
  usequant = FALSE,
)
cont = FALSE,
rm.const = TRUE,
k = 2,
power = 2,
base = 0.95,
split.prob = "polynomial",
ntree = 10L,
ndpost = 1,
nskip = 200,
keepvery = 1L,
printevery = 100L,
verbose = FALSE
)

Arguments

x A matrix or a data frame of predictors values with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with \( q \) levels in a data frame, it is replaced with \( q \) dummy variables.

y A vector of response (continuous or binary) values.
nabc The number of ABC samples, i.e., the number of subsets sampled from the model space.
tolerance A number between 0 and 1; the nabc subsets are ranked by MSE in ascending order if the response variable is continuous (or by mean log loss (MLL) if the response variable is binary), and the top tolerance*100% of the subsets are accepted by ABC for selection.
threshold A number between 0 and 1; within the ABC accepted subsets, predictors with MPVIP exceeding threshold are selected.

beta.params A vector with two positive numbers; the spike-and-slab prior on the model space is assumed to be a beta-binomial prior, i.e., \( \theta \) Beta(beta.params[1], beta.params[2]) and each predictor is included into a model by Bernoulli(\( \theta \)); only used when beta.theta=NA.

beta.theta A number between 0 and 1; the probability that a predictor is included into a model; if beta.theta=NA, it is sampled from Beta(beta.params[1], beta.params[2]).

split.ratio A number between 0 and 1; the data set \((x, y)\) is split into a training set and a testing set according to the split.ratio.

probit A Boolean argument indicating whether the response variable is binary or continuous; probit=FALSE (by default) means that the response variable is continuous.

true.idx (Optional) A vector of indices of the true relevant predictors; if true.idx is provided and analysis=TRUE, metrics including precision, recall and F1 score are returned.

analysis A Boolean argument indicating whether to perform variable selection; if analysis=TRUE, the best model selected by ABC Bayesian forest is returned; if analysis=FALSE, only return the visited models and their corresponding model errors.

sparse A Boolean argument indicating whether to perform DART or BART.
xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. \(xinfo=matrix(0.0,0,0)\) indicates the cut-points are specified by BART.

numcut The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to \(ncol(x)\) is required, where the \(i\)-th element gives the number of cut-points for the \(i\)-th predictor in \(x\). If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of \(x\). If usequants=TRUE, then \(\min(\text{numcut}, \text{the number of unique values in the corresponding column of } x - 1)\) cut-point values are used.

usequants A Boolean argument indicating how the cut-points in \(xinfo\) are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont A Boolean argument indicating whether to assume all predictors are continuous.

rm.const A Boolean argument indicating whether to remove constant predictors.

k The number of prior standard deviations that \(E(Y|x) = f(x)\) is away from \(+/- .5\). The response (y) is internally scaled to the range from \(-.5\) to \(.5\). The bigger k is, the more conservative the fitting will be.

power The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

base The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth \(d\) is \(\text{base}^d\).

split.prob A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

ntree The number of trees in the ensemble.

ndpost The number of posterior samples returned.

nskip The number of posterior samples burned in.

keepevery Every keepevery posterior sample is kept to be returned to the user.

printevery As the MCMC runs, a message is printed every printevery iterations.

verbose A Boolean argument indicating whether any messages are printed out.

Details

At each iteration of the algorithm, the data set is randomly split into a training set and a testing set according to a certain split ratio. The algorithm proceeds by sampling a subset from the spike-and-slab prior on the model space, fitting a BART model on the training set only with the predictors in the subset, and computing the root mean squared errors (RMSE) for the test set based on a posterior sample from the fitted BART model. Only those subsets that result in a low RMSE on the test set are kept for selection. ABC Bayesian forest selects predictors based on their marginal posterior variable inclusion probabilities (MPVIPS) which are estimated by computing the proportion of ABC accepted BART posterior samples that use the predictor at least one time. Given the MPVIPS, predictors with MPVIP exceeding a pre-specified threshold are selected.

See Liu, Rockova and Wang (2021) or Section 2.2.4 in Luo and Daniels (2021) for details.
Value

The function abc.vs() returns a list with the following components.

- **theta**: The probability that a predictor is included into a model.
- **models**: A matrix with nabc rows and ncol(x) columns; each row corresponds to a ABC model (or subset); if the (i, j)-th element is 1, it means that the j-th predictor is included in the i-th ABC model; if the (i, j)-th element is 0, it means that the j-th predictor is not included in the i-th ABC model.
- **actual.models**: A matrix with nabc rows and ncol(x) columns; each row corresponds to an ABC BART posterior sample; if the (i, j)-th element is 1, it means that the j-th predictor is used as a split variable at least one time in the BART posterior sample of the i-th ABC model; if the (i, j)-th element is 0, it means that the j-th predictor is not used as a split variable in the BART posterior sample of the i-th ABC model.
- **model.errors**: The vector of MSEs (or MLLs if the response variable is binary) for the nabc ABC models.
- **idx**: The vector of indices (in terms of the row numbers of models) of the ABC accepted models which are the top tolerance*100% of the nabc ABC models when ranked by MSE or MLL in ascending order; only returned when analysis=TRUE.
- **top.models**: A matrix with length(idx) rows and ncol(x) columns, representing the ABC accepted models; top.models=models[idx, ]; only returned when analysis=TRUE.
- **top.actual.models**: A matrix with length(idx) rows and ncol(x) columns, representing the ABC accepted BART posterior samples; top.models=actual.models[idx, ]; only returned when analysis=TRUE.
- **mip**: The vector of marginal posterior variable inclusion probabilities; only returned when analysis=TRUE.
- **best.model**: The vector of predictors selected by ABC Bayesian forest; only returned when analysis=TRUE.
- **precision**: The precision score for the ABC Bayesian forest; only returned when analysis=TRUE and true.idx is provided.
- **recall**: The recall score for the ABC Bayesian forest; only returned when analysis=TRUE and true.idx is provided.
- **f1**: The F1 score for the ABC Bayesian forest; only returned when analysis=TRUE and true.idx is provided.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


**See Also**

`permute.vs`, `medianInclusion.vs` and `mc.backward.vs`.

**Examples**

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## test abc.vs() function
res = abc.vs(data$X, data$Y, nabc=100, tolerance=0.1, threshold=0.25, beta.params=c(1.0, 1.0), split.ratio=0.5, probit=FALSE, true.idx=c(1,2,6:8), ntree=10, ndpost=1, nskip=200, analysis=TRUE)
```

---

**bartModelMatrix**

Create a matrix out of a vector or data frame

**Description**

The external BART functions (e.g. `wbart()`) operate on matrices in memory. Therefore, if the user submits a vector or data frame, then this function converts it to a matrix. Also, it determines the number of cut points necessary for each column when asked to do so. This function is inherited from the CRAN package 'BART'.

**Usage**

```r
bartModelMatrix(
  X,
  numcut = 0L,
  usequants = FALSE,
  type = 7,
  rm.const = FALSE,
  cont = FALSE,
  xinfo = NULL
)
```
Arguments

- **X**
  A vector or data frame where the matrix is created.

- **numcut**
  The maximum number of cut points to consider. If `numcut=0`, then return a matrix; otherwise, return a list containing a matrix `X`, a vector `numcut` and a list `xinfo`.

- **usequants**
  A Boolean argument indicating the way to generate cut points. If `usequants=FALSE`, then the cut points in `xinfo` are generated uniformly; otherwise, the quantiles are used for the cut points.

- **type**
  An integer between 1 and 9 determining which algorithm is employed in the function `quantile()`.

- **rm.const**
  A Boolean argument indicating whether to remove constant variables.

- **cont**
  A Boolean argument indicating whether to assume all variables are continuous.

- **xinfo**
  A list (matrix) where the items (rows) are the predictors and the contents (columns) of the items are the cut points. If `xinfo=NULL`, BART will choose `xinfo` for the user.

Value

The function `bartModelMatrix()` returns a list with the following components.

- **X**
  A matrix with rows corresponding to observations and columns corresponding to predictors (after dummification).

- **numcut**
  A vector of `nrow(X)` integers with each indicating the number of cut points for the corresponding predictor.

- **rm.const**
  A vector of indicators for the predictors (after dummification) used in BART; when the indicator is negative, it refers to remove that predictor.

- **xinfo**
  A list (matrix) where the items (rows) are the predictors and the contents (columns) of the items are the cut points.

- **grp**
  A vector of group indices for predictors. For example, if 2 appears 3 times in grp, the second predictor of X is a categorical predictor with 3 levels.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

`wbart` and `pbart`.

Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## test bartModelMatrix() function
res = bartModelMatrix(data$X, numcut=100, usequants=FALSE, cont=FALSE, rm.const=TRUE)
```

checkerboard  Generate data for an example of Zhu, Zeng and Kosorok (2015)

Description


Usage

`checkerboard(n, p, sigma, binary)`

Arguments

- `n` The number of observations.
- `p` The number of predictors.
- `sigma` The error variance.
- `binary` A boolean argument: `binary = TRUE` indicates that binary responses are generated and `binary = FALSE` indicates that continuous responses are generated.

Details

Sample the predictors $x_1, \ldots, x_p$ from Normal($0$, $\Sigma$) with $\Sigma_{j,k} = 0.3^{|j-k|}$, $j, k = 1, \ldots, p$. If `binary = FALSE`, sample the continuous response $y$ from Normal($f_0(x), \sigma^2$), where

$$f_0(x) = 2x_1 * x_4 + 2x_7 * x_{10}.$$ 

If `binary = TRUE`, sample the binary response $y$ from Bernoulli($\Phi(f_0(x))$) where $f_0$ is defined above and $\Phi$ is the cumulative density function of the standard normal distribution.
Value

Return a list with the following components.

- **X**: An n by p data frame representing predictors values, with each row corresponding an observation.
- **Y**: A vector of length n representing response values.
- **f0**: A vector of length n representing the values of \( f_0(x) \).
- **sigma**: The error variance which is only returned when \( binary = FALSE \).
- **prob**: A vector of length n representing the values of \( \Phi(f_0(x)) \), which is only returned when \( binary = TRUE \).

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


Examples

```r
data = checkerboard(100, 10, 1, FALSE)
```

**friedman**  
*Generate data for an example of Friedman (1991)*

Description


Usage

```r
friedman(n, p, sigma, binary)
```

Arguments

- **n**: The number of observations.
- **p**: The number of predictors.
- **sigma**: The error variance.
- **binary**: A boolean argument: \( binary = TRUE \) indicates that binary responses are generated and \( binary = FALSE \) indicates that continuous responses are generated.
Details

Sample the predictors $x_1, \ldots, x_p$ from Uniform(0, 1) independently. If binary = FALSE, sample the continuous response $y$ from Normal($f_0(x), \sigma^2$), where

$$f_0(x) = 10\sin(\pi x_1 \ast x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5.$$  

If binary = TRUE, sample the binary response $y$ from Bernoulli(\Phi(f_0(x))) where $f_0$ is defined above and $\Phi$ is the cumulative density function of the standard normal distribution.

Value

Return a list with the following components.

- **X**: An n by p data frame representing predictors values, with each row corresponding an observation.
- **Y**: A vector of length n representing response values.
- **f0**: A vector of length n representing the values of $f_0(x)$.
- **sigma**: The error variance which is only returned when binary = FALSE.
- **prob**: A vector of length n representing the values of $\Phi(f_0(x))$, which is only returned when binary = TRUE.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


Examples

```r
data = friedman(100, 10, 1, FALSE)
```

Description

This function implements the variable selection approach proposed in Liu, Rockova and Wang (2021) with parallel computation. Rockova and Pas (2020) introduce a spike-and-forest prior which wraps the BART prior with a spike-and-slab prior on the model space. Due to intractable marginal likelihood, Liu, Rockova and Wang (2021) propose an approximate Bayesian computation (ABC) sampling method based on data-splitting to help sample from the model space with higher ABC acceptance rate.

Unlike the function `abc.vs()` which sequentially evaluates ABC models, this function evaluates the ABC models in parallel.
Usage

mc.abc.vs(
  x,
  y,
  nabc = 1000,
  tolerance = 0.1,
  threshold = 0.25,
  beta.params = c(1, 1),
  split.ratio = 0.5,
  probit = FALSE,
  true.idx = NULL,
  sparse = FALSE,
  xinfo = matrix(0, 0, 0),
  numcut = 100L,
  usequants = FALSE,
  cont = FALSE,
  rm.const = TRUE,
  k = 2,
  power = 2,
  base = 0.95,
  split.prob = "polynomial",
  ntree = 10L,
  ndpost = 1,
  nskip = 200,
  keepevery = 1L,
  printevery = 100L,
  verbose = FALSE,
  mc.cores = 2L,
  nice = 19L,
  seed = 99L
)

Arguments

x          A matrix or a data frame of predictors values with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with $q$ levels in a data frame, it is replaced with $q$ dummy variables.

y          A vector of response (continuous or binary) values.

nabc       The number of ABC samples, i.e., the number of subsets sampled from the model space.

tolerance  A number between 0 and 1; the nabc subsets are ranked by MSE in ascending order if the response variable is continuous (or by mean log loss (MLL) if the response variable is binary), and the top tolerance*100% of the subsets are accepted by ABC for selection.

threshold  A number between 0 and 1; within the ABC accepted subsets, predictors with MPVIP exceeding threshold are selected.
beta.params: A vector with two positive numbers; the spike-and-slab prior on the model space is assumed to be a beta-binomial prior, i.e., $\theta \sim \text{Beta}(\beta\text{.params}[1], \beta\text{.params}[2])$ and each predictor is included into a model by Bernoulli($\theta$).

split.ratio: A number between 0 and 1; the data set ($x, y$) is split into a training set and a testing set according to the split.ratio.

probit: A Boolean argument indicating whether the response variable is binary or continuous; probit=FALSE (by default) means that the response variable is continuous.

true.idx: (Optional) A vector of indices of the true relevant predictors; if true.idx is provided, metrics including precision, recall and F1 score are returned.

sparse: A Boolean argument indicating whether to perform DART or BART.

xinfo: A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. xinfo=matrix(0.0,0,0) indicates the cut-points are specified by BART.

numcut: The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to ncol(x) is required, where the i-th element gives the number of cut-points for the i-th predictor in x. If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of x. If usequants=TRUE, then min(numcut, the number of unique values in the corresponding column of x - 1) cut-point values are used.

usequants: A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont: A Boolean argument indicating whether to assume all predictors are continuous.

rm.const: A Boolean argument indicating whether to remove constant predictors.

k: The number of prior standard deviations that $E(Y|x) = f(x)$ is away from $+/-.5$. The response ($y$) is internally scaled to the range from $-.5$ to $.5$. The bigger k is, the more conservative the fitting will be.

power: The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

base: The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth $d$ is base$^d$.

split.prob: A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

ntree: The number of trees in the ensemble.

ndpost: The number of posterior samples returned.

nskip: The number of posterior samples burned in.

keepevery: Every keepevery posterior sample is kept to be returned to the user.

printevery: As the MCMC runs, a message is printed every printevery iterations.
mc.abc.vs

verbose A Boolean argument indicating whether any messages are printed out.
mc.cores The number of cores to employ in parallel.
nice Set the job niceness. The default niceness is 19 and niceness goes from 0 (highest) to 19 (lowest).
seed Seed required for reproducible MCMC.

Details
At each iteration of the algorithm, the data set is randomly split into a training set and a testing set according to a certain split ratio. The algorithm proceeds by sampling a subset from the spike-and-slab prior on the model space, fitting a BART model on the training set only with the predictors in the subset, and computing the root mean squared errors (RMSE) for the test set based on a posterior sample from the fitted BART model. Only those subsets that result in a low RMSE on the test set are kept for selection. ABC Bayesian forest selects predictors based on their marginal posterior variable inclusion probabilities (MPVIPs) which are estimated by computing the proportion of ABC accepted BART posterior samples that use the predictor at least one time. Given the MPVIPs, predictors with MPVIP exceeding a pre-specified threshold are selected.
See Liu, Rockova and Wang (2021) or Section 2.2.4 in Luo and Daniels (2021) for details.

Value
The function mc.abc.vs() returns a list with the following components.

theta The probability that a predictor is included into a model.
models A matrix with nabc rows and ncol(x) columns; each row corresponds to a ABC model (or subset); if the (i,j)-th element is 1, it means that the j-th predictor is included in the i-th ABC model; if the (i,j)-th element is 0, it means that the j-th predictor is not included in the i-th ABC model.
actual.models A matrix with nabc rows and ncol(x) columns; each row corresponds to a ABC BART posterior sample; if the (i,j)-th element is 1, it means that the j-th predictor is used as a split variable at least one time in the BART posterior sample of the i-th ABC model; if the (i,j)-th element is 0, it means that the j-th predictor is not used as a split variable in the BART posterior sample of the i-th ABC model.
model.errors The vector of MSEs (or MLLs if the response variable is binary) for the nabc ABC models.
idx The vector of indices (in terms of the row numbers of models) of the ABC accepted models which are the top tolerance*100% of the nabc ABC models when ranked by MSE or MLL in ascending order.
top.models A matrix with length(idx) rows and ncol(x) columns, representing the ABC accepted models; top.models=models[idx, ].
top.actual.models A matrix with length(idx) rows and ncol(x) columns, representing the ABC accepted BART posterior samples; top.models=actual.models[idx, ].
mip The vector of marginal posterior variable inclusion probabilities.
best.model The vector of predictors selected by ABC Bayesian forest.
mc.abc.vs

precision  The precision score for the ABC Bayesian forest; only returned when true.idx is provided.
recall     The recall score for the ABC Bayesian forest; only returned when true.idx is provided.
f1         The F1 score for the ABC Bayesian forest; only returned when true.idx is provided.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

abc.vs.

Examples

## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## parallel::mparallel/mccollect do not exist on windows
if(.Platform$OS.type=='unix') {
  ## test mc.abc.vs() function
  res = mc.abc.vs(data$X, data$Y, nabc=100, tolerance=0.1, threshold=0.25,
                  beta.params=c(1.0, 1.0), split.ratio=0.5, probit=FALSE, true.idx=c(1,2,6:8),
                  ntree=10, ndpost=1, nskip=200, mc.cores=2)
mc.backward.vs  
*Backward selection with two filters (using parallel computation)*

**Description**

This function implements the backward variable selection approach for BART (see Algorithm 2 in Luo and Daniels (2021) for details). Parallel computation is used within each step of the backward selection approach.

**Usage**

```r
mc.backward.vs(
  x,
  y,
  split.ratio = 0.8,
  probit = FALSE,
  true.idx = NULL,
  xinfo = matrix(0, 0, 0),
  numcut = 100L,
  usequants = FALSE,
  cont = FALSE,
  rm.const = TRUE,
  k = 2,
  power = 2,
  base = 0.95,
  split.prob = "polynomial",
  ntree = 50L,
  ndpost = 1000,
  nskip = 1000,
  keepevery = 1L,
  printevery = 100L,
  verbose = FALSE,
  mc.cores = 2L,
  nice = 19L,
  seed = 99L
)
```

**Arguments**

- **x**
  A matrix or a data frame of predictors values with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with \( q \) levels in a data frame, it is replaced with \( q \) dummy variables.

- **y**
  A vector of response (continuous or binary) values.

- **split.ratio**
  A number between 0 and 1; the data set \((x, y)\) is split into a training set and a testing set according to the \( \text{split.ratio} \).
mc.backward.vs

probit A Boolean argument indicating whether the response variable is binary or continuous; probit=FALSE (by default) means that the response variable is continuous.

true.idx (Optional) A vector of indices of the true relevant predictors; if provided, metrics including precision, recall and F1 score are returned.

xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. xinfo=matrix(0.0,0,0) indicates the cut-points are specified by BART.

numcut The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to ncol(x) is required, where the i-th element gives the number of cut-points for the i-th predictor in x. If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of x. If usequants=TRUE, then min(numcut, the number of unique values in the corresponding column of x - 1) cut-point values are used.

usequants A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont A Boolean argument indicating whether to assume all predictors are continuous.

rm.const A Boolean argument indicating whether to remove constant predictors.

k The number of prior standard deviations that E(Y|x) = f(x) is away from +/- .5. The response (y) is internally scaled to the range from -.5 to .5. The bigger k is, the more conservative the fitting will be.

power The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

base The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth d is base^d.

split.prob A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

ntree The number of trees in the ensemble.

ndpost The number of posterior samples returned.

nskip The number of posterior samples burned in.

keepevery Every keepevery posterior sample is kept to be returned to the user.

printevery As the MCMC runs, a message is printed every printevery iterations.

verbose A Boolean argument indicating whether any messages are printed out.

mc.cores The number of cores to employ in parallel.

nice Set the job niceness. The default niceness is 19 and niceness goes from 0 (highest) to 19 (lowest).

seed Seed required for reproducible MCMC.
Details

The backward selection starts with the full model with all the predictors, followed by comparing the deletion of each predictor using mean squared error (MSE) if the response variable is continuous (or mean log loss (MLL) if the response variable is binary) and then deleting the predictor whose loss gives the smallest MSE (or MLL). This process is repeated until there is only one predictor in the model and ultimately returns ncol(x) "winner" models with different model sizes ranging from 1 to ncol(x).

Given the ncol(x) "winner" models, the one with the largest expected log pointwise predictive density based on leave-one-out (LOO) cross validation is the best model. See Section 3.3 in Luo and Daniels (2021) for details.

If true.idx is provided, the precision, recall and F1 scores are returned.

Value

The function mc.backward.vs() returns a list with the following components.

best.model.names

The vector of column names of the predictors selected by the backward selection approach.

best.model.cols

The vector of column indices of the predictors selected by the backward selection approach.

best.model.order

The step where the best model is located.

models

The list of winner models from each step of the backward selection procedure; length equals ncol(x).

model.errors

The vector of MSEs (or MLLs if the response variable is binary) for the ncol(x) winner models.

elpd.loos

The vector of LOO scores for the ncol(x) winner models.

all.models

The list of all the evaluated models.

all.model.errors

The vector of MSEs (or MLLs if the response variable is binary) for all the evaluated models.

precision

The precision score for the backward selection approach; only returned if true.idx is provided.

recall

The recall score for the backward selection approach; only returned if true.idx is provided.

f1

The F1 score for the backward selection approach; only returned if true.idx is provided.

all.models.idx

The vector of Boolean arguments indicating whether the corresponding model in all.models is acceptable or not; a model containing all the relevant predictors is an acceptable model; only returned if true.idx is provided.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.
mc.cores.openmp

References


See Also

`permute.vs`, `medianInclusion.vs` and `abc.vs`.

Examples

```r
## simulate data (Scenario C.C.1. in Luo and Daniels (2021))
set.seed(123)
data = friedman(100, 5, 1, FALSE)
## parallel::mcparallel/mccollect do not exist on windows
if(.Platform$OS.type=='unix') {
  ## test mc.backward.vs() function
  res = mc.backward.vs(data$X, data$Y, split.ratio=0.8, probit=FALSE,
                        true.idx=c(1:5), ntree=10, ndpost=100, nskip=100, mc.cores=2)
}
```

mc.cores.openmp Detecting OpenMP

Description

This function is inherited from the CRAN R package ‘BART’ and was designed for OpenMP. For example, the `pwbart` function can use OpenMP or the ‘parallel’ R package for multi-threading. On UNIX/Unix-like systems, OpenMP, if available, is discovered at install time. However, we know of no GPL licensed code available to detect OpenMP on Windows (for Artistic licensed OpenMP detection code on Windows, see the Bioconductor R package ‘rGADEM’). To determine whether OpenMP is available at run time, we provide the function documented here.

Usage

`mc.cores.openmp()`

Value

This function returns 0 when OpenMP is not available; otherwise, an integer greater than 0 is returned when OpenMP is available (1 is returned unless you are running in a multi-threaded process).
Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

`pwbart`.

Examples

```r
mc.cores.openmp()
```

---

**mc.pbart**

Probit BART for binary responses with parallel computation

Description

BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.

For a binary response \( y \) and a \( p \)-dimensional vector of predictors \( x = (x_1, ..., x_p)' \), probit BART models \( y \) and \( x \) using

\[
P(Y = 1|x) = \Phi[f(x)],
\]

where \( \Phi \) is the CDF of the standard normal distribution and \( f \) is a sum of Bayesian regression trees function.

The function `mc.pbart()` is inherited from the CRAN R package `BART` and is a variant of the function `pbart()` with parallel computation.
Usage

\texttt{mc.pbart(}
\texttt{x.train,}
\texttt{y.train,}
\texttt{x.test = matrix(0, 0L, 0L),}
\texttt{sparse = FALSE,}
\texttt{theta = 0,}
\texttt{omega = 1,}
\texttt{a = 0.5,}
\texttt{b = 1,}
\texttt{augment = FALSE,}
\texttt{rho = NULL,}
\texttt{xinfo = matrix(0, 0, 0),}
\texttt{numcut = 100L,}
\texttt{usequants = FALSE,}
\texttt{cont = FALSE,}
\texttt{rm.const = TRUE,}
\texttt{k = 2,}
\texttt{power = 2,}
\texttt{base = 0.95,}
\texttt{split.prob = "polynomial",}
\texttt{binaryOffset = NULL,}
\texttt{ntree = 50L,}
\texttt{ndpost = 1000L,}
\texttt{nskip = 100L,}
\texttt{keepevery = 1L,}
\texttt{printevery = 100L,}
\texttt{keeptrainfits = TRUE,}
\texttt{transposed = FALSE,}
\texttt{verbose = FALSE,}
\texttt{mc.cores = 2L,}
\texttt{nice = 19L,}
\texttt{seed = 99L}
\texttt{)}

Arguments

\texttt{x.train} \hspace{1cm} A matrix or a data frame of predictors values (for training) with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with \( q \) levels in a data frame, it is replaced with \( q \) dummy variables.

\texttt{y.train} \hspace{1cm} A vector of continuous response values for training.

\texttt{x.test} \hspace{1cm} A matrix or a data frame of predictors values for testing, which has the same structure as \texttt{x.train}.

\texttt{sparse} \hspace{1cm} A Boolean argument indicating whether to replace the discrete uniform distribution for selecting a split variable with a categorical distribution whose event probabilities follow a Dirichlet distribution (see Linero (2018) for details).
theta  Set theta parameter; zero means random.
omega  Set omega parameter; zero means random.
a    A sparse parameter of Beta(a, b) hyper-prior where 0.5 <= a <= 1; a lower value induces more sparsity.
b    A sparse parameter of Beta(a, b) hyper-prior; typically, b = 1.
augment  A Boolean argument indicating whether data augmentation is performed in the variable selection procedure of Linero (2018).
rho  A sparse parameter; typically $\rho = p$ where $p$ is the number of predictors.
xinfo  A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. xinfo=matrix(0.0,0,0) indicates the cut-points are specified by BART.
numcut  The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to ncol(x.train) is required, where the $i$-th element gives the number of cut-points for the $i$-th predictor in x.train. If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of x.train. If usequants=TRUE, then min(numcut, the number of unique values in the corresponding column of x.train - 1) cut-point values are used.
usequants  A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.
cont  A Boolean argument indicating whether to assume all predictors are continuous.
rm.const  A Boolean argument indicating whether to remove constant predictors.
k    The number of prior standard deviations that $E(Y|x) = f(x)$ is away from $+/- .5$. The response (y.train) is internally scaled to the range from $-.5$ to $.5$. The bigger $k$ is, the more conservative the fitting will be.
power  The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".
base  The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth $d$ is base$^d$.
split.prob  A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.
binaryOffset  The binary offset term in the probit BART model, i.e., $P(Y = 1|x) = \Phi[f(x) + binaryOffset]$.
ntree  The number of trees in the ensemble.
ndpost  The number of posterior samples returned.
nskip  The number of posterior samples burned in.
keepevery  Every keepevery posterior sample is kept to be returned to the user.
printevery  As the MCMC runs, a message is printed every printevery iterations.
mc.pbart

keeptrainfits A Boolean argument indicating whether to keep yhat.train or not.
transposed A Boolean argument indicating whether the matrices x.train and x.test are transposed.
verbose A Boolean argument indicating whether any messages are printed out.
mc.cores The number of cores to employ in parallel.
nice Set the job niceness. The default niceness is 19 and niceness goes from 0 (highest) to 19 (lowest).
seed Seed required for reproducible MCMC.

Details
This function is inherited from BART::mc.pbart() and is a variant of the function pbart() with parallel computation. While the original features of BART::pbart() are preserved, two modifications are made.

The first modification is to provide two types of split probability for BART. One split probability is proposed in Chipman et al. (2010) and defined as

\[ p(d) = \gamma \times (1 + d)^{-\beta}, \]

where \( d \) is the depth of the node, \( \gamma \in (0, 1) \) and \( \beta \in (0, \infty) \). The other split probability is proposed by Rockova and Saha (2019) and defined as

\[ p(d) = \gamma^d, \]

where \( \gamma \in (1/n, 1/2) \). BART with the second split probability is proved to achieve the optimal posterior contraction.

The second modification is to provide five types of variable importance measures (vip, within.type.vip, pvip, varprob.mean and mi) in the return object, for the sake of the existence of mixed-type predictors.

Value
The function pbart() returns an object of type pbart which essentially is a list consisting of the following components.

yhat.train A matrix with ndpost rows and nrow(x.train) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a training data point. The \((i,j)\)-th element of the matrix is \( f^* (x) + binaryOffset \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th training data point. Burn-in posterior samples are dropped.

prob.train A matrix with the same structure as yhat.train and the \((i,j)\)-th element of prob.train is \( \Phi[f^* (x) + binaryOffset] \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th training data point.

prob.train.mean A vector which is colMeans(prob.train).

yhat.test A matrix with ndpost rows and nrow(x.test) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a test data point. The \((i,j)\)-th element of the matrix is \( f^* (x) + binaryOffset \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th test data point. Burn-in posterior samples are dropped.
prob.test  A matrix with the same structure as yhat.test and the \((i, j)\)-th element of prob.test is \(\Phi[f^* (x) + binaryOffset]\) for the \(i\)-th kept draw of \(f\) and the \(j\)-th test data point.

prob.test.mean  A vector which is colMeans(prob.test).

varcount  A matrix with \(\text{ndpost}\) rows and \(n\text{col}(x.\text{train})\) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i, j)\)-th element is the number of times that the \(j\)-th predictor is used as a split variable in the \(i\)-th posterior sample.

varprob  A matrix with \(\text{ndpost}\) rows and \(n\text{col}(x.\text{train})\) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i, j)\)-th element is the split probability of the \(j\)-th predictor in the \(i\)-th posterior sample. Only useful when DART is fit, i.e., \(\text{sparse=TRUE}\).

treedraws  A list containing the posterior samples of the ensembles (trees structures, split variables and split values); Can be used for prediction.

proc.time  The process time of running the function \text{wbart()}.

vip  A vector of variable inclusion proportions (VIP) proposed in Chipman et al. (2010).

within.type.vip  A vector of within-type VIPs proposed in Luo and Daniels (2021).

pvip  A vector of marginal posterior variable inclusion probabilities (PVIP) proposed in Linero (2018); Only useful when DART is fit, i.e., \(\text{sparse=TRUE}\).

varprob.mean  A vector of posterior split probabilities (PSP) proposed in Linero (2018); Only useful when DART is fit, i.e., \(\text{sparse=TRUE}\).

mr.vecs  A list of \(n\text{col}(x.\text{train})\) sub-lists with each corresponding to a predictor; Each sub-list contains \(\text{ndpost}\) vectors with each vector containing the (birth) Metropolis ratios for splits using the predictor as the split variable in that posterior sample.

mr.mean  A matrix with \(\text{ndpost}\) rows and \(n\text{col}(x.\text{train})\) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i, j)\)-th element is the average Metropolis acceptance ratio per splitting rule using the \(j\)-th predictor in the \(i\)-th posterior sample.

mi  A vector of Metropolis importance (MI) proposed in Luo and Daniels (2021).

rm.const  A vector of indicators for the predictors (after dummification) used in BART; when the indicator is negative, it refers to remove that predictor.

binaryOffset  The binary offset term used in BART.

ndpost  The number of posterior samples returned.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.
mc.permute.vs

References


See Also

pbart.

Examples

```r
## simulate data (Scenario B.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, TRUE)
## parallel::mcmclapply does not exist on windows
if(.Platform$OS.type=="unix") {
  ## test mc.pbart() function
  res = mc.pbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100, mc.cores=2)
}
```

mc.permute.vs

*Permutation-based variable selection approach with parallel computation*

Description

This function implements the permutation-based variable selection approach for BART (see Algorithm 1 in Luo and Daniels (2021) for details) with parallel computation used in computing the null variable importance scores. Three types of variable importance measures are considered: BART variable inclusion proportions (VIP), BART within-type variable inclusion proportions (within-type VIP) and BART Metropolis Importance (MI).

The permutation-based variable selection approach using BART VIP as the variable importance measure is proposed by Bleich et al. (2014). BART within-type VIP and BART MI are proposed by Luo and Daniels (2021), for the sake of the existence of mixed-type predictors and the goal of allowing more relevant predictors into the model.
Usage

mc.permute.vs(
  x.train,  # A matrix or a data frame of predictors values with each row corresponding to
  y.train,  # an observation and each column corresponding to a predictor. If a predictor is a
  probit = FALSE,  # factor with q levels in a data frame, it is replaced with q dummy variables.
  npermute = 100L,  # A vector of response (continuous or binary) values.
  nreps = 10L,  # A Boolean argument indicating whether the response variable is binary or con-
  alpha = 0.05,  # tinuous; probit=FALSE (by default) means that the response variable is contin-
  true.idx = NULL,  # nuous.
  plot = TRUE,  # The number of permutations for estimating the null distributions of the variable
  n.var.plot = Inf,  # importance scores.
  xinfo = matrix(0, 0, 0),  # The number of replications for obtaining the averaged (or median) variable
  numcut = 100L,  # importance scores based on the original data set.
  usequants = FALSE,  # A number between 0 and 1; a predictor is selected if its averaged (or median)
  cont = FALSE,  # variable importance score exceeds the $1 - \alpha$ quantile of the corresponding null
  rm.const = TRUE,  # distribution.
  k = 2,  # A matrix or a data frame of predictors values with each row corresponding to
  power = 2,  # an observation and each column corresponding to a predictor. If a predictor is a
  base = 0.95,  # factor with q levels in a data frame, it is replaced with q dummy variables.
  split.prob = "polynomial",  # A vector of response (continuous or binary) values.
  ntree = 20L,  # A Boolean argument indicating whether the response variable is binary or con-
  ndpost = 1000,  # tinuous; probit=FALSE (by default) means that the response variable is contin-
  nskip = 1000,  # nuous.
  keepevery = 1L,  # The number of permutations for estimating the null distributions of the variable
  verbose = FALSE,  # importance scores.
  mc.cores = 2L,  # The number of replications for obtaining the averaged (or median) variable
  nice = 19L,  # importance scores based on the original data set.
  seed = 99L)

Arguments

x.train A matrix or a data frame of predictors values with each row corresponding to
an observation and each column corresponding to a predictor. If a predictor is a
factor with q levels in a data frame, it is replaced with q dummy variables.

y.train A vector of response (continuous or binary) values.

probit A Boolean argument indicating whether the response variable is binary or con-
tinuous; probit=FALSE (by default) means that the response variable is contin-
uous.

npermute The number of permutations for estimating the null distributions of the variable
importance scores.

nreps The number of replications for obtaining the averaged (or median) variable
importance scores based on the original data set.

alpha A number between 0 and 1; a predictor is selected if its averaged (or median)
variable importance score exceeds the $1 - \alpha$ quantile of the corresponding null
distribution.
true.idx  (Optional) A vector of indices of the true relevant predictors; if provided, metrics including precision, recall and F1 score will be returned.

plot  (Optional) A Boolean argument indicating whether plots are returned or not.

n.var.plot  The number of variables to be plotted.

xinfo  A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. xinfo=matrix(0,0,0) indicates the cut-points are specified by BART.

numcut  The number of possible cut-points: If a single number is given, this is used for all predictors; Otherwise a vector with length equal to ncol(x.train) is required, where the i-th element gives the number of cut-points for the i-th predictor in x.train. If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of x.train. If usequants=TRUE, then min(numcut, the number of unique values in the corresponding column of x.train - 1) cut-point values are used.

usequants  A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont  A Boolean argument indicating whether to assume all predictors are continuous.

rm.const  A Boolean argument indicating whether to remove constant predictors.

k  The number of prior standard deviations that $E(Y|x) = f(x)$ is away from $+/-.5$. The response (y.train) is internally scaled to the range from $-.5$ to $.5$. The bigger k is, the more conservative the fitting will be.

power  The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

base  The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth $d$ is base$^d$.

split.prob  A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

ntree  The number of trees in the ensemble.

ndpost  The number of posterior samples returned.

nskip  The number of posterior samples burned in.

keepevery  Every keepevery posterior sample is kept to be returned to the user.

verbose  A Boolean argument indicating whether any messages are printed out.

mc.cores  The number of cores to employ in parallel.

nice  Set the job niceness. The default niceness is 19 and niceness goes from 0 (highest) to 19 (lowest).

seed  Seed required for reproducible MCMC.
Details

The detailed algorithm can be found in Algorithm 1 in Luo and Daniels (2021). The permutation-based variable selection approach using within-type VIP as the variable importance measure is only used when the predictors are of mixed-type; otherwise, it is the same as the one using VIP as the variable importance measure.

If `true.idx` is provided, the precision, recall and F1 scores will be returned for the three (or two if the predictors are of the same type) methods.

If `plot=TRUE`, three (or two if the predictors are of the same type) plots showing which predictors are selected are generated.

Value

The function `mc.permute.vs()` returns three (or two if the predictors are of the same type) plots if `plot=TRUE` and a list with the following components.

- **vip.imp.cols**
  - The vector of column indices of the predictors selected by the approach using VIP as the variable importance score.

- **vip.imp.names**
  - The vector of column names of the predictors selected by the approach using VIP as the variable importance score.

- **avg.vip**
  - The vector (length=`ncol(x.train)`) of the averaged VIPs based on the original data set; `avg.vip=colMeans(avg.vip.mtx)`.

- **avg.vip.mtx**
  - A matrix of VIPs based on the original data set, with each row corresponding to a repetition and each column corresponding to a predictor.

- **permute.vips**
  - A matrix of VIPs based on the null data sets, with each row corresponding to a permutation (null data set) and each column corresponding to a predictor.

- **within.type.vip.imp.cols**
  - The vector of column indices of the predictors selected by the approach using within-type VIP as the variable importance score.

- **within.type.vip.imp.names**
  - The vector of column names of the predictors selected by the approach using within-type VIP as the variable importance score.

- **avg.within.type.vip**
  - The vector (length=`ncol(x.train)`) of the averaged within-type VIPs based on the original data set; `avg.within.type.vip=colMeans(avg.within.type.vip.mtx)`.

- **avg.within.type.vip.mtx**
  - A matrix of within-type VIPs based on the original data set, with each row corresponding to a repetition and each column corresponding to a predictor.

- **permute.within.type.vips**
  - A matrix of within VIPs based on the null data sets, with each row corresponding to a permutation (null data set) and each column corresponding to a predictor.

- **mi.imp.cols**
  - The vector of column indices of the predictors selected by the approach using MI as the variable importance score.

- **mi.imp.names**
  - The vector of column names of the predictors selected by the approach using MI as the variable importance score.

- **median.mi**
  - The vector (length=`ncol(x.train)`) of the median MIs based on the original data set; `median.mi=colMeans(median.mi.mtx)`.
mc.permute.vs

median.mi.mtx  A matrix of MIs based on the original data set, with each row corresponding to a repetition and each column corresponding to a predictor.

permute.mis  A matrix of MIs based on the null data sets, with each row corresponding to a permutation (null data set) and each column corresponding to a predictor.

true.idx  A vector of indices of the true relevant predictors; only returned if true.idx is provided as inputs.

vip.precision  The precision score for the approach using VIP as the variable importance score; only returned if true.idx is provided.

vip.recall  The recall score for the approach using VIP as the variable importance score; only returned if true.idx is provided.

vip.f1  The F1 score for the approach using VIP as the variable importance score; only returned if true.idx is provided.

wt.vip.precision  The precision score for the approach using within-VIP as the variable importance score; only returned when the predictors are of the same type and true.idx is provided.

wt.vip.recall  The recall score for the approach using within-VIP as the variable importance score; only returned when the predictors are of the same type and true.idx is provided.

wt.vip.f1  The F1 score for the approach using within-VIP as the variable importance score; only returned when the predictors are of the same type and true.idx is provided.

mi.precision  The precision score for the approach using MI as the variable importance score; only returned if true.idx is provided.

mi.recall  The recall score for the approach using MI as the variable importance score; only returned if true.idx is provided.

mi.f1  The F1 score for the approach using MI as the variable importance score; only returned if true.idx is provided.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


mc.pwbart

Predicting new observations based on a previously fitted BART model with parallel computation

Description

BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.
For a continuous response $y$ and a $p$-dimensional vector of predictors $x = (x_1, ..., x_p)'$, BART models $y$ and $x$ using

$$ y = f(x) + \epsilon, $$

where $f$ is a sum of Bayesian regression trees function and $\epsilon \sim N(0, \sigma^2)$.
For a binary response $y$, probit BART models $y$ and $x$ using

$$ P(Y = 1|x) = \Phi[f(x)], $$

where $\Phi$ is the CDF of the standard normal distribution and $f$ is a sum of Bayesian regression trees function.
The function `mc.pwbart()` is inherited from the CRAN R package 'BART'.

Usage

```r
mc.pwbart(
  x.test,
  treedraws,
  rm.const,
  mu = 0,
  mc.cores = 2L,
  transposed = FALSE,
  dodraws = TRUE,
  nice = 19L
)
```

See Also

`permute.vs`, `medianInclusion.vs`, `mc.backward.vs` and `abc.vs`. 

Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## parallel::mcparallel/mccollect do not exist on windows
if(.Platform$OS.type=="unix") {
  ## test mc.permute.vs() function
  res = mc.permute.vs(data$X, data$Y, probit=FALSE, npermute=100, nreps=10, alpha=0.05,
    true.idx=c(1, 2, 6:8), plot=FALSE, ntree=10, ndpost=100, nskip=100, mc.cores=2)
}
```
Arguments

- **x.test**: A matrix or a data frame of predictors values for prediction with each row corresponding to an observation and each column corresponding to a predictor.

- **treedraws**: A list which is the $treedraws$ returned from the function `wbart()` or `pbart()`.

- **rm.const**: A vector which is the $rm.const$ returned from the function `wbart()` or `pbart()`.

- **mu**: Mean to add on to \( y \) prediction.

- **mc.cores**: The number of threads to utilize.

- **transposed**: A Boolean argument indicating whether the matrix `x.test` is transposed. When running `pwbart()` or `mc.pwbart()` in parallel, it is more memory-efficient to transpose `x.test` prior to calling the internal versions of these functions.

- **dodraws**: A Boolean argument indicating whether to return the draws themselves (the default), or whether to return the mean of the draws as specified by `dodraws=FALSE`.

- **nice**: Set the job niceness. The default niceness is 19 and niceness goes from 0 (highest) to 19 (lowest).

Value

Returns the predictions for `x.test`. If `dodraws=TRUE`, return a matrix of prediction with each row corresponding to a draw and each column corresponding to a new observation; if `dodraws=FALSE`, return a vector of predictions which are the mean of the draws.

Author(s)

Chuji Luo: `<cjluo@ufl.edu>` and Michael J. Daniels: `<daniels@ufl.edu>`.

References


See Also

- `wbart`, `pbart` and `mc.pwbart`. 
Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## run wbart() function
res = wbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100)
## parallel::mcparallel/mccollect do not exist on windows
if(.Platform$OS.type=='unix') {
## test pwbart() function
x.test = mixone(5, 10, 1, FALSE)$X
pred = mc.pwbart(x.test, res$treedraws, res$rm.const, mu=mean(data$Y), mc.cores=2)
}
```

mc.wbart  

BART for continuous responses with parallel computation

Description

BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.

For a continuous response \( y \) and a \( p \)-dimensional vector of predictors \( x = (x_1, ..., x_p)' \), BART models \( y \) and \( x \) using

\[
y = f(x) + \epsilon,
\]

where \( f \) is a sum of Bayesian regression trees function and \( \epsilon \sim N(0, \sigma^2) \).

The function `mc.wbart()` is inherited from the CRAN R package `BART` and is a variant of the function `wbart()` with parallel computation.

Usage

```r
mc.wbart(
x.train,  
y.train,  
x.test = matrix(0, 0, 0),  
sparse = FALSE,  
theta = 0,  
omega = 1,  
a = 0.5,  
b = 1,  
 augment = FALSE,  
rho = NULL,  
xinfo = matrix(0, 0, 0),  
numcut = 100L,  
usequants = FALSE,  
cont = FALSE,  
rm.const = TRUE,  
power = 2,
```
mc.wbart

base = 0.95,
split.prob = "polynomial",
k = 2,
sigmaf = NA,
sigest = NA,
sigdf = 3,
sigquant = 0.9,
lambda = NA,
fmean = mean(y.train),
w = rep(1, length(y.train)),
ntree = 200L,
ndpost = 1000L,
nskip = 100L,
keepevery = 1L,
printevery = 100L,
keeptrainfits = TRUE,
transposed = FALSE,
verbose = FALSE,
mc.cores = 2L,
nice = 19L,
seed = 99L
)

Arguments

x.train A matrix or a data frame of predictors values (for training) with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with q levels in a data frame, it is replaced with q dummy variables.

y.train A vector of continuous response values for training.

x.test A matrix or a data frame of predictors values for testing, which has the same structure as x.train.

sparse A Boolean argument indicating whether to replace the discrete uniform distribution for selecting a split variable with a categorical distribution whose event probabilities follow a Dirichlet distribution (see Linero (2018) for details).

theta Set theta parameter; zero means random.

omega Set omega parameter; zero means random.

a A sparse parameter of Beta(a, b) hyper-prior where 0.5 <= a <= 1; a lower value induces more sparsity.

b A sparse parameter of Beta(a, b) hyper-prior; typically, b = 1.

augment A Boolean argument indicating whether data augmentation is performed in the variable selection procedure of Linero (2018).

rho A sparse parameter; typically ρ = p where p is the number of predictors.

xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. xinfo=matrix(0,0,0) indicates the cut-points are specified by BART.
numcut
The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to ncol(x.train) is required, where the i-th element gives the number of cut-points for the i-th predictor in x.train. If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of x.train. If usequants=TRUE, then min(numcut, the number of unique values in the corresponding column of x.train - 1) cut-point values are used.

usequants
A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont
A Boolean argument indicating whether to assume all predictors are continuous.

rm.const
A Boolean argument indicating whether to remove constant predictors.

power
The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

base
The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth d is base^d.

split.prob
A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

k
The number of prior standard deviations that E(Y|x) = f(x) is away from +/- .5. The response (y.train) is internally scaled to the range from -.5 to .5. The bigger k is, the more conservative the fitting will be.

sigf
The standard deviation of f.

sigest
A rough estimate of the error standard deviation, the square of which follows an inverse chi-squared prior. If sigest=NA, the rough estimate will be the usual least square estimator; Otherwise, the supplied value will be used.

sigdf
The degrees of freedom for the error variance prior.

sigquant
The quantile of the error variance prior, where sigest is placed. The closer the quantile is to 1, the more aggressive the fit will be.

lambda
The scale parameter of the error variance prior.

fmean
BART operates on y.train centered by fmean.

w
A vector of weights which multiply the standard deviation.

ntree
The number of trees in the ensemble.

ndpost
The number of posterior samples returned.

nskip
The number of posterior samples burned in.

keepevery
Every keepevery posterior sample is kept to be returned to the user.

printevery
As the MCMC runs, a message is printed every printevery iterations.

keeptrainfits
A Boolean argument indicating whether to keep yhat.train or not.

transposed
A Boolean argument indicating whether the matrices x.train and x.test are transposed.
mc.wbart

verbose  A Boolean argument indicating whether any messages are printed out.
mc.cores  The number of cores to employ in parallel.
nice  Set the job niceness. The default niceness is 19 and niceness goes from 0 (highest) to 19 (lowest).
seed  Seed required for reproducible MCMC.

Details

This function is inherited from BART::mc.wbart() and is a variant of the function wbart() with parallel computation.
While the original features of BART::wbart() are preserved, two modifications are made.
The first modification is to provide two types of split probability for BART. One split probability is proposed in Chipman et al. (2010) and defined as
\[ p(d) = \gamma \ast (1 + d)^{-\beta}, \]
where \( d \) is the depth of the node, \( \gamma \in (0, 1) \) and \( \beta \in (0, \infty) \). The other split probability is proposed by Rockova and Saha (2019) and defined as
\[ p(d) = \gamma^d, \]
where \( \gamma \in (1/n, 1/2) \). BART with the second split probability is proved to achieve the optimal posterior contraction.
The second modification is to provide five types of variable importance measures (vip, within.type.vip, pvip, varprob.mean and mi) in the return object, for the sake of the existence of mixed-type predictors.

Value

The function mc.wbart() returns an object of type wbart which essentially is a list consisting of the following components.
sigma  A vector with \( nskip \ast ndpost \ast keepevery \) posterior samples of \( \sigma \).
yhat.train.mean  \text{colMeans(yhat.train)}. 
yhat.train  A matrix with \( ndpost \) rows and \( nrow(x.train) \) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a training data point. The \((i,j)\)-th element of the matrix is \( f^*(x) \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th training data point. Burn-in posterior samples are dropped.
yhat.test.mean  \text{colMeans(yhat.test)}. 
yhat.test  A matrix with \( ndpost \) rows and \( nrow(x.test) \) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a test data point. The \((i,j)\)-th element of the matrix is \( f^*(x) \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th test data point. Burn-in posterior samples are dropped.
varcount  A matrix with \( ndpost \) rows and \( ncol(x.train) \) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i,j)\)-th element is the number of times that the \( j \)-th predictor is used as a split variable in the \( i \)-th posterior sample.
varprob  A matrix with ndpost rows and ncol(x.train) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i,j)\)-th element is the split probability of the \(j\)-th predictor in the \(i\)-th posterior sample. Only useful when DART is fit, i.e., sparse=TRUE.

treedraws  A list containing the posterior samples of the ensembles (trees structures, split variables and split values); Can be used for prediction.

proc.time  The process time of running the function wbart().

mu  BART operates on y.train centered by fmean.

mr.vecs  A list of ncol(x.train) sub-lists with each corresponding to a predictor; Each sub-list contains ndpost vectors with each vector containing the (birth) Metropolis ratios for splits using the predictor as the split variable in that posterior sample.

vip  A vector of variable inclusion proportions (VIP) proposed in Chipman et al. (2010).

within.type.vip  A vector of within-type VIPs proposed in Luo and Daniels (2021).

pvip  A vector of marginal posterior variable inclusion probabilities (PVIP) proposed in Linero (2018); Only useful when DART is fit, i.e., sparse=TRUE.

varprob.mean  A vector of posterior split probabilities (PSP) proposed in Linero (2018); Only useful when DART is fit, i.e., sparse=TRUE.

mr.mean  A matrix with ndpost rows and ncol(x.train) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i,j)\)-th element is the average Metropolis acceptance ratio per splitting rule using the \(j\)-th predictor in the \(i\)-th posterior sample.

mi  A vector of Metropolis importance (MI) proposed in Luo and Daniels (2021).

rm.const  A vector of indicators for the predictors (after dummification) used in BART; when the indicator is negative, it refers to remove that predictor.

ndpost  The number of posterior samples returned.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


### medianInclusion.vs

**Variable selection with DART**

#### Description

This function implements the variable selection approach proposed in Linero (2018). Linero (2018) proposes DART, a variant of BART, which replaces the discrete uniform distribution for selecting a split variable with a categorical distribution of which the event probabilities follow a Dirichlet distribution. DART estimates the marginal posterior variable inclusion probability (MPVIP) for a predictor by the proportion of the posterior samples of the trees structures where the predictor is used as a split variable at least once, and selects predictors with MPVIP at least 0.5, yielding a median probability model.

#### Usage

```r
medianInclusion.vs(x.train, y.train, probit = FALSE, vip.selection = TRUE, true.idx = NULL, plot = FALSE, num.var.plot = Inf, theta = 0, omega = 1, a = 0.5, b = 1, augment = FALSE, rho = NULL, xinfo = matrix(0, 0, 0), numcut = 100L, usequants = FALSE, cont = FALSE)
```

#### Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## parallel::mcparallel/mccollect do not exist on windows
if(.Platform$OS.type=='unix') {
  ## test mc.wbart() function
  res = mc.wbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100, mc.cores=2)
}
```
rm.const = TRUE,
power = 2,
base = 0.95,
split.prob = "polynomial",
k = 2,
ntree = 20L,
ndpost = 1000L,
nskip = 1000L,
keepevery = 1L,
printhevery = 100L,
verbose = FALSE)
)

Arguments

x.train A matrix or a data frame of predictors values with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with \( q \) levels in a data frame, it is replaced with \( q \) dummy variables.

y.train A vector of response (continuous or binary) values.

probit A Boolean argument indicating whether the response variable is binary or continuous; probit=FALSE (by default) means that the response variable is continuous.

vip.selection A Boolean argument indicating whether to select predictors using BART VIPs.

true.idx (Optional) A vector of indices of the true relevant predictors; if provided, metrics including precision, recall and F1 score are returned.

plot (Optional) A Boolean argument indicating whether plots are returned or not.

num.var.plot The number of variables to be plotted.

theta Set \( \theta \) parameter; zero means random.

omega Set \( \omega \) parameter; zero means random.

\( a \) A sparse parameter of \( \text{Beta}(a,b) \) hyper-prior where \( 0.5 < a < 1 \); a lower value induces more sparsity.

\( b \) A sparse parameter of \( \text{Beta}(a,b) \) hyper-prior; typically, \( b = 1 \).

augment A Boolean argument indicating whether data augmentation is performed in the variable selection procedure of Linero (2018).

\( \rho \) A sparse parameter; typically \( \rho = p \) where \( p \) is the number of predictors.

xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. \( xinfo=\text{matrix}(0.0,0,0) \) indicates the cut-points are specified by BART.

numcut The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to \( \text{ncol}(x.train) \) is required, where the \( i \)-th element gives the number of cut-points for the \( i \)-th predictor in \( x.train \). If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of \( x.train \). If usequants=TRUE, then \( \min(\text{numcut}, \text{the number of unique values in the corresponding column of } x.train - 1) \) cut-point values are used.
usequants A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont A Boolean argument indicating whether to assume all predictors are continuous.

rm.const A Boolean argument indicating whether to remove constant predictors.

power The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

base The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth \( d \) is \( \text{base}^d \).

split.prob A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

k The number of prior standard deviations that \( E(Y|x) = f(x) \) is away from \(+/- .5\). The response (y.train) is internally scaled to the range from \(-.5\) to \(.5\). The bigger \( k \) is, the more conservative the fitting will be.

ntree The number of trees in the ensemble.

ndpost The number of posterior samples returned.

nskip The number of posterior samples burned in.

keepevery Every keepevery posterior sample is kept to be returned to the user.

printevery As the MCMC runs, a message is printed every printevery iterations.

verbose A Boolean argument indicating whether any messages are printed out.

Details
See Linero (2018) or Section 2.2.3 in Luo and Daniels (2021) for details.

If vip.selection=TRUE, this function also does variable selection by selecting variables whose BART VIP exceeds 1/ncol(x.train).

If true.idx is provided, the precision, recall and F1 scores are returned.

If plot=TRUE, plots showing which predictors are selected are generated.

Value
The function medianInclusion.vs() returns two (or one if vip.selection=FALSE) plots if plot=TRUE and a list with the following components.

dart.pvip The vector of DART MPVIPs.

dart.pvip.imp.names The vector of column names of the predictors with DART MPVIP at least 0.5.

dart.pvip.imp.cols The vector of column indices of the predictors with DART MPVIP at least 0.5.

dart.precision The precision score for the DART approach; only returned if true.idx is provided.

dart.recall The recall score for the DART approach; only returned if true.idx is provided.
medianInclusion.vs

dart.f1  The F1 score for the DART approach; only returned if true.idx is provided.
bart.vip  The vector of BART VIPs; only returned if vip.selection=TRUE.
bart.vip.imp.names  The vector of column names of the predictors with BART VIP exceeding 1/ncol(x.train); only returned if vip.selection=TRUE.
bart.vip.imp.cols  The vector of column indices of the predictors with BART VIP exceeding 1/ncol(x.train); only returned if vip.selection=TRUE.
bart.precision  The precision score for the BART approach; only returned if vip.selection=TRUE and true.idx is provided.
bartrecall  The recall score for the BART approach; only returned if vip.selection=TRUE and true.idx is provided.
bart.f1  The F1 score for the BART approach; only returned if vip.selection=TRUE and true.idx is provided.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

permute.vs, mc.backward.vs and abc.vs.

Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## test medianInclusion.vs() function
res = medianInclusion.vs(data[,1], data[,2], probit=FALSE, vip.selection=TRUE, true.idx=c(1, 2, 6:8), plot=FALSE, ntree=10, ndpost=100, nskip=100, verbose=FALSE)
```
**mixone**

*Generate data with independent and mixed-type predictors*

## Description
Generate data including responses and predictors values, of which predictors are independent and of mixed types.

## Usage

```r
mixone(n, p, sigma, binary)
```

## Arguments

- `n` The number of observations.
- `p` The number of predictors.
- `sigma` The error variance.
- `binary` A boolean argument: `binary = TRUE` indicates that binary responses are generated and `binary = FALSE` indicates that continuous responses are generated.

## Details
Sample the predictors $x_1, \ldots, x_{\text{ceiling}(p/2)}$ from Bernoulli(0.5) independently and $x_{\text{ceiling}(p/2)+1}, \ldots, x_p$ from Uniform(0, 1) independently. If `binary = FALSE`, sample the continuous response $y$ from Normal($f_0(x), \sigma^2$), where

$$f_0(x) = 10sin(\pi x_{\text{ceiling}(p/2) + 1} * x_{\text{ceiling}(p/2) + 2}) + 20(x_{\text{ceiling}(p/2) + 3} - 0.5)^2 + 10x_1 + 5x_2.$$  

If `binary = TRUE`, sample the binary response $y$ from Bernoulli($\Phi(f_0(x))$) where $f_0$ is defined above and $\Phi$ is the cumulative density function of the standard normal distribution.

## Value
Return a list with the following components.

- `X` An n by p data frame representing predictors values, with each row corresponding an observation.
- `Y` A vector of length n representing response values.
- `f0` A vector of length n representing the values of $f_0(x)$.
- `sigma` The error variance which is only returned when `binary = FALSE`.
- `prob` A vector of length n representing the values of $\Phi(f_0(x))$, which is only returned when `binary = TRUE`.

## Author(s)

Chuji Luo: `<cjluo@ufl.edu>` and Michael J. Daniels: `<daniels@ufl.edu>`.
References


Examples

data = mixone(100, 10, 1, FALSE)

mixtwo

Generate data with correlated and mixed-type predictors

Description

Generate data including responses and predictors values, of which predictors are correlated and of mixed types.

Usage

mixtwo(n, sigma, binary)

Arguments

n The number of observations.
sigma The error variance.
binary A boolean argument: binary = TRUE indicates that binary responses are generated and binary = FALSE indicates that continuous responses are generated.

Details

Sample the predictors \(x_1, ..., x_{20}\) from Bernoulli(0.2) independently, \(x_{21}, ..., x_{40}\) from Bernoulli(0.5) independently, and \(x_{41}, ..., x_{84}\) from a multivariate normal distribution with mean 0, variance 1 and correlation 0.3. If binary = FALSE, sample the continuous response \(y\) from \(\text{Normal}(f_0(x), \sigma^2)\), where

\[
f_0(x) = -4 + x_1 + \sin(\pi x_1 * x_{44}) - x_{21} + 0.6x_{41} * x_{42} - exp[-2(x_{42} + 1)^2] - x_{43}^2 + 0.5x_{44}.
\]

If binary = TRUE, sample the binary response \(y\) from Bernoulli(\(\Phi(f_0(x))\)) where \(f_0\) is defined above and \(\Phi\) is the cumulative density function of the standard normal distribution.

Value

Return a list with the following components.

\(X\) An \(n\) by \(p\) data frame representing predictors values, with each row corresponding an observation.

\(Y\) A vector of length \(n\) representing response values.

\(f_0\) A vector of length \(n\) representing the values of \(f_0(x)\).

\(\sigma\) The error variance which is only returned when binary = FALSE.

\(\Phi\) A vector of length \(n\) representing the values of \(\Phi(f_0(x))\), which is only returned when binary = TRUE.
Author(s)
Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References

Examples
data = mixtwo(100, 1, FALSE)

pbart  Probit BART for binary responses with Normal latents

Description
BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.
For a binary response $y$ and a $p$–dimensional vector of predictors $x = (x_1, ..., x_p)'$, probit BART models $y$ and $x$ using

$$P(Y = 1 | x) = \Phi[f(x)],$$

where $\Phi$ is the CDF of the standard normal distribution and $f$ is a sum of Bayesian regression trees function.
The function pbart() is inherited from the CRAN R package 'BART' and two modifications are made for the splitting probability and variable importance (see Details).

Usage
pbart(
  x.train,
  y.train,
  x.test = matrix(0, 0, 0),
  sparse = FALSE,
  theta = 0,
  omega = 1,
  a = 0.5,
  b = 1,
  augment = FALSE,
  rho = NULL,
  xinfo = matrix(0, 0, 0),
  numcut = 100L,
  usequants = FALSE,
  cont = FALSE,
  rm.const = TRUE,
  grp = NULL,
Arguments

x.train A matrix or a data frame of predictors values (for training) with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with $q$ levels in a data frame, it is replaced with $q$ dummy variables.

y.train A vector of continuous response values for training.

x.test A matrix or a data frame of predictors values for testing, which has the same structure as x.train.

sparse A Boolean argument indicating whether to replace the discrete uniform distribution for selecting a split variable with a categorical distribution whose event probabilities follow a Dirichlet distribution (see Linero (2018) for details).

theta Set theta parameter; zero means random.

omega Set omega parameter; zero means random.

a A sparse parameter of $Beta(a, b)$ hyper-prior where $0.5 \leq a \leq 1$; a lower value induces more sparsity.

b A sparse parameter of $Beta(a, b)$ hyper-prior; typically, $b = 1$.

augment A Boolean argument indicating whether data augmentation is performed in the variable selection procedure of Linero (2018).

rho A sparse parameter; typically $\rho = p$ where $p$ is the number of predictors.

xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. $xinfo=matrix(0.0,0,0)$ indicates the cut-points are specified by BART.

numcut The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to $\text{ncol}(x.train)$ is required, where the $i$–th element gives the number of cut-points for the $i$–th
predictor in \( x.train \). If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of \( x.train \). If usequants=TRUE, then \( \min(\text{numcut}, \text{the number of unique values in the corresponding column of } x.train - 1) \) cut-point values are used.

**usequants**  
A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

**cont**  
A Boolean argument indicating whether to assume all predictors are continuous.

**rm.const**  
A Boolean argument indicating whether to remove constant predictors.

**grp**  
A vector of group indices for predictors. For example, if 2 appears 3 times in grp, the second predictor of x.train is a categorical predictor with 3 levels. grp is required if transposed=TRUE.

**xnames**  
Column names of x.train. xnames is required if transposed=TRUE.

**categorical.idx**  
A vector of the column indices of categorical predictors in x.train. categorical.idx is required if transposed=TRUE.

**k**  
The number of prior standard deviations that \( E(Y|x) = f(x) \) is away from \(+/- .5\). The response (y.train) is internally scaled to the range from \(-.5\) to \(.5\). The bigger k is, the more conservative the fitting will be.

**power**  
The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

**base**  
The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth \( d \) is \( \text{base}^d \).

**split.prob**  
A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

**binaryOffset**  
The binary offset term in the probit BART model, i.e., \( P(Y = 1|x) = \Phi[f(x) + \text{binaryOffset}] \).

**ntree**  
The number of trees in the ensemble.

**ndpost**  
The number of posterior samples returned.

**nskip**  
The number of posterior samples burned in.

**keepevery**  
Every keepevery posterior sample is kept to be returned to the user.

**nkeeptrain**  
The number of posterior samples returned for the train data.

**nkeeptest**  
The number of posterior samples returned for the test data.

**nkeeptreedraws**  
The number of posterior samples returned for the tree draws.

**printevery**  
As the MCMC runs, a message is printed every printevery iterations.

**transposed**  
A Boolean argument indicating whether the matrices x.train and x.test are transposed.

**verbose**  
A Boolean argument indicating whether any messages are printed out.
Details

This function is inherited from BART::pbart(). While the original features of BART::pbart() are preserved, two modifications are made. The first modification is to provide two types of split probability for BART. One split probability is proposed in Chipman et al. (2010) and defined as

\[ p(d) = \gamma * (1 + d)^{-\beta}, \]

where \( d \) is the depth of the node, \( \gamma \in (0, 1) \) and \( \beta \in (0, \infty) \). The other split probability is proposed by Rockova and Saha (2019) and defined as

\[ p(d) = \gamma^d, \]

where \( \gamma \in (1/n, 1/2) \). BART with the second split probability is proved to achieve the optimal posterior contraction.

The second modification is to provide five types of variable importance measures (vip, within.type.vip, pvip, var.prob.mean and mi) in the return object, for the sake of the existence of mixed-type predictors.

Value

The function pbart() returns an object of type pbart which essentially is a list consisting of the following components.

- **yhat.train**  
  A matrix with \( ndpost \) rows and \( nrow(x.train) \) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a training data point. The \((i,j)\)-th element of the matrix is \( f^*(x) + \text{binaryOffset} \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th training data point. Burn-in posterior samples are dropped.

- **prob.train**  
  A matrix with the same structure as yhat.train and the \((i,j)\)-th element of prob.train is \( \Phi[f^*(x) + \text{binaryOffset}] \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th training data point.

- **prob.train.mean**  
  A vector which is colMeans(prob.train).

- **yhat.test**  
  A matrix with \( ndpost \) rows and \( nrow(x.test) \) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a test data point. The \((i,j)\)-th element of the matrix is \( f^*(x) + \text{binaryOffset} \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th test data point. Burn-in posterior samples are dropped.

- **prob.test**  
  A matrix with the same structure as yhat.test and the \((i,j)\)-th element of prob.test is \( \Phi[f^*(x) + \text{binaryOffset}] \) for the \( i \)-th kept draw of \( f \) and the \( j \)-th test data point.

- **prob.test.mean**  
  A vector which is colMeans(prob.test).

- **varcount**  
  A matrix with \( ndpost \) rows and \( ncol(x.train) \) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i,j)\)-th element is the number of times that the \( j \)-th predictor is used as a split variable in the \( i \)-th posterior sample.
varprob  A matrix with \( n_{\text{dpost}} \) rows and \( ncol(x_train) \) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i,j)\)-th element is the split probability of the \( j \)-th predictor in the \( i \)-th posterior sample. Only useful when DART is fit, i.e., \( \text{sparse}=\text{TRUE} \).

treedraws  A list containing the posterior samples of the ensembles (trees structures, split variables and split values); Can be used for prediction.

proc.time  The process time of running the function \( \text{wbart()} \).

vip  A vector of variable inclusion proportions (VIP) proposed in Chipman et al. (2010).

within.type.vip  A vector of within-type VIPs proposed in Luo and Daniels (2021).

pvip  A vector of marginal posterior variable inclusion probabilities (PVIP) proposed in Linero (2018); Only useful when DART is fit, i.e., \( \text{sparse}=\text{TRUE} \).

varprob.mean  A vector of posterior split probabilities (PSP) proposed in Linero (2018); Only useful when DART is fit, i.e., \( \text{sparse}=\text{TRUE} \).

mr.vecs  A list of \( ncol(x_train) \) sub-lists with each corresponding to a predictor; Each sub-list contains \( n_{\text{dpost}} \) vectors with each vector containing the (birth) Metropolis ratios for splits using the predictor as the split variable in that posterior sample.

mr.mean  A matrix with \( n_{\text{dpost}} \) rows and \( ncol(x.train) \) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i,j)\)-th element is the average Metropolis acceptance ratio per splitting rule using the \( j \)-th predictor in the \( i \)-th posterior sample.

mi  A vector of Metropolis importance (MI) proposed in Luo and Daniels (2021).

rm.const  A vector of indicators for the predictors (after dummification) used in BART; when the indicator is negative, it refers to remove that predictor.

binaryOffset  The binary offset term used in BART.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

mc.pbart, wbart and pwbart.

Examples

```r
## simulate data (Scenario B.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, TRUE)
## test pbart() function
res = pbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100)
```

---

**permute.vs**

Permutation-based variable selection approach

**Description**

This function implements the permutation-based variable selection approach for BART (see Algorithm 1 in Luo and Daniels (2021) for details). Three types of variable importance measures are considered: BART variable inclusion proportions (VIP), BART within-type variable inclusion proportions (within-type VIP) and BART Metropolis Importance (MI).

The permutation-based variable selection approach using BART VIP as the variable importance measure is proposed by Bleich et al. (2014). BART within-type VIP and BART MI are proposed by Luo and Daniels (2021), for the sake of the existence of mixed-type predictors and the goal of allowing more relevant predictors into the model.

**Usage**

```r
permute.vs(
x.train,
y.train,
probit = FALSE,
npermute = 100L,
nreps = 10L,
alpha = 0.05,
true.idx = NULL,
plot = TRUE,
n.var.plot = Inf,
xinfo = matrix(0, 0, 0),
numcut = 100L,
usequants = FALSE,
cont = FALSE,
rm.const = TRUE,
k = 2,
power = 2,
base = 0.95,
split.prob = "polynomial",
```
Arguments

x.train A matrix or a data frame of predictors values with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with \( q \) levels in a data frame, it is replaced with \( q \) dummy variables.

y.train A vector of response (continuous or binary) values.

probit A Boolean argument indicating whether the response variable is binary or continuous; \texttt{probit}=FALSE (by default) means that the response variable is continuous.

npermute The number of permutations for estimating the null distributions of the variable importance scores.

nreps The number of replications for obtaining the averaged (or median) variable importance scores based on the original data set.

alpha A number between 0 and 1; a predictor is selected if its averaged (or median) variable importance score exceeds the \( 1 - \alpha \) quantile of the corresponding null distribution.

true.idx (Optional) A vector of indices of the true relevant predictors; if provided, metrics including precision, recall and F1 score will be returned.

plot (Optional) A Boolean argument indicating whether plots are returned or not.

n.var.plot The number of variables to be plotted.

xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. \texttt{xinfo=matrix(0.0,0,0)} indicates the cut-points are specified by BART.

numcut The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to \texttt{ncol(x.train)} is required, where the \( i \)-th element gives the number of cut-points for the \( i \)-th predictor in \texttt{x.train}. If \texttt{usequants=FALSE}, \texttt{numcut} equally spaced cut-points are used to cover the range of values in the corresponding column of \texttt{x.train}. If \texttt{usequants=TRUE}, then \texttt{min(numcut, the number of unique values in the corresponding column of x.train - 1)} cut-point values are used.

usequants A Boolean argument indicating how the cut-points in \texttt{xinfo} are generated; If \texttt{usequants=TRUE}, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont A Boolean argument indicating whether to assume all predictors are continuous.

rm.const A Boolean argument indicating whether to remove constant predictors.
The number of prior standard deviations that $E(Y|x) = f(x)$ is away from $+/-.5$. The response ($y.train$) is internally scaled to the range from $-.5$ to $.5$. The bigger $k$ is, the more conservative the fitting will be.

The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".

The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth $d$ is $\text{base}^d$.

A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.

The number of trees in the ensemble.

The number of posterior samples returned.

The number of posterior samples burned in.

Every keepevery posterior sample is kept to be returned to the user.

As the MCMC runs, a message is printed every printevery iterations.

A Boolean argument indicating whether any messages are printed out.

The detailed algorithm can be found in Algorithm 1 in Luo and Daniels (2021). The permutation-based variable selection approach using within-type VIP as the variable importance measure is only used when the predictors are of mixed-type; otherwise, it is the same as the one using VIP as the variable importance measure. If true.idx is provided, the precision, recall and F1 scores will be returned for the three (or two if the predictors are of the same type) methods. If plot=TRUE, three (or two if the predictors are of the same type) plots showing which predictors are selected are generated.

The function permute.vs() returns three (or two if the predictors are of the same type) plots if plot=TRUE and a list with the following components.

The vector of column indices of the predictors selected by the approach using VIP as the variable importance score.

The vector of column names of the predictors selected by the approach using VIP as the variable importance score.

The vector (length=ncol(x.train)) of the averaged VIPs based on the original data set; avg.vip=colMeans(avg.vip.mtx).

A matrix of VIPs based on the original data set, with each row corresponding to a repetition and each column corresponding to a predictor.

A matrix of VIPs based on the null data sets, with each row corresponding to a permutation (null data set) and each column corresponding to a predictor.
within.type.vip.imp.cols
The vector of column indices of the predictors selected by the approach using
within-type VIP as the variable importance score.

within.type.vip.imp.names
The vector of column names of the predictors selected by the approach using
within-type VIP as the variable importance score.

avg.within.type.vip
The vector (length=ncol(x.train)) of the averaged within-type VIPs based on
the original data set; avg.within.type.vip=colMeans(avg.within.type.vip.mtx).

avg.within.type.vip.mtx
A matrix of within-type VIPs based on the original data set, with each row
corresponding to a repetition and each column corresponding to a predictor.

permute.within.type.vips
A matrix of within VIPs based on the null data sets, with each row corresponding
to a permutation (null data set) and each column corresponding to a predictor.

varcounts
A list of nreps+npermute elements; Each element is a ndpost by ncol(x.train)
matrix with each row corresponding to a draw of the ensemble and each column
corresponding to a predictor; The (i,j)-th element of the matrix is the number
of times that the j-th predictor is used as a split variable in the i-th posterior
sample; The first nreps elements correspond to nreps repetitions and the latter
npermute elements correspond to npermute permutations.

mi.imp.cols
The vector of column indices of the predictors selected by the approach using
MI as the variable importance score.

mi.imp.names
The vector of column names of the predictors selected by the approach using MI
as the variable importance score.

median.mi
The vector (length=ncol(x.train)) of the median MIs based on the original
data set; median.mi=colMeans(median.mi.mtx).

median.mi.mtx
A matrix of MIs based on the original data set, with each row corresponding to
a repetition and each column corresponding to a predictor.

permute.mis
A matrix of MIs based on the null data sets, with each row corresponding to a
permutation (null data set) and each column corresponding to a predictor.

true.idx
A vector of indices of the true relevant predictors; only returned if true.idx is
provided as inputs.

vip.precision
The precision score for the approach using VIP as the variable importance score;
only returned if true.idx is provided.

vip.recall
The recall score for the approach using VIP as the variable importance score;
only returned if true.idx is provided.

vip.f1
The F1 score for the approach using VIP as the variable importance score; only
returned if true.idx is provided.

wt.vip.precision
The precision score for the approach using within-VIP as the variable importance
score; only returned when the predictors are of the same type and true.idx is
provided.
permute.vs

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wt.vip.recall</td>
<td>The recall score for the approach using within-VIP as the variable importance score; only returned when the predictors are of the same type and true.idx is provided.</td>
</tr>
<tr>
<td>wt.vip.f1</td>
<td>The F1 score for the approach using within-VIP as the variable importance score; only returned when the predictors are of the same type and true.idx is provided.</td>
</tr>
<tr>
<td>mi.precision</td>
<td>The precision score for the approach using MI as the variable importance score; only returned if true.idx is provided.</td>
</tr>
<tr>
<td>mi.recall</td>
<td>The recall score for the approach using MI as the variable importance score; only returned if true.idx is provided.</td>
</tr>
<tr>
<td>mi.f1</td>
<td>The F1 score for the approach using MI as the variable importance score; only returned if true.idx is provided.</td>
</tr>
</tbody>
</table>

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

mc.permute.vs, medianInclusion.vs, mc.backward.vs and abc.vs.

Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## test permute.vs() function
res = permute.vs(data$X, data$Y, probit=FALSE, npermute=100, nreps=10, alpha=0.05, true.idx=c(1,2,6:8), plot=FALSE, ntree=10, ndpost=100, nskip=100)
```
predict.pbart

### Predict new observations with a fitted BART model

**Description**

BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.

For a binary response $y$, probit BART models $y$ and $x$ using

$$P(Y = 1|x) = \Phi[f(x)],$$

where $\Phi$ is the CDF of the standard normal distribution and $f$ is a sum of Bayesian regression trees function.

This function uses S3 method for the class pbart and is inherited from the CRAN R package 'BART'.

**Usage**

```r
## S3 method for class 'pbart'
predict(object, newdata, mc.cores = 1, openmp = (mc.cores.openmp() > 0), ...)
```

**Arguments**

- `object` An object of class pbart, returned from the function pbart().
- `newdata` A matrix of predictors with rows corresponding to new observations.
- `mc.cores` The number of threads to utilize.
- `openmp` A Boolean argument dictating whether OpenMP is utilized for parallel processing. This depends on whether OpenMP is available on your system which, by default, is verified with the function mc.cores.openmp().
- `...` Other arguments passed on to the function pwbart().

**Value**

Returns a matrix of prediction for newdata, whose rows correspond to draws and columns correspond to observations.

**Author(s)**

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

**References**


predict.wbart


See Also

pwbart and pbart.

Examples

```r
## simulate data (Scenario B.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, TRUE)
## run pbart() function
res = pbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100)
## test predict.pbart() function
newdata = mixone(5, 10, 1, TRUE)$X
pred = predict(res, newdata)
```

**predict.wbart**

*Predict new observations with a fitted BART model*

**Description**

BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees. For a continuous response $y$ and a $p$–dimensional vector of predictors $x = (x_1, \ldots, x_p)'$, BART models $y$ and $x$ using

$$y = f(x) + \epsilon,$$

where $f$ is a sum of Bayesian regression trees function and $\epsilon \sim N(0, \sigma^2)$. This function uses S3 method for the class wbart and is inherited from the CRAN R package 'BART'.

**Usage**

```r
## S3 method for class 'wbart'
predict(object, newdata, mc.cores = 1, openmp = (mc.cores.openmp() > 0), ...)
```
**Arguments**

- **object** An object of class `wbart`, returned from the function `wbart()`.
- **newdata** A matrix of predictors with rows corresponding to new observations.
- **mc.cores** The number of threads to utilize.
- **openmp** A Boolean argument dictating whether OpenMP is utilized for parallel processing. This depends on whether OpenMP is available on your system which, by default, is verified with the function `mc.cores.openmp()`.
- ... Other arguments passed on to the function `pwbart()`.

**Value**

Returns a matrix of prediction for `newdata`, whose rows correspond to draws and columns correspond to observations.

**Author(s)**

Chuji Luo: `<cjluo@ufl.edu>` and Michael J. Daniels: `<daniels@ufl.edu>`.

**References**


**See Also**

- `pwbart` and `wbart`.

**Examples**

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## run wbart() function
res = wbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100)
## test predict.wbart() function
newdata = mixone(5, 10, 1, FALSE)$X
pred = predict(res, newdata)
```
BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.

For a continuous response \( y \) and a \( p \)-dimensional vector of predictors \( x = (x_1, ..., x_p)' \), BART models \( y \) and \( x \) using

\[
y = f(x) + \epsilon,
\]

where \( f \) is a sum of Bayesian regression trees function and \( \epsilon \sim N(0, \sigma^2) \).

For a binary response \( y \), probit BART models \( y \) and \( x \) using

\[
P(Y = 1|x) = \Phi[f(x)],
\]

where \( \Phi \) is the CDF of the standard normal distribution and \( f \) is a sum of Bayesian regression trees function.

The function \( \text{pwbart}() \) is inherited from the CRAN R package 'BART'.

**Usage**

\[
\text{pwbart}(\text{x.test, treedraws, rm.const, mu = 0, mc.cores = 1L, transposed = FALSE, dodraws = TRUE, verbose = FALSE})
\]

**Arguments**

- **x.test**: A matrix or a data frame of predictors values for prediction with each row corresponding to an observation and each column corresponding to a predictor.
- **treedraws**: A list which is the \$treedraws returned from the function \( \text{wbart}() \) or \( \text{pbart}() \).
- **rm.const**: A vector which is the \$rm.const returned from the function \( \text{wbart}() \) or \( \text{pbart}() \).
- **mu**: Mean to add on to \( y \) prediction.
- **mc.cores**: The number of threads to utilize.
- **transposed**: A Boolean argument indicating whether the matrix \( x.\text{test} \) is transposed. When running \( \text{pwbart}() \) or \( \text{mc.pwbart}() \) in parallel, it is more memory-efficient to transpose \( x.\text{test} \) prior to calling the internal versions of these functions.
- **dodraws**: A Boolean argument indicating whether to return the draws themselves (the default), or whether to return the mean of the draws as specified by \( \text{dodraws} = \text{FALSE} \).
- **verbose**: A Boolean argument indicating whether any messages are printed out.
Value

Returns the predictions for x.test. If dodraws=TRUE, return a matrix of prediction with each row corresponding to a draw and each column corresponding to a new observation; if dodraws=FALSE, return a vector of predictions which are the mean of the draws.

Author(s)

Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also

`wbart`, `pbart` and `mc.pwbart`.

Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## run wbart() function
res = wbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100)
## test pwbart() function
x.test = mixone(5, 10, 1, FALSE)$X
pred = pwbart(x.test, res$treedraws, res$rm.const, mu=mean(data$Y))
```

---

**wbart**

*BART for continuous responses*
Description

BART is a Bayesian approach to nonparametric function estimation and inference using a sum of trees.

For a continuous response \( y \) and a \( p \)-dimensional vector of predictors \( x = (x_1, ..., x_p)' \), BART models \( y \) and \( x \) using

\[
y = f(x) + \epsilon,
\]

where \( f \) is a sum of Bayesian regression trees function and \( \epsilon \sim N(0, \sigma^2) \).

The function \texttt{wbart()} is inherited from the CRAN R package \textquote{BART} and two modifications are made for the splitting probability and variable importance (see Details).

Usage

\texttt{wbart(}
  \texttt{x.train,}
  \texttt{y.train,}
  \texttt{x.test = matrix(0, 0, 0),}
  \texttt{sparse = FALSE,}
  \texttt{theta = 0,}
  \texttt{omega = 1,}
  \texttt{a = 0.5,}
  \texttt{b = 1,}
  \texttt{augment = FALSE,}
  \texttt{rho = NULL,}
  \texttt{xinfo = matrix(0, 0, 0),}
  \texttt{numcut = 100L,}
  \texttt{usequants = FALSE,}
  \texttt{cont = FALSE,}
  \texttt{rm.const = TRUE,}
  \texttt{grp = NULL,}
  \texttt{xnames = NULL,}
  \texttt{categorical.idx = NULL,}
  \texttt{power = 2,}
  \texttt{base = -1,}
  \texttt{split.prob = "polynomial",}
  \texttt{k = 2,}
  \texttt{sigmf = NA,}
  \texttt{sigest = NA,}
  \texttt{sigdf = 3,}
  \texttt{sigquant = 0.9,}
  \texttt{lambda = NA,}
  \texttt{fmean = mean(y.train),}
  \texttt{w = rep(1, length(y.train)),}
  \texttt{ntree = 200L,}
  \texttt{ndpost = 1000L,}
  \texttt{nskip = 1000L,}
  \texttt{keepevery = 1L,}
  \texttt{nnkeeptrain = ndpost,}
  \texttt{nnkeeptrain = ndpost,}
\texttt{)}
Arguments

x.train A matrix or a data frame of predictors values (for training) with each row corresponding to an observation and each column corresponding to a predictor. If a predictor is a factor with q levels in a data frame, it is replaced with q dummy variables.

y.train A vector of continuous response values for training.

x.test A matrix or a data frame of predictors values for testing, which has the same structure as x.train.

sparse A Boolean argument indicating whether to replace the discrete uniform distribution for selecting a split variable with a categorical distribution whose event probabilities follow a Dirichlet distribution (see Linero (2018) for details).

theta Set theta parameter; zero means random.

omega Set omega parameter; zero means random.

a A sparse parameter of Beta(a, b) hyper-prior where 0.5 <= a <= 1; a lower value induces more sparsity.

b A sparse parameter of Beta(a, b) hyper-prior; typically, b = 1.

augment A Boolean argument indicating whether data augmentation is performed in the variable selection procedure of Linero (2018).

rho A sparse parameter; typically ρ = p where p is the number of predictors.

xinfo A matrix of cut-points with each row corresponding to a predictor and each column corresponding to a cut-point. xinfo=matrix(0.0,0,0) indicates the cut-points are specified by BART.

numcut The number of possible cut-points; If a single number is given, this is used for all predictors; Otherwise a vector with length equal to ncol(x.train) is required, where the i-th element gives the number of cut-points for the i-th predictor in x.train. If usequants=FALSE, numcut equally spaced cut-points are used to cover the range of values in the corresponding column of x.train. If usequants=TRUE, then min(numcut, the number of unique values in the corresponding column of x.train - 1) cut-point values are used.

usequants A Boolean argument indicating how the cut-points in xinfo are generated; If usequants=TRUE, uniform quantiles are used for the cut-points; Otherwise, the cut-points are generated uniformly.

cont A Boolean argument indicating whether to assume all predictors are continuous.

rm.const A Boolean argument indicating whether to remove constant predictors.

grp A vector of group indices for predictors. For example, if 2 appears 3 times in grp, the second predictor of x.train is a categorical predictor with 3 levels. grp is required if transposed=TRUE.
xnames: Column names of x.train. xnames is required if transposed=TRUE.
categorical.idx: A vector of the column indices of categorical predictors in x.train. categorical.idx is required if transposed=TRUE.
power: The power parameter of the polynomial splitting probability for the tree prior. Only used if split.prob="polynomial".
base: The base parameter of the polynomial splitting probability for the tree prior if split.prob="polynomial"; if split.prob="exponential", the probability of splitting a node at depth $d$ is $\text{base}^d$.
split.prob: A string indicating what kind of splitting probability is used for the tree prior. If split.prob="polynomial", the splitting probability in Chipman et al. (2010) is used; If split.prob="exponential", the splitting probability in Rockova and Saha (2019) is used.
k: The number of prior standard deviations that $E(Y|x) = f(x)$ is away from $+/- .5$. The response (y.train) is internally scaled to the range from $-.5$ to $.5$. The bigger $k$ is, the more conservative the fitting will be.
sigf: The standard deviation of $f$.
sigest: A rough estimate of the error standard deviation, the square of which follows an inverse chi-squared prior. If sigest=NA, the rough estimate will be the usual least square estimator; Otherwise, the supplied value will be used.
sigdf: The degrees of freedom for the error variance prior.
sigquant: The quantile of the error variance prior, where sigest is placed. The closer the quantile is to 1, the more aggressive the fit will be.
lambda: The scale parameter of the error variance prior.
fmean: BART operates on y.train centered by fmean.
w: A vector of weights which multiply the standard deviation.
ntree: The number of trees in the ensemble.
ndpost: The number of posterior samples returned.
nskip: The number of posterior samples burned in.
keepevery: Every keepevery posterior sample is kept to be returned to the user.
nkeeptest: The number of posterior samples returned for the train data.
nkeeptest: The number of posterior samples returned for the test data.
nkeeptreedraws: The number of posterior samples returned for the tree draws.
printevery: As the MCMC runs, a message is printed every printevery iterations.
transposed: A Boolean argument indicating whether the matrices x.train and x.test are transposed.
verbose: A Boolean argument indicating whether any messages are printed out.
Details

This function is inherited from BART::wbart(). While the original features of BART::wbart() are preserved, two modifications are made.

The first modification is to provide two types of split probability for BART. One split probability is proposed in Chipman et al. (2010) and defined as

\[ p(d) = \gamma \ast (1 + d)^{-\beta}, \]

where \( d \) is the depth of the node, \( \gamma \in (0, 1) \) and \( \beta \in (0, \infty) \). The other split probability is proposed by Rockova and Saha (2019) and defined as

\[ p(d) = \gamma^d, \]

where \( \gamma \in (1/n, 1/2) \). BART with the second split probability is proved to achieve the optimal posterior contraction.

The second modification is to provide five types of variable importance measures (vip, within.type.vip, pvip, varprob.mean and mi) in the return object, for the sake of the existence of mixed-type predictors.

Value

The function wbart() returns an object of type wbart which essentially is a list consisting of the following components.

sigma A vector with nskip+ndpost*keepevery posterior samples of \( \sigma \).

yhat.train.mean colMeans(yhat.train).

yhat.train A matrix with ndpost rows and nrow(x.train) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a training data point. The \((i, j)\)-th element of the matrix is \( f^*(x) \) for the i-th kept draw of \( f \) and the j-th training data point. Burn-in posterior samples are dropped.

yhat.test.mean colMeans(yhat.test).

yhat.test A matrix with ndpost rows and nrow(x.test) columns with each row corresponding to a draw \( f^* \) from the posterior of \( f \) and each column corresponding to a test data point. The \((i, j)\)-th element of the matrix is \( f^*(x) \) for the i-th kept draw of \( f \) and the j-th test data point. Burn-in posterior samples are dropped.

varcount A matrix with ndpost rows and ncol(x.train) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i, j)\)-th element is the number of times that the j-th predictor is used as a split variable in the i-th posterior sample.

varprob A matrix with ndpost rows and ncol(x.train) columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The \((i, j)\)-th element is the split probability of the j-th predictor in the i-th posterior sample. Only useful when DART is fit, i.e., sparse=TRUE.

treedraws A list containing the posterior samples of the ensembles (trees structures, split variables and split values); Can be used for prediction.
proc.time  The process time of running the function `wbart()`.

mu  BART operates on `y.train` centered by `fmean`.

mr.vecs  A list of `ncol(x.train)` sub-lists with each corresponding to a predictor; Each sub-list contains `ndpost` vectors with each vector containing the (birth) Metropolis ratios for splits using the predictor as the split variable in that posterior sample.

vip  A vector of variable inclusion proportions (VIP) proposed in Chipman et al. (2010).

within.type.vip  A vector of within-type VIPs proposed in Luo and Daniels (2021).

pvip  A vector of marginal posterior variable inclusion probabilities (PVIP) proposed in Linero (2018); Only useful when DART is fit, i.e., `sparse=TRUE`.

varprob.mean  A vector of posterior split probabilities (PSP) proposed in Linero (2018); Only useful when DART is fit, i.e., `sparse=TRUE`.

mr.mean  A matrix with `ndpost` rows and `ncol(x.train)` columns with each row corresponding to a draw of the ensemble and each column corresponding to a predictor. The `(i,j)`-th element is the average Metropolis acceptance ratio per splitting rule using the `j`-th predictor in the `i`-th posterior sample.

mi  A vector of Metropolis importance (MI) proposed in Luo and Daniels (2021).

rm.const  A vector of indicators for the predictors (after dummification) used in BART; when the indicator is negative, it refers to remove that predictor.

Author(s)
Chuji Luo: <cjluo@ufl.edu> and Michael J. Daniels: <daniels@ufl.edu>.

References


See Also
`mc.wbart`, `pbart` and `pwbart`. 
Examples

```r
## simulate data (Scenario C.M.1. in Luo and Daniels (2021))
set.seed(123)
data = mixone(100, 10, 1, FALSE)
## test wbart() function
res = wbart(data$X, data$Y, ntree=10, nskip=100, ndpost=100)
```
**Index**

* **BART**  
  BartMixVs-package, 2

* **Non-Parametric Regression**  
  BartMixVs-package, 2

* **Variable Selection**  
  BartMixVs-package, 2

abc.vs, 3, 15, 19, 30, 40, 52

BartMixVs (BartMixVs-package), 2  
BartMixVs-package, 2  
bartModelMatrix, 7

checkerboard, 9

friedman, 10

mc.abc.vs, 11  
mc.backward.vs, 7, 16, 30, 40, 52  
mc.cores.openmp, 19  
mc.pbart, 20, 48  
mc.permute.vs, 25, 52  
mc.pwbart, 30, 31, 57  
mc.wbart, 32, 62  
medianInclusion.vs, 7, 19, 30, 37, 52  
mixone, 41  
mixtwo, 42

pbart, 9, 25, 31, 43, 54, 57, 62  
permute.vs, 7, 19, 30, 40, 48  
predict.pbart, 53  
predict.wbart, 54  
pwbart, 20, 48, 54, 55, 56, 62

wbart, 9, 31, 37, 48, 55, 57, 57