Package ‘nftbart’

March 29, 2022

**Type** Package

**Title** Nonparametric Failure Time Bayesian Additive Regression Trees

**Version** 1.3

**Date** 2022-03-26

**Author** Rodney Sparapani [aut, cre],
Robert McCulloch [aut],
Matthew Pratola [ctb],
Hugh Chipman [ctb]

**Maintainer** Rodney Sparapani <rsparapa@mcw.edu>

**Description** Nonparametric Failure Time (NFT) Bayesian Additive Regression Trees (BART): Time-to-event Machine Learning with Heteroskedastic Bayesian Additive Regression Trees (HBART) and Low Information Omnibus (LIO) Dirichlet Process Mixtures (DPM). An NFT BART model is of the form $Y = \mu + f(x) + s(x) E$ where functions $f$ and $s$ have BART and HBART priors, respectively, while $E$ is a nonparametric error distribution due to a DPM LIO prior hierarchy. See the following for a technical description of the model <https://www.mcw.edu/-/media/MCW/Departments/Biostatistics/tr72.pdf?la=en>.

**License** GPL (>= 2)

**Depends** R (>= 3.6), survival, nnet

**Imports** Rcpp

**Suggests** knitr, rmarkdown

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-03-29 19:30:02 UTC

**R topics documented:**

- bartModelMatrix .......................................................... 2
- CDimpute ................................................................. 3
- galaxy ................................................................. 4
bartModelMatrix

Create a matrix out of a vector or data.frame

Description

The compiled functions of this package 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 cutpoints necessary for each column when asked to do so.

Usage

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

Arguments

- **X**: A vector or data.frame to create the matrix from.
- **numcut**: The maximum number of cutpoints to consider. If `numcut=0`, then just return a matrix; otherwise, return a list.
- **usequants**: If `usequants` is FALSE, then the cutpoints in `xinfo` are generated uniformly; otherwise, if TRUE, quantiles are used for the cutpoints.
- **type**: Determines which quantile algorithm is employed.
- **rm.const**: Whether or not to remove constant variables.
- **cont**: Whether or not to assume all variables are continuous.
- **xicuts**: To specify your own cut-points, use the `xicuts` argument.
- **rm.vars**: The variables that you want removed.

Value

If `numcut==0` (the default), then a matrix of the covariates is returned; otherwise, a list is returned with the following values.

- **X**: A matrix of the covariates with n rows and p columns.
- **numcut**: A vector of length p of the number of cut-points for each covariate.
- **grp**: A vector that corresponds to variables in the input data.frame that were translated into dummy columns in the output matrix, i.e., for each input variable in order, there is a number in the vector corresponding to the number of output columns created for it.
CDimpute

Cold-deck missing imputation

Description

This function imputes missing data.

Usage

CDimpute(x.train, x.test=matrix(0, 0, 0), impute.bin=NULL)

Arguments

x.train The training matrix.
x.test The testing matrix, if given.
impute.bin An index of the columns to avoid imputing which will be handled by BART internally.
Details

We call this method cold-decking in analogy to hot-decking. Hot-decking was a method commonly employed with US Census data in the early computing era. For a particular respondent, missing data was imputed by randomly selecting from the responses of their neighbors since it is assumed that the values are likely similar. In our case, we make no assumptions about which values may, or may not, be nearby. We simply take a random sample from the matrix rows to impute the missing data. If the training and testing matrices are the same, then they receive the same imputation.

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.train</td>
<td>The imputed training matrix.</td>
</tr>
<tr>
<td>x.test</td>
<td>The imputed testing matrix.</td>
</tr>
<tr>
<td>miss.train</td>
<td>A summary of the missing variables for training.</td>
</tr>
<tr>
<td>miss.test</td>
<td>A summary of the missing variables for testing.</td>
</tr>
<tr>
<td>impute.flag</td>
<td>Whether impute.bin columns were, or were not, imputed.</td>
</tr>
<tr>
<td>same</td>
<td>Whether x.train and x.test are identical.</td>
</tr>
</tbody>
</table>

Description

This data set consider physical information on velocities (km/second) for 82 galaxies reported by Roeder (1990). These are drawn from six well-separated conic sections of the Corona Borealis region.

Usage

```
data(galaxy)
```

Format

A data frame with 82 observations on the following variable.

- **speed**: a numeric vector giving the speed of galaxies ((km/second))

Source


References

Examples

data(galaxy)
## maybe str(galaxy); plot(galaxy) ...

---

lung NCCTG Lung Cancer Data

Description

Survival for 228 patients with advanced lung cancer was recorded up to a median of roughly one year by the North Central Cancer Treatment Group. Performance scores rate how well the patient can perform usual daily activities.

Format

<table>
<thead>
<tr>
<th>inst: Institution code</th>
</tr>
</thead>
<tbody>
<tr>
<td>time: Survival time in days</td>
</tr>
<tr>
<td>status: censoring status 1=censored, 2=dead</td>
</tr>
<tr>
<td>age: Age in years</td>
</tr>
<tr>
<td>sex: Male=1 Female=2</td>
</tr>
<tr>
<td>ph.ecog: ECOG performance score (0=good 5=dead)</td>
</tr>
<tr>
<td>ph.karno: Karnofsky performance score (bad=0-good=100) rated by physician</td>
</tr>
<tr>
<td>pat.karno: Karnofsky performance score as rated by patient</td>
</tr>
<tr>
<td>meal.cal: Calories consumed at meals</td>
</tr>
<tr>
<td>wt.loss: Weight loss in last six months</td>
</tr>
</tbody>
</table>

Source

Terry Therneau

References


Examples

data(lung)
nft

Fit NFT BART models.

Description

The `nft()` function is for fitting NFT BART (Nonparametric Failure Time Bayesian Additive Regression Tree) models.

Usage

```r
nft(
## data
x.train, times, delta=NULL, x.test=matrix(nrow=0, ncol=0),
impute.bin=NULL, impute.prob=NULL,
## multi-threading
tc=1,
##MCMC
nskip=1000, ndpost=2000, nadapt=1000, adaptevery=100,
chv = cor(x.train, method="spearman"),
pbd=c(0.7, 0.7), pb=c(0.5, 0.5),
stepwpert=c(0.1, 0.1), probchv=c(0.1, 0.1),
minnumbot=c(5, 5),
## BART and HBART prior parameters
ntree=c(50, 10), numcut=100, xicuts=NULL,
power=c(2, 2), base=c(0.95, 0.95),
## f function
k=5, sigmaf=NA, dist='weibull',
## s function
sigmav=NULL, total.lambda=NA, total.nu=10,
## survival analysis
K=100, events=NULL,
## DPM LIO
drawDPM=1L,
alpha=1, alpha.a=1, alpha.b=0.1, alpha.draw=1,
neal.m=2, constrain=1,
m0=0, k0.a=1.5, k0.b=7.5, k0=1, k0.draw=1,
a0=1.5, b0.a=0.5, b0.b=1, b0=1, b0.draw=1,
## misc
printevery=100)
```

Arguments

- `x.train`: nxp matrix of predictor variables for the training data.
- `times`: nx1 vector of the observed times for the training data.
delta \quad n x 1 \text{ vector of the time type for the training data: 0, for right-censoring; 1, for an event; and, 2, for left-censoring.}

x.test \quad m x p \text{ matrix of predictor variables for the test set.}

impute.bin \quad \text{Indices of the columns of } x.\text{train to be imputed.}

impute.prob \quad n x 1 \text{ vector of prior probabilities for imputation.}

tc \quad \text{Number of OpenMP threads to use.}

nskip \quad \text{Number of MCMC iterations to burn-in and discard.}

ndpost \quad \text{Number of MCMC iterations kept after burn-in.}

nadapt \quad \text{Number of MCMC iterations for adaptation prior to burn-in.}

adaptevery \quad \text{Adapt MCMC proposal distributions every } adaptevery \text{ iteration.}

chv \quad \text{Predictor correlation matrix used as a pre-conditioner for MCMC change-of-variable proposals.}

pbd \quad \text{Probability of performing a birth/death proposal, otherwise perform a rotate proposal.}

pb \quad \text{Probability of performing a birth proposal given that we choose to perform a birth/death proposal.}

stepwpert \quad \text{Initial width of proposal distribution for peturbing cut-points.}

probchv \quad \text{Probability of performing a change-of-variable proposal. Otherwise, only do a perturb proposal.}

minnumbot \quad \text{Minimum number of observations required in leaf (terminal) nodes.}

ntree \quad \text{Vector of length two for the number of trees used for the mean model and the number of trees used for the variance model.}

numcut \quad \text{Number of cutpoints to use for each predictor variable.}

xicuts \quad \text{More detailed construction of cut-points can be specified by the xicuts function and provided here.}

power \quad \text{Power parameter in the tree depth penalizing prior.}

base \quad \text{Base parameter in the tree depth penalizing prior.}

k \quad \text{Prior hyperparameter for the mean model.}

sigmaf \quad \text{SD of } y.\text{train desired for f function leaf prior.}

dist \quad \text{Distribution to be passed to intercept-only AFT model to center } y.\text{train.}

sigmav \quad \text{Initialization of square-root of variance parameter.}

total.lambda \quad \text{A rudimentary estimate of the process standard deviation. Used in calibrating the variance prior.}

total.nu \quad \text{Shape parameter for the variance prior.}

K \quad \text{Number of grid points for which to estimate survival probability.}

events \quad \text{Grid points for which to estimate survival probability.}

drawDPM \quad \text{Whether to utilize DPM or not.}

alpha \quad \text{Initial value of DPM concentration parameter.}

alpha.a \quad \text{Gamma prior parameter setting for DPM concentration parameter where } E[\alpha]=\text{alpha.a/alpha.b.
alpha.b  See alpha.a above.
alpha.draw  Whether to draw alpha or it is fixed at the initial value.
neal.m  The number of additional atoms for Neal 2000 DPM algorithm 8.
constrain  Whether to perform constrained DPM or unconstrained.
m0  Center of the error distribution: defaults to zero.
k0.a  First Gamma prior argument for k0.
k0.b  Second Gamma prior argument for k0.
k0  Initial value of k0.
k0.draw  Whether to fix k0 or draw it if from the DPM LIO prior hierarchy: k0~Gamma(k0.a,k0.b), i.e., E[k0]=k0.a/k0.b.
a0  First Gamma prior argument for tau.
b0.a  First Gamma prior argument for b0.
b0.b  Second Gamma prior argument for b0.
b0  Initial value of b0.
b0.draw  Whether to fix b0 or draw it from the DPM LIO prior hierarchy: b0~Gamma(b0.a,b0.b), i.e., E[b0]=b0.a/b0.b.
printevery  Outputs MCMC algorithm status every printevery iterations.

Details

nft() is the function to fit time-to-event data. The most general form of the model allowed is
Y(x) = mu + f(x) + s(x)Z where E follows a nonparametric error distribution by default.
The nft() function returns a fit object of S3 class type nft that is essentially a list containing the
following items.

Value

ots, oid, ovar, oc, otheta
   These are XPtrs to the BART f(x) objects in RAM that are only available for
   fits generated in the current R session.
sts, sid, svar, sc, sttheta
   Similarly, these are XPtrs to the HBART s(x) objects.
fmu  The constant mu.
f.train, s.train
   The trained f(x) and s(x) respectively: matrices with ndpost rows and n
   columns.
f.train.mean, s.train.mean
   The posterior mean of the trained f(x) and s(x) respectively: vectors of length
   n.
f.trees, s.trees
   Character strings representing the trained fits of f(x) and s(x) respectively to
   facilitate usage of the predict function when XPtrs are unavailable.
dpalpha  The draws of the DPM concentration parameter alpha.
dpn, dpn. The number of atom clusters per DPM, $J$, for all draws including burn-in and excluding burn-in respectively.

dpmu. The draws of the DPM parameter $\mu[i]$ where $i = 1, \ldots, n$ indexes subjects: a matrix with $ndpost$ rows and $n$ columns.

dpmu. The draws of the DPM parameter $\mu[j]$ where $j = 1, \ldots, J$ indexes atom clusters: a matrix with $ndpost$ rows and $J$ columns.

dpwt. The weights for efficient DPM calculations by atom clusters (as opposed to subjects) for use with $dpmu$. (and $dpsd$; see below): a matrix with $ndpost$ rows and $J$ columns.

dpsd, dpsd. Similarly, the draws of the DPM parameter $\tau[i]$ transformed into the standard deviation $\sigma[i]$ for convenience.

dpC The indices $j$ for each subject $i$ corresponding to their shared atom cluster.

z.train The data values/augmentation draws of $\log t$.

f.tmind/f.tavgd/f.tmaxd
The min/average/max tier degree of trees in the $f$ ensemble.

s.tmind/s.tavgd/s.tmaxd
The min/average/max tier degree of trees in the $s$ ensemble.

f.varcount, s.varcount
Variable importance counts of branch decision rules for each $x$ of $f$ and $s$ respectively: matrices with $ndpost$ rows and $p$ columns.

f.varcount.mean, s.varcount.mean
Similarly, the posterior mean of the variable importance counts for each $x$ of $f$ and $s$ respectively: vectors of length $p$.

f.varprob, s.varprob
Similarly, re-weighting the posterior mean of the variable importance counts as sum-to-one probabilities for each $x$ of $f$ and $s$ respectively: vectors of length $p$.

LPML The log Pseudo-Marginal Likelihood as typically calculated for right-/left-censoring.

pred The object returned from the predict function where $x$.test=$x$.train in order to calculate the soffset item that is needed to use predict when $XPtrs$ are not available.

soffset See pred above.

aft The AFT model fit used to initialize NFT BART.

elapsed The elapsed time of the run in seconds.

Author(s)
Rodney Sparapani: <r.sparapa@mcw.edu>

References
See Also

predict.nft

Examples

B=getOption('mc.cores', 1)

data(lung)
str(lung)
N=length(lung$status)

#lung$status: 1=censored, 2=dead
#delta: 0=censored, 1=dead
delta=lung$status-1
table(delta)

## this study reports time in days rather than weeks or months

times=lung$time
times=times/7  ## weeks
summary(times)

## matrix of covariates
x.train=cbind(lung[, -(1:3)])
## lung$sex: Male=1 Female=2

## token run just to test installation
post=nft(x.train, times, delta, tc=B, K=0,
          nskip=0, ndpost=1, nadapt=1, adaptevery=1)

file.='lung.rds'
if(file.exists(file.)) {
  post=readRDS(file.)
  XPtr=FALSE
} else {
  set.seed(99)
  post=nft(x.train, times, delta, tc=B, K=0)
  XPtr=TRUE
  ##saveRDS(post, file.)
}

x.test = rbind(x.train, x.train)
x.test[, 2]=rep(1:2, each=N)
K=75
events=seq(0, 150, length.out=K+1)
pred = predict(post, x.test, K=K, events=events[-1],
               XPtr=XPtr, tc=B, FPD=TRUE)

plot(events, c(1, pred$surv.fpd.mean[1:K]), type='l', col=4,
     ylim=0:1,
     xlab=expression(italic(t)), sub='weeks',
The function `predict.nft()` is the main function for drawing posterior predictive realizations at new inputs using a fitted model stored in a `nft` object returned from `nft()`. 

### Usage

```r
## S3 method for class 'nft'
predict(
  object,
  x.test=object$x.train,
  tc=1,  ##OpenMP thread count
  XPtr=TRUE,
  K=0,
  events=object$events,
  FPD=FALSE,
  probs=c(0.025, 0.975),
  take.logs=TRUE,
  na.rm=FALSE,
  fmu=object$fmu,
  soffset=object$soffset,
  drawMuTau=object$drawMuTau,
  ...)  ## etc.
)
```

### Arguments

- `object` Object of type `nft` from a previous call to `nft()`. 

---

**Description**

The function `predict.nft()` is the main function for drawing posterior predictive realizations at new inputs using a fitted model stored in a `nft` object returned from `nft()`. 

### Example

```r
ylab=expression(italic(S)(italic(t), italic(x)))
lines(events, c(1, pred$surv.fpd.upper[1:K]), lty=2, lwd=2, col=4)
lines(events, c(1, pred$surv.fpd.lower[1:K]), lty=2, lwd=2, col=4)
lines(events, c(1, pred$surv.fpd.mean[K+1:K]), lwd=2, col=2)
lines(events, c(1, pred$surv.fpd.upper[K+1:K]), lty=2, lwd=2, col=2)
lines(events, c(1, pred$surv.fpd.lower[K+1:K]), lty=2, lwd=2, col=2)
legend('topright', c('Adv. lung cancer\n\nmortality example',
                     'M', 'F'), lwd=2, col=c(0, 4, 2), lty=1)
```
predict.nft

x.test
New input settings in the form of a matrix at which to construct predictions. Defaults to the training inputs.
tc
Number of OpenMP threads to use for parallel computing.
XPtr
If object was created during the currently running R process, then (via an Rcpp XPtr) the BART/HBART tree ensemble objects can be accessed in RAM; otherwise, those objects will need to be loaded from their string encodings.
K
The length of the grid of time-points to be used for survival predictions. Set to zero to avoid these calculations which can be time-consuming for large data sets.
events
You can specify the grid of time-points; otherwise, they are derived from quantiles of the augmented event times.
FPD
Whether to yield the usual predictions or marginal predictions calculated by the partial dependence function.
probs
A vector of length two containing the lower and upper quantiles to be calculated for the predictions.
take.logs
Whether or not to take logarithms.
na.rm
Whether NA values should be removed from the summaries.
fmu
BART centering parameter for the test data. Defaults to the value used by nft() when training the model.
soffset
HBART centering parameter for the test data. Defaults to the value used by nft() when training the model.
drawMuTau
Whether to use NFT BART with, or without, DPM.
...
The et cetera objects passed to the predict method. Currently, it has no functionality.

Details
predict.nft() is the main function for calculating posterior predictions and uncertainties once a model has been fit by nft().
Returns a list with the following entries.

Value
f.test
Posterior realizations of the mean function stored in a matrix. Omitted if partial dependence functions are performed since these will typically be large.
s.test
Posterior realizations of the SD function stored in a matrix. Omitted if partial dependence functions are performed since these will typically be large.
f.test.mean
Posterior predictive mean of mean function.
f.test.lower
Posterior predictive lower quantile of mean function.
f.test.upper
Posterior predictive upper quantile of mean function.
s.test.mean
Posterior predictive mean of SD function.
s.test.lower
Posterior predictive lower quantile of SD function.
s.test.upper
Posterior predictive upper quantile of SD function.
xicuts

 surv.fpd      Survival function posterior draws on a grid of time-points by the partial dependence function when requested.
 surv.fpd.mean Survival function estimates on a grid of time-points by the partial dependence function when requested.
 surv.fpd.lower Survival function lower quantiles on a grid of time-points by the partial dependence function when requested.
 surv.fpd.upper Survival function upper quantiles on a grid of time-points by the partial dependence function when requested.

Author(s)

Rodney Sparapani: rsparapa@mcw.edu

See Also

nft

Description

This function allows you to create a list that specifies the cut-points for the covariates.

Usage

xicuts(x.train, transposed=FALSE, numcut=100)

Arguments

x.train The training matrix to derive cut-points from.
transposed Whether or not the matrix has been transposed yet.
numcut The number of cut-points to create.

Details

The cut-points are generated uniformly from min. to max., i.e., the distribution of the data is ignored.

Value

An object is returned of type BARTcutinfo which is essentially a list.
Index

* datasets
  galaxy, 4
  lung, 5

bartModelMatrix, 2

cancer (lung), 5
CDimpute, 3

galaxy, 4
lung, 5
nft, 6, 13
predict.nft, 10, 11
xicuts, 3, 13