Package ‘trtf’

April 22, 2020

Title Transformation Trees and Forests
Version 0.3-7
Date 2020-04-22

Depends mlt (>= 1.0-2), partykit (>= 1.2-1)
Imports Formula, sandwich, grid, stats, variables, libcoin, utils
Suggests survival, TH.data, coin

URL http://ctm.R-forge.R-project.org
License GPL-2

R topics documented:

trtf-package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
traforest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
traftree . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Index 8
General Information on the \texttt{trtf} Package

Description

The \texttt{trtf} package implements transformation trees and transformation forests as described in Hothorn and Zeileis (2017).

Example applications of transformation trees and forests can be replicated using \texttt{demo("applications")} and \texttt{demo("BMI")}. Figure 1 in Hothorn and Zeileis (2017) can be reproduced by \texttt{demo("QRF")}.

Source code of simulation experiments is available in directory \texttt{trtf/inst/sim}.


Author(s)

This package is authored by Torsten Hothorn \texttt{<Torsten.Hothorn@R-project.org>}.

References


\begin{itemize}
\item \texttt{traforest} (\texttt{traforest})
\end{itemize}

\textbf{Description}

Partitioned and aggregated transformation models

\textbf{Usage}

\begin{verbatim}
traforest(object, parm = 1:length(coef(object)), reparm = NULL,
update = TRUE, min_update = length(coef(object)) * 2,
mltargs = list(maxit = 10000), ...)
## S3 method for class 'traforest'
predict(object, newdata, mnewdata = data.frame(1), K = 20, q = NULL,
type = c("weights", "node", "coef", "trafo", "distribution", "survivor", "density",
"logdensity", "hazard", "loghazard", "cumhazard", "quantile"),
OOB = FALSE, simplify = FALSE, trace = FALSE, updatestart = FALSE,
applyfun = NULL, cores = NULL, ...)
## S3 method for class 'traforest'
logLik(object, newdata, weights = NULL, OOB = FALSE, coef = NULL, ...)
\end{verbatim}
Arguments

object: an object of class `ctm` or `mlt` specifying the abstract model to be partitioned.
parm: parameters of `object` those corresponding score is used for finding partitions.
reparm: optional matrix of contrasts for reparameterisation of the scores. `teststat = "quadratic"` is invariant to this operation but `teststat = "max"` might be more powerful for example when formulating an implicit into an explicit intercept term.
mltargs: arguments to `mlt` for fitting the transformation models.
update: logical, if TRUE, models and thus scores are updated in every node. If FALSE, the model and scores are computed once in the root node. The latter option is faster but less accurate.
min_update: number of observations necessary to refit the model in a node. If less observations are available, the parameters from the parent node will be reused.
newdata: an optional data frame of observations for the forest.
mnewdata: an optional data frame of observations for the model.
K: number of grid points to generate (in the absence of `q`).
q: quantiles at which to evaluate the model.
type: type of prediction or plot to generate.
OOB: compute out-of-bag predictions.
simplify: simplify predictions (if possible).
trace: a logical indicating if a progress bar shall be printed while the predictions are computed.
updatetime: try to be smart about starting values for computing predictions (experimental).
applyfun: an optional `lapply`-style function with arguments `function(X,FUN,...)` for looping over `newdata`. The default is to use the basic `lapply` function unless the `cores` argument is specified (see below).
cores: numeric. If set to an integer the `applyfun` is set to `mclapply` with the desired number of `cores`.
weights: an optional vector of weights.
coef: an optional matrix of precomputed coefficients for `newdata` (using `predict`). Helps to compute the coefficients once for later reuse (different weights, for example).
...: arguments to `cforest`, at least formula and data.

Details

Conditional inference trees are used for partitioning likelihood-based transformation models as described in Hothorn and Zeileis (2017). The method can be seen in action in Hothorn (2018) and the corresponding code is available as demo("BMI").

Value

An object of class `traforest` with corresponding `logLik` and `predict` methods.
References


Examples

### Example: Personalised Medicine Using Partitioned and Aggregated Cox-Models


based on infrastructure in the mlt R add-on package described in

https://cran.r-project.org/web/packages/mlt.docreg/vignettes/mlt.pdf

library("trtf")
library("survival")

### German Breast Cancer Study Group 2 data set
data("GBSG2", package = "TH.data")

### set-up Cox model with overall treatment effect in hormonal therapy
yvar <- numeric_var("y", support = c(100, 2000), bounds = c(0, Inf))
By <- Bernstein_basis(yvar, order = 5, ui = "incre")
m <- ctm(response = By, shifting = ~ horTh, todistr = "MinExt", data = GBSG2)
GBSG2$y <- with(GBSG2, Surv(time, cens))

### overall log-hazard ratio
cof(cmod <- mlt(m, data = GBSG2))["horThyes"]

### Not run:

### estimate age-dependent Cox models (here ignoring all other covariates)
ctrl <- ctree_control(minsplit = 50, minbucket = 20, mincriterion = 0)
set.seed(290875)
tf_cmod <- traforest(m, formula = y ~ horTh | age, control = ctrl,
ntree = 50, mtry = 1, trace = TRUE, data = GBSG2)

### plot age-dependent treatment effects vs. overall treatment effect
nd <- data.frame(age = 30:70)
cf <- predict(tf_cmod, newdata = nd, type = "coef")
nd$logHR <- sapply(cf, function(x) x["horThyes"])
plot(logHR ~ age, data = nd, pch = 19, xlab = "Age", ylab = "log-Hazard Ratio")
abline(h = coef(cmod <- mlt(m, data = GBSG2))["horThyes"])

### treatment most beneficial in very young patients
### NOTE: scale of log-hazard ratios depends on
### corresponding baseline hazard function which differs
### across age; interpretation of positive / negative treatment effect is,
### however, save.
### mclapply doesn't work in Windows
if (.Platform$OS.type != "windows") {

### computing predictions: predicted coefficients
cf1 <- predict(tf_cmod, newdata = nd, type = "coef")
### speedup with plenty of RAM and 4 cores
cf2 <- predict(tf_cmod, newdata = nd, cores = 4, type = "coef")
### memory-efficient with low RAM and _one_ core
cf3 <- predict(tf_cmod, newdata = nd, cores = 4, applyfun = lapply, type = "coef")
all.equal(cf1, cf2)
all.equal(cf1, cf3)
}

## End(Not run)

---

## trafotree  
*Transformation Trees*

### Description

Partitioned transformation models

### Usage

```r
trafotree(object, parm = 1:length(coef(object)), reparm = NULL, 
  min_update = length(coef(object)) * 2, 
  mltargs = list(maxit = 10000), ...)
```

### S3 method for class 'trafotree'

```r
predict(object, newdata, K = 20, q = NULL, 
  type = c("node", "coef", "trafo" ,"distribution", "survivor", "density", 
  "logdensity", "hazard", "loghazard", "cumhazard", "quantile"), 
  perm = NULL, ...)
```

### S3 method for class 'trafotree'

```r
logLik(object, newdata, weights = NULL, perm = NULL, ...)
```

### Arguments

- **object**: an object of class `ctm` or `mlt` specifying the abstract model to be partitioned. For `predict` and `logLik`, object is an object of class `trafotree`.
- **parm**: parameters of object those corresponding score is used for finding partitions.
- **reparm**: optional matrix of contrasts for reparameterisation of the scores. `teststat = "quadratic"` is invariant to this operation but `teststat = "max"` might be more powerful for example when formulating an implicit into an explicit intercept term.
min_update  number of observations necessary to refit the model in a node. If less observations are available, the parameters from the parent node will be reused.

mltargs arguments to mlt for fitting the transformation models.

ewndata an optional data frame of observations.

K number of grid points to generate (in the absence of q).

q quantiles at which to evaluate the model.

type type of prediction or plot to generate.

weights an optional vector of weights.

perm a vector of integers specifying the variables to be permuted prior before splitting (i.e., for computing permutation variable importances). The default NULL doesn’t alter the data, see fitted_node.

... arguments to ctree, at least formula and data.

Details

Conditional inference trees are used for partitioning likelihood-based transformation models as described in Hothorn and Zeileis (2017). The method can be seen in action in Hothorn (2018) and the corresponding code is available as demo("BMI"). demo("applications") performs transformation tree analyses for some standard benchmarking problems.

Value

An object of class trafotree with corresponding plot, logLik and predict methods.

References


Examples

### Example: Stratified Medicine Using Partitioned Cox-Models
### based on infrastructure in the mlt R add-on package described in
### https://cran.r-project.org/web/packages/mlt.docreg/vignettes/mlt.pdf

library("trtf")
library("survival")
### German Breast Cancer Study Group 2 data set
data("GBSG2", package = "TH.data")
# set-up Cox model with overall treatment effect in hormonal therapy

```r
yvar <- numeric_var("y", support = c(100, 2000), bounds = c(0, Inf))
By <- Bernstein_basis(yvar, order = 5, ui = "incre")
m <- ctm(response = By, shifting = ~ horTh, todistr = "MinExt", data = GBSG2)
GBSG2$y <- with(GBSG2, Surv(time, cens))
```

### overall log-hazard ratio

```r
coef(cmod <- mlt(m, data = GBSG2))['horThyes']
```

### roughly the same as

```r
coef(coxph(y ~ horTh, data = GBSG2))
```

### partition the model, ie both the baseline hazard function AND the treatment effect

```r
(part_cmod <- trafotree(m, formula = y ~ horTh | age + menostat + tsize + tgrade + pnodes + progrec + estrec, data = GBSG2))
```

### compare the log-likelihoods

```r
logLik(cmod)
logLik(part_cmod)
```

### stronger effects in nodes 2 and 4 and no effect in node 5

```r
coef(part_cmod)[, "horThyes"]
```

### plot the conditional survivor functions; blue is untreated and green is hormonal therapy

```r
nd <- data.frame(horTh = sort(unique(GBSG2$horTh)))
plot(part_cmod, newdata = nd, 
      tp_args = list(type = "survivor", col = c("cadetblue3", "chartreuse4")))
```

### same model, but with explicit intercept term and max-type statistic

```r
K <- diag(length(coef(m)) - 1)
K[upper.tri(K)] <- 1
K <- cbind(rbind(K, 0), 0)
K[nrow(K), nrow(K)] <- 1

### horThyes is not touched, 6th parameter is intercept

```r
coef(cmod)
```

```r
(part_cmod_max <- trafotree(m, formula = y ~ horTh | age + menostat + tsize + tgrade + pnodes + progrec + estrec, data = GBSG2, reparm = K, 
control = ctree_control(teststat = "max")))
```

### the trees (and log-likelihoods are the same) but the p-values are sometimes much smaller in the latter tree

```r
cbind(format.pval(info_node(node_party(part_cmod))$criterion["p.value",]), 
      format.pval(info_node(node_party(part_cmod_max))$criterion["p.value",]))
```
Index

★Topic **package**
   trtf-package, 2

★Topic **trees**
   trafotree, 5
   trafotree, 5

 cforest, 3
 ctm, 3, 5
 ctree, 6

 fitted_node, 6

 logLik.traforest (traforest), 2
 logLik.trafotree (trafotree), 5

 mlt, 3, 5, 6

 predict.traforest (traforest), 2
 predict.trafotree (trafotree), 5

 trafotree, 2
 trafotree, 5
 trtf-package, 2