Package 'RLT'

April 28, 2023

Version 3.2.6

Date 2023-04-28

Title Reinforcement Learning Trees

Suggests randomForest, survival

Description Random forest with a variety of additional features for regression, classification and survival analysis.
The features include: parallel computing with OpenMP, embedded model for selecting the splitting variable,
based on Zhu, Zeng & Kosorok (2015) <doi:10.1080/01621459.2015.1036994>, subject weight, variable weight,
tracking subjects used in each tree, etc.

License GPL (>= 2)

URL https://cran.r-project.org/package=RLT

NeedsCompilation yes

Repository CRAN

Encoding UTF-8

RoxygenNote 7.2.3

Author Ruoqing Zhu [aut, cre, cph] (<https://orcid.org/0000-0002-0753-5716>)

Maintainer Ruoqing Zhu <teazrq@gmail.com>

Date/Publication 2023-04-28 10:50:02 UTC

R topics documented:

MuteRate *Muting rate*

Description

Get the muting rate based on sample size N and dimension P. This is an experimental feature. When P is too small, this is not recommended.

Usage

```
MuteRate(N, P, speed = NULL, info = FALSE)
```

Arguments

N	sample size
P	dimension
speed	Muting speed: moderate or aggressive
info	Whether to output detailed information

Value

A suggested muting rate

Examples

```
MuteRate(500, 100, speed = "aggressive")
```

predict.RLT *Prediction function for reinforcement learning trees*

Description

Predict future subjects with a fitted RLT model

Usage

```
## S3 method for class 'RLT'
predict(object, testx, ...)
```

Arguments

object	A fitted RLT object
testx	Testing data
...	...

Value

The predicted values. For survival model, it returns the fitted survival functions

Examples

```
x = matrix(rnorm(100), ncol = 10)
y = rowMeans(x)
fit = RLT(x, y, ntrees = 5)
predict(fit, x)
```

`print.RLT` *Print a RLT object*

Description

Print a RLT object

Usage

```
## S3 method for class 'RLT'
print(x, ...)
```

Arguments

`x`	A fitted RLT object
`...`	...

Value

No return value

Examples

```
x = matrix(rnorm(100), ncol = 10)
y = rowMeans(x)
fit = RLT(x, y, ntrees = 5)
fit
```

RLT *Reinforcement Learning Trees*

Description

Fit models for regression, classification and survival analysis using reinforced splitting rules

Usage

```
RLT(
  x,
  y,
  censor = NULL,
  model = "regression",
  print.summary = 0,
  use.cores = 1,
  ntrees = if (reinforcement) 100 else 500,
  mtry = max(1, as.integer(ncol(x)/3)),
  nmin = max(1, as.integer(log(nrow(x)))),
  alpha = 0.4,
  split.gen = "random",
  nsplit = 1,
  resample.prob = 0.9,
  replacement = TRUE,
  npermute = 1,
  select.method = "var",
  subject.weight = NULL,
  variable.weight = NULL,
  track.obs = FALSE,
  importance = TRUE,
  reinforcement = FALSE,
  muting = -1,
 muting.percent = if (reinforcement) MuteRate(nrow(x), ncol(x), speed = "aggressive",
    info = FALSE) else 0,
  protect = as.integer(log(ncol(x))),
  combsplit = 1,
  combsplit.th = 0.25,
  random.select = 0,
  embed.n.th = 4 * nmin,
  embed.ntrees = max(1, -atan(0.01 * (ncol(x) - 500))/pi * 100 + 50),
  embed.resample.prob = 0.8,
  embed.mtry = 1/2,
  embed.nmin = as.integer(nrow(x)^(1/3)),
  embed.split.gen = "random",
  embed.nsplit = 1
)
```

Arguments

x	A matrix or data.frame for features
y	Response variable, a numeric/factor vector or a Surv object
censor	The censoring indicator if survival model is used
model	The model type: `regression`, `classification` or `survival`
print.summary	Whether summary should be printed
use.cores	Number of cores
ntrees	Number of trees, `ntrees = 100` if use reinforcement, `ntrees = 1000` otherwise
mtry	Number of variables used at each internal node, only for `reinforcement = FALSE`
nmin	Minimum number of observations required in an internal node to perform a split. Set this to twice of the desired terminal node size.
alpha	Minimum number of observations required for each child node as a portion of the parent node. Must be within `(0, 0.5]`.
split.gen	How the cutting points are generated
nsplit	Number of random cutting points to compare for each variable at an internal node
resample.prob	Proportion of in-bag samples
replacement	Whether the in-bag samples are sampled with replacement
npermute	Number of imputations (currently not implemented, saved for future use)
select.method	Method to compare different splits
subject.weight	Subject weights
variable.weight	Variable weights when randomly sample `mtry` to select the splitting rule
track.obs	Track which terminal node the observation belongs to
importance	Should importance measures be calculated
reinforcement	If reinforcement splitting rules should be used. There are default values for all tuning parameters under this feature.
muting	Muting method, `-1` for muting by proportion, positive for muting by count
muting.percent	Only for `muting = -1` the proportion of muting
protect	Number of protected variables that will not be muted. These variables are adaptively selected for each tree.
combsplit	Number of variables used in a combination split. `combsplit = 1` gives regular binary split; `combsplit > 1` produces linear combination splits.
combsplit.th	The minimum threshold (as a relative measurement compared to the best variable) for a variable to be used in the combination split.
random.select	Randomly select a variable from the top variable in the linear combination as the splitting rule.
embed.n.th	Number of observations to stop the embedded model and choose randomly from the current protected variables.

`embed.ntrees` Number of embedded trees

`embed.resample.prob` Proportion of in-bag samples for embedded trees

`embed.mtry` Number of variables used for embedded trees, as proportion

`embed.nmin` Terminal node size for embedded trees

`embed.split.gen` How the cutting points are generated in the embedded trees

`embed.nsplit` Number of random cutting points for embedded trees

Value

A `RLT` object; a list consisting of

`FittedTrees` Fitted tree structure

`FittedSurv, timepoints` Terminal node survival estimation and all time points, if survival model is used

`AllError` All out-of-bag errors, if `importance = TRUE`

`VarImp` Variable importance measures, if `importance = TRUE`

`ObsTrack` Registration of each observation in each fitted tree

`...` All the tuning parameters are saved in the fitted `RLT` object

References

Zhu, R., Zeng, D., & Kosorok, M. R. (2015) "Reinforcement Learning Trees." Journal of the American Statistical Association. 110(512), 1770-1784.

Zhu, R., & Kosorok, M. R. (2012). Recursively imputed survival trees. Journal of the American Statistical Association, 107(497), 331-340.

Examples

```
N = 600
P = 100

X = matrix(runif(N*P), N, P)
Y = rowSums(X[,1:5]) + rnorm(N)

trainx = X[1:200,]
trainy = Y[1:200]
testx = X[-c(1:200),]
testy = Y[-c(1:200)]

# Regular ensemble trees (Extremely Randomized Trees, Geurts, et. al., 2006)

RLT.fit = RLT(trainx, trainy, model = "regression", use.cores = 6)

barplot(RLT.fit$VarImp)
RLT.pred = predict(RLT.fit, testx)
```

```
mean((RLT.pred$Prediction - testy)^2)

# Reinforcement Learning Trees, using an embedded model to find the splitting rule

Mark0 = proc.time()
RLT.fit = RLT(trainx, trainy, model = "regression", use.cores = 6, ntrees = 100,
              importance = TRUE, reinforcement = TRUE, combsplit = 3, embed.ntrees = 25)
proc.time() - Mark0

barplot(RLT.fit$VarImp)
RLT.pred = predict(RLT.fit, testx)
mean((RLT.pred$Prediction - testy)^2)
```

Index