Package: RLT (via r-universe)

September 2, 2024

Type Package

Title Reinforcement Learning Trees

Version 4.2.6

Description Random forest with a variety of additional features for regression, classification, survival analysis and graphical model. New features include parallel computing with OpenMP, reproduciblity with random seeds, variance and confidence band estimations, embedded model for selecting splitting varibles and constructing linear combination splits, observaton and variable weights, setting and tracking subjects used in each tree, etc.

License GPL (>= 2)

Encoding UTF-8

LazyData TRUE

Imports Rcpp (>= 1.0.9), stats, utils, Matrix, orthoDr, glmnet

LinkingTo Rcpp, RcppArmadillo, dqrng, BH, sitmo

URL `https://cran.r-project.org/package=RLT`, `https://teazrq.github.io/RLT/`

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Repository https://teazrq.r-universe.dev

RemoteUrl https://github.com/teazrq/rlt

RemoteRef HEAD

RemoteSha de8b11f0717280b94cec79fddd9ee29dd3725873

Contents

`cindex` *C-index*

Description

Calculate c-index for survival data

Usage

```
cindex(y, censor, pred)
```

Arguments

`y`	survival time
`censor`	The censoring indicator if survival model is used
`pred`	the predicted value for each subject

Value

c-index

`forest.kernel` `random forest kernel`

Description

```
Get random forest induced kernel weight matrix of testing samples
             or between any two sets of data. This is an experimental feature.
             Use at your own risk.
```

Usage

```
forest.kernel(
  object,
  X1 = NULL,
  X2 = NULL,
  vs.train = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	A fitted RLT object.
`X1`	The dataset for prediction. This calculates an $n_1 \times n_1$ kernel matrix of `X1`.
`X2`	The dataset for reference/training. If `X2` is supplied, then calculate an $n_1 \times n_2$ kernel matrix. If `vs.train` is used, then this must be the original training data.
`vs.train`	To calculate the kernel weights with respect to the training data. This is slightly different than supplying the training data to `X2` due to re-samplings of the training process. To use this feature, you must specify `resample.track = TRUE` in `param.control` when fitting the forest
`verbose`	Whether fitting should be printed.
`...`	... Additional arguments.

Value

A kernel matrix that contains kernel weights for each observation in `X1` with respect to `X1`

`get.one.tree` *Print a single tree*

Description

Print a single fitted tree from a forest object

Usage

```
get.one.tree(x, tree = 1, ...)
```

Arguments

`x`	A fitted RLT object
`tree`	the tree number, starting from 1 to `ntrees`.
`...`	...

get.surv.band

`get.surv.band`

Description

```
Calculate the survival function (two-sided) confidence band from
             a RLT survival prediction.
```

Usage

```
get.surv.band(
  x,
  i = 0,
  alpha = 0.05,
  approach = "naive-mc",
  nsim = 1000,
  r = 3,
  ...
)
```

Arguments

`x`	A RLT prediction object. This must be an object calculated from a forest with `var.ready = TRUE`.
`i`	Observation number in the prediction. Default to calculate all ($i = 0$)
`alpha`	alpha level for interval $(\alpha/2, 1-\alpha/2)$
`approach`	What approach is used to calculate the confidence band. Can be • `naive-mc`: positive-definite projection of the covariance matrix. the confidence band is non-smooth • `smoothed-mc`: use a smoothed marginal variance to perform the Monte Carlo approximation of the critical value. This is only recommended for large number of time points. • `smoothed-lr`: use a smoothed low-rank approximation of the covariance matrix and apply an adaptive Bonferroni correction to derive the critical values. Note that this approach relies on the assumption of the smoothness and low rank of the covariance matrix.
`nsim`	number of simulations for estimating the Monte Carlo critical value. Set this to be a large number. Default is 1000.
`r`	maximum number of ranks used in the `smoothed-lr` approximation. Usually 5 is enough for approximating the covariance matrix due to smoothness.
`...`	...

mytest *mytest*

Description

my function

Usage

```
mytest(n, ...)
```

Arguments

`n`	n
`...`	other arguments

Value

output

predict.RLT *prediction using RLT*

Description

Predict the outcome (regression, classification or survival) using a fitted RLT object

Usage

```
## S3 method for class 'RLT'
predict(
  object,
  testx = NULL,
  var.est = FALSE,
  keep.all = FALSE,
  ncores = 1,
  verbose = 0,
  ...
)
```

Arguments

`object`	A fitted RLT object
`testx`	The testing samples, must have the same structure as the training samples
`var.est`	Whether to estimate the variance of each testing data. The original forest must be fitted with `var.ready = TRUE`. For survival forests, calculates the covariance matrix over all observed time points and calculates critical value for the confidence band.
`keep.all`	whether to keep the prediction from all trees. Warning: this can occupy a large storage space, especially in survival model
`ncores`	number of cores
`verbose`	print additional information
`...`	...

Value

A `RLT` prediction object, constructed as a list consisting

`Prediction`	Prediction
`Variance`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`

For Survival Forests

`hazard`	predicted hazard functions
`CumHazard`	predicted cumulative hazard function
`Survival`	predicted survival function
`Allhazard`	if `keep.all = TRUE`, the predicted hazard function for each observation and each tree
`AllCHF`	if `keep.all = TRUE`, the predicted cumulative hazard function for each observation and each tree
`Cov`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`. For each test subject, a matrix of size NFail$\times$NFail where NFail is the number of observed failure times in the training data
`Var`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`. Marginal variance for each subject
`timepoints`	ordered observed failure times from the training data
`MarginalVar`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`. Marginal variance for each subject from the Cov matrix projected to the nearest positive definite matrix
`MarginalVarSmooth`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`. Marginal variance for each subject from the Cov matrix projected to the nearest positive definite matrix and then smoothed using Gaussian kernel smoothing
`CVproj`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`. Critical values to calculate confidence bands around cumulative hazard predictions at several confidence levels. Calculated using `MarginalVar`

`CVprojSmooth`	if `var.est = TRUE` and the fitted object is `var.ready = TRUE`. Critical values to calculate confidence bands around cumulative hazard predictions at several confidence levels. Calculated using `MarginalVarSmooth`

`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
`...`	...

`RLT`

`Reinforcement Learning Trees`

Description

```
        Fit models for regression, classification and survival
                     analysis using reinforced splitting rules. The model
                     fits regular random forest models by default unless the
                     parameter \code{reinforcement} is set to `"TRUE"`. Using
                     \code{reinforcement = TRUE} activates embedded model for
                     splitting variable selection and allows linear combination
                     split. To specify parameters of embedded models, see
                     definition of \code{param.control} for details.
```

Usage

```
RLT(
  x,
  y,
  censor = NULL,
  model = NULL,
  ntrees = if (reinforcement) 100 else 500,
  mtry = max(1, as.integer(ncol(x)/3)),
  nmin = max(1, as.integer(log(nrow(x)))),
  split.gen = "random",
  nsplit = 1,
  resample.replace = TRUE,
  resample.prob = if (resample.replace) 1 else 0.8,
  resample.preset = NULL,
  obs.w = NULL,
  var.w = NULL,
  importance = FALSE,
  reinforcement = FALSE,
  param.control = list(),
  ncores = 0,
  verbose = 0,
  seed = NULL,
  ...
)
```

Arguments

`x`	A `matrix` or `data.frame` of features. If `x` is a data.frame, then all factors are treated as categorical variables, which will go through an exhaustive search of splitting criteria.
`y`	Response variable. a `numeric/factor` vector.
`censor`	Censoring indicator if survival model is used.
`model`	The model type: `"regression"`, `"classification"`, `"quantile"`, `"survival"` or `"graph"`.
`ntrees`	Number of trees, `ntrees = 100` if reinforcement is used and `ntrees = 1000` otherwise.
`mtry`	Number of randomly selected variables used at each internal node.
`nmin`	Terminal node size. Splitting will stop when the internal node size is less equal to `nmin`.
`split.gen`	How the cutting points are generated: `"random"`, `"rank"` or `"best"`. If minimum child node size is enforced (`alpha` $> 0$), then `"rank"` and `"best"` should be used.
`nsplit`	Number of random cutting points to compare for each variable at an internal node.
`resample.replace`	Whether the in-bag samples are obtained with replacement.
`resample.prob`	Proportion of in-bag samples.
`resample.preset`	A pre-specified matrix for in-bag data indicator/count matrix. It must be an $n\times$ `ntrees` matrix with integer entries. Positive number indicates the number of copies of that observation (row) in the corresponding tree (column); zero indicates out-of-bag; negative values indicates not being used in either. Extremely large counts should be avoided. The sum of each column should not exceed n.

`obs.w`	Observation weights. The weights will be used for calculating the splitting scores, such as a weighted variance reduction or weighted gini index. But they will not be used for sampling observations. In that case, one can pre-specify `resample.preset` instead for balanced sampling, etc. For survival analysis, observation weights are not implemented in the `"logrank"` or `"suplogrank"` tests, due to the difficulty of calculating the variance of test statistic. However, it is used in the `"coxgrad"` splitting rule. For other models, this feature is currently not available.
`var.w`	Variable weights. If this is supplied, the default is to perform weighted sampling of `mtry` variables. For other usage, see the details of `split.rule` in `param.control`.
`importance`	Whether to calculate variable importance measures. When set to `"TRUE"` (or `"permute"`), the calculation follows Breiman's original permutation strategy. If set to `"distribute"`, then it sends the oob data to both child nodes with weights proportional to their sample sizes. Hence the final prediction is a weighted average of all possible terminal nodes that a perturbed observation could fall into. This feature is currently only available in regression and classification models.
`reinforcement`	Should reinforcement splitting rule be used. Default is `"FALSE"`, i.e., regular random forests with marginal search of splitting variable. When it is activated, an embedded model is fitted to find the best splitting variable or a linear combination of them, if `linear.comb` $> 1$. They can also be specified in `param.control`.
`param.control`	A list of additional parameters. This can be used to specify other features in a random forest or set embedded model parameters for reinforcement splitting rules. Using `reinforcement = TRUE` will automatically generate some default tuning for the embedded model. This mode is currently only available in regression. They are not necessarily optimized. • `embed.ntrees`: number of trees in the embedded model • `embed.mtry`: number or proportion of variables • `embed.nmin`: terminal node size • `embed.split.gen` random cutting point search method (`"random"`, `"rank"` or `"best"`) • `embed.nsplit` number of random cutting points • `embed.resample.replace` whether to sample with replacement • `embed.resample.prob`: proportion of samples (of the internal node) in the embedded model • `embed.mute` muting rate • `embed.protect` number of protected variables • `embed.threshold` threshold, as a fraction of the best VI, for being included in the protected set at an internal node.

```
 \code{linear.comb} is a separate feature that can be
activated with or without using reinforcement. It creates
linear combination of features as the splitting rule.
      Currently only available for regression.
      \itemize{
```

```
\item In reinforcement mode, a linear combination is created
    using the top continuous variables from the embedded
    model. If a categorical variable is the best, then
     a regular split will be used. The splitting point
    will be searched based on \code{split.rule} of the
            model.
\item In non-reinforcement mode, a marginal screening
    is performed and the top features are used to construct
    the linear combination. This is an experimental feature.
      }

\code{split.rule} is used to specify the criteria used
to compare different splittings. Here are the available
      choices. The first one is the default:
      \itemize{
\item Regression: `"var"` (variance reduction); `"pca"`
    and `"sir"` can be used for linear combination splits
      \item Classification: `"gini"` (gini index)
\item Survival: `"logrank"` (log-rank test), `"suplogrank"`,
            `"coxgrad"`.
   \item Quantile: `"ks"` (Kolmogorov-Smirnov test)
\item Graph: `"spectral"` (spectral embedding with variance
      reduction)
      }

\code{resample.track} indicates whether to keep track
      of the observations used in each tree.

\code{var.ready} this is a feature to allow calculating variance
(hence confidence intervals) of the random forest prediction.
Currently only available for regression (Xu, Zhu & Shao, 2023)
and confidence band in survival models (Formentini, Liang & Zhu, 2023).
 Please note that this only perpares the model fitting
 so that it is ready for the calculation. To obtain the
confidence intervals, please see the prediction function.
Specifying \code{var.ready = TRUE} has the following effect
if these parameters are not already provided. For details
 of their restrictions, please see the orignal paper.
      \itemize{
\item \code{resample.preset} is constructed automatically
    \item \code{resample.replace} is set to `FALSE`
    \item \code{resample.prob} is set to \eqn{n / 2}
      \item \code{resample.track} is set to `TRUE`
      }

 It is recommended to use a very large \code{ntrees},
e.g, 10000 or larger. For \code{resample.prob} greater
  than \eqn{n / 2}, one should consider the bootstrap
```

```
                    approach in Xu, Zhu & Shao (2023).

                  \code{alpha} force a minimum proportion of samples
                    (of the parent node) in each child node.

                \code{failcount} specifies the unique number of failure
                time points used in survival model. By default, all failure
                time points will be used. A smaller number may speed up
                the computation. The time points will be chosen uniformly
                on the quantiles of failure times, while must include the
                    minimum and the maximum.
```

`ncores`	Number of CPU logical cores. Default is 0 (using all available cores).
`verbose`	Whether info should be printed.
`seed`	Random seed number to replicate a previously fitted forest. Internally, the `xoshiro256++` generator is used. If not specified, a seed will be generated automatically and recorded.
`...`	Additional arguments.

Value

A `RLT` fitted object, constructed as a list consisting

- FittedForestFitted tree structures
- VarImpVariable importance measures, if `importance = TRUE`
- PredictionOut-of-bag prediction
- ErrorOut-of-bag prediction error, adaptive to the model type
- ObsTrackProvided if `resample.track = TRUE`, `var.ready = TRUE`, or if `resample.preset` was supplied. This is an `n` $\times$ `ntrees` matrix that has the same meaning as `resample.preset`.

For classification forests, these items are further provided or will replace the regression version

- NClassThe number of classes
- ProbOut-of-bag predicted probability

For survival forests, these items are further provided or will replace the regression version

- timepointsordered observed failure times
- NFailThe number of observed failure times
- PredictionOut-of-bag prediciton of hazard function

References

- Zhu, R., Zeng, D., & Kosorok, M. R. (2015) "Reinforcement Learning Trees." Journal of the American Statistical Association. 110(512), 1770-1784.
- Xu, T., Zhu, R., & Shao, X. (2023) "On Variance Estimation of Random Forests with Infinite-Order U-statistics." arXiv preprint arXiv:2202.09008.
- Formentini, S. E., Wei L., & Zhu, R. (2022) "Confidence Band Estimation for Survival Random Forests." arXiv preprint arXiv:2204.12038.

Index