

Package ‘tuneRanger’

March 20, 2024

Type Package

Title Tune Random Forest of the 'ranger' Package

Description Tuning random forest with one line. The package is mainly based on the packages 'ranger' and 'mlrMBO'.

Version 0.7

License GPL-3

Encoding UTF-8

Depends R (>= 3.0.2), ranger (>= 0.8.0), mlrMBO (>= 1.1.1), parallel (>= 3.3.3), lhs (>= 0.14)

Imports mlr (>= 2.11), smooof (>= 1.5.1), ParamHelpers (>= 1.10), methods (>= 3.3.3), BBmisc (>= 1.11), DiceKriging (>= 1.5.5)

RoxygenNote 7.3.1

Suggests survival, testthat

NeedsCompilation no

Author Philipp Probst [aut, cre],
Simon Klau [ctb]

Maintainer Philipp Probst <philipp_probst@gmx.de>

Repository CRAN

Date/Publication 2024-03-20 22:20:02 UTC

R topics documented:

estimateTimeTuneRanger	2
restartTuneRanger	3
tuneMtryFast	4
tuneRanger	5

Index	8
--------------	----------

estimateTimeTuneRanger

estimateTimeTuneRanger

Description

estimateTimeTuneRanger

Usage

```
estimateTimeTuneRanger(  
  task,  
  iters = 100,  
  num.threads = 1,  
  num.trees = 1000,  
  respect.unordered.factors = "order"  
)
```

Arguments

task	The mlr task created by makeClassifTask or makeRegrTask.
iters	Number of iterations.
num.threads	Number of threads. Default is 1.
num.trees	Number of trees.
respect.unordered.factors	Handling of unordered factor covariates. One of 'ignore', 'order' and 'partition'. 'order' is the default.

Value

estimated time for the tuning procedure

Examples

```
estimateTimeTuneRanger(iris.task)
```

```
restartTuneRanger      restartTuneRanger
```

Description

Restarts the tuning process if an error occurred.

Usage

```
restartTuneRanger(save.file.path = "optpath.RData", task, measure = NULL)
```

Arguments

`save.file.path` File name in the current working directory to which interim results were saved by `tuneRanger`.

`task` The mlr task created by `makeClassifTask` or `makeRegrTask`.

`measure` Performance measure that was already used in the original `tuneRanger` process.

Value

A list with elements

- `recommended.pars` Recommended hyperparameters.
- `results` A data.frame with all evaluated hyperparameters and performance and time results for each run.

No model is build.

Examples

```
## Not run:
library(tuneRanger)
library(mlr)

# iris is a bit nonsense here
# A mlr task has to be created in order to use the package
# the already existing iris task is used here
estimateTimeTuneRanger(iris.task)
# temporarily file name to save results
path = tempfile()
res = tuneRanger(iris.task, measure = list(multiclass.brier), num.trees = 1000,
  num.threads = 8, iters = 70, save.file.path = path)

# Mean of best 5 % of the results
res

# Restart after failing in one of the iterations:
res = restartTuneRanger(save.file.path = path, iris.task,
```

```
measure = list(multiclass.brier))
## End(Not run)
```

tuneMtryFast

tuneMtryFast

Description

Similar to tuneRF in [randomForest](#) but for [ranger](#).

Usage

```
tuneMtryFast(
  formula = NULL,
  data = NULL,
  dependent.variable.name = NULL,
  mtryStart = floor(sqrt(ncol(data) - 1)),
  num.treesTry = 50,
  stepFactor = 2,
  improve = 0.05,
  trace = TRUE,
  plot = TRUE,
  doBest = FALSE,
  ...
)
```

Arguments

formula	Object of class formula or character describing the model to fit. Interaction terms supported only for numerical variables.
data	Training data of class data.frame, matrix, dgCMatrix (Matrix) or gwaal.data (GenABEL).
dependent.variable.name	Name of dependent variable, needed if no formula given. For survival forests this is the time variable.
mtryStart	starting value of mtry; default is the same as in ranger
num.treesTry	number of trees used at the tuning step
stepFactor	at each iteration, mtry is inflated (or deflated) by this value
improve	the (relative) improvement in OOB error must be by this much for the search to continue
trace	whether to print the progress of the search
plot	whether to plot the OOB error as function of mtry
doBest	whether to run a forest using the optimal mtry found
...	options to be given to ranger

Details

Provides fast tuning for the `mtry` hyperparameter.

Starting with the default value of `mtry`, search for the optimal value (with respect to Out-of-Bag error estimate) of `mtry` for `randomForest`.

Value

If `doBest=FALSE` (default), it returns a matrix whose first column contains the `mtry` values searched, and the second column the corresponding OOB error.

If `doBest=TRUE`, it returns the [ranger](#) object produced with the optimal `mtry`.

Examples

```
library(tuneRanger)

data(iris)
res <- tuneMtryFast(Species ~ ., data = iris, stepFactor = 1.5)
```

tuneRanger

tuneRanger

Description

Automatic tuning of random forests of the [ranger](#) package with one line of code.

Usage

```
tuneRanger(
  task,
  measure = NULL,
  iters = 70,
  iters.warmup = 30,
  time.budget = NULL,
  num.threads = NULL,
  num.trees = 1000,
  parameters = list(replace = FALSE, respect.unordered.factors = "order"),
  tune.parameters = c("mtry", "min.node.size", "sample.fraction"),
  save.file.path = NULL,
  build.final.model = TRUE,
  show.info = getOption("mlrMBO.show.info", TRUE)
)
```

Arguments

<code>task</code>	The mlr task created by makeClassifTask , makeRegrTask or makeSurvTask .
<code>measure</code>	Performance measure to evaluate/optimize. Default is brier score for classification and mse for regression. Can be changed to accuracy, AUC or logarithmic loss by setting it to <code>list(acc)</code> , <code>list(auc)</code> or <code>list(logloss)</code> . Other possible performance measures from mlr can be looked up in the mlr tutorial .
<code>iters</code>	Number of iterations. Default is 70.
<code>iters.warmup</code>	Number of iterations for the warmup. Default is 30.
<code>time.budget</code>	Running time budget in seconds. Note that the actual mbo run can take more time since the condition is checked after each iteration. The default NULL means: There is no time budget.
<code>num.threads</code>	Number of threads. Default is number of CPUs available.
<code>num.trees</code>	Number of trees.
<code>parameters</code>	Optional list of fixed named parameters that should be passed to ranger .
<code>tune.parameters</code>	Optional character vector of parameters that should be tuned. Default is <code>mtry</code> , <code>min.node.size</code> and <code>sample.fraction</code> . Additionally <code>replace</code> and <code>respect.unordered.factors</code> can be included in the tuning process.
<code>save.file.path</code>	File to which interim results are saved (e.g. "optpath.RData") in the current working directory. Default is NULL, which does not save the results. If a file was specified and one iteration fails the algorithm can be started again with restartTuneRanger .
<code>build.final.model</code>	[logical(1)] Should the best found model be fitted on the complete dataset? Default is TRUE.
<code>show.info</code>	Verbose mlrMBO output on console? Default is TRUE.

Details

Model based optimization is used as tuning strategy and the three parameters `min.node.size`, `sample.fraction` and `mtry` are tuned at once. Out-of-bag predictions are used for evaluation, which makes it much faster than other packages and tuning strategies that use for example 5-fold cross-validation. Classification as well as regression is supported. The measure that should be optimized can be chosen from the list of measures in mlr: [mlr tutorial](#)

Value

A list with elements	
<code>recommended.pars</code>	Recommended hyperparameters.
<code>results</code>	A data.frame with all evaluated hyperparameters and performance and time results for each run.
<code>model</code>	The final model if <code>build.final.model</code> set to TRUE.

See Also

[estimateTimeTuneRanger](#) for time estimation and [restartTuneRanger](#) for continuing the algorithm if there was an error.

Examples

```
## Not run:
library(tuneRanger)
library(mlr)

# A mlr task has to be created in order to use the package
data(iris)
iris.task = makeClassifTask(data = iris, target = "Species")

# Estimate runtime
estimateTimeTuneRanger(iris.task)
# Tuning
res = tuneRanger(iris.task, measure = list(multiclass.brier), num.trees = 1000,
  num.threads = 2, iters = 70, save.file.path = NULL)

# Mean of best 5 % of the results
res
# Model with the new tuned hyperparameters
res$model
# Prediction
predict(res$model, newdata = iris[1:10,])
## End(Not run)
```

Index

`estimateTimeTuneRanger`, 2, 7

`makeClassifTask`, 3, 6

`makeRegrTask`, 3, 6

`makeSurvTask`, 6

`randomForest`, 4

`ranger`, 4–6

`restartTuneRanger`, 3, 6, 7

`tuneMtryFast`, 4

`tuneRanger`, 3, 5