Package 'mlr3tuningspaces'

April 12, 2022

Title Search Spaces for Hyperparameter Tuning

Version 0.2.0

Description Collection of search spaces for hyperparameter tuning. Includes various search spaces that can be directly applied on an `mlr3` learner. Additionally, meta information about the search space can be queried.

License LGPL-3

URL https://mlr3tuningspaces.mlr-org.com, https://github.com/mlr-org/mlr3tuningspaces

BugReports https://github.com/mlr-org/mlr3tuningspaces/issues

Depends mlr3tuning (>= 0.12.1), R (>= 3.1.0)

Imports checkmate (>= 2.0.0), data.table (>= 1.14.0), mlr3 (>= 0.11.0), mlr3misc (>= 0.9.1), paradox (>= 0.7.1), R6 (>= 2.5.0)

Suggests e1071 (>= 1.7-6), glmnet (>= 4.1-2), kknn (>= 1.3.1), mlr3learners (>= 0.4.5), ranger (>= 0.12.1), rpart (>= 4.1-15), testthat (>= 3.0.0), xgboost (>= 1.4.1.1)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.2

Collate 'mlr_tuning_spaces.R' 'TuningSpace.R' 'bibentries.R' 'helper.R' 'sugar.R' 'tuning_spaces_default.R' 'tuning_spaces_rbv2.R' 'zzz.R'

NeedsCompilation no

Author Marc Becker [cre, aut] (<https://orcid.org/0000-0002-8115-0400>), Michel Lang [ctb] (<https://orcid.org/0000-0001-9754-0393>)

Maintainer Marc Becker <marcbecker@posteo.de>

Repository CRAN

Date/Publication 2022-04-12 07:12:34 UTC

R topics documented:

`mlr3tuningspaces-package`
mlr3tuningspaces: Search Spaces for Hyperparameter Tuning

Description

Collection of search spaces for hyperparameter tuning. Includes various search spaces that can be directly applied on an ‘mlr3‘ learner. Additionally, meta information about the search space can be queried.

Author(s)

Maintainer: Marc Becker <marcbecker@posteo.de> (ORCID)

Other contributors:

- Michel Lang <michellang@gmail.com> (ORCID) [contributor]

See Also

Useful links:

- https://mlr3tuningspaces.mlr-org.com
- https://github.com/mlr-org/mlr3tuningspaces
- Report bugs at https://github.com/mlr-org/mlr3tuningspaces/issues

lts *Syntactic Sugar for Tuning Space Construction*

Description

This function complements mlr_tuning_spaces with functions in the spirit of mlr3::mlr_sugar.

Usage

```
lts(x)

## S3 method for class 'character'
lts(x)

## S3 method for class 'Learner'
lts(x)

ltss(x)
```

Arguments

x	(character() | mlr3::Learner) If character, key passed the dictionary to retrieve the tuning space. If mlr3::Learner, default tuning space is added to the learner.

Value

- TuningSpace for lts()
- list of TuningSpace for ltss()

Examples

```
lts("classif.ranger.default")
```

mlr_tuning_spaces *Dictionary of Tuning Spaces*

Description

A simple mlr3misc::Dictionary storing objects of class TuningSpace. Each tuning space has an associated help page, see mlr_tuning_spaces_[id].

Format

R6::R6Class object inheriting from mlr3misc::Dictionary.

Methods

See mlr3misc::Dictionary.

S3 methods

- `as.data.table(dict,...,objects = FALSE)`
  mlr3misc::Dictionary -> `data.table::data.table()`
  Returns a `data.table::data.table()` with fields "key", "label", "learner", and "n_values" as columns. If `objects` is set to `TRUE`, the constructed objects are returned in the list column named `object`.

Examples

```
as.data.table(mlr_tuning_spaces)
mlr_tuning_spaces$get("classif.ranger.default")
lts("classif.ranger.default")
```

TuningSpace *Tuning Spaces*

Description

This class defines a tuning space for hyperparameter tuning.

For tuning, it is important to create a search space that defines the range over which hyperparameters should be tuned. `TuningSpace` object consists of search spaces from peer-reviewed articles which work well for a wide range of data sets.

The $values field stores a list of paradox::TuneToken which define the search space. These tokens can be assigned to the $values slot of a learner's paradox::ParamSet. When the learner is tuned, the tokens are used to create the search space.

S3 Methods

- `as.data.table.TuningSpace(x)`
  Returns a tabular view of the tuning space.
  TuningSpace -> `data.table::data.table()`

  - `x` (TuningSpace)

Public fields

`id (character(1))`
Identifier of the object.

`values (list())`
List of paradox::TuneToken that describe the tuning space and fixed parameter values.

`tags (character())`
Arbitrary tags to group and filter tuning space e.g. `"classification"` or `"regression"`.

`learner (character(1))`
: mlr3::Learner of the tuning space.

`package (character(1))`
: Packages which provide the Learner, e.g. **mlr3learners** for the learner mlr3learners::LearnerClassifRanger which interfaces the **ranger** package.

`label (character(1))`
: Label for this object. Can be used in tables, plot and text output instead of the ID.

`man (character(1))`
: String in the format [pkg]::[topic] pointing to a manual page for this object. The referenced help package can be opened via method $help().

Methods

Public methods:

- `TuningSpace$new()`
- `TuningSpace$get_learner()`
- `TuningSpace$format()`
- `TuningSpace$help()`
- `TuningSpace$print()`
- `TuningSpace$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
TuningSpace$new(
  id,
  values,
  tags,
  learner,
  package = character(),
  label = NA_character_,
  man = NA_character_
)
```

Arguments:

`id (character(1))`
: Identifier for the new instance.

`values (list())`
: List of paradox::TuneToken that describe the tuning space and fixed parameter values.

`tags (character())`
: Tags to group and filter tuning spaces e.g. `"classification"` or `"regression"`.

`learner (character(1))`
: mlr3::Learner of the tuning space.

`package (character())`
: Packages which provide the Learner, e.g. **mlr3learners** for the learner mlr3learners::LearnerClassifRanger which interfaces the **ranger** package.

`label (character(1))`
: Label for the new instance. Can be used in tables, plot and text output instead of the ID.

`man (character(1))`
String in the format [pkg]::[topic] pointing to a manual page for for the new instance. The referenced help package can be opened via method $help().

Method `get_learner()`**:** Returns a learner with TuneToken set in parameter set.

Usage:

```
TuningSpace$get_learner(...)
```

Arguments:

`...` (named 'list()')
Passed to `mlr3::lrn()`. Named arguments passed to the constructor, to be set as parameters in the paradox::ParamSet, or to be set as public field. See `mlr3misc::dictionary_sugar_get()` for more details.

Returns: mlr3::Learner

Method `format()`**:** Helper for print outputs.

Usage:

```
TuningSpace$format()
```

Method `help()`**:** Opens the corresponding help page referenced by field $man.

Usage:

```
TuningSpace$help()
```

Method `print()`**:** Printer.

Usage:

```
TuningSpace$print(...)
```

Arguments:

`...` (ignored).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TuningSpace$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(mlr3tuning)

# get default tuning space of rpart learner
tuning_space = lts("classif.rpart.default")

# get learner and set tuning space
learner = lrn("classif.rpart")
learner$param_set$values = tuning_space$values

# tune learner
```

```
instance = tune(
 method = "random_search",
 task = tsk("pima"),
 learner = learner,
 resampling = rsmp ("holdout"),
 measure = msr("classif.ce"),
 term_evals = 10)

instance$result
```

`tuning_spaces_default` *Default Tuning Spaces*

Description

Tuning spaces from the Bischl (2021) article.

kknn tuning space

- k $[1, 50]$
- distance $[1, 5]$
- kernel [“rectangular”, “optimal”, “epanechnikov”, “biweight”, “triweight”, “cos”, “inv”, “gaussian”, “rank”]

ranger tuning space

- mtry.ratio $[0, 1]$
- replace [TRUE,FALSE]
- sample.fraction $[0.1, 1]$
- num.trees $[1, 2000]$

rpart tuning space

- minsplit $[2, 128]$
- minbucket $[1, 64]$
- cp $[1e-04, 0.1]$

svm tuning space

- cost $[1e-04, 10000]$
- kernel [“polynomial”, “radial”, “sigmoid”, “linear”]
- degree $[2, 5]$
- gamma $[1e-04, 10000]$

xgboost tuning space

- eta $[1e-04, 1]$
- nrounds $[1, 5000]$
- max_depth $[1, 20]$
- colsample_bytree $[0.1, 1]$
- colsample_bylevel $[0.1, 1]$
- lambda $[0.001, 1000]$
- alpha $[0.001, 1000]$
- subsample $[0.1, 1]$

Source

Bischl B, Binder M, Lang M, Pielok T, Richter J, Coors S, Thomas J, Ullmann T, Becker M, Boulesteix A, Deng D, Lindauer M (2021). "Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges." 2107.05847, https://arxiv.org/abs/2107.05847.

`tuning_spaces_rbv2` *RandomBot Tuning Spaces*

Description

Tuning spaces from the Kuehn (2018) article.

glmnet tuning space

- alpha $[0, 1]$
- s $[1e-04, 1000]$

kknn tuning space

- k $[1, 30]$

ranger tuning space

- num.trees $[1, 2000]$
- replace [TRUE,FALSE]
- sample.fraction $[0.1, 1]$
- mtry.ratio $[0, 1]$
- respect.unordered.factors ["ignore", "order", "partition"]
- min.node.size $[1, 100]$
- splitrule ["gini", "extratrees"]
- num.random.splits $[1, 100]$

`mtry.power` is replaced by `mtry.ratio`.

rpart tuning space

- cp $[1e-04, 1]$
- maxdepth $[1, 30]$
- minbucket $[1, 100]$
- minsplit $[1, 100]$

svm tuning space

- kernel ["linear", "polynomial", "radial"]
- cost $[1e-04, 1000]$
- gamma $[1e-04, 1000]$
- tolerance $[1e-04, 2]$
- degree $[2, 5]$

xgboost tuning space

- booster ["gblinear", "gbtree", "dart"]
- nrounds $[2, 8]$
- eta $[1e-04, 1]$
- gamma $[1e-05, 7]$
- lambda $[1e-04, 1000]$
- alpha $[1e-04, 1000]$
- subsample $[0.1, 1]$
- max_depth $[1, 15]$
- min_child_weight $[1, 100]$
- colsample_bytree $[0.01, 1]$
- colsample_bylevel $[0.01, 1]$
- rate_drop $[0, 1]$
- skip_drop $[0, 1]$

Source

Kuehn D, Probst P, Thomas J, Bischl B (2018). "Automatic Exploration of Machine Learning Experiments on OpenML." 1806.10961, https://arxiv.org/abs/1806.10961.

Index