

Package: mlr3tuningspaces (via r-universe)

January 24, 2025

Title Search Spaces for 'mlr3'

Version 0.5.2

Description Collection of search spaces for hyperparameter optimization in the 'mlr3' ecosystem. It features ready-to-use search spaces for many popular machine learning algorithms. The search spaces are from scientific articles and work for a wide range of data sets.

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 (>= 0.15.0), R (>= 3.1.0)

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

Suggests e1071 (>= 1.7-6), bbotk, glmnet (>= 4.1-2), kkn (>= 1.3.1), mlr3learners (>= 0.4.5), mlr3pipelines, 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 7.3.2

Collate 'mlr_tuning_spaces.R' 'TuningSpace.R' 'bibentries.R' 'sugar.R'
'tuning_spaces_default.R' 'tuning_spaces_rbv1.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 2024-11-22 20:40:02 UTC

Contents

mlr3tuningspaces-package	2
lts	3
mlr_tuning_spaces	4
mlr_tuning_spaces_default	5
mlr_tuning_spaces_rbv1	6
mlr_tuning_spaces_rbv2	7
TuningSpace	9

Index	13
--------------	-----------

mlr3tuningspaces-package

mlr3tuningspaces: Search Spaces for 'mlr3'

Description

Collection of search spaces for hyperparameter optimization in the 'mlr3' ecosystem. It features ready-to-use search spaces for many popular machine learning algorithms. The search spaces are from scientific articles and work for a wide range of data sets.

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**

Function to retrieve [TuningSpace](#) objects from [mlr_tuning_spaces](#) and further, allows a [mlr3::Learner](#) to be directly configured with a search space. This function belongs to [mlr3::mlr_sugar](#) family.

Usage

```
lts(x, ...)

## S3 method for class 'missing'
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.
...	(named list of paradox::TuneToken NULL) Pass paradox::TuneToken to add or overwrite parameters in the tuning space. Use NULL to remove parameters (see examples).

Value

[TuningSpace](#) if x is [character\(\)](#). [mlr3::Learner](#) if x is [mlr3::Learner](#). Or a list of objects for the [ltss\(\)](#) function.

missing, [mlr_tuning_spaces](#) dictionary

a [character](#), [TuningSpace](#)

a [mlr3::Learner](#), [mlr3::Learner](#) with [paradox::TuneToken](#)

a [list\(\)](#), list of [TuningSpace](#) or [mlr3::Learner](#)

Examples

```
# load tuning space
lts("classif.rpart.default")
```

```

# load tuning space and add parameter
lts("classif.rpart.default", maxdepth = to_tune(1, 15))

# load tuning space and remove parameter
lts("classif.rpart.default", minsplit = NULL)

# load tuning space and overwrite parameter
lts("classif.rpart.default", minsplit = to_tune(32, 128))

# load learner and apply tuning space in one go
lts(lrn("classif.rpart"))

# load learner, overwrite parameter and apply tuning space
lts(lrn("classif.rpart"), minsplit = to_tune(32, 128))

# load multiple tuning spaces
ltss(c("classif.rpart.default", "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")

```

`mlr_tuning_spaces_default`*Default Tuning Spaces*

Description

Tuning spaces from the Bischl (2023) article.

glmnet tuning space

- `s` [$1e - 04$, 10000] Logscale
- `alpha` [0, 1]

kknn tuning space

- `k` [1, 50] Logscale
- `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] Logscale
- `minbucket` [1, 64] Logscale
- `cp` [$1e - 04$, 0.1] Logscale

svm tuning space

- `cost` [$1e - 04$, 10000] Logscale
- `kernel` ["polynomial", "radial", "sigmoid", "linear"]
- `degree` [2, 5]
- `gamma` [$1e - 04$, 10000] Logscale

xgboost tuning space

- eta [$1e - 04$, 1] Logscale
- nrounds [1, 5000]
- max_depth [1, 20]
- colsample_bytree [0.1, 1]
- colsample_bylevel [0.1, 1]
- lambda [0.001, 1000] Logscale
- alpha [0.001, 1000] Logscale
- 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 (2023). “Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges.”

mlr_tuning_spaces_rbv1

RandomBot Tuning Spaces

Description

Tuning spaces from the Kuehn (2018) article.

glmnet tuning space

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

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”]
- min.node.size [1, 100]

The tuning space of the ranger learner is slightly different from the original paper. The hyperparameter `mtry.power` is replaced by `mtry.ratio` and `min.node.size` is explored in a range from 1 to 100.

rpart tuning space

- cp [0, 1]
- maxdepth [1, 30]
- minbucket [1, 60]
- minsplit [1, 60]

svm tuning space

- kernel [“linear”, “polynomial”, “radial”]
- cost [$1e - 04$, 1000] Logscale
- gamma [$1e - 04$, 1000] Logscale
- degree [2, 5]

xgboost tuning space

- nrounds [1, 5000]
- eta [$1e - 04$, 1] Logscale
- subsample [0, 1]
- booster [“gblinear”, “gbtree”, “dart”]
- max_depth [1, 15]
- min_child_weight [1, 100] Logscale
- colsample_bytree [0, 1]
- colsample_bylevel [0, 1]
- lambda [$1e - 04$, 1000] Logscale
- alpha [$1e - 04$, 1000] Logscale

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>.

mlr_tuning_spaces_rbv2

RandomBot V2 Tuning Spaces

Description

Tuning spaces from the Binder (2020) article.

glmnet tuning space

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

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] Logscale
- maxdepth [1, 30]
- minbucket [1, 100]
- minsplit [1, 100]

svm tuning space

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

xgboost tuning space

- booster ["gblinear", "gbtree", "dart"]
- nrounds [7, 2981] Logscale
- eta [$1e - 04$, 1] Logscale
- gamma [$1e - 05$, 7] Logscale
- lambda [$1e - 04$, 1000] Logscale
- alpha [$1e - 04$, 1000] Logscale
- subsample [0.1, 1]
- max_depth [1, 15]
- min_child_weight [1, 100] Logscale

- `colsample_bytree` [0.01, 1]
- `colsample_bylevel` [0.01, 1]
- `rate_drop` [0, 1]
- `skip_drop` [0, 1]

Source

Binder M, Pfisterer F, Bischl B (2020). "Collecting Empirical Data About Hyperparameters for Data Driven AutoML." https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_63.pdf.

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 `mlr3::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 [mlr3::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 [paradox::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(...)
```

Arguments:

```
... (ignored).
```

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")

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

# Tune learner
instance = tune(
  trn("random_search"),
  task = tsk("pima"),
```

```
learner = learner,  
resampling = rsmp ("holdout"),  
measure = msr("classif.ce"),  
term_evals = 10)  
  
instance$result  
  
library(mlr3pipelines)  
  
# Set tuning space in a pipeline  
graph_learner = as_learner(po("subsample") %>%  
  lts(lrn("classif.rpart")))
```

Index

- * **Dictionary**
 - mlr_tuning_spaces, 4
- * **TuningSpace**
 - mlr_tuning_spaces, 4
- * **datasets**
 - mlr_tuning_spaces, 4

data.table::data.table(), 4, 9

lts, 3

ltss (lts), 3

mlr3::Learner, 3, 9–11

mlr3::mlr_sugar, 3

mlr3learners::LearnerClassifRanger, 9, 10

mlr3misc::Dictionary, 4

mlr3tuningspaces

- (mlr3tuningspaces-package), 2

mlr3tuningspaces-package, 2

mlr_tuning_spaces, 3, 4

mlr_tuning_spaces_classif.glmnet.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_classif.glmnet.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_classif.glmnet.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_classif.kknn.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_classif.kknn.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_classif.kknn.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_classif.ranger.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_classif.ranger.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_classif.ranger.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_classif.rpart.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_classif.rpart.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_classif.rpart.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_classif.svm.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_classif.svm.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_classif.svm.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_classif.xgboost.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_classif.xgboost.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_classif.xgboost.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_default, 5

mlr_tuning_spaces_rbv1, 6

mlr_tuning_spaces_rbv2, 7

mlr_tuning_spaces_regr.glmnet.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_regr.glmnet.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_regr.glmnet.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_regr.kknn.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_regr.kknn.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_regr.kknn.rbv2

- (mlr_tuning_spaces_rbv2), 7

mlr_tuning_spaces_regr.ranger.default

- (mlr_tuning_spaces_default), 5

mlr_tuning_spaces_regr.ranger.rbv1

- (mlr_tuning_spaces_rbv1), 6

mlr_tuning_spaces_regr.ranger.rbv2

- (mlr_tuning_spaces_rbv2), 7

`mlr_tuning_spaces_regr.rpart.default`
 (`mlr_tuning_spaces_default`), [5](#)
`mlr_tuning_spaces_regr.rpart.rbv1`
 (`mlr_tuning_spaces_rbv1`), [6](#)
`mlr_tuning_spaces_regr.rpart.rbv2`
 (`mlr_tuning_spaces_rbv2`), [7](#)
`mlr_tuning_spaces_regr.svm.default`
 (`mlr_tuning_spaces_default`), [5](#)
`mlr_tuning_spaces_regr.svm.rbv1`
 (`mlr_tuning_spaces_rbv1`), [6](#)
`mlr_tuning_spaces_regr.svm.rbv2`
 (`mlr_tuning_spaces_rbv2`), [7](#)
`mlr_tuning_spaces_regr.xgboost.default`
 (`mlr_tuning_spaces_default`), [5](#)
`mlr_tuning_spaces_regr.xgboost.rbv1`
 (`mlr_tuning_spaces_rbv1`), [6](#)
`mlr_tuning_spaces_regr.xgboost.rbv2`
 (`mlr_tuning_spaces_rbv2`), [7](#)

`paradox::ParamSet`, [9](#), [11](#)
`paradox::TuneToken`, [3](#), [9](#), [10](#)

`R6`, [10](#)
`R6::R6Class`, [4](#)

`TuningSpace`, [3](#), [4](#), [9](#), [9](#)