Package 'mlrintermbo'

May 27, 2024

Title Model-Based Optimization for 'mlr3' Through 'mlrMBO'

Description
The 'mlrMBO' package can ordinarily not be used for optimization within 'mlr3', because of incompatibilities of their respective class systems. 'mlrintermbo' offers a compatibility interface that provides 'mlrMBO' as an 'mlr3tuning' 'Tuner' object, for tuning of machine learning algorithms within 'mlr3', as well as a 'bbotk' 'Optimizer' object for optimization of general objective functions using the 'bbotk' black box optimization framework. The control parameters of 'mlrMBO' are faithfully reproduced as a 'paradox' 'ParamSet'.

URL `https://github.com/mb706/mlrintermbo`

BugReports `https://github.com/mb706/mlrintermbo/issues`

License LGPL-3

Encoding UTF-8

Imports backports, checkmate, data.table, mlr3misc (>= 0.1.4), paradox, R6, lhs, callr, bbotk, mlr3tuning

Suggests mlr, ParamHelpers, testthat, rgenoud, DiceKriging, emoa, cmaesr, randomForest, smoof, lgr, mlr3, mlr3learners, mlr3pipelines, mlrMBO, ranger, rpart, mco

ByteCompile yes

Version 0.5.1

RoxygenNote 7.3.1

Collate 'utils.R' 'CapsuledMlr3Learner.R' 'ParamHelpersParamSet.R' 'optimize.R' 'paramset.R' 'TunerInterMBO.R' 'surrogates.R' 'zzz.R'

NeedsCompilation no

Author Martin Binder [aut, cre]

Maintainer Martin Binder <`developer.mb706@doublecaret.com`>

Repository CRAN

Date/Publication 2024-05-27 18:40:03 UTC

R topics documented:

`mlrintermbo-package` *mlrintermbo: Model-Based Optimization for 'mlr3' Through 'mlrMBO'*

Description

The 'mlrMBO' package can ordinarily not be used for optimization within 'mlr3', because of incompatibilities of their respective class systems. 'mlrintermbo' offers a compatibility interface that provides 'mlrMBO' as an 'mlr3tuning' 'Tuner' object, for tuning of machine learning algorithms within 'mlr3', as well as a 'bbotk' 'Optimizer' object for optimization of general objective functions using the 'bbotk' black box optimization framework. The control parameters of 'mlrMBO' are faithfully reproduced as a 'paradox' 'ParamSet'.

Author(s)

Maintainer: Martin Binder <`developer.mb706@doublecaret.com`>

See Also

Useful links:

- `https://github.com/mb706/mlrintermbo`
- Report bugs at `https://github.com/mb706/mlrintermbo/issues`

`makeMlr3Surrogate` *Create Surrogate Learner*

Description

Creates the default mlrMBO surrogate learners as an `mlr3::Learner`.

This imitates the behaviour of mlrCPO when no `learner` argument is given to `mbo()` / `initSMBO()`.

Usage

```
makeMlr3Surrogate(
  is.numeric = TRUE,
  is.noisy = TRUE,
  has.dependencies = !is.numeric
)
```

Arguments

`is.numeric`	(`logical(1)`) Whether only numeric parameters are present. If so, a `LearnerRegrKM` (**DiceKriging** package) is constructed. Otherwise a `LearnerRegrRanger` (random forest from the **ranger** package) is constructed. Default is `TRUE`.
`is.noisy`	(`logical(1)`) Whether to use nugget estimation. Only considered when `is.numeric` is `TRUE`. Default is `TRUE`.
`has.dependencies`	(`logical(1)`) Whether to anticipate missing values in the surrogate model design. This adds out-of-range imputation to the model. If more elaborate imputation is desired, it may be desirable to set this to `FALSE` and instead perform custom imputation using **mlr3pipelines**. Default is `!numeric`.

Examples

```
# DiceKriging Learner:
makeMlr3Surrogate()

# mlr3pipelines Graph: imputation %>>% 'ranger' (randomForest):
makeMlr3Surrogate(is.numeric = FALSE)

# just the 'ranger' Learner:
makeMlr3Surrogate(is.numeric = FALSE, has.dependencies = FALSE)
```

`OptimizerInterMBO` *Tuner and Optimizer using mlrMBO*

Description

mlrMBO tuning object.

mlrMBO must not be loaded directly into R when using mlr3, for various reasons. TunerInterMBO and OptimizerInterMBO take care that this does not happen.

Format

R6::R6Class object inheriting from mlr3tuning::Tuner or bbotk::Optimizer.

Construction

To optimize an objective (using the `bbotk` package), use the `OptimizerInterMBO` object, ideally obtained through the `bbotk::opt()` function: `opt("intermbo")`.

To tune a machine learning method represented by a mlr3::Learner object, use the `TunerInterMBO` obtained ideally through `mlr3tuning::tnr()`: `tnr("intermbo")`.

Both have following optional arguments:

- `n.objectives` :: `integer(1)`
  Number of objectives to optimize. Default is 1 for ordinary ("single objective") optimization, but can be breater than 1 for multi-objective optimization. See `mlrMBO::setMBOControlMultiObj()` for details on multi-objective optimization in `mlrMBO`.
- `on.surrogate.error` :: `character(1)`
  What to do when fitting or predicting the surrogate model fails. One of `"stop"` (throw error), `"warn"`, and `"quiet"`(ignore and propose a random point).
  The surrogate model may fail sometimes, for example when the size of the initial design is too small or when the objective function returns constant values. In practice this is usually safe to ignore for single iterations (i.e. `"warn"` or `"quiet"`), but be aware that MBO effectively degrades to random search when the surrogate model fails for all iterations.

Configuration Parameters

The `ParamSet` of the optimizer / tuner reflects the possible configuration options of mlrMBO. The control parameters map directly to the arguments of `mlrMBO::makeMBOControl()`, `mlrMBO::setMBOControlInfill()`, `mlrMBO::setMBOControlMultiObj()`, `mlrMBO::setMBOControlMultiPoint()`, and `mlrMBO::setMBOControlTerminati`

Examples

```
library("paradox")
library("bbotk")

# silly example function: minimize x^2 for -1 < x < 1
domain <- ps(x = p_dbl(lower = -1, upper = 1))
codomain <- ps(y = p_dbl(tags = "minimize"))
objective <- ObjectiveRFun$new(function(xs) list(y = xs$x^2), domain, codomain)

# initialize instance
instance <- OptimInstanceSingleCrit$new(objective, domain, trm("evals", n_evals = 6))

# use intermbo optimizer
#
# Also warn on surrogate model errors
# (this is the default and can be omitted)
optser <- opt("intermbo", on.surrogate.error = "warn")

# optimizer has hyperparameters from mlrMBO
optser$param_set$values$final.method <- "best.predicted"

# optimization happens here.
optser$optimize(instance)

instance$result
```

Index