Package ‘RLoptimal’

December 21, 2024

Type Package

Title Optimal Adaptive Allocation Using Deep Reinforcement Learning

Version 1.2.0

Description An implementation to compute an optimal adaptive allocation rule using deep reinforcement learning in a dose-response study (Matsuura et al. (2022) <doi:10.1002/sim.9247>).
The adaptive allocation rule can directly optimize a performance metric, such as power, accuracy of the estimated target dose, or mean absolute error over the estimated dose-response curve.

URL `https://github.com/MatsuuraKentaro/RLoptimal`

BugReports `https://github.com/MatsuuraKentaro/RLoptimal/issues`

VignetteBuilder knitr

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Imports DoseFinding, glue, R6, reticulate, stats, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Collate 'timer.R' 'train_algo.R' 'utils.R' 'allocation_rule.R' 'generate_setup_code.R' 'rl_dnn_config.R' 'rl_config_set.R' 'learn_allocation_rule.R' 'setup_python.R' 'zzz.R' 'simulate_one_trial.R' 'adjust_significance_level.R'

Config/testthat/edition 3

NeedsCompilation no

Author Kentaro Matsuura [aut, cre, cph] (<`https://orcid.org/0000-0001-5262-055X`>), Koji Makiyama [aut, ctb]

Maintainer Kentaro Matsuura <`matsuurakentaro55@gmail.com`>

Repository CRAN

Date/Publication 2024-12-21 03:10:02 UTC

Contents

`adjust_significance_level` *Adjust Significance Level on a Simulation Basis*

Description

Adjust Significance Level on a Simulation Basis

Usage

```
adjust_significance_level(
  allocation_rule,
  models,
  N_total,
  N_ini,
  N_block,
  outcome_type = c("continuous", "binary"),
  sd_normal = NULL,
  alpha = 0.025,
  n_sim = 10000L,
  seed = NULL
)
```

Arguments

`allocation_rule`	An object of class AllocationRule specifying an obtained optimal adaptive allocation rule.
`models`	An object of class Mods specifying assumed dose-response models. This is used in the MCPMod method at the end of this study.
`N_total`	A positive integer value. The total number of subjects.
`N_ini`	A positive integer vector in which each element is greater than or equal to 2. The number of subjects initially assigned to each dose.

`N_block`	A positive integer value. The number of subjects allocated adaptively in each round.
`outcome_type`	A character value specifying the outcome type. Possible values are "continuous" (default), and "binary".
`sd_normal`	A positive numeric value. The standard deviation of the observation noise. When `outcome_type` is "continuous", `sd_normal` must be specified.
`alpha`	A positive numeric value. The original significance level. Default is 0.025.
`n_sim`	A positive integer value. The number of simulation studies to calculate the adjusted significance level. Default is 10000.
`seed`	An integer value. Random seed for data generation in the simulation studies.

Value

A positive numeric value specifying adjusted significance level.

Examples

```
library(RLoptimal)

doses <- c(0, 2, 4, 6, 8)

models <- DoseFinding::Mods(
  doses = doses, maxEff = 1.65,
  linear = NULL, emax = 0.79, sigEmax = c(4, 5)
)

## Not run:
allocation_rule <- learn_allocation_rule(
  models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10, Delta = 1.3,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  seed = 123, rl_config = rl_config_set(iter = 1000),
  alpha = 0.025
)

# Simulation-based adjustment of the significance level using `allocation_rule`
adjusted_alpha <- adjust_significance_level(
  allocation_rule, models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  alpha = 0.025, n_sim = 10000, seed = 123
)
## End(Not run)
```

`AllocationRule` *Allocation Rule Class*

Description

This class represents an allocation rule that generates a next allocation.

Public fields

`policy` The RLlib policy that is a Python object.

`dir` Directory path of the allocation rule (policy).

`dirpath` Full path to the directory of the allocation rule.

`created_at` Created time of this object.

`info` Information when learning the allocation rule.

`input` Inputs for learning the allocation rule.

`log` The log of scores during the learning of the allocation rule.

`checkpoints` The integer vector of iteration counts for checkpoints.

`checkpoints_paths` The paths to the directories where each checkpoint is stored.

Methods

Public methods:

- `AllocationRule$new()`
- `AllocationRule$opt_allocation_probs()`
- `AllocationRule$resume_learning()`
- `AllocationRule$set_info()`
- `AllocationRule$print()`
- `AllocationRule$clone()`

Method `new()`**:** Create a new AllocationRule object.

Usage:

```
AllocationRule$new(dir = "latest", base_dir = "allocation_rules")
```

Arguments:

`dir` A character value. A directory name or path where an allocation rule is outputted. By default, the latest allocation rule is searched in 'base_dir'.

`base_dir` A character value. A directory path that is used as the parent directory if the 'dir' argument is a directory name and is not used otherwise.

Method `opt_allocation_probs()`**:** Compute optimal allocation probabilities using the obtained allocation rule for dose and response data.

Usage:

```
AllocationRule$opt_allocation_probs(data_doses, data_resps)
```

Arguments:

`data_doses` A numeric vector. The doses actually administered to each subject in your clinical trial. It must include all previous doses.

`data_resps` A numeric vector. The values of responses corresponding to each subject for the 'data_doses' argument.

Returns: A vector of the probabilities of the doses.

Method `resume_learning():` Resume learning the allocation rule. This function updates the original AllocationRule object.

Usage:

```
AllocationRule$resume_learning(iter)
```

Arguments:

`iter` A number of additional iterations.

Returns: An updated AllocationRule object.

Method `set_info():` Set information when learning the allocation rule.

Usage:

```
AllocationRule$set_info(info, input, log, checkpoints)
```

Arguments:

`info` Information when learning the allocation rule.

`input` Inputs for learning the allocation rule.

`log` The log of scores during the learning of the allocation rule.

`checkpoints` The paths to the directories where each checkpoint is stored.

Method `print():` Print function for AllocationRule object

Usage:

```
AllocationRule$print()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

clean_python_settings *Clean the Python Virtual Environment*

Description

Clean the Python Virtual Environment

Usage

```
clean_python_settings(envname = "r-RLoptimal")
```

Arguments

envname	Python virtual environment name.

learn_allocation_rule *Build an Optimal Adaptive Allocation Rule using Reinforcement Learning*

Description

Build an Optimal Adaptive Allocation Rule using Reinforcement Learning

Usage

```
learn_allocation_rule(
  models,
  N_total,
  N_ini,
  N_block,
  Delta,
  outcome_type = c("continuous", "binary"),
  sd_normal = NULL,
  optimization_metric = c("MAE", "power", "TD", "power and MAE"),
  rl_models = models,
  rl_models_prior = NULL,
  seed = NULL,
  rl_config = rl_config_set(),
  alpha = 0.025,
  selModel = c("AIC", "maxT", "aveAIC"),
  Delta_range = c(0.9, 1.1) * Delta,
  output_dir = format(Sys.time(), "%Y%m%d_%H%M%S"),
  output_base_dir = "allocation_rules",
  checkpoint_dir = "checkpoints"
)
```

Arguments

models	An object of class Mods specifying assumed dose-response models. When outcome_type is "binary", models should be specified on the logit scale.
N_total	A positive integer value. The total number of subjects.
N_ini	A positive integer vector in which each element is greater than or equal to 2. The number of subjects initially assigned to each dose.
N_block	A positive integer value. The number of subjects allocated adaptively in each round.
Delta	A positive numeric value. The clinically relevant target effect. When outcome_type is "binary", Delta should be specified on the logit scale. See TD for details.
outcome_type	A character value specifying the outcome type. Possible values are "continuous" (default), and "binary".
sd_normal	A positive numeric value. The standard deviation of the observation noise. When outcome_type is "continuous", sd_normal must be specified.
optimization_metric	A character value specifying the metric to optimize. Possible values are "MAE" (default), "power", "TD", or "power and MAE". See Section 2.2 of the original paper for details. "power and MAE" shows performance between "power" and "MAE" by setting the reward based on MAE to 0 when not significant.
rl_models	An object of class Mods. True dose-response models in simulations for reinforcement learning. The default is the same as the 'models' argument. Empirically, the inclusion of a wide variety of models tends to stabilize performance (See RL-MAE incl. exp in the supporting information of the original paper).
rl_models_prior	A positive numeric vector. The probability or weight with which each model in rl_models is selected as the true model in the simulation. The default is NULL, which specifies equal probability for each model.
seed	An integer value. Random seed for reinforcement learning.
rl_config	A list. Other settings for reinforcement learning. See rl_config_set for details.
alpha	A positive numeric value. The significance level. Default is 0.025.
selModel	A character value specifying the model selection criterion for dose estimation. Possible values are "AIC" (default), "maxT", or "aveAIC". See MCPMod for details.
Delta_range	A numeric vector of length 2. The lower and upper bounds of Delta where the estimated target dose is correct. Default is c(0.9, 1.1) * Delta.
output_dir	A character value. Directory name or path to store the built allocation rule. Default is the current datetime.
output_base_dir	A character value. Parent directory path where the built allocation rule will be stored. Valid only if 'output_dir' does not contain '/'. Default is "allocation_rules".
checkpoint_dir	A character value. Parent directory path to save checkpoints. It enables you to resume learning from that point onwards. Default is "checkpoints".

Value

An AllocationRule object.

Examples

```
library(RLoptimal)

doses <- c(0, 2, 4, 6, 8)

# We build the dose-response models to be used in the MCPMod method,
# which we plan to execute at the end of the clinical trial.
models <- DoseFinding::Mods(
  doses = doses, maxEff = 1.65,
  linear = NULL, emax = 0.79, sigEmax = c(4, 5)
)

# We obtain an optimal adaptive allocation rule by executing
# `learn_allocation_rule()` with the `models`.
## Not run:
allocation_rule <- learn_allocation_rule(
  models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10, Delta = 1.3,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  seed = 123, rl_config = rl_config_set(iter = 1000),
  alpha = 0.025
)
## End(Not run)

# It is recommended that the models used in reinforcement learning include
# possible models in addition to the models used in the MCPMod method.
# Here, we add the exponential model according to the supporting information
# in the original paper.
rl_models <- DoseFinding::Mods(
  doses = doses, maxEff = 1.65,
  linear = NULL, emax = 0.79, sigEmax = c(4, 5), exponential = 1
)

# Then, we specify the argument `rl_models` in `learn_allocation_rule` function.
## Not run:
allocation_rule <- learn_allocation_rule(
  models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10, Delta = 1.3,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  seed = 123, rl_models = rl_models, rl_config = rl_config_set(iter = 1000),
  alpha = 0.025
)
## End(Not run)
```

`rl_config_set` *Configuration of Reinforcement Learning*

Description

Mainly settings for the arguments of the training() function. Not compatible with the new API stack introduced in Ray 2.10.0.

Usage

```
rl_config_set(
  iter = 1000L,
  save_start_iter = NULL,
  save_every_iter = NULL,
  cores = 4L,
  gamma = 1,
  lr = 5e-05,
  train_batch_size = 10000L,
  model = rl_dnn_config(),
  sgd_minibatch_size = 200L,
  num_sgd_iter = 20L,
  ...
)
```

Arguments

`iter`	A positive integer value. Number of iterations.
`save_start_iter, save_every_iter`	An integer value. Save checkpoints every 'save_every_iter' iterations starting from 'save_start_iter' or later.
`cores`	A positive integer value. Number of CPU cores used for learning.
`gamma`	A positive numeric value. Discount factor of the Markov decision process. Default is 1.0 (not discount).
`lr`	A positive numeric value. Learning rate (default 5e-5). You can set a learning schedule instead of a learning rate.
`train_batch_size`	A positive integer value. Training batch size. Deprecated on the new API stack.
`model`	A list. Arguments passed into the policy model. See rl_dnn_config for details.
`sgd_minibatch_size`	A positive integer value. Total SGD batch size across all devices for SGD. Deprecated on the new API stack.
`num_sgd_iter`	A positive integer value. Number of SGD iterations in each outer loop.
`...`	Other settings for training(). See the arguments of the training() function in the source code of RLlib. https://github.com/ray-project/ray/blob/master/rllib/algorithms/algorithm_config.py https://github.com/ray-project/ray/blob/master/rllib/algorithms/ppo/ppo.py

Value

A list of reinforcement learning configuration parameters

Examples

```
## Not run:
allocation_rule <- learn_allocation_rule(
  models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10, Delta = 1.3,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  seed = 123,
  # We change `iter` to 200 and `cores` for reinforcement learning to 2
  rl_config = rl_config_set(iter = 200, cores = 2),
  alpha = 0.025
)
## End(Not run)
```

`rl_dnn_config` *DNN Configuration for Reinforcement Learning*

Description

DNN (deep neural network) configuration for reinforcement learning. For detail, see Section 3.2.6 of the original paper.

Usage

```
rl_dnn_config(
  fcnet_hiddens = c(256L, 256L),
  fcnet_activation = c("relu", "tanh", "swish", "silu", "linear"),
  ...
)
```

Arguments

`fcnet_hiddens`	A positive integer vector. Numbers of units of the intermediate layers.
`fcnet_activation`	A character value specifying the activation function. Possible values are "ReLU" (default), "tanh", "Swish" (or "SiLU"), or "linear".
`...`	Other configurations. See source code of RLlib. https://github.com/ray-project/ray/blob/master/rllib/mode

Value

A list of DNN configuration parameters

Examples

```
## Not run:
allocation_rule <- learn_allocation_rule(
  models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10, Delta = 1.3,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  seed = 123,
  # We change iter to 200 and cores to 8
  rl_config = rl_config_set(
    iter = 1000,
    # We change the DNN model
    model = rl_dnn_config(fcnet_hiddens = c(512L, 512L), fcnet_activation = "tanh")
  ),
  alpha = 0.025
)
## End(Not run)
```

`setup_python` *Setting up a Python Virtual Environment*

Description

Setting up a Python virtual environment for the Ray package, which includes the RLlib library for reinforcement learning.

Usage

```
setup_python(envname = "r-RLoptimal")
```

Arguments

`envname`	Python virtual environment name.

`simulate_one_trial` *Simulate One Trial Using an Obtained Optimal Adaptive Allocation Rule*

Description

Simulate One Trial Using an Obtained Optimal Adaptive Allocation Rule

Usage

```
simulate_one_trial(
  allocation_rule,
  models,
  true_response,
  N_total,
  N_ini,
  N_block,
  Delta,
  outcome_type = c("continuous", "binary"),
  sd_normal = NULL,
  alpha = 0.025,
  selModel = c("AIC", "maxT", "aveAIC"),
  seed = NULL,
  eval_type = c("all", "pVal")
)
```

Arguments

`allocation_rule`	An object of class AllocationRule specifying an obtained optimal adaptive allocation rule.
`models`	An object of class Mods specifying assumed dose-response models. When `outcome_type` is "binary", `models` should be specified on the logit scale. This is used in the MCPMod method at the end of this trial.
`true_response`	A numeric vector specifying the true response values of the true model. When `outcome_type` is "binary", `true_response` should be specified on the logit scale.
`N_total`	A positive integer value. The total number of subjects.
`N_ini`	A positive integer vector in which each element is greater than or equal to 2. The number of subjects initially assigned to each dose.
`N_block`	A positive integer value. The number of subjects allocated adaptively in each round.
`Delta`	A positive numeric value. The clinically relevant target effect. When `outcome_type` is "binary", `Delta` should be specified on the logit scale. See TD for details.
`outcome_type`	A character value specifying the outcome type. Possible values are "continuous" (default), and "binary".
`sd_normal`	A positive numeric value. The standard deviation of the observation noise. When `outcome_type` is "continuous", `sd_normal` must be specified.
`alpha`	A positive numeric value. The significance level. Default is 0.025.
`selModel`	A character value specifying the model selection criterion for dose estimation. Possible values are "AIC" (default), "maxT", or "aveAIC". See MCPMod for details.
`seed`	An integer value. Random seed for data generation in this trial.

`eval_type`	A character value specifying the evaluation type. Possible values are "all" (default) and "pVal". "all" returns all metrics, which contain the minimum p value, the selected model name, the estimated target dose, and the MAE. "pVal" returns only the minimum p value without fitting models.

Value

A list which contains the minimum p value, the selected model name, the estimated target dose, the MAE, and the proportions of subjects allocated to each dose.

Examples

```
library(RLoptimal)

doses <- c(0, 2, 4, 6, 8)

models <- DoseFinding::Mods(
  doses = doses, maxEff = 1.65,
  linear = NULL, emax = 0.79, sigEmax = c(4, 5)
)

## Not run:
allocation_rule <- learn_allocation_rule(
  models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10, Delta = 1.3,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  seed = 123, rl_config = rl_config_set(iter = 1000),
  alpha = 0.025
)

# Simulation-based adjustment of the significance level using `allocation_rule`
adjusted_alpha <- adjust_significance_level(
  allocation_rule, models,
  N_total = 150, N_ini = rep(10, 5), N_block = 10,
  outcome_type = "continuous", sd_normal = sqrt(4.5),
  alpha = 0.025, n_sim = 10000, seed = 123
)
## End(Not run)

eval_models <- DoseFinding::Mods(
  doses = doses, maxEff = 1.65,
  linear = NULL, emax = 0.79, sigEmax = c(4, 5), exponential = 1, quadratic = - 1/12
)
true_response_matrix <- DoseFinding::getResp(eval_models, doses = doses)
true_response_list <- as.list(data.frame(true_response_matrix, check.names = FALSE))

true_model_name <- "emax"

# Simulate one trial using the obtained `allocation_rule` When the true model is "emax"
## Not run:
res_one <- simulate_one_trial(
  allocation_rule, models,
  true_response = true_response_list[[true_model_name]],
  N_total = 150, N_ini = rep(10, 5), N_block = 10,
  Delta = 1.3, outcome_type = "continuous", sd_normal = sqrt(4.5),
  alpha = adjusted_alpha, seed = simID, eval_type = "all"
)
## End(Not run)
```

Index