Package ‘gym’

October 25, 2016

Version 0.1.0

Title Provides Access to the OpenAI Gym API

Description OpenAI Gym is a open-source Python toolkit for developing and comparing reinforcement learning algorithms. This is a wrapper for the OpenAI Gym API, and enables access to an ever-growing variety of environments.

For more details on OpenAI Gym, please see here: <https://github.com/openai/gym>. For more details on the OpenAI Gym API specification, please see here: <https://github.com/openai/gym-http-api>.

License MIT + file LICENSE

LazyData true

Depends R (>= 3.3.1)

Imports httr, jsonlite

URL https://github.com/paulhendricks/gym-R

BugReports https://github.com/paulhendricks/gym-R/issues

RoxygenNote 5.0.1

Suggests testthat

NeedsCompilation no

Author Paul Hendricks [aut, cre]

Maintainer Paul Hendricks <paul.hendricks.2013@owu.edu>

Repository CRAN

Date/Publication 2016-10-25 00:57:54

R topics documented:

create_GymClient .......................................................... 2
env_action_space_contains ........................................ 3
env_action_space_info ................................................ 3
env_action_space_sample .......................................... 4
env_close ................................................................. 5
env_create .............................................................. 6
create_GymClient

Create a GymClient instance.

Description

This function instantiates a GymClient instance to integrate with an OpenAI Gym server.

Usage

create_GymClient(remote_base)

Arguments

remote_base The URL of the OpenAI gym server. This value is usually "http://127.0.0.1:5000".

Value

An instance of class "GymClient"; this object has "remote_base" as an attribute.

Examples

```r
# Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)

# End(Not run)
```
env_action_space_contains

Evaluate whether an action is a member of an environments's action space.

Description

Evaluate whether an action is a member of an environments's action space.

Usage

env_action_space_contains(x, instance_id, action)

Arguments

x
An instance of class "GymClient"; this object has "remote_base" as an attribute.

instance_id
A short identifier (such as "3c657dbe") for the environment instance.

action
An action to take in the environment.

Value

A boolean atomic vector of length one indicating if the action is a member of an environments’s action space.

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
action <- env_action_space_sample(client, instance_id)
env_action_space_contains(client, instance_id, action)
## End(Not run)
```

env_action_space_info

Get information (name and dimensions/bounds) of the environments’s action space.

Description

Get information (name and dimensions/bounds) of the environments’s action space.
Usage

env_action_space_info(x, instance_id)

Arguments

x An instance of class "GymClient"; this object has "remote_base" as an attribute.

instance_id A short identifier (such as "3c657dbc") for the environment instance.

Value

A list containing "name" (such as "Discrete"), and additional dimensional info (such as "n") which varies from space to space.

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
env_action_space_info(client, instance_id)
## End(Not run)
```

---

env_action_space_sample

Sample an action from the environments's action space.

Description

Sample an action from the environments's action space.

Usage

env_action_space_sample(x, instance_id)

Arguments

x An instance of class "GymClient"; this object has "remote_base" as an attribute.

instance_id A short identifier (such as "3c657dbc") for the environment instance.

Value

An action sampled from a space (such as "Discrete"), which varies from space to space.
env_close

Examples

## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
env_action_space_sample(client, instance_id)

## End(Not run)

---

env_close

Flush all monitor data to disk.

Description

Flush all monitor data to disk.

Usage

env_close(x, instance_id)

Arguments

- x: An instance of class "GymClient"; this object has "remote_base" as an attribute.
- instance_id: A short identifier (such as "3c657dbc") for the environment instance.

Value

NULL.

Examples

## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
env_close(client, instance_id)

## End(Not run)
env_create

Create an instance of the specified environment.

Description

Create an instance of the specified environment.

Usage

env_create(x, env_id)

Arguments

- x: An instance of class "GymClient"; this object has "remote_base" as an attribute.
- env_id: A short identifier (such as "3c657dbc") for the created environment instance. The instance_id is used in future API calls to identify the environment to be manipulated.

Value

A short identifier (such as "3c657dbc") for the created environment instance. The instance_id is used in future API calls to identify the environment to be manipulated.

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
env_create(client, env_id)
## End(Not run)
```

env_list_all

List all environments running on the server.

Description

List all environments running on the server.

Usage

env_list_all(x)

Arguments

- x: An instance of class "GymClient"; this object has "remote_base" as an attribute.
**Value**

A list mapping instance_id to env_id e.g. `list("3c657dbc" = "CartPole-v0")` for every env on the server.

**Examples**

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_list_all(client)

## End(Not run)
```

---

**Description**

Flush all monitor data to disk.

**Usage**

```r
env_monitor_close(x, instance_id)
```

**Arguments**

- `x`: An instance of class "GymClient"; this object has "remote_base" as an attribute.
- `instance_id`: A short identifier (such as "3c657dbc") for the environment instance.

**Value**

NULL.

**Examples**

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
env_monitor_close(client, instance_id)

## End(Not run)
```
env_monitor_start  
Start monitoring.

Description
Start monitoring.

Usage
env_monitor_start(x, instance_id, directory, force = FALSE, resume = FALSE)

Arguments
x  
An instance of class "GymClient"; this object has "remote_base" as an attribute.

instance_id  
A short identifier (such as "3c657dbc") for the environment instance.

directory  
The directory to write the training data to. Defaults to FALSE.

force  
Clear out existing training data from this directory (by deleting every file pre-fixed with "openaigym"). Defaults to NULL.

resume  
Retain the training data already in this directory, which will be merged with our new data. Defaults to FALSE.

Value
NULL.

Examples
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
outdir <- "/tmp/random-agent-results"
env_monitor_start(client, instance_id, outdir, force = TRUE, resume = FALSE)

## End(Not run)
env_observation_space_info

Get information (name and dimensions/bounds) of the environment’s observation space.

Description
Get information (name and dimensions/bounds) of the environment’s observation space.

Usage

env_observation_space_info(x, instance_id)

Arguments

x
An instance of class "GymClient"; this object has "remote_base" as an attribute.

instance_id
A short identifier (such as "3c657dbc") for the environment instance.

Value
A list containing "name" (such as "Discrete"), and additional dimensional info (such as "n") which varies from space to space.

Examples

## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
environ_observation_space_info(client, instance_id)

## End(Not run)

env_reset

Reset the state of the environment and return an initial observation.

Description
Reset the state of the environment and return an initial observation.

Usage

env_reset(x, instance_id)
env_step

Arguments

x An instance of class "GymClient"; this object has "remote_base" as an attribute.
instance_id A short identifier (such as "3c657dbc") for the environment instance.

Value

The initial observation of the space.

Examples

## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
env_reset(client, instance_id)

## End(Not run)

---

env_step Step though an environment using an action.

Description

Step though an environment using an action.

Usage

env_step(x, instance_id, action, render = FALSE)

Arguments

x An instance of class "GymClient"; this object has "remote_base" as an attribute.
instance_id A short identifier (such as "3c657dbc") for the environment instance.
action An action to take in the environment.
render Whether to render the environment. Defaults to FALSE.

Value

A list consisting of the following: action; an action to take in the environment, observation; an agent's observation of the current environment, reward; the amount of reward returned after previous action, done; whether the episode has ended, and info; a list containing auxiliary diagnostic information.
get_request

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
env_id <- "CartPole-v0"
instance_id <- env_create(client, env_id)
action <- env_action_space_sample(client, instance_id)
env_step(client, instance_id, action)

## End(Not run)
```

get_request Submit a GET request to an OpenAI Gym server.

Description

Submit a GET request to an OpenAI Gym server.

Usage

```r
get_request(x, route, data = NULL)
```

Arguments

- `x`: An instance of class "GymClient"; this object has "remote_base" as an attribute.
- `route`: The URL path or endpoint.
- `data`: URL query arguments. Default value is NULL.

Value

If the response code is 200 or 204, a parsed response. Else, a server error or raised exception.

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
route <- "/v1/envs/"
get_request(client, route)

## End(Not run)
```

gym gym: Provides Access to the OpenAI Gym API

Description

gym: Provides Access to the OpenAI Gym API
parse_server_error_or_raise_for_status

Parse the server error or raise for status.

Description
Parse the server error or raise for status.

Usage
parse_server_error_or_raise_for_status(response)

Arguments
response A response object from \texttt{httr::POST} or \texttt{httr::GET}.

Value
If the response code is 200 or 204, a parsed response. Else, a server error or raised exception.

Examples
\begin{verbatim}
## Not run:
b2 <- "http://httpbin.org/post"
response <- \texttt{httr::POST(b2, body = \texttt{"A simple text string"})}
parse_server_error_or_raise_for_status(response)
## End(Not run)
\end{verbatim}

post_request
Submit a POST request to an OpenAI Gym server.

Description
Submit a POST request to an OpenAI Gym server.

Usage
post_request(x, route, data = \texttt{NULL})

Arguments

\begin{itemize}
\item \texttt{x} An instance of class "GymClient"; this object has "remote_base" as an attribute.
\item route The URL path or endpoint.
\item data URL query arguments. Default value is \texttt{NULL}.
\end{itemize}
Value

If the response code is 200 or 204, a parsed response. Else, a server error or raised exception.

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
route <- "/v1/envs/"
env_id <- "CartPole-v0"
data <- list(env_id = env_id)
post_request(client, route, data)

## End(Not run)
```

print.GymClient

Represent a GymClient instance on the command line.

Description

Represent a GymClient instance on the command line.

Usage

```r
## S3 method for class 'GymClient'
print(x, ...)
```

Arguments

- `x` An instance of class "GymClient"; this object has "remote_base" as an attribute.
- `...` Further arguments passed to or from other methods.

Value

`x` A GymClient instance.

Examples

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
print(client)

## End(Not run)
```
random_discrete_agent  A sample random discrete agent.

Description
A sample random discrete agent.

Usage
random_discrete_agent(n)

Arguments
n The number of discrete action spaces available.

Value
NULL.

Examples
agent <- random_discrete_agent(10)

shutdown_server  Request a server shutdown.

Description
Request a server shutdown.

Usage
shutdown_server(x)

Arguments
x An instance of class "GymClient"; this object has "remote_base" as an attribute.

Value
NULL Currently used by the integration tests to repeatedly create and destroy fresh copies of the server running in a separate thread.
**Examples**

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
shutdown_server(client)

## End(Not run)
```

---

**Description**

Flush all monitor data to disk.

**Usage**

```r
upload(x, training_dir, api_key = NULL, algorithm_id = NULL)
```

**Arguments**

- `x`: An instance of class "GymClient"; this object has "remote_base" as an attribute.
- `training_dir`: A directory containing the results of a training run.
- `api_key`: Your OpenAI API key.
- `algorithm_id`: An arbitrary string indicating the particular version of the algorithm (including choices of parameters) you are running.

**Value**

`NULL`.

**Examples**

```r
## Not run:
remote_base <- "http://127.0.0.1:5000"
client <- create_GymClient(remote_base)
outdir <- "/tmp/random-agent-results"
upload(client, outdir)

## End(Not run)
```
Index

create_gymclient, 2

env_action_space_contains, 3
env_action_space_info, 3
env_action_space_sample, 4
env_close, 5
env_create, 6
env_list_all, 6
env_monitor_close, 7
env_monitor_start, 8
env_observation_space_info, 9
env_reset, 9
env_step, 10

get_request, 11
gym, 11
gym-package (gym), 11

parse_server_error_or_raise_for_status, 12
post_request, 12
print.GymClient, 13

random_discrete_agent, 14
shutdow_server, 14

upload, 15