Package 'mafR'

July 22, 2025

Type Package

Title Interface for Masked Autoregressive Flows

Description Interfaces the Python library 'zuko' implementing Masked Autoregressive Flows. See Rozet, Divo and Schnake (2023) <doi:10.5281/zenodo.7625672> and Papamakarios, Pavlakou and Murray (2017) <doi:10.48550/arXiv.1705.07057>.

Encoding UTF-8

Version 1.1.6

Date 2024-09-25

Imports reticulate

Depends R (>= 3.6.0)

License GPL (>= 2)

ByteCompile true

URL https://github.com/f-rousset/mafR

NeedsCompilation no

Author Jean-Michel Marin [aut, cph],
François Rousset [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-4670-0371>)

Maintainer François Rousset <francois.rousset@umontpellier.fr>

Repository CRAN

Date/Publication 2024-09-26 10:50:03 UTC

Contents

.r_to_torch 2
control_py_env 2
get_py_MAF_handle 3
mafR 4

Index **6**

.r_to_torch *Utility to manage torch tensors*

Description

(Currently not used nor exported) utility converting an R object to a torch tensor.

Usage

```
.r_to_torch(x, py_handle, device)
```

Arguments

x	An R object suitable for use in `reticulate::r_to_py(x)` (this being as indefinite as the r_to_py documentation in this respect.)
py_handle	The return value of get_py_MAF_handle, or possibly more generally an environment with (at least) elements `torch` and `device` defined as in such a return value.
device	Character: `"cpu"`; or a GPU backend, either `"cuda"` (or `"cuda:0"`, etc.) or `"mps"` depending on system capabilities.

Value

`r_to_torch` returns a 32-bit floating-point **torch** tensor allocated on the given `device`.

Examples

```
my_env <- list2env(list(is_set=FALSE),parent = emptyenv())
my_env <- get_py_MAF_handle(my_env, reset=FALSE, torch_device="cpu")
```

control_py_env *Python controls*

Description

Interface to control variables in a Python environment possibly used by Infusion. Currently the only implemented control is that of the **torch** random seed.

Usage

```
control_py_env(py_handle, seed = NULL)
```

Arguments

py_handle	An R environment that provides access to a Python evaluation environment, as produced by get_py_MAF_handle
seed	Numeric: passed (as integer value) to `torch.random.manual_seed`.

Value

Returns NULL invisibly.

Examples

```
## Initialization of Python session:
my_env <- list2env(list(is_set=FALSE),parent = emptyenv())
py_handle <- get_py_MAF_handle(my_env, reset=FALSE, torch_device="cpu")

if (inherits(py_handle,"environment"))  control_py_env(py_handle, seed=0L)
```

---

get_py_MAF_handle *Utilities to manage Python environment and torch tensors*

Description

Utility initializing a Python environment for running `zuko.flows.MAF` and retrieving it.

Usage

```
get_py_MAF_handle(envir, reset=FALSE, torch_device="cpu", GPU_mem=NULL,
                  verbose = TRUE)
```

Arguments

`envir`	An environment (in the R sense) initialized as shown in the Examples.
`reset`	Boolean: Whether to reinitialize the Python session or not.
`torch_device`	Character: `"cpu"`; or a GPU backend, either `"cuda"` (or `"cuda:0"`, etc.) or `"mps"` depending on system capabilities.
`GPU_mem`	For development purposes (effect is complicated). An amount of (dedicated) GPU memory, in bytes.
`verbose`	Boolean. Whether to print some messages or not.

Value

If successful, `get_py_MAF_handle` returns the modified input environment. If sourcing the Python code provided by **mafR** failed (presumably from trying to use an improperly set-up Python environment), the error condition message is returned.

Examples

```
# Initialization of Python session:
my_env <- list2env(list(is_set=FALSE),parent = emptyenv())
my_env <- get_py_MAF_handle(my_env, reset=FALSE, torch_device="cpu")

if (inherits(my_env,"environment"))  {
    # => provides access to:
```

```
    my_env$torch # Imported Python package (result of reticulate::import("torch"))
    my_env$device # the torch_device
    # and to internal definitions for MAF training
}
```

---

mafR *Interface for masked autoregressive flows*

Description

This wraps Python procedures to train Masked Autoregressive Flows (MAFs, Paramakarios et al. 2017) using the Python package zuko. It has been tested with version 1.1.0 and 1.2.0 of that package. Note that objects created by its version 1.2.0 cannot be read with its version 1.1.0 (i.e., when saved in and read from pickle files).

The simplest portable way to get **mafR** working may be to install it in a conda environment. Below is a complete installation recipe. More information about alternative installation procedure may be found on the Git repository for **mafR**, https://github.com/f-rousset/mafR.

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm ~/miniconda3/miniconda.sh

~/miniconda3/bin/conda init bash
conda create --name maf-conda python==3.10
conda activate maf-conda

pip install zuko

conda install R
conda install conda-forge::r-gmp
conda install conda-forge::gsl
```

and, in an R session within the maf-conda environment:

```
install.packages("reticulate")
library(reticulate)
use_condaenv(condaenv="maf-conda", conda="~/miniconda3/bin/conda")
install.packages("mafR")

# 'mafR' was first designed for use with 'Infusion':
install.packages("Infusion")
install.packages("Rmixmod") # only a Suggested dependency of Infusion, but needed.
```

References

Papamakarios, G., D. Sterratt, and I. Murray. 2019. Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows. Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, PMLR 89:837-848, 2019. https://doi.org/10.48550/arXiv.1705.07057 ; https://proceedings.mlr.press/v89/papamakarios19a.html

Rozet, F., Divo, F., Schnake, S (2023) Zuko: Normalizing flows in PyTorch. https://doi.org/10.5281/zenodo.7625672

Index

.r_to_torch, 2

control_py_env, 2

get_py_MAF_handle, *2*, 3

mafR, 4
mafR-package *(*mafR*)*, 4

r_to_py, *2*