Package 'reservoirnet'

April 4, 2023

Type Package

Title Reservoir Computing and Echo State Networks

Version 0.2.0

Date 2023-03-13

SystemRequirements Python (>= 3.7)

Description A simple user-friendly library based on the 'python' module 'reservoirpy'. It provides a flexible interface to implement efficient Reservoir Computing (RC) architectures with a particular focus on Echo State Networks (ESN). Some of its features are: offline and online training, parallel implementation, sparse matrix computation, fast spectral initialization, advanced learning rules (e.g. Intrinsic Plasticity) etc. It also makes possible to easily create complex architectures with multiple reservoirs (e.g. deep reservoirs), readouts, and complex feedback loops. Moreover, graphical tools are included to easily explore hyperparameters. Finally, it includes several tutorials exploring time series forecasting, classification and hyperparameter tuning. For more information about 'reservoirpy', please see Trouvain et al. (2020) <doi:10.1007/978-3-030-61616-8_40>. This package was developed in the framework of the University of Bordeaux's IdEx ``Investments for the Future" program / RRI PHDS.

Config/reticulate list(packages = list(list(package = ``reservoirpy", pip=TRUE)))

License GPL (>= 3)

Repository CRAN

URL https://github.com/reservoirpy

Depends R (>= 3.6)

RoxygenNote 7.2.3

Encoding UTF-8

Imports reticulate, testthat (>= 3.0.0), rlang, ggplot2, ggpubr, janitor, dplyr, magrittr, methods

Suggests rmarkdown, knitr, covr, kableExtra, slider, tibble, tidyr

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Thomas Ferte [aut, cre, trl],
Kalidou Ba [aut, trl],
Nathan Trouvain [aut],
Rodolphe Thiebaut [aut],
Xavier Hinaut [aut],
Boris Hejblum [aut, trl]

Maintainer Thomas Ferte <thomas.ferte@u-bordeaux.fr>

Date/Publication 2023-04-04 11:40:02 UTC

R topics documented:

createNode 2
dfCovid 4
generate_data 5
install_reservoirpy 6
link 7
plot.reservoir_predict_seq 8
plot_2x2_perf 9
plot_marginal_perf 10
plot_perf_22 10
predict_seq 11
print.summary.reservoirR_fit 12
random_search_hyperparam 13
reservoirR_fit 14
rloguniform 15
summary.reservoirR_fit 15
summary.reservoir_predict_seq 16
%>>% 17

Index **18**

`createNode` *Function to create some node*

Description

Function to create some node

Usage

```
createNode(
  nodeType = c("Ridge"),
  units = NULL,
  lr = 1,
  sr = NULL,
  otputDim = NULL,
  inputDim = NULL,
  name = NULL,
  ridge = 0,
  inputBias = TRUE,
  input_scaling = TRUE,
  input_connectivity = 0.1,
  rc_connectivity = 0.1,
  activation = "tanh",
  dtype = "float64",
  seed = NULL,
  ...
)
```

Arguments

nodeType	Type of node. Default is "Ridge".
units	(int) optional Number of reservoir units. If None, the number of units will be infered from the W matrix shape.
lr	(float) default to 1.0 Neurons leak rate. Must be in :math:[0, 1].
sr	(float) optional Spectral radius of recurrent weight matrix.
otputDim	Output dimension of the Node. Dimension of its state.
inputDim	Input dimension of the Node.
name	Name of the Node. It must be a unique identifier.
ridge	float, default to 0.0. L2 regularization parameter.
inputBias	bool, default to TRUE. If TRUE, then a bias parameter will be learned along with output weights.
input_scaling	float or array-like of shapes (features), default to 1.0. Input gain. An array of the same dimension as the inputs can be used to set up different input scaling for each feature.
input_connectivity	float, default to 0.1. Connectivity of input neurons, i.e. ratio of input neurons connected to reservoir neurons. Must be between 0 and 1.
rc_connectivity	float, default to 0.1. Connectivity of recurrent weight matrix, i.e. ratio of reservoir neurons connected to other reservoir neurons, including themselves. Must be between 0 and 1.
activation	str 'tanh'. Reservoir units activation function. Should be a activationsfunc function name ('tanh', 'identity', 'sigmoid', 'relu', 'softmax', 'softplus').

dtype	Numerical type for node parameters
seed	set random seed
...	Others params

Value

A node generated by reservoirpy python module.

Examples

```
if(interactive()){
readout <- reservoirnet::createNode("Ridge")
}
```

`dfCovid` *Datagouv covid-19 dataset*

Description

A dataset containing the data from datagouv.fr concerning covid-19 infections in Aquitaine. Data related to hospitalizations can be found at Santé publique France - Data downloaded at https://www.data.gouv.fr/fr/datasets/r/0 6780-452d-9b8c-ae244ad529b3, update from 26/01/2023. Data related to RT-PCR can be found at Santé publique France - Data downloaded at https://www.data.gouv.fr/fr/datasets/r/10639654-3864-48ac-b024-d772c218c4c1, update from 26/01/2023.

Usage

```
data(dfCovid)
```

Format

A data frame with 962 rows and 4 variables

Details

- date. The date
- hosp. Number of person hospitalized with SARS-CoV-2 in Aquitaine.
- Positive. Number of person with a positive RT-PCR in Aquitaine.
- Tested. Number of person with a RT-PCR in Aquitaine.

generate_data *Load data from the* Japanese vowels *or the* Mackey-Glass

Description

Mackey-Glass time series [8]_ [9]_, computed from the Mackey-Glass delayed differential equation:

Usage

```
generate_data(
  dataset = c("japanese_vowels", "mackey_glass", "both"),
  one_hot_encode = TRUE,
  repeat_targets = FALSE,
  reload = FALSE,
  n_timesteps,
  tau = 17,
  a = 0.2,
  b = 0.1,
  n = 10,
  x0 = 1.2,
  h = 1
)
```

Arguments

dataset	(String) take value in array [japanese_vowels,mackey_glass]
one_hot_encode	(bool), default to True. If True, returns class label as a one-hot encoded vector.
repeat_targets	(bool), default to False. If True, repeat the target label or vector along the time axis of the corresponding sample.
reload	(bool), default to False If True, re-download data from remote repository. Else, if a cached version of the dataset exists, use the cached dataset.
n_timesteps	(int) Number of time steps to compute.
tau	(int), default to 17 Time delay :math:‘\tau‘ of Mackey-Glass equation. By defaults, equals to 17. Other values can change the choatic behaviour of the timeseries.
a	(float) default to 0.2 :math:‘a‘ parameter of the equation.
b	(float) default to 0.1 :math:‘b‘ parameter of the equation.
n	(int) default to 10 :math:‘n‘ parameter of the equation.
x0	(float), optional, default to 1.2 Initial condition of the timeseries.
h	(float), default to 1.0 Time delta between two discrete timesteps.

Value

array of shape (n_timesteps, 1) Mackey-Glass timeseries.

Examples

```
if(interactive()){
japanese_vowels <- generate_data(dataset="japanese_vowels")
timeSerie <- generate_data(dataset = "mackey_glass",n_timesteps = 2500)
res =generate_data(dataset <- "both",n_timesteps = 2500)
}
```

`install_reservoirpy` *Install reservoirpy*

Description

Install reservoirpy

Usage

```
install_reservoirpy(envname = "r-reticulate", method = "auto")
```

Arguments

`envname`	`str` name of environment. Default is R-reticulate
`method`	`str` type of environment type (`virtualenv`, `conda`). Default is auto (`virtualenv` is not available on `Windows`)

Value

A NULL object after installing reservoirpy python module.

Examples

```
## Not run:
reservoirnet::install_reservoirpy()

## End(Not run)
```

link *Link two :py:class:~.Node instances to form a :py:class:~.Model instance.* node1 *output will be used as input for* node2 *in the created model. This is similar to a function composition operation:*

Description

Link two :py:class:~.Node instances to form a :py:class:~.Model instance. node1 output will be used as input for node2 in the created model. This is similar to a function composition operation:

Usage

```
link(node1, node2, name = NULL)
```

Arguments

node1	(Node) or (list_of_Node) Nodes or lists of nodes to link.
node2	(Node) or (list_of_Node) Nodes or lists of nodes to link.
name	(str) optional Name for the chaining Model.

Details

Can update the state of the node several times

Value

A reservoir model linking node1 and node2.

Examples

```
if(reticulate::py_module_available("reservoirpy")){
reservoir <- reservoirnet::createNode(nodeType = "Reservoir",
                                      seed = 1,
                                      units = 100,
                                      lr = 0.7,
                                      sr = 1,
                                      input_scaling = 1)
readout <- reservoirnet::createNode(nodeType = "Ridge", ridge = 0.1)
model <- reservoirnet::link(reservoir, readout)
}
```

plot.reservoir_predict_seq *plot.reservoir_predict_seq*

Description

plot.reservoir_predict_seq

Usage

```
## S3 method for class 'reservoir_predict_seq'
plot(x, ..., vec_nodes = c(1:20), vec_time = NULL)
```

Arguments

x	A reservoir_predict_seq object
...	deprecated
vec_nodes	Number of nodes to plot
vec_time	Time to plot

Value

A ggplot

Examples

```
if(reticulate::py_module_available("reservoirpy")){
reservoir <- reservoirnet::createNode(nodeType = "Reservoir",
                                      seed = 1,
                                      units = 100,
                                      lr = 0.7,
                                      sr = 1,
                                      input_scaling = 1)
X <- matrix(data = rnorm(100), ncol = 4)
reservoir_state_stand <- reservoirnet::predict_seq(node = reservoir, X = X)
plot(reservoir_state_stand)
summary(reservoir_state_stand)
}
```

plot_2x2_perf *plot_2x2_perf*

Description

Plot 2x2 combinations of the hyperparameters.

Usage

```
plot_2x2_perf(
  dfPerf,
  perf_lab = "Median relative error",
  legend_position = "bottom",
  trans = "log10"
)
```

Arguments

dfPerf	The performance dataframe which should have the columns : perf, ridge, input_scaling, leaking_rate, spectral_radius. Where perf is the performance metric
perf_lab	The label of the performance metric.
legend_position	Position of legend passed to ggarrange
trans	The transformation (default is "log10")

Value

A mutliple 2x2 plots.

Examples

```
dfPerf <-
data.frame(
  perf = runif(n = 10),
  ridge = runif(n = 10),
  input_scaling = runif(n = 10),
  leaking_rate = runif(n = 10)
)
reservoirnet::plot_2x2_perf(dfPerf = dfPerf)
```

plot_marginal_perf *plot_marginal_perf*

Description

get marginal performance from dfPerf

Usage

```
plot_marginal_perf(dfPerf, color_cut = 10, perf_lab = "Median relative error")
```

Arguments

dfPerf	The performance dataframe which should have the columns : perf, ridge, input_scaling, leaking_rate, spectral_radius. Where perf is the performance metric
color_cut	The cutting point to highlight best values (default = 10)
perf_lab	The label of the performance metric.

Value

A plot with 4 facets

Examples

```
dfPerf <-
data.frame(
  perf = runif(n = 10),
  ridge = runif(n = 10),
  input_scaling = runif(n = 10),
  leaking_rate = runif(n = 10)
)
reservoirnet::plot_marginal_perf(dfPerf = dfPerf, color_cut = 2)
```

plot_perf_22 *plot_perf_22*

Description

Unit plot for 2x2 function

Usage

```
plot_perf_22(x, y, dfPerf, perf_lab, trans = "log10")
```

Arguments

x	The x feature
y	The y feature
dfPerf	The performance dataframe which should have the columns : perf, ridge, input_scaling, leaking_rate, spectral_radius. Where perf is the performance metric
perf_lab	The label of the performance metric.
trans	The transformation (default is "log10")

Value

A 2x2 plot

Examples

```
dfPerf <-
data.frame(
  perf = runif(n = 10),
  ridge = runif(n = 10),
  input_scaling = runif(n = 10),
  leaking_rate = runif(n = 10)
)
reservoirnet::plot_perf_22(
  dfPerf = dfPerf,
  x = "ridge",
  y = "input_scaling",
  perf_lab = "MSE"
)
```

---

predict_seq *Run the node-forward function on a sequence of data*

Description

Run the node-forward function on a sequence of data

Usage

```
predict_seq(node, X, formState = NULL, stateful = TRUE, reset = FALSE)
```

Arguments

node	node
X	array-like of shape (`[n_inputs]`, `timesteps`, `input_dim`) A sequence of data of shape (timesteps, features).

formState	array of shape (1, output_dim), optional Node state value to use at begining of computation.
stateful	bool, default to TRUE If True, Node state will be updated by this operation.
reset	bool, default to FALSE If True, Node state will be reset to zero before this operation.

Details

Can update the state of the node several times

Value

An object of class reservoir_predict_seq. This object is a numeric vector containing the matrix of the prediction of the reservoir. It is either the forecast of the ridge layer or the node state of the reservoir if no ridge layer is given.

Examples

```
if(reticulate::py_module_available("reservoirpy")){
reservoir <- reservoirnet::createNode(nodeType = "Reservoir",
                                      seed = 1,
                                      units = 100,
                                      lr = 0.7,
                                      sr = 1,
                                      input_scaling = 1)
X <- matrix(data = rnorm(100), ncol = 4)
reservoir_state_stand <- reservoirnet::predict_seq(node = reservoir, X = X)
plot(reservoir_state_stand)
summary(reservoir_state_stand)
}
```

---

print.summary.reservoirR_fit

reservoirR_fit print summary

Description

print S3 method for summary.reservoirR_fit object

Usage

```
## S3 method for class 'summary.reservoirR_fit'
print(x, ...)
```

Arguments

x	an object of class summary.reservoirR_fit to print.
...	further arguments.

Value

A NULL object which shows the model setting to perform the reservoir fit.

Examples

```
if(reticulate::py_module_available("reservoirpy")){
}
```

---

random_search_hyperparam *random_search_hyperparam*

Description

Generate a hyperparameter simulation table using functions as input.

Usage

```
random_search_hyperparam(
  n = 100,
 ls_fct = list(ridge = function(n) 1e-05, input_scaling = function(n) 1, spectral_radius
  = function(n) rloguniform(n = n, min = 0.01, max = 10), leaking_rate = function(n)
    rloguniform(n = n, min = 0.001, max = 1))
)
```

Arguments

n	Number of search
ls_fct	A list of functions

Value

A dataframe of size n x 4. Each row is a different set of hyperparameters.

Examples

```
random_search_hyperparam(
  n = 100,
  ls_fct = list(
    ridge = function(n)
      1e-5,
    input_scaling = function(n)
      1,
    spectral_radius = function(n)
      rloguniform(n = n, min = 1e-2, max = 10),
    leaking_rate = function(n)
      rloguniform(n = n, min = 1e-3, max = 1)
  )
)
```

---

reservoirR_fit *Offline fitting method of a Node*

Description

Offline fitting method of a Node

Usage

```
reservoirR_fit(node, X, Y, warmup = 0, stateful = FALSE, reset = FALSE)
```

Arguments

node	node
X	array-like of shape [n_inputs], [series], timesteps, input_dim), optional Input sequences dataset. If None, the method will try to fit the parameters of the Node using the precomputed values returned by previous call of :py:meth:partial_fit.
Y	array-like of shape ([series], timesteps, output_dim), optional Teacher signals dataset. If None, the method will try to fit the parameters of the Node using the precomputed values returned by previous call of :py:meth: partial_fit, or to fit the Node in an unsupervised way, if possible.
warmup	: int, default to 0 Number of timesteps to consider as warmup and discard at the begining of each timeseries before training.
stateful	is boolen
reset	is boolean. Should the node status be reset before fitting.

Value

A fitted reservoir of class reservoiR_fit containing the fitted model.

Examples

```
if(reticulate::py_module_available("reservoirpy")){
}
```

`rloguniform` *rloguniform*

Description

Simulate a log-uniform distribution

Usage

```
rloguniform(n, min = 10^-1, max = 10^2)
```

Arguments

`n`	number of sample
`min`	minimum of the distribution
`max`	maximum of the distribution

Value

A vector of simulated values

Examples

```
rloguniform(n = 1)
```

`summary.reservoirR_fit` *reservoirR_fit summary*

Description

summary S3 method for reservoirR_fit object

Usage

```
## S3 method for class 'reservoirR_fit'
summary(object, ...)
```

Arguments

object	an object of class reservoirR_fit to summarized.
...	further arguments.

Value

a list object

Examples

```
if(reticulate::py_module_available("reservoirpy")){
}
```

---

summary.reservoir_predict_seq

summary.reservoir_predict_seq

Description

summary.reservoir_predict_seq

Usage

```
## S3 method for class 'reservoir_predict_seq'
summary(object, ...)
```

Arguments

object	A reservoir_predict_seq object
...	Additional argument (unused)

Value

A dataframe with node activation

Examples

```
if(reticulate::py_module_available("reservoirpy")){
reservoir <- reservoirnet::createNode(nodeType = "Reservoir",
                                      seed = 1,
                                      units = 100,
                                      lr = 0.7,
                                      sr = 1,
                                      input_scaling = 1)
X <- matrix(data = rnorm(100), ncol = 4)
reservoir_state_stand <- reservoirnet::predict_seq(node = reservoir, X = X)
plot(reservoir_state_stand)
```

```
summary(reservoir_state_stand)
}
```

`%>>%` *Takes two nodes and applies python operator >>*

Description

A port of the >> "chevron" operator from reservoirpy.

Usage

```
node1 %>>% node2
```

Arguments

node1	a Node or a list of Nodes
node2	a Node or a list of Nodes

Value

A node or a list of nodes.

Examples

```
if(interactive()){
  source <- reservoirnet::createNode("Input")
  reservoir <- reservoirnet::createNode("Reservoir", units = 100, lr=0.1, sr=0.9)
  source %>>% reservoir

  readout <- reservoirnet::createNode("Ridge")
  list(source %>>% reservoir, source) %>>% readout
}
```

Index

∗ **datasets**
dfCovid, 4
%>>%, 17

chevron *(%>>%)*, 17
createNode, 2

dfCovid, 4

generate_data, 5

install_reservoirpy, 6

link, 7

plot.reservoir_predict_seq, 8
plot_2x2_perf, 9
plot_marginal_perf, 10
plot_perf_22, 10
predict_seq, 11
print.summary.reservoirR_fit, 12

random_search_hyperparam, 13
reservoirR_fit, 14
rloguniform, 15

summary.reservoir_predict_seq, 16
summary.reservoirR_fit, 15