Time Series Generator

Release 0.2.8

Daniel Kaminski de Souza

Oct 02, 2021

CONTENTS

1 Description **3**

2 Installation **5**

3 Usage **7**
- 3.1 Main Functionality . . . 7
- 3.2 Candidate Improvement . . . 7

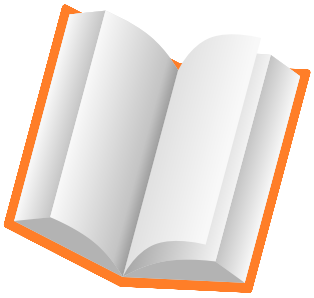
4 Documentation for the Code **9**
- 4.1 Modules . . . 9
- 4.2 Indices and tables . . . 10

Python Module Index **11**

Index **13**

Welcome to Time Series Generator's documentation!

This documents the python package sourced from the following repository.



Read Online

Download PDF

Download EPUB

CHAPTER

ONE

DESCRIPTION

Emulates Teras Tensorflow TimeSeriesGenerator functionality presenting a candidate solution for the direct multi-step outputs limitation in Keras version.

CHAPTER

TWO

INSTALLATION

```
pip install time-series-generator
```

CHAPTER

THREE

USAGE

3.1 Main Functionality

3.2 Candidate Improvement

Addition of the keyworded argument `length_output`.

```
# define dataset
series = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
target = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
# define generator
n_input = 2
n_output = 2
generator = TimeseriesGenerator(series, target, length=n_input, length_output=n_output,␣
↪batch_size=1)
# print each sample
for i in range(len(generator)):
   x, y = generator[i]
   print('%s => %s' % (x, y))
```

3.2.1 Output

```
[[1 2]] => [[3 4]]
[[2 3]] => [[4 5]]
[[3 4]] => [[5 6]]
[[4 5]] => [[6 7]]
[[5 6]] => [[7 8]]
[[6 7]] => [[8 9]]
[[7 8]] => [[9 10]]
```

CHAPTER

FOUR

DOCUMENTATION FOR THE CODE

4.1 Modules

4.1.1 Time Series Generator module

class `time_series_generator.time_series_generator.`**`TimeseriesGenerator`**(*data*, *targets*, *length*, *sampling_rate=1*, *length_output=1*, *sampling_rate_output=1*, *stride=1*, *start_index=0*, *end_index=None*, *shuffle=False*, *reverse=False*, *batch_size=9223372036854775807*, *augmentation=0*, *overlap=0*)

Bases: `object`

Utility class for generating batches of temporal data.

This class takes in a sequence of data-points gathered at equal intervals, along with time series parameters such as stride, length of history, etc., to produce batches for training/validation.

 # Arguments

data: Indexable generator (such as list or Numpy array) containing consecutive data points (timesteps). The data should be at 2D, and axis 0 is expected to be the time dimension.

targets: Targets corresponding to timesteps in *data*. It should have same length as *data*.

length: Length of the output sequences (in number of timesteps). sampling_rate: Period between successive individual timesteps

within sequences. For rate *r*, timesteps *data[i]*, *data[i-r]*, … *data[i - length]* are used for create a sample sequence.

stride: Period between successive output sequences. For stride *s*, consecutive output samples would be centered around *data[i]*, *data[i+s]*, *data[i+2*s]*, etc.

start_index: Data points earlier than *start_index* will not be used in the output sequences. This is useful to reserve part of the data for test or validation.

end_index: Data points later than *end_index* will not be used in the output sequences. This is useful to reserve part of the data for test or validation.

shuffle: Whether to shuffle output samples, or instead draw them in chronological order.

reverse: Boolean: if *true*, timesteps in each output sample will be in reverse chronological order.

batch_size: Number of timeseries samples in each batch (except maybe the last one).

 # Returns A [Sequence](/utils/#sequence) instance.

Examples

```python from keras.preprocessing.sequence import TimeseriesGenerator import numpy as np

data = np.array([[i] for i in range(50)]) targets = np.array([[i] for i in range(50)])

**data_gen = TimeseriesGenerator(data, targets,** length=10, sampling_rate=2, batch_size=2)

assert len(data_gen) == 20

batch_0 = data_gen[0] x, y = batch_0 assert np.array_equal(x,

**np.array([[[0], [2], [4], [6], [8]],** [[1], [3], [5], [7], [9]]]))

**assert np.array_equal(y,** np.array([[10], [11]]))

```

get_config()

Returns the TimeseriesGenerator configuration as Python dictionary.

 # Returns A Python dictionary with the TimeseriesGenerator configuration.

to_json(***kwargs*)

Returns a JSON string containing the timeseries generator configuration. To load a generator from a JSON string, use *keras.preprocessing.sequence.timeseries_generator_from_json(json_string)*.

 # Arguments

****kwargs: Additional keyword arguments** to be passed to *json.dumps()*.

 # Returns A JSON string containing the tokenizer configuration.

time_series_generator.time_series_generator.**timeseries_generator_from_json**(*json_string*)

Parses a JSON timeseries generator configuration file and returns a timeseries generator instance.

 # Arguments

json_string: JSON string encoding a timeseries generator configuration.

 # Returns A Keras TimeseriesGenerator instance

4.2 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

time_series_generator.time_series_generator, 9

INDEX

G

`get_config()` (*time_series_generator.time_series_generator.TimeseriesGenerator method*), 10

M

`module`
`time_series_generator.time_series_generator`, 9

T

`time_series_generator.time_series_generator`
`module`, 9
`timeseries_generator_from_json()` (*in module time_series_generator.time_series_generator*), 10
`TimeseriesGenerator` (*class in time_series_generator.time_series_generator*), 9
`to_json()` (*time_series_generator.time_series_generator.TimeseriesGenerator method*), 10