rhos

Release 0.1+11.gdfb6690

Claudio Satriano

Apr 04, 2023

CONTENTS:

rhos (recursive high order statistics) is a python module to compute recursive mean, variance and high-order statistics on 1D signals.

Each function is implemented in pure python (functions ending in `_py`, useful for algorithm reference), as well as using a faster approach based on `scipy.signal.lfilter()`.

CHAPTER

ONE

FUNCTIONS

Recursive high-order statistics for Python.

copyright
2022 Claudio Satriano <satriano@ipgp.fr>

license
GNU Lesser General Public License, Version 3 (https://www.gnu.org/copyleft/lesser.html)

`rhos.rec_mean_py`(*signal*, *C*)

Recursive mean of a signal, pure Python implementation.

μ[i] = C·signal[i] + (1-C)·μ[i-1]

Parameters

- **signal** (`ArrayLike`) – signal to compute recursive mean for
- **C** (`float`) – decay constant, in the [0, 1] interval

Returns
the recursive mean, with the same length than signal

Return type
`numpy.ndarray`

Raises
ValueError – if C is not in the [0, 1] interval

> **Warning:** This is a pure python reference implementation. Use `rec_mean()` for a faster implementation.

`rhos.rec_mean`(*signal*, *C*)

Recursive mean of a signal.

μ[i] = C·signal[i] + (1-C)·μ[i-1]

Parameters

- **signal** (`ArrayLike`) – signal to compute recursive mean for
- **C** (`float`) – decay constant, in the [0, 1] interval

Returns
the recursive mean, with the same length than signal

Return type
`numpy.ndarray`

Raises
ValueError – if C is not in the [0, 1] interval

Note: Fast implementation, using `scipy.signal.lfilter()`.

`rhos.`**`rec_variance_py`**(*signal*, *C*, *definition=0*)

Recursive variance of a signal, pure Python implementation.

Defined as in Poiata *et al.* [2016] (definition 0):

$\sigma^2[i] = C\cdot(signal[i]-\mu[i-1])^2 + (1-C)\cdot\sigma^2[i-1]$

Or, defined as in Langet *et al.* [2014] (definition 1):

$\sigma^2[i] = C\cdot(signal[i]-\mu[i])^2 + (1-C)\cdot\sigma^2[i-1]$

For both definitions:

$\mu[i] = C\cdot signal[i] + (1-C)\cdot\mu[i-1]$

Parameters

- **signal** (`ArrayLike`) – signal to compute recursive variance for
- **C** (`float`) – decay constant, in the [0, 1] interval
- **definition** (`int`) – which formula to use

Returns
the recursive variance, with the same length than signal

Return type
`numpy.ndarray`

Raises

- **ValueError** – if C is not in the [0, 1] interval
- **ValueError** – if definition is not 0 or 1

Warning: This is a pure python reference implementation. Use *`rec_variance()`* for a faster implementation.

`rhos.`**`rec_variance`**(*signal*, *C*, *definition=0*)

Recursive variance of a signal.

Defined as in Poiata *et al.* [2016] (definition 0):

$\sigma^2[i] = C\cdot(signal[i]-\mu[i-1])^2 + (1-C)\cdot\sigma^2[i-1]$

Or, defined as in Langet *et al.* [2014] (definition 1):

$\sigma^2[i] = C\cdot(signal[i]-\mu[i])^2 + (1-C)\cdot\sigma^2[i-1]$

For both definitions:

$\mu[i] = C\cdot signal[i] + (1-C)\cdot\mu[i-1]$

Parameters

- **signal** (`ArrayLike`) – signal to compute recursive variance for

- **C** (`float`) – decay constant, in the [0, 1] interval
- **definition** (`int`) – which formula to use

Returns
the recursive variance, with the same length than signal

Return type
`numpy.ndarray`

Raises

- **ValueError** – if C is not in the [0, 1] interval
- **ValueError** – if definition is not 0 or 1

Note: Fast implementation, using `scipy.signal.lfilter()`.

`rhos.`**`rec_hos_py`**(*signal*, *C*, *order=4*, *var_min=-1*, *definition=0*)

Recursive high order statistics (hos) of a signal, pure Python implementation.

Defined as in BackTrackBB (definition 0):

$$hos[i] = C\cdot(signal[i]-\mu[i-1])^n / (\sigma^2[i])^{n/2} + (1-C)\cdot hos[i-1]$$

with

$$\sigma^2[i] = C\cdot(signal[i]-\mu[i-1])^2 + (1-C)\cdot\sigma^2[i-1]$$

Note: This is the actual implementation in the BackTrackBB source code, which does not correspond to equation 7 of Poiata *et al.* [2016] or equation 1 of Poiata *et al.* [2018].

Or, defined as in Langet *et al.* [2014] (definition 1):

$$hos[i] = C\cdot(signal[i]-\mu[i])^n / (\sigma^2[i])^{n/2} + (1-C)\cdot hos[i-1]$$

with

$$\sigma^2[i] = C\cdot(signal[i]-\mu[i])^2 + (1-C)\cdot\sigma^2[i-1]$$

For both definitions:

$$\mu[i] = C\cdot signal[i] + (1-C)\cdot\mu[i-1]$$

Parameters

- **signal** (`ArrayLike`) – signal to compute recursive hos for
- **C** (`float`) – decay constant, in the [0, 1] interval
- **order** (`int`) – hos order
- **var_min** (`float`) – values of variance σ^2 (hos denominator) smaller than *var_min* will be replaced by *var_min*
- **definition** (`int`) – which formula to use

Returns
the recursive hos, with the same length than signal

Return type
`numpy.ndarray`

Raises

- **ValueError** – if C is not in the [0, 1] interval
- **ValueError** – if definition is not 0 or 1

> **Warning:** This is a pure python reference implementation. Use `rec_hos()` for a faster implementation.

`rhos.rec_hos`(*signal*, *C*, *order=4*, *var_min=-1*, *definition=0*)

Recursive high order statistics (hos) of a signal.

Defined as in BackTrackBB (definition 0):

$$hos[i] = C\cdot(signal[i]-\mu[i-1])^n / (\sigma^2[i])^{n/2} + (1-C)\cdot hos[i-1]$$

with

$$\sigma^2[i] = C\cdot(signal[i]-\mu[i-1])^2 + (1-C)\cdot\sigma^2[i-1]$$

Note: This is the actual implementation in the BackTrackBB source code, which does not correspond to equation 7 of Poiata *et al.* [2016] or equation 1 of Poiata *et al.* [2018].

Or, defined as in Langet *et al.* [2014] (definition 1):

$$hos[i] = C\cdot(signal[i]-\mu[i])^n / (\sigma^2[i])^{n/2} + (1-C)\cdot hos[i-1]$$

with

$$\sigma^2[i] = C\cdot(signal[i]-\mu[i])^2 + (1-C)\cdot\sigma^2[i-1]$$

For both definitions:

$$\mu[i] = C\cdot signal[i] + (1-C)\cdot\mu[i-1]$$

Parameters

- **signal** (`ArrayLike`) – signal to compute recursive hos for
- **C** (`float`) – decay constant, in the [0, 1] interval
- **order** (`int`) – hos order
- **var_min** (`float`) – values of variance σ^2 (hos denominator) smaller than *var_min* will be replaced by *var_min*
- **definition** (`int`) – which formula to use

Returns

the recursive hos, with the same length than signal

Return type

`numpy.ndarray`

Raises

- **ValueError** – if C is not in the [0, 1] interval
- **ValueError** – if definition is not 0 or 1

Note: Fast implementation, using `scipy.signal.lfilter()`.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] N. Poiata, C. Satriano, J.-P. Vilotte, P. Bernard, and K. Obara. Multiband array detection and location of seismic sources recorded by dense seismic networks. *Geophysical Journal International*, 205(3):1548–1573, 2016. doi:10.1093/gji/ggw071.

[2] N. Langet, A. Maggi, A. Michelini, and F. Brenguier. Continuous kurtosis-based migration for seismic event detection and location, with application to piton de la fournaise volcano, la reunion. *Bulletin of the Seismological Society of America*, 104(1):229–246, 2014. doi:10.1785/0120130107.

[3] N. Poiata, J.-P. Vilotte, P. Bernard, C. Satriano, and K. Obara. Imaging different components of a tectonic tremor sequence in southwestern japan using an automatic statistical detection and location method. *Geophysical Journal International*, 213(3):2193–2213, 2018. doi:10.1093/gji/ggy070.

PYTHON MODULE INDEX

r

INDEX

M

R