Package ‘VMDecomp’

July 4, 2022

Type Package
Title Variational Mode Decomposition
Version 1.0.1
Date 2022-07-04
Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

BugReports https://github.com/mlampros/VMDecomp/issues
URL https://github.com/mlampros/VMDecomp


License GPL-3
Encoding UTF-8
Copyright inst/COPYRIGHTS

SystemRequirements libarmadillo: apt-get install -y libarmadillo-dev
       (deb), libblas: apt-get install -y libblas-dev (deb),
       liblapack: apt-get install -y liblapack-dev (deb),
       libarpack++2: apt-get install -y libarpack++2-dev (deb),
       gfortran: apt-get install -y gfortran (deb)

Depends R(>= 3.5.0)
Imports Rcpp (>= 1.0.8.3), data.table, glue
LinkingTo Rcpp, RcppArmadillo
Suggests OpenImageR, R.matlab, testthat (>= 3.0.0), rmarkdown, knitr
RoxygenNote 7.1.2
Config/testthat/edition 3
Sample Arrhythmia Data from MIT Boston's Beth Israel Hospital (BIH) Database

Description
Sample arrhythmia data from the MIT-BIH Arrhythmia Database

Usage

data(arrhythmia)

Format
An object of class data.table (inherits from data.frame) with 10000 rows and 2 columns.

Details
The data includes two columns "MLII" and "V1". According to https://www.physionet.org/files/mitdb/1.0.0/mitdbdir/intro.htm, "In most records, the upper signal is a modified limb lead II (MLII), obtained by placing the electrodes on the chest. The lower signal is usually a modified lead V1 (occasionally V2 or V5, and in one instance V4)."

The data was downloaded after installing the "wfdb" Python package. The Python code used to save the sample data is the following:

import wfdb
import pandas as pd
sample_annotat_200 = wfdb.rdrecord('200', sampfrom = 0, sampto = 10000, pn_dir = 'mitdb')
arrhythmia = pd.DataFrame(sample_annotat_200.p_signal, columns = sample_annotat_200.sig_name)

References


https://physionet.org/content/mitdb/1.0.0/
https://www.physionet.org/files/mitdb/1.0.0/mitdbdir/intro.htm
https://github.com/MIT-LCP/wfdb-python

Examples

require(VMDecomp)
data(arrhythmia)

estimate_k_modes signal_1d, cor_thresh, default_vmd_params, min_K = 2, seed = 1, verbose = FALSE

estimate_k_modes Estimation of Intrinsic Mode Function (IMF) Number in Variational Mode Decomposition

Description

Estimation of Intrinsic Mode Function (IMF) Number in Variational Mode Decomposition

Usage

estimate_k_modes( signal_1d, cor_thresh, default_vmd_params, min_K = 2, seed = 1, verbose = FALSE )
**estimate_k_modes**

**Arguments**

- `signal_1d` a numeric vector specifying the 1-dimensional input signal
- `cor_thresh` a numeric value specifying the minimum (positive or negative) correlation coefficient threshold where decomposition will be stopped (a value between 0.0 and 1.0)
- `default_vmd_params` a list of parameters consisting of the (remaining) Variational Mode Decomposition default parameters (except for 'data' and 'K')
- `min_K` a numeric value specifying the minimum value of the K (modes) parameter (from which decomposition starts)
- `seed` a numeric value specifying the seed (for reproducibility purposes)
- `verbose` a boolean. If TRUE then information will be printed in the console

**Details**

Correlation Coefficient Method:

- Correlation coefficient (CC) between the mode components and the original signal will be obtained. Decomposition will be stopped when the minimum correlation coefficient is less than the given threshold, and then the value of K will be determined

**Value**

a numeric value specifying the optimal K parameter

**References**

https://doi.org/10.1155/2020/8304903

**Examples**

```r
## Not run:
require(VMDecomp)
data(arrhythmia)

default_vmd_params = list(alpha = 2000,
                          tau = 0,
                          DC = FALSE,
                          init = 1,
                          tol = 1e-6)

res_k = estimate_k_modes(signal_1d = arrhythmia[['MLII']],
cor_thresh = 0.1,
default_vmd_params = default_vmd_params,
min_K = 2,
seed = 1,
verbose = TRUE)
```
**Description**

Variational Mode Decomposition (1- or 2-dimensional)

**Usage**

```r
vmd(data, alpha, tau, K, DC, init, tol, verbose = FALSE)
```

**Arguments**

- `data`: either a vector or a matrix (of type numeric or integer)
- `alpha`: a numeric value specifying the balancing parameter of the data-fidelity constraint
- `tau`: a numeric value specifying the time-step of the dual ascent (pick 0 for noise-slab)
- `K`: a numeric value specifying the number of modes to be recovered
- `DC`: a boolean. If true the first mode is put and kept at DC (0-freq)
- `init`: a numeric value. This parameter differs depending on the input 'data' parameter (1-dimensional and 2-dimensional). See the details section for more information
- `tol`: a numeric value specifying the tolerance of convergence criterion (typically this parameter is around 1e-6 for the 1-dimensional and 1e-7 for the 2-dimensional data)
- `verbose`: a boolean. If TRUE then information will be printed in the console

**Details**

The 'init' parameter takes the following values for,

- **1-dimensional data**:
  - 0 = all omegas start at 0
  - 1 = all omegas start uniformly distributed
  - 2 = all omegas initialized randomly

- **2-dimensional data**:
  - 0 = all omegas start at 0
  - 1 = all omegas start initialized randomly
Value

a list object of length three which includes the

• 'u' (collection of decomposed modes)
• 'u_hat' (spectra of the modes)
• 'omega' (estimated mode center-frequencies) objects

References

https://math.montana.edu/dzosso/code/

Examples

require(VMDecomp)

#. ............
# 1-dimensional
#. ............

N = 250
set.seed(1)
rand_unif = runif(n = N, min = 0, max = 1.0)

f_sig1 = 6 * rand_unif
f_sig2 = cos(x = 8 * pi * rand_unif)
f_sig3 = 0.5 * cos(x = 40 * pi * rand_unif)

f_sig = f_sig1 + f_sig2 + f_sig3

alpha = 2000
tau = 0
K = 3
DC = FALSE
init = 1
tol = 1e-6

set.seed(2)
res_ld = vmd(data = f_sig,
    alpha = alpha,
tau = tau,
K = K,
DC = DC,
init = init,
tol = tol,
verbose = FALSE)

#. ............
# 2-dimensional
#. ............
rows_cols = 10

set.seed(3)
data = matrix(runif(rows_cols^2), rows_cols, rows_cols)
alpha = 5000
tau = 0.25
K = 2
DC = TRUE
init = 1
tol = 1e-7

set.seed(4)
res_2d = vmd(data = data, alpha = alpha, tau = tau, K = K, DC = DC, init = init, tol = tol, verbose = FALSE)
Index

* datasets
  arrhythmia, 2

arrhythmia, 2

estimate_k_modes, 3

vmd, 5