Package: GSD (via r-universe)

September 9, 2024

Type Package

Title Graph Signal Decomposition

Version 1.0.0

Date 2024-02-04

Author Hyeonglae Cho [aut], Hee-Seok Oh [aut], Donghoh Kim [aut, cre]

Maintainer Donghoh Kim <donghoh.kim@gmail.com>

Description Graph signals residing on the vertices of a graph have recently gained prominence in research in various fields. Many methodologies have been proposed to analyze graph signals by adapting classical signal processing tools. Recently, several notable graph signal decomposition methods have been proposed, which include graph Fourier decomposition based on graph Fourier transform, graph empirical mode decomposition, and statistical graph empirical mode decomposition. This package efficiently implements multiscale analysis applicable to various fields, and offers an effective tool for visualizing and decomposing graph signals. For the detailed methodology, see Ortega et al. (2018) <doi:10.1109/JPROC.2018.2820126>, Shuman et al. (2013) <doi:10.1109/MSP.2012.2235192>, Tremblay et al. (2014) <https://www.eurasip.org/Proceedings/Eusipco/Eusipco2014/HTML/papers/1569922141.pdf>, and Cho et al. (2024) ``Statistical graph empirical mode decomposition by graph denoising and boundary treatment".

Depends R (>= 3.5.0), igraph, Matrix, EbayesThresh, ggplot2

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation no

Repository https://dkimstatlab.r-universe.dev

RemoteUrl https://github.com/dkimstatlab/gsd

RemoteRef HEAD

RemoteSha 71ddd75408a59c577b91db0a0986124b79b3e0af

Contents

`adjmatrix` *Weighted Adjacency Matrix*

Description

This function produces weighted adjacency matrix by Gaussian kernel.

Usage

```
adjmatrix(xy, method = c("dist", "neighbor"), alpha)
```

Arguments

`xy`	matrix or data.frame containing vertex coordinate `x`, `y`.
`method`	When `method="dist"`, edge weights are calculated by Gaussian kernel for connecting vertices within distance `alpha`. When `method="neighbor"`, edge weights are calcaulated by Gaussian kernel for connecting `alpha` neighboring vertices.
`alpha`	specifies distance between vertices when `method="dist"`, and the number of neighboring vertices when `method="neighbor"`.

Details

This function produces a sparse weighted adjacency matrix by Gaussian kernel based on the distance between vertices.

Value

a sparse weighted adjacency matrix

References

Zeng, J., Cheung, G., and Ortega, A. (2017). Bipartite approximation for graph wavelet signal decomposition. *IEEE Transactions on Signal Processing*, **65(20)**, 5466–5480. doi:10.1109/TSP.2017.2733489

See Also

gsignal, gplot.

Examples

```
## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A1 <- adjmatrix(xy, method = "dist", 0.04)

## weighted adjacency matrix by Gaussian kernel
## for connecting seven neighboring vertices
A2 <- adjmatrix(xy, method="neighbor", 7)
```

---

gextrema *Finding Local Extrema of a Graph Signal*

Description

This function finds the local extrema of a graph signal identifying the edge between neighboring vertices.

Usage

```
gextrema(ad_mat, signal)
```

Arguments

ad_mat	an weighted adjacency matrix.
signal	a graph signal.

Details

This function finds the local extrema of a graph signal identifying the edge between neighboring vertices.

Value

maxima_list	vertex index of local maxima of a signal.
minima_list	vertex index of local minima of a signal.
n_extrema	the number of local maxima and local minima of a signal.

References

Tremblay, N., Borgnat, P., and Flandrin, P. (2014). Graph empirical mode decomposition. *22nd European Signal Processing Conference (EUSIPCO)*, 2350–2354.

See Also

ginterpolating, gsmoothing, sgemd.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# local extrema
gextrema(A, signal)

# local extrema using graph object
gextrema(as_adjacency_matrix(gsig, attr="weight"), V(gsig)$z)
```

---

gfdecomp *Graph Fourier Decomposition*

Description

This function performs the graph Fourier decomposition.

Usage

```
gfdecomp(graph, K)
```

Arguments

graph	an **igraph** graph object with vertex attributes of coordinates x, y, a signal z, and edge attribute of weight.
K	specifies the number of frequency components.

Details

This function performs the graph Fourier decomposition.

Value

fc	list of frequency components according to the frequencies with fc[[1]] the lowest-frequency component.
residue	residue signal after extracting frequency components.

References

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., and Vandergheynst, P. (2018). Graph signal processing: overview, challenges, and applications. *Proceedings of the IEEE 106*, 808–828. doi:10.1109/JPROC.2018.2820126

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, **30(3)**, 83–98. doi:10.1109/MSP.2012.2235192

See Also

sgemd.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)
```

```
# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# display of absolute values of the graph Fourier coefficients vs the eigenvalues
gftplot(gsig)
gftplot(gsig, K=5, size=3)
outgft <- gftplot(gsig, K=5, plot=FALSE)
outgft$eigenvalues

# graph Fourier decomposition
out <- gfdecomp(gsig, K=4)
names(out)

# display of a signal, the extracted low- and high-frequency components by GFD
gplot(gsig, size=3)
gplot(gsig, out$fc[[1]]+out$fc[[2]], size=3)
gplot(gsig, out$fc[[3]]+out$fc[[4]], size=3)
```

`gftplot` *Plot of the absolute values of the graph Fourier coefficients vs the eigenvalues*

Description

This function displays the absolute values of the graph Fourier coefficients vs the eigenvalues.

Usage

```
gftplot(graph, signal = NULL, K = NULL, size = 1, plot=TRUE)
```

Arguments

`graph`	an **igraph** graph object with vertex attributes of coordinates `x`, `y`, a signal `z`, and edge attribute of `weight`.
`signal`	specifies a signal for the graph Fourier transform. When `signal=NULL`, a signal `z` of object `graph` is used.
`K`	specifies the number of frequency components.
`size`	specifies point size.
`plot`	specifies whether plot is displayed.

Details

This function displays the absolute values of the graph Fourier coefficients vs the eigenvalues for `signal`. The red color denotes the nonnegative graph Fourier coefficients, and the blue color indicates the negative graph Fourier coefficients.

Value

If `plot=TRUE`, plot of the absolute values of the graph Fourier coefficients vs the eigenvalues for `signal` over a graph using package **ggplot2**. If `plot=FALSE`, a list with components:

`absgFCoeffs`	the absolute values of the graph Fourier coefficients in decreasing order.
`eigenvalues`	the eigenvalues according to the absolute values of the graph Fourier coefficients.

See Also

`gsignal`, `gfdecomp`.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# display a signal over graph
gplot(gsig, size=3)

# display of absolute values of the graph Fourier coefficients vs the eigenvalues
# for signal
```

```
gftplot(gsig)

gftplot(gsig, K=5, size=3)
out <- gftplot(gsig, K=5, plot=FALSE)
names(out)

## signal3
# high-frequency component
signal11 <- c(outer(sin(6*pi*x - 0.5*pi), sin(6*pi*y - 0.5*pi)))

# low-frequency component
signal22 <- c(outer(sin(2*pi*x - 0.5*pi), sin(2*pi*y - 0.5*pi)))

# composite signal
signal00 <- signal11 + signal22

# noisy signal
signal3 <- signal00 + rnorm(900, 0, sqrt(var(signal00) / 5))

# display signal3 over graph
gplot(gsig, signal=signal3, size=3)

# display of absolute values of the graph Fourier coefficients vs the eigenvalues
# for signal3
gftplot(gsig, signal=signal3)
gftplot(gsig, signal=signal3, K=10, size=2)
```

`ginterpolating` *Interpolation of a Graph Signal*

Description

This function interpolates a graph signal utilizing the Laplacian matrix.

Usage

```
ginterpolating(ad_mat, signal, vertices)
```

Arguments

`ad_mat`	an weighted adjacency matrix.
`signal`	a graph signal.
`vertices`	specifies vertices for the observed signal. A signal on `vertices` and Laplacian matrix is used for interpolating a signal outside `vertices`.

Details

This function interpolates a graph signal utilizing the Laplacian matrix.

Value

a signal with interpolated points.

References

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., and Vandergheynst, P. (2018). Graph signal processing: overview, challenges, and applications. *Proceedings of the IEEE 106*, 808–828. doi:10.1109/JPROC.2018.2820126

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, **30(3)**, 83–98. doi:10.1109/MSP.2012.2235192

Tremblay, N., Borgnat, P., and Flandrin, P. (2014). Graph empirical mode decomposition. *22nd European Signal Processing Conference (EUSIPCO)*, 2350–2354.

Zeng, J., Cheung, G., and Ortega, A. (2017). Bipartite approximation for graph wavelet signal decomposition. *IEEE Transactions on Signal Processing*, **65(20)**, 5466–5480. doi:10.1109/TSP.2017.2733489

See Also

gextrema, gsmoothing, sgemd.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# local extrema using graph object
```

```
extremaout <- gextrema(as_adjacency_matrix(gsig, attr="weight"), V(gsig)$z)
maxima <- extremaout$maxima_list; minima <- extremaout$minima_list

# Interpolation of upper, lower and mean envelope
uenvelope <- ginterpolating(as_adjacency_matrix(gsig, attr="weight"),
    V(gsig)$z, maxima)
lenvelope <- ginterpolating(as_adjacency_matrix(gsig, attr="weight"),
    V(gsig)$z, minima)
menvelope <- (uenvelope + lenvelope) / 2

# display a graph signal
gplot(gsig, size=3, legend=FALSE)

# display mean envelope
gplot(gsig, menvelope, size=3, legend=FALSE)
```

`gplot` *Plot of a Graph Signal*

Description

This function displays a signal on a graph using a color palette.

Usage

```
gplot(graph, signal = NULL, size = 1, limits = range(V(graph)$z),
    gpalette = NULL, legend = TRUE)
```

Arguments

`graph`	an **igraph** graph object with vertex attributes of coordinates `x`, `y`, a signal `z`, and edge attribute of `weight`.
`signal`	specifies a signal to be displayed over object `graph` using a color palette. When `signal=NULL`, a signal `z` of object `graph` is used.
`size`	specifies point size of vertex.
`limits`	specifies color palette limits.
`gpalette`	specifies a character vector of color palette. When `gpalette=NULL`, `c('#00008D', '#002AFF', '#00D4FF', '#2AFFD4', '#FFFF00', '#FF8D00', '#FF0000')` is used as default palette.
`legend`	if `legend=FALSE`, the legend is not included.

Details

This function displays a signal on a graph using a color palette.

Value

plot of a signal over a graph using package **ggplot2**.

See Also

gsignal, adjmatrix.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# display a signal over graph with legend
gplot(gsig, size=3, legend=TRUE)

# display a signal over graph without legend
gplotout <- gplot(gsig, size=3, legend=FALSE)
gplotout

# adding labels using ggplot2 package
gplotout +
  theme(axis.title=element_text(),
    plot.title=element_text(hjust = 0.5, vjust = 0)) +
    labs(x="x", y="y", title="visualization of a composite signal")

# deleting axis title, text and ticks using ggplot2 package
gplotout +
  theme(axis.title=element_blank(),
        axis.text=element_blank(),
        axis.ticks=element_blank())

# display high-frequency component
gplot(gsig, signal1, size=3, legend=FALSE)
```

```
# display low-frequency component
gplot(gsig, signal2, size=3, legend=FALSE)
```

---

gsignal *Graph Object with a Signal*

Description

This function constructs an **igraph** graph object with several vertex and edge attributes.

Usage

```
gsignal(vertex, edge, edgetype = c("matrix", "list"))
```

Arguments

vertex	matrix or data.frame containing vertex information. The first two columns are vertex coordinate x, y, and the third column is a signal z on each vertex.
edge	When edgetype="matrix", a square weighted adjacency matrix. This can be a sparse matrix created with **Matrix** package. When edgetype="list", matrix or data.frame of edge list with three columns. The first two columns are edge lists and the third column is an edge weight.
edgetype	edges and weights information are provided by "matrix" or "list".

Details

This function constructs an **igraph** graph object with vertex and edge attributes.

Value

an **igraph** graph object. The vertex attributes are vertex coordinate x, y, and a signal z on each vertex. The edge attribute is weight. These vertex and edge attributes can be identified by names(vertex_attr()) and names(edge_attr()), and are accessible by V(), E(), as_edgelist(), as_adjacency_matrix() or other **igraph** functions.

See Also

adjmatrix, gplot.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)
```

```
## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# vertex and edge attribute
names(vertex_attr(gsig)); names(edge_attr(gsig))

# edge list
# as_edgelist(gsig, name=FALSE)

# weighted adjacency matrix
# as_adjacency_matrix(gsig, attr="weight")

# display a noisy graph signal
gplot(gsig, size=3)

# display a composite graph signal
gplot(gsig, signal0, size=3)

# display high-frequency component
gplot(gsig, signal1, size=3)

# display low-frequency component
gplot(gsig, signal2, size=3)
```

---

gsmoothing *Smoothing a Graph Signal*

Description

This function denoises a graph signal.

Usage

```
gsmoothing(ad_mat, signal)
```

Arguments

ad_mat	an weighted adjacency matrix.
signal	a graph signal.

Details

This function denoises a graph signal utilizing the graph Fourier transform and empirical Bayes thresholding.

Value

a denoised signal.

References

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., and Vandergheynst, P. (2018). Graph signal processing: overview, challenges, and applications. *Proceedings of the IEEE 106*, 808–828. doi:10.1109/JPROC.2018.2820126

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, **30(3)**, 83–98. doi:10.1109/MSP.2012.2235192

Johnstone, I. and Silverman, B.~W. (2004). Needles and straw in haystacks: empirical Bayes estimates of possibly sparse sequences. *The Annals of Statistics*, **32**, 594–1649. doi:10.1214/009053604000000030

See Also

gextrema, ginterpolating, sgemd.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2
```

```
# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# local extrema using graph object
extremaout <- gextrema(as_adjacency_matrix(gsig, attr="weight"), V(gsig)$z)
maxima <- extremaout$maxima_list; minima <- extremaout$minima_list

# Interpolation of upper, lower and mean envelope
uenvelope <- ginterpolating(as_adjacency_matrix(gsig, attr="weight"),
  V(gsig)$z, maxima)
lenvelope <- ginterpolating(as_adjacency_matrix(gsig, attr="weight"),
  V(gsig)$z, minima)

# smoothing upper, lower and mean envelope
suenvelope <- gsmoothing(A, uenvelope)
slenvelope <- gsmoothing(A, lenvelope)

smenvelope <- (suenvelope + slenvelope) / 2

# display a graph signal
gplot(gsig, size=3, legend=FALSE)

# display mean envelope
gplot(gsig, smenvelope, size=3, legend=FALSE)
```

---

gsubway *Seoul Subway Ridership Data*

Description

This **igraph** graph object represents the number of subway passengers in Seoul, Korea for January 2021. Consider the subway stations as vertices and railroads between the stations as the edges. The number of passengers for each station is regarded as the graph signal. Seoul subway ridership data can be obtained at `https://www.seoulmetro.co.kr`.

Usage

```
data(gsubway)
```

Format

igraph graph object. The vertex attributes are longitude `x`, latitude `y`, and the number of passengers `z` for each station.

Examples

```
data(gsubway)

# attributes
names(vertex_attr(gsubway)); names(edge_attr(gsubway)); names(graph_attr(gsubway))

# standardizing the graph signal
V(gsubway)$z <- c(scale(V(gsubway)$z))

# statistical graph empirical mode decomposition (SGEMD) with boundary treatment
out <- sgemd(gsubway, nimf=1, smoothing=TRUE, boundary=TRUE, connweight="graph")

# display of a signal, denoised signal by SGEMD
gplot(gsubway, size=3)
gplot(gsubway, out$residue, size=3)
```

---

sgemd *Statistical Graph Empirical Mode Decomposition*

Description

This function performs statistical graph empirical mode decomposition.

Usage

```
sgemd(graph, nimf, smoothing = FALSE, smlevels = c(1),
    boundary = FALSE, reflperc = 0.3, reflaver = FALSE,
    connperc = 0.05, connweight = "boundary",
    tol = 0.1^3, max.sift = 50, verbose = FALSE)
```

Arguments

graph	an **igraph** graph object with vertex attributes of coordinates x, y, a signal z, and edge attribute of weight.
nimf	specifies the maximum number of intrinsic mode functions (IMF).
smoothing	specifies whether intrinsic mode functions are constructed by interpolating or smoothing envelopes. When smoothing =TRUE, denoise envelopes utilizing the graph Fourier transform and empirical Bayes thresholding.
smlevels	specifies which level of the IMF is obtained by smoothing other than interpolation.
boundary	When boundary=TRUE, a given graph is reflected for boundary treatment.
reflperc	expand a graph by adding specified percentage of a graph at the boundary when boundary=TRUE.
reflaver	specifies the method assigning signal to reflected vertices. When reflaver=TRUE, the signal on reflected vertices is produced by averaging signals on neighboring vertices on a given graph. Otherwise, signal on reflected vertices is the same to the signal on a given graph.

connperc	specifies percentage of a graph for connecting a given graph and reflected graph when boundary=TRUE.
connweight	specifies the method assigning the edge weights for connecting a given graph and reflected graph when boundary=TRUE. The edge weights are calculated by Gaussian kernel considering the relative distance between vertices. When connweight="graph", the relative distance is calculated based on the maximum distance of all the neighboring edges of a given graph. When connweight="boundary", the relative distance is calculated based on the maximum distance of the connected vertices between a given graph and reflected graph.
tol	tolerance for stopping rule of sifting.
max.sift	the maximum number of sifting.
verbose	specifies whether sifting steps are displayed.

Details

This function performs statistical graph empirical mode decomposition utilizing extrema detection of a graph signal.

Value

imf	list of IMF's according to the frequencies with imf[[1]] the highest-frequency IMF.
residue	residue signal after extracting IMF's.
nimf	the number of IMF's.
n_extrema	Each row specifies the number of local maxima and local minima of the remaining signal after extracting the i-th IMF. The first row represents the number of local maxima and local minima of a given signal.

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.- C., Tung, C. C., and Liu, H. H. (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, **454(1971)**, 903—995. doi:10.1098/rspa.1998.0193

Johnstone, I. and Silverman, B.~W. (2004). Needles and straw in haystacks: empirical Bayes estimates of possibly sparse sequences. *The Annals of Statistics*, **32**, 594–1649. doi:10.1214/009053604000000030

Kim, D., Kim, K. O., and Oh, H.-S. (2012a). Extending the scope of empirical mode decomposition by smoothing. *EURASIP Journal on Advances in Signal Processing*, **2012**, 1–17. doi:10.1186/168761802012168

Kim, D., Park, M., and Oh, H.-S. (2012b). Bidimensional statistical empirical mode decomposition. *IEEE Signal Processing Letters*, **19(4)**, 191–194. doi:10.1109/LSP.2012.2186566

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., and Vandergheynst, P. (2018). Graph signal processing: overview, challenges, and applications. *Proceedings of the IEEE 106*, 808–828. doi:10.1109/JPROC.2018.2820126

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, **30(3)**, 83–98. doi:10.1109/MSP.2012.2235192

Tremblay, N., Borgnat, P., and Flandrin, P. (2014). Graph empirical mode decomposition. *22nd European Signal Processing Conference (EUSIPCO)*, 2350–2354

Zeng, J., Cheung, G., and Ortega, A. (2017). Bipartite approximation for graph wavelet signal decomposition. *IEEE Transactions on Signal Processing*, **65(20)**, 5466–5480. doi:10.1109/TSP.2017.2733489

See Also

gextrema, gsmoothing, ginterpolating.

Examples

```
#### example : composite of two components having different frequencies

## define vertex coordinate
x <- y <- seq(0, 1, length=30)
xy <- expand.grid(x=x, y=y)

## weighted adjacency matrix by Gaussian kernel
## for connecting vertices within distance 0.04
A <- adjmatrix(xy, method = "dist", 0.04)

## signal
# high-frequency component
signal1 <- rep(sin(12.5*pi*x - 1.25*pi), 30)

# low-frequency component
signal2 <- rep(sin(5*pi*x - 0.5*pi), 30)

# composite signal
signal0 <- signal1 + signal2

# noisy signal with SNR(signal-to-noise ratio)=5
signal <- signal0 + rnorm(900, 0, sqrt(var(signal0) / 5))

# graph with signal
gsig <- gsignal(vertex = cbind(xy, signal), edge = A, edgetype = "matrix")

# graph empirical mode decomposition (GEMD) without boundary treatment
out1 <- sgemd(gsig, nimf=3, smoothing=FALSE, boundary=FALSE)

# denoised signal by GEMD
dsignal1 <- out1$imf[[2]] + out1$imf[[3]] + out1$residue


# statistical graph empirical mode decomposition (SGEMD) with boundary treatment
out2 <- sgemd(gsig, nimf=3, smoothing=TRUE, boundary=TRUE)
names(out2)
```

```
# denoised signal by SGEMD
dsignal2 <- out2$imf[[2]] + out2$imf[[3]] + out2$residue

# display of a signal, denoised signal, imf2, imf3 and residue by SGEMD
gplot(gsig, size=3)
gplot(gsig, dsignal2, size=3)
gplot(gsig, out2$imf[[2]], size=3)
gplot(gsig, out2$imf[[3]], size=3)
gplot(gsig, out2$residue, size=3)
```

Index