Package ‘CNLTreg’

July 18, 2018

Type Package

Title Complex-Valued Wavelet Lifting for Signal Denoising

Version 0.1-2

Date 2018-07-18

Author Matt Nunes [aut, cre],
       Marina Knight [aut],
       Jean Hamilton [ctb],
       Piotr Fryzlewicz [ctb]

Maintainer Matt Nunes <nunesrpackages@gmail.com>

Description

License GPL-2

Depends adlift, miscTools, nlt

Suggests MASS

NeedsCompilation no

Repository CRAN

Date/Publication 2018-07-18 12:40:08 UTC

R topics documented:

CNLTreg-package .......................................................... 2
cnl.reg ................................................................. 3
denoisepermC .......................................................... 5
denoisepermCh ......................................................... 7
fwtuppermC ............................................................ 9
mthreshC .............................................................. 11
orthpredfilters ......................................................... 14

Index 16
Description


Details

The DESCRIPTION file:

Package:       CNLTreg
Type:          Package
Title:         Complex-Valued Wavelet Lifting for Signal Denoising
Version:       0.1-2
Date:          2018-07-18
Author:        Matt Nunes [aut, cre], Marina Knight [aut], Jean Hamilton [ctb], Piotr Fryzlewicz [ctb]
Maintainer:    Matt Nunes <nunesrpackages@gmail.com>
License:       GPL-2
Depends:       adlift, miscTools, nlt
Suggests:      MASS

Index of help topics:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNLTreg-package</td>
<td>Complex-Valued Wavelet Lifting for Signal Denoising</td>
</tr>
<tr>
<td>cnlt.reg</td>
<td>Performs 'nondecimated' complex-valued wavelet lifting for signal denoising</td>
</tr>
<tr>
<td>denoisepermC</td>
<td>Denoises a signal using the complex-valued lifting transform and multivariate soft thresholding</td>
</tr>
<tr>
<td>denoisepermCh</td>
<td>Denoises a signal using the complex-valued lifting transform and multivariate soft thresholding and heteroscedastic variance computation</td>
</tr>
<tr>
<td>fwtpermpermC</td>
<td>Forward complex wavelet lifting transform</td>
</tr>
<tr>
<td>mthreshC</td>
<td>Function to perform 'multiwavelet style' level-dependent soft thresholding for complex-valued wavelet coefficients</td>
</tr>
<tr>
<td>orthpredfilters</td>
<td>Computes orthogonal filters</td>
</tr>
</tbody>
</table>

The main routines of the package are denoisepermC and cnlt.reg which perform complex-valued
lifting-based denoising, using a single or a multiple (chosen) number of lifting trajectories, respectively.

Author(s)
Matt Nunes [aut, cre], Marina Knight [aut], Jean Hamilton [ctb], Piotr Fryzlewicz [ctb]
Maintainer: Matt Nunes <nunesrpackages@gmail.com>

References

For related literature on the lifting methodology adopted in the technique, see


See Also
denoise denoiseperm nlt

---

**cnlt.reg**

*Performs ‘nondecimated’ complex-valued wavelet lifting for signal denoising*

**Description**
The transform-threshold-invert procedure for signal denoising is dependent on the trajectory (lifting order) used in the forward lifting transform. This procedure uses trajectory bootstrapping and averaging of estimates to gain denoising performance

**Usage**
```
cnlt.reg(x, f, p, returnall = FALSE, nkeep = 2, ...)
```

**Arguments**
- **x** Vector of any length (not necessarily equally spaced) that gives the grid on which the signal is observed.
- **f** Vector of the same length as x that gives the signal values corresponding to the x-locations.
- **p** Number of trajectories to be used by the nondecimated lifting algorithm.
returnall  Indicates whether the function returns useful variables or just the denoised data-points.
nkeep   Number of scaling points we want at the end of the transform. The usual choice is nkeep=2.
...     Any other arguments to be passed to denoisepermC, see the function documentation for more details.

Details

Essentially, this function applies the complex wavelet lifting denoising procedure denoisepermC P times, each with a different random lifting trajectory. This results in P estimates of the (unknown) true signal. The average of these estimators is the proposed estimator.

Value

If returnall=FALSE, the estimate of the function after denoising. If returnall=TRUE, a list with components:

vec A matrix of dimension P x (n - nkeep), each row corresponding to a different lifting trajectory.
aveghat Estimated signal after removing the noise.

Warning

Using a large number of trajectories for long datasets could take a long time!

Author(s)

Matt Nunes

References


For the real-valued equivalent procedure, see also

See Also

denoisepermCh, fwtppermC, mthreshC, nlt
denoisepermC

Examples

library(adlift)

# construct an (irregular) observation grid
x<-runif(256)

#construct the true, normally unknown, signal
g<-make.signal2("blocks",x=x)

#generate noise with mean 0 and signal-to-noise ratio 5
noise<-rnorm(256,mean=0,sd=sqrt(var(g))/5)

#generate a noisy version of g
f<-g+noise

# decide on a number of random trajectories to be used (e.g. J=5 below), and apply
# the nondecimated lifting transform to the noisy signal (x,f):
## Not run:
est<-cnlt.reg(x,f,J=50,LocalPred=AdaptPred,neighbours=1,returnall=FALSE)
## End(Not run)

denoisepermC  Denoises a signal using the complex-valued lifting transform and multivari ate soft thresholding

Description

Denoises an input signal contaminated by noise. First the signal is decomposed using the complex-valued lifting scheme (see fwtnpermC) using an order of point removal. The resulting complex-valued wavelet coefficients are then thresholded using a soft thresholding rule on the details’ magnitude. The transform is inverted and an estimate of the noisy signal is obtained.

Usage

denoisepermC(x, f, returnall = FALSE, sdtype = "adlift", verbose = FALSE, ...)

Arguments

x  Vector of any length (not necessarily equally spaced) that gives the grid on which the signal is observed.

f  Vector of the same length as x that gives the signal values corresponding to the x-locations.
denoisepermC

returnall  Indicates whether the function returns useful variables or just the denoised datapoints.
sdtype   Options are either "adlift" or "complex", indicating whether the noise variance is estimated with the average of the mean absolute deviations of both real and imaginary components of the finest wavelet coefficients, or just the real component, as in denoise.
verbose  Indicates whether useful messages should be printed to the console during the procedure.
...  Any other arguments to be passed to fwtpermpermC, see documentation for this function for more details.

Details

After the complex lifting transform is applied, the wavelet coefficients are divided into artificial levels. The details from the lifting scheme have different variances, and will therefore be normalized to have the same variance as the noise, by using the lifting matrix. Those normalized details falling into the finest artificial level will be used for estimating the standard deviation of the noise that contaminated the signal. The variable sdtype is used for this estimate, see Appendix B of Hamilton et al. (2018) for more details. Using this estimate, the normalized details can then be thresholded and un-normalized. The transform is then inverted to give an estimate of the signal.

Value

If returnall=FALSE, the estimate of the function after denoising. If returnall=TRUE, a list with components:

fhat  Estimated signal after removing the noise.
w    This is the matrix associated to the modified lifting transform.
indsd  Vector giving the standard deviations of the detail and scaling coefficients.
al  List giving the split of points between the artificial levels.
sd  Estimated standard deviation of the noise.

Author(s)

Matt Nunes, Marina Knight

References


See Also
denoisepermCh,fwtpermpermC,mthreshC,cnlt.reg
denoisepermCh

Examples

library(adlift)

# construct an (irregular) observation grid
x<-runif(256)

# construct the true, normally unknown, signal
G<-make.signal2("blocks",x=x)

# generate noise with mean 0 and signal-to-noise ratio 5
noise<-rnorm(256,mean=0, sd=sqrt(var(G))/5)

# generate a noisy version of G
f<-G+noise

# perform the complex-valued denoising procedure to the noisy signal (x,f):
est<-denoisepermC(x,f,LocalPred=AdaptPred,neigh=1,returnall=FALSE)

denoisepermCh Denoises a signal using the complex-valued lifting transform and multivariate soft thresholding and heteroscedastic variance computation

Description

Denoises an input signal contaminated by noise. First the signal is decomposed using the complex-valued lifting scheme (see fwtppermC) using an order of point removal. A sliding window approach is then used on these wavelet coefficients to estimate a local noise variance. The resulting complex-valued wavelet coefficients are then thresholded using a soft thresholding rule on the details' magnitude. The transform is inverted and an estimate of the noisy signal is obtained.

Usage

denoisepermCh(x, f, returnall = FALSE, verbose = FALSE, ...)

Arguments

x Vector of any length (not necessarily equally spaced) that gives the grid on which the signal is observed.
f Vector of the same length as x that gives the signal values corresponding to the x-locations.
returnall Indicates whether the function returns useful variables or just the denoised datapoints.
verbose Indicates whether useful messages should be printed to the console during the procedure.
... Any other arguments to be passed to fwtpperm and fwtppermC.
Details

After the complex lifting transform is applied, the wavelet coefficients are divided into artificial levels. The details from the lifting scheme have different variances, and will therefore be normalized to have the same variance as the noise, by using the lifting matrix. A sliding window is used to compute a local 'heteroscedastic' noise variance by taking the MAD of those normalized details falling into the window, see Nunes et al. (2006) for more details. Given the noise estimates for each observation, the normalized details can then be thresholded and un-normalized. The transform is then inverted to give an estimate of the signal.

Value

If `returnall=FALSE`, the estimate of the function after denoising. If `returnall=TRUE`, a list with components:

- `fhat` Estimated signal after removing the noise.
- `w` This is the matrix associated to the modified lifting transform.
- `al` List giving the split of points between the artificial levels.
- `sd` Estimated heteroscedastic standard deviation of the noise.

Author(s)

Matt Nunes, Marina Knight

References


See Also

denoisepermC, fwtinpermC, fwtinperm, heterovar, mthreshC

Examples

```r
library(MASS) # where the motorcycle data lives

mcycleu<-mcycle[which(duplicated(mcycle$times)=='FALSE'),]
time<-mcycleu[,1]
accel<-mcycleu[,2]

set.seed(200)
```
**Description**

Performs the complex-valued lifting transform on a signal with grid \( x \) and corresponding function values \( f \).

**Usage**

\[
\text{fwtnppermC}(x, f, \text{LocalPred} = \text{LinearPred}, \text{neighbours} = 1, \\
\text{intercept} = \text{TRUE}, \text{closest} = \text{FALSE}, \text{nkeep} = 2, \\
\text{mod} = \text{sample}(1:\text{length}(x), (\text{length}(x) - \text{nkeep}), \text{FALSE}))
\]

**Arguments**

- **x**
  A vector of grid values. Can be of any length, not necessarily equally spaced.

- **f**
  A vector of function values corresponding to \( x \). Must be of the same length as \( x \).

- **LocalPred**
  The type of regression to be performed in the prediction lifting step. Possible options are LinearPred, QuadPred, CubicPred, Adaptpred and AdaptNeigh.

- **neighbours**
  The number of neighbours over which the regression is performed at each step. If closest is FALSE, then this in fact denotes the number of neighbours on each side of the removed point.

- **intercept**
  Indicates whether or not the regression prediction includes an intercept.

- **closest**
  Refers to the configuration of the chosen neighbours. If closest is FALSE, the neighbours will be chosen symmetrically around the removed point. Otherwise, the closest neighbours (in distance) will be chosen.

- **nkeep**
  The number of scaling coefficients to be kept in the final representation of the initial signal. This must be at least two.

- **mod**
  Vector of length (length(\( x \)) - \text{nkeep}). This gives the trajectory for the lifting algorithm to follow, i.e. it gives the order of point removal.

**Details**

Given \( n \) points on a line, \( x \), each with a corresponding \( envf \) value this function computes the complex-valued lifting transform of the \((x,f)\) data. This is similar in spirit to the real-valued lifting transform in \text{fwtnpperm}, except that the algorithm constructs *two* orthogonally linked prediction filters, as in Section 2.2 of Hamilton et al. (2018). A summary of the procedure is as follows:

**Step One.** Compute "integrals" associated to each point, representing the intervals that each grid-point \( x_i \) spans.

Then for each point index in the lifting trajectory \text{mod},
Step Two(a). The neighbours of the removed point are identified using the specified neighbour configuration. The value of $f$ at the removed point is predicted using the specified regression over the neighbours, unless an adaptive procedure is chosen. In this case, the algorithm chooses the regression which produces the minimal detail coefficient (in magnitude) from a range of regression types (see `AdaptPred` or `AdaptNeigh` for more information). In either case, the regression specifies a local filter of the function values over the neighbourhood, $L$.

Step Two(b). A second filter, $M$, is then constructed orthogonal to $L$, such that it has unit norm, see Hamilton et al. (2018) for more details.

The differences between the removed point’s $f$ value and the predictions using the two filters are computed, which constitute the real and imaginary parts of the complex-valued wavelet coefficient. This coefficient is then stored.

Step Three. The integrals and the scaling function values (neighbouring `coeffv` values) are updated according to the filter $L$.

The algorithm continues until all points in `mod` are removed.

**Value**

- `coeff` matrix of detail and scaling coefficients in the wavelet decomposition of the signal; first column: real component, second column: imaginary component.
- `lengthsremove` vector of interval lengths corresponding to the points removed during the transform (in `removelist`).
- `pointsin` indices into $X$ of the scaling coefficients in the wavelet decomposition. These are the indices of the $X$ values which remain after all points in `removelist` have been predicted and removed. This has length `nkeep`.
- `removelist` a vector of indices into $X$ of the lifted coefficients during the transform (in the order of removal).
- `gamlist` a list of all the prediction weights used at each step of the transform; each list entry is a matrix of two rows, corresponding to the filters $L$ and $M$.
- `alphalist` a list of the update coefficients used in the update step of the decomposition.
- $w$ The complex-valued lifting matrix associated to the transform.
- `reo` An index into the observations indicating a reordering to give 1:n. This is reported for convenience for other functions, and is not intended for use by the user.
- `coeffv` vector of complex-valued detail and scaling coefficients in the wavelet decomposition of the signal; contains the same information as `coeff`.
- `ialpha` Vector of "irregularity degree" measures corresponding to each lifting step of the transform. Note that this is returned for convenience in other functions, and is not intended for use by the user.

**Author(s)**

Matt Nunes, Marina Knight
mthreshC

References


For related literature on the lifting methodology adopted in the technique, see


See Also

AdaptNeigh, AdaptPred, CubicPred, denoisepermC, denoisepermCh, LinearPred, orthpredfilters, QuadPred

Examples

library(adlif)

# construct an (irregular) observation grid
x<-runif(256)

# construct a signal
f<-make.signal2("blocks",x=x)

fwd<-fwtnpermC(x,f,LocalPred=AdaptPred,neigh=1,closest=FALSE)

# have a look at the complex-valued coefficients and the removal trajectory:

fwd$coeffv

fwd$removelist

mthreshC

*Function to perform 'multiwavelet style' level-dependent soft thresholding for complex-valued wavelet coefficients*

Description

This function uses chi^2 statistics similar to Barber and Nason (2004) to threshold wavelet coefficients based on their magnitude

Usage

mthreshC(coeffs, Sigma, r1, po, ali, verbose = FALSE)
Arguments

coeffv  A matrix of complex-valued wavelet coefficients (columns are real and imaginary parts of the coefficients respectively).
Sigma  An array of dimension $2 \times 2 \times n$ describing the covariance between real and imaginary parts of the wavelet coefficients. In particular, $\Sigma[\cdot, i]$ represents the covariance between real and imaginary parts of the $i$th lifted wavelet coefficient (see r1 argument).
rl  The removelist (trajectory of lifted points) corresponding to a forward lifting transform.
po  A vector of indices describing the unlifted scaling coefficients in a forward lifting transform.
ali  A list of indices of observations, each entry corresponding to an ’artificial level’ (finest to coarsest), see artlev for more details.
verbose  Indicates whether helpful messages should be printed to the console during the procedure.

Details

The procedure in Downie and Silverman (1998) or Barber and Nason (2004) makes use of the magnitude of wavelet coefficients to threshold them. In particular, the covariance between the components of the wavelet coefficients (contained in Sigma) is taken into account to compute a thresholding statistic, the distribution of which is chi-squared distributed, see cthresh for more details. These statistics are then compared with level-dependent universal thresholds computed by counting the number of coefficients in specific artificial levels.

Value

A list with the following components:

chi  the vector of chi-squared statistics used in the thresholding procedure.
coeffvt  the matrix of thresholded coefficients, columns representing the real and imaginary components respectively.

Author(s)

Matt Nunes, Marina Knight

References


See Also

cthresh, denoisepermC, denoisepermCh
Examples

library(adlift)

set.seed(100)

# construct an (irregular) sampling structure:
x<-sort(runif(200))
g<-make.signal2("bumps",x=x)

# generate IID noise with a particular sd
noise<-rnorm(200,0,sd=0.5)
f<-g+noise

# perform forward complex lifting transform
out<-fwtpermC(x,f,LocalPred=LinearPred,neigh=1)

# have a look at some of the coefficients
out$coeffv[1:10]

# extract lifting matrix and induced lifting variances
W <- out$W
Gpre<-tcrossprod(W,Conj(W))

indsd<-sqrt(diag(Gpre))

# now estimate noise sd using the first artificial level:
al<-artlev(out$lengthsremove,out$removelist)
fine<-(out$coeffv/indsd)[al[[1]]]

varest<-mad(fine)^2

# now compute coefficient covariance structure, see
# Hamilton et al. (2018), Appendix B

C = varest * tcrossprod(W)
G = varest * Gpre
P = Conj(G) - t(Conj(C))
Sigma <- array(0, dim = c(2, 2, length(out$coeffv)))
Sigma[1, 1, ] <- diag(Re(G + C)/2)
Sigma[2, 2, ] <- diag(Re(G - C)/2)
Sigma[1, 2, ] <- -diag(Im(G - C)/2)
Sigma[2, 1, ] <- diag(Im(G + C)/2)
orthpredfilters

Computes orthogonal filters

Description
Given a filter L, finds a second filter M, orthogonal to L and with unit norm

Usage
orthpredfilters(filter = c(0.5, 1, 0.5))

Arguments
filter An initial filter L.

Details
See Hamilton et al. (2018), section 2.2.

Value
A matrix with two rows, the first row corresponding to L, the second corresponding to the orthogonal filter M.

Warning
At present only works with odd length filters

Author(s)
Marina Knight, Matt Nunes

References

See Also
fwtnppermC
Examples

# create a vector representing a filter for one neighbour either side of a removed point
# (equally weighted):

L = c(0.5, 1, 0.5)

# now work out a unit-norm filter orthogonal to L

out <- orthpredfilters(L)

# M should be the second row:

out[2,]
Index

*Topic **manip**
  mthreshC, 11
  orthpredfilters, 14

*Topic **methods**
  cnlt.reg, 3
  denoisepermC, 5
  denoisepermCh, 7
  fwtpppermC, 9

*Topic **package**
  CNLTreg-package, 2

*Topic **regression**
  cnlt.reg, 3
  denoisepermC, 5
  denoisepermCh, 7

AdaptNeigh, 10, 11
AdaptPred, 10, 11
artlev, 12

cnlt.reg, 2, 3, 6
CNLTreg (CNLTreg-package), 2
CNLTreg-package, 2
cthresh, 12
CubicPred, 11

denoise, 3, 6
denoiseperm, 3
denoisepermC, 2, 4, 5, 8, 11, 12
denoisepermCh, 4, 6, 7, 11, 12

fwtppperm, 7–9
fwtpppermC, 4–8, 9, 14

heterovar, 8
LinearPred, 11
mthreshC, 4, 6, 8, 11
nlt, 3, 4
orthpredfilters, 11, 14