Package ‘nlt’

February 20, 2015

Type  Package
Title  A nondecimated lifting transform for signal denoising
Version  2.1-3
Date  11/11/2012
Author  Marina Knight, Matt Nunes
Maintainer  Matt Nunes <m.nunes@lancs.ac.uk>
Depends  EbayesThresh, adlift (>= 1.3)
Description  Uses a modified lifting algorithm on which it builds the nondecimated lifting transform. It has applications in wavelet shrinkage.
License  GPL
Repository  CRAN
Date/Publication  2012-11-11 14:19:57

R topics documented:

  denoiseperm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  fwtupperm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  nlt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  transmatdualperm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Index

denoiseperm

Denoise a signal using the modified lifting transform and empirical Bayes thresholding

1
Description

Denoises an input signal contaminated by noise. First the signal is decomposed using the modified lifting scheme (coded in `fwtnperm`) using a prespecified order, known as path or trajectory, of point removal. Once the signal is decomposed into wavelet coefficients (or details), these are subjected to an empirical Bayes shrinkage procedure in order to remove the noise, the transform is inverted and an estimate of the noisy signal is obtained.

Usage

```r
denoiseperm(x, f, pred=LinearPred, neigh=1, int=TRUE, clo=FALSE, keep=2,
rule = "median", per = sample(1:length(x),(length(x)-keep),FALSE),returnall=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.
- **pred** The type of regression to be used in the prediction step of the modified lifting algorithm. Choices are linear, quadratic or cubic (respectively, `LinearPred`, `QuadPred` or `CubicPred`), or two adaptive procedure which automatically choose the degree used in regression, (`AdapPred` or `AdaptNeigh`).
- **neigh** Number of neighbours to be used in order to construct the neighbourhood of each point that has to be removed. If `clo=FALSE`, this gives the number of neighbours on each side of the removed point.
- **int** Specifies whether (int=TRUE) or not (int=FALSE) an intercept is to be used in the regression curve. For pred=AdapPred or AdaptNeigh, the algorithm automatically makes this choice.
- **clo** If (clo=TRUE) or (clo=FALSE), then at each step the neighbours are in closest, respectively symmetrical configuration.
- **keep** Number of scaling points we want at the end of the transform. The usual choice is keep=2.
- **rule** The type of Bayesian shrinkage technique, with possible choices posterior median ("median") or posterior mean ("mean").
- **per** Vector of length (length(x)-keep) which gives the order of point removal in the lifting algorithm.
- **returnall** Indicates whether the function returns useful variables or just the denoised datapoints.

Details

Once the modified lifting transform is applied, the wavelet coefficients are divided into artificial levels. The details obtained by means of a lifting scheme have different variances, and will therefore be normalized to have the same variance as the noise. Those normalized details falling into the finest artificial level will be used for estimating the standard deviation of the noise that contaminated the signal. Using this estimate, the normalized details can then be shrunk and un-normalized.
(using package 'EbayesThresh'), and the transform inverted (using the function invtnp of package 'adlift') to give an estimate of the signal. The choices for pred can be found in the package 'adlift'.

**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.
- inds: 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.

**Note**

Use this function together with the "adlift" and "EbayesThresh" packages available from CRAN.

**Author(s)**

Marina Knight (marina.knight@bristol.ac.uk)

**References**

See the paper 'A "nondecimated" lifting transform' by Knight, M.I. and Nason, G.P. (2008) for further details.

**See Also**

fwtnpperm, fwtnppperm, and also invtnp of package 'adlift'

**Examples**

```r
#construct a grid
x<-runif(256)

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

#now generate noise (here with mean 0 and signal-to-noise ratio 3)
noise<-rnorm(256,mean=0, sd=sqrt(var(g))/3)

#obtain a noisy version of the true signal g
f<-g+noise

#construct the trajectory which will indicate the order of point removal that will be followed by the modified lifting algorithm
#vec below gives the first (length(x)-keep) entries of a random permutation of (1:length(x))
vec<-sample(1:256,254,FALSE)
```
# denoise the signal (x,f) by applying the modified lifting transform following the removal order in vec and using 
# and neighbourhoods of size 2 in symmetrical configuration
# the details are then thresholded using posterior medians and the algorithm inverted
# the proposed estimate of g is given by out$\hat{f}$
out<-denoiseperm(xLf, pred=AdapPred, neigh=1, int=TRUE, clo=FALSE, keep=2, rule="median", per=vec)

**Description**

Performs the lifting transform on a signal with grid input and corresponding function values f. There is a unique correspondence between the grid values and the function values. Can also cope with length vector input instead of gridpoint vector input.

**Usage**

```r
fwtnpperm (input, f, LocalPred = LinearPred, neighbours = 1,
intercept = TRUE, closest = FALSE, nkeep = 2, initboundhandl = "reflect", mod =
sample(1:length(input), (length(input) - nkeep), FALSE),
doW = FALSE, varonly = FALSE)
```

**Arguments**

- **input**: A vector of grid values. Can be of any length, not necessarily equally spaced.
- **f**: A vector of function values corresponding to input. Must be of the same length as input.
- **LocalPred**: The type of regression to be performed. Possible options are LinearPred, QuadPred, CubicPred, AdapPred 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 curve 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 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.
- **initboundhandl**: Variable specifying how to handle the boundary at the start of the transform. Possible values are "reflect" - the intervals corresponding to the first and last datapoints are taken to have the respective grid values as midpoints; and "stop" - the first and last intervals have the first and last grid values (respectively) as outer endpoints.
- **mod**: Vector of length (length(x)-keep). It gives the trajectory for the modified lifting algorithm to follow, i.e. it gives the order of point removal.
...
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 \texttt{removelist} have been predicted and removed. This has length \texttt{nkeep}. \n
\texttt{removelist} \a vector of indices into \( X \) of the lifted coefficients during the transform (in the order of removal). \n
\texttt{neighbrs} \a list of indices into \( X \). Each list entry gives the indices of the neighbours of the removed point used at that particular step of the transform. \n
\texttt{neighbours} \the user-specified number of neighbours used in the prediction step of the transform. \n
\texttt{gamlist} \a list of all the prediction weights used at each step of the transform. \n
\texttt{alphalist} \a list of the update coefficients used in the update step of the decomposition. \n
\texttt{schemehist} \a vector of character strings indicating the type of regression used at each step of the transform. \n
\texttt{interhist} \a boolean vector indicating whether or not an intercept was used in the regression curve at each step. \n
\texttt{clolist} \a boolean vector showing whether or not the neighbours were symmetrical (FALSE) about the removed point during the transform. This is NULL except when \texttt{LocalPred=AdaptNeigh}. \n
\section*{Author(s)} \n
Matt Nunes (m.nunes@lancs.ac.uk), Marina.Knight \n
\section*{See Also} \n
AdaptNeigh, AdaptPred, CubicPred, fwttnpmp, invtnp, LinearPred, QuadPred. \n
\section*{Examples} \n
\begin{verbatim} \# \# Generate some one-dimensional data: 100 observations. \# \input <- runif(100) \f <- input^2 - 3*input \# \# Compute fwttnp function on this data \# \vec <- sample(1:100,98,FALSE) \# \texttt{out} <- fwttnpperm(input,f,LocalPred=AdaptPred,neighbours=2,closest=TRUE,mod=vec) \# \# That's it. \# \end{verbatim}
Denoise a signal using a nondecimated lifting transform

Description

Starting with a noise-contaminated signal, we decompose it using a 'nondecimated' lifting algorithm (i.e. by applying the modified lifting transform following several random paths), shrink all the obtained detail coefficients and invert each transform to give an estimated signal. The average of all these estimates is the final proposal for estimating the true (unknown) signal.

Usage

\[ \text{nlt}(x, f, J, \text{Pred = AdapPred}, \text{neighbours = 1, closest = FALSE, intercept = TRUE, nkeep = 2, trule = "median"}) \]

Arguments

- **x**: Vector of any length (possibly irregularly spaced) that gives the grid locations where the signal is observed.
- **f**: Vector of the same length as \( x \) that gives the signal values corresponding to the \( x \)-locations.
- **J**: Number of trajectories to be used by the nondecimated lifting algorithm.
- **Pred**: The type of regression to be used in the prediction step of the modified lifting algorithm. Choices are linear, quadratic or cubic (respectively, \text{linearpred}, \text{quadpred} or \text{cubicpred}), or two adaptive procedures which automatically choose the degree used in regression, (\text{adaptpred} or \text{adaptneigh}).
- **neighbours**: Number of neighbours to be used for defining the neighbourhood of each point that has to be removed. If (\text{closest}=\text{FALSE}), then this gives the number of neighbours to be used on each side of the removed point.
- **closest**: If (\text{closest}=\text{TRUE}) or (\text{closest}=\text{FALSE}), then at each step the neighbours are in closest, respectively symmetrical configuration.
- **intercept**: Specifies whether (\text{intercept=TRUE}) or not (\text{intercept=FALSE}) an intercept is to be used in the regression curve. For \text{Pred=AdapPred} or \text{AdaptNeigh}, the algorithm automatically makes this choice.
- **nkeep**: Number of scaling points we want at the end of the application of the transform. The usual choice is \text{nkeep}=2.
- **trule**: The type of Bayesian shrinkage technique, with possible choices posterior median ("median") or posterior mean ("mean").
- **verbose**: A boolean indicating whether extra information should be printed.
- **do.orig**: A boolean indicating whether the original \text{adlift} algorithm should also be computed.
- **returnall**: A boolean indicating whether the function returns useful variables or just the denoised datapoints.
Details

Essentially, this function applies $J$ times the modified lifting algorithm (that can be found in `fwtnpperm`), and removes the noise from all sets of detail coefficients by using empirical Bayes shrinkage (of package `EbayesThresh`). Inverting (by means of the function `invtnp` of the package `adlift`) each transform consequently results in $J$ estimates of the (unknown) true signal. The average of these estimators is our proposed estimator. The functions that appear as choices for `Pred` can be found in the package `adlift`.

Value

- **vec**: Matrix whose rows give the trajectories to be used by the nondecimated lifting algorithm.
- **ghatnat**: Vector that gives the estimated true signal given by denoising using the lifting scheme that establishes its own order for removing the points (but with the same specification for prediction stage and neighbourhood as the modified algorithm), rather than a randomly generated order.
- **aveghat**: Estimated signal, obtained as the average of the individual estimates from the random trajectory runs.

Note

Use this function together with the "adlift" and "EbayesThresh" packages available from CRAN.

Author(s)

Marina Knight (marina.knight@bristol.ac.uk)

References

See the paper 'A "nondecimated" lifting transform.' by Knight, M.I. and Nason, G.P. (2009) for further details.

See Also

denoiseperm, fwtnpperm, fwtnpperm, and also `invtnp` of package 'adlift'

Examples

```r
# construct the 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
```
transmatdualperm

# decide on a number of random trajectories to be used (below J=100, in paper J=20, 30), and apply the nondecimated 
# below we apply the modified lifting transform J times, each time following a different path, and using adaptive p 
# in closest configuration; all details are then thresholded using posterior medians and the algorithms inverted 
# the aggregate estimator of g proposed by our method is found in out$aveghat
out<-nlt(x,f,J=10,Pred=AdapPred,neighbours=2,closest=TRUE,intercept=TRUE,nkeep=2,rule="median")

transmatdualperm  transmatdualperm

Description

Works out the transform matrix for a particular prediction scheme and neighbourhood structure.

Usage

transmatdualperm(x, f, Pred = AdapPred, neigh = 1, int = TRUE, clo = 
TRUE, keep = 2, perm = 
sample(1:length(x), (length(x)-keep), FALSE), varonly=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.
Pred The type of regression to be performed. Possible options are LinearPred, 
QuadPred, CubicPred, AdaptPred and AdapPred.
neigh The number of neighbours over which the regression is performed at each step. 
If clo is false, then this in fact denotes the number of neighbours on each side 
of the removed point.
int Indicates whether or not the regression curve includes an intercept.
clo Refers to the configuration of the chosen neighbours. If clo is false, the neigh-
bours will be chosen symmetrically around the removed point. Otherwise, the 
closest neighbours will be chosen.
keep The number of scaling coefficients to be kept in the final representation of the 
initial signal. This must be at least two.
perm Vector of length (length(x)-keep). It gives the trajectory for the modified lifting 
algorithm to follow, i.e. it gives the order of point removal.
varonly A boolean variable indicating whether only the coefficient variances should be 
returned, i.e. just the diagonal of the transform matrix \( \mathbf{W}_{\text{new}} \).

Details

The function uses \texttt{amatdual} to form the refinement matrices \( A_j \), from which the augmented matrices \( T_j \) are constructed. This process is iterated, to find the transform matrix (the top level augmented 
matrix). The rows and columns of this matrix are then reordered to be in the order of \texttt{out$coeff}, 
i.e. so that the columns correspond to \( f_1 \ldots f_n \).
Value

out the output from the forward transform.
wnew the matrix associated to the wavelet transform.
x the original gridpoint vector.

Note

This function has been left in the package for completeness. However, the transform matrix is (optionally) computed within the forward lifting transform function \texttt{fwt\textsc{np}}.

Author(s)

Matt Nunes (<m.nunes@lancs.ac.uk>), Marina Knight

See Also

\texttt{fwt\textsc{np}}

Examples

```r
x1<-runif(10)
y1<-make.signal2("doppler",x=x1)
#
vec<-sample(1:10,8,FALSE)

a<-transmatdualperm(x1,y1,AdaptNeigh,2,TRUE,TRUE,2,perm=vec)
#
a$wnew
#
#the transform matrix for this adaptive lifting scheme
Index

*Topic array
  transmatdualperm, 9
*Topic methods
  fwtntpperm, 4
*Topic nonparametric
  denoiseperm, 1
  nlt, 7

AdaptNeigh, 6, 9
AdaptPred, 6, 9

CubicPred, 6, 9

denoiseperm, 1, 8

fwtnp, 10
fwtnpmp, 6
fwtnpperm, 2, 3, 4, 8

invtnp, 6

LinearPred, 6, 9

nlt, 7

QuadPred, 6, 9

transmatdualperm, 9