Package 'nlt'

July 22, 2025

Type Package

Title A Nondecimated Lifting Transform for Signal Denoising

Version 2.2-2

Date 2025-04-08

Author Marina Knight [aut],
Matt Nunes [aut, cre]

Maintainer Matt Nunes <nunesrpackages@gmail.com>

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 2025-04-09 07:50:02 UTC

NeedsCompilation no

Contents

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

Description

Denoises an input signal contaminated by noise. First the signal is decomposed using the modified lifting scheme (coded in fwtnpperm) 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

```
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, (AdaptPred 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=AdaptPred 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 coeficients 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.
`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.

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

Examples

```
# 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 adaptive prediction
# 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$fhat$coeff

out<-denoiseperm(x,f,pred=AdaptPred,neigh=1,int=TRUE,clo=FALSE,keep=2,rule="median",per=vec)
```

`fwtnpperm` *fwtnpperm*

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

```
fwtnpperm (input, f, LocalPred = LinearPred, neighbours = 1,
intercept = TRUE,closest = FALSE, nkeep = 2, initboundhandl = "reflect", mod =
    sample(1:length(input), (length(input) - nkeep), FALSE),
    do.W = 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`, `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 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.
do.W	A boolean indicating whether the transform matrix should be computed and returned.
varonly	A boolean indicating whether only the coefficient variances should be returned (if do.W=TRUE).

Details

Given n points on a line, input, each with a corresponding f value this algorithm computes a lifting transform of the (input,f) data. If lengths are inputted (inputtype="lengths"), then the gridpoints are taken to be the left endpoints of the intervals defined by the lengths inputted. Step One. Order the grid values so that corresponding intervals can be constructed.

Step Two. Compute "integrals" for each point. For each point its integral is the length of the interval associated to the gridpoint.

Step Three. Identify the point to remove as that with the smallest integral. Generally, we remove points in order of smallest to largest integral. The integrals of neighbours of removed points change at each step.

Step Four(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 curve over the neighbours, unless an adaptive procedure is chosen. In this case, the algorithm adjusts itself. The difference between the removed point's f value and the prediction is computed: this is the wavelet coefficient for the removed point. The difference replaces the function value in the vector coeff at the removed point's location. In this way wavelet coefficients gradually overwrite (scaling) function values in coeff.

Step Four(b). The integrals and the scaling function values (other coeff values) of neighbours of the removed point are updated. The values of the rest of the scaling coefficients are unaffected.

Step Five. Return to step 3 but in the identification of a point to remove the updated integrals are used.

The algorithm continues until as many points as desired are removed. If do.W=TRUE, the predict and update lifting steps are used to propogate coefficient contributions to the transform matrix W. If varonly=TRUE, only the (detail and scaling) coefficient variances are returned. After each lifting step, the coefficient variance is computed and the transform matrix row corresponding to the lifted coefficient is deleted for the next stage (minimal storage efficiency). The transform matrix is not returned (i.e. W=NULL).

Value

X	data vector of the grid used in the transform.

coeff	vector of detail and scaling coefficients in the wavelet decomposition of the signal.
origlengths	vector of initial interval lengths corresponding to the gridpoints.
lengths	vector of (updated) interval lengths at the end of the transform. This is of length nkeep.
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).
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.
neighbours	the user-specified number of neighbours used in the prediction step of the transform.
gamlist	a list of all the prediction weights used at each step of the transform.
alphalist	a list of the update coefficients used in the update step of the decomposition.
schemehist	a vector of character strings indicating the type of regression used at each step of the transform.
interhist	a boolean vector indicating whether or not an intercept was used in the regression curve at each step.
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 LocalPred=AdaptNeigh.

Author(s)

Matt Nunes (<nunesrpackages@gmail.com>), Marina.Knight

See Also

AdaptNeigh, AdaptPred, CubicPred, fwtnpmp, invtnp, LinearPred, QuadPred

Examples

```
#
# Generate some one-dimensional data: 100 observations.
#
input <- runif(100)
f <- input^2 - 3*input
#
# Compute fwtnp function on this data
#
vec<-sample(1:100,98,FALSE)
```

```
out <- fwtnpperm(input,f,LocalPred=AdaptPred,neighbours=2,closest=TRUE,mod=vec)
#
# That's it.
#
```

---

nlt *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

```
nlt(x, f, J, Pred = AdaptPred, neighbours = 1, closest = FALSE, intercept = TRUE,
nkeep = 2, trule = "median",verbose = TRUE,do.orig = FALSE, returnall = FALSE)
```

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, LinearPred, QuadPred or CubicPred), or two adaptive procedures which automatically choose the degree used in regression, (AdaptPred or AdaptNeigh).
neighbours	Number of neighbours to be used for defining the neighbourhood of each point that has to be removed. If (closest=FALSE), then this gives the number of neighbours to be used on each side of the removed point.
closest	If (closest=TRUE) or (closest=FALSE), then at each step the neighbours are in closest, respectively symmetrical configuration.
intercept	Specifies whether (intercept=TRUE) or not (intercept=FALSE) an intercept is to be used in the regression curve. For Pred=AdaptPred or 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 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 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

```
# 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

# decide on a number of random trajectories to be used (below J=100, in paper J=20,30), and apply
# the nondecimated lifting transform to the noisy signal (x,f)
#
# below we apply the modified lifting transform J times, each time following a different path,
# and using adaptive prediction with neighbourhoods of size 2 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=AdaptPred,neighbours=2,closest=TRUE,intercept=TRUE,nkeep=2,trule="median")
```

`transmatdualperm` *transmatdualperm*

Description

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

Usage

```
transmatdualperm(x, f, Pred = AdaptNeigh, 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 `AdaptNeigh`.
`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 neighbours 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 `Wnew`.

Details

The function uses `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 `out$coeff`, i.e. so that the columns correspond to $f_1 \dots f_n$.

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 `fwtnp`.

Author(s)

Matt Nunes (<nunesrpackages@gmail.com>), Marina Knight

See Also

`fwtnp`

Examples

```
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