Package 'CVThresh'

January 20, 2025

Title Level-Dependent Cross-Validation Thresholding

Version 1.1.2

Date 2022-05-02

Author Donghoh Kim <donghoh.kim@gmail.com>,
Hee-Seok Oh <heeseok@stats.snu.ac.kr>

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

Depends R (>= 2.15.1), wavethresh (>= 4.6.1), EbayesThresh (>= 1.3.2)

Description The level-dependent cross-validation method is implemented
for the selection of thresholding
value in wavelet shrinkage. This procedure is implemented
by coupling a conventional cross validation with an
imputation method due to a limitation of data length,
a power of 2. It can be easily applied to classical
leave-one-out and k-fold cross validation.
Since the procedure is computationally fast,
a level-dependent cross validation can be performed for
wavelet shrinkage of various data such as a data
with correlated errors.

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-02 03:20:02 UTC

Contents

`cvimpute.by.wavelet` *Imputation by wavelet*

Description

This function performs imputation for test dataset of cross-validation given test dataset index and initial values.

Usage

```
cvimpute.by.wavelet(y, impute.index, impute.tol=0.1^3,
    impute.maxiter=100, impute.vscale="independent",
    filter.number=10, family="DaubLeAsymm", ll=3)
```

Arguments

`y`	observation
`impute.index`	test dataset index for cross-validation
`impute.tol`	tolerance for imputation
`impute.maxiter`	maximum iteration for imputation
`impute.vscale`	specifies whether variance is adjusted level-by-level or not. “level" or “independent"
`filter.number`	specifies the smoothness of wavelet in the decomposition (argument of WaveThresh)
`family`	specifies the family of wavelets “DaubExPhase" or “DaubLeAsymm" (argument of WaveThresh)
`ll`	specifies the lowest level to be thresholded

Details

In wavelet context, test dataset of cross-validation is missing values. Based on h-likelihood concept and penalized least squares, this function performs imputation by wavelet for missing dataset, given the missing dataset. Lee and Nelder (1996, 2001) introduced the hierarchical likelihood as an extended likelihood for general models that include unobserved random variables such as missing. Following Lee and Nelder (1996, 2001), treat the missing values as random parameters and it has been known that a wavelet shrinkage estimator can be formulated by penalized least squares problem (Antoniadis and Fan, 2001). This arguments lead to the iterative algorithm for imputing the missing values based on wavelet shrinkage.

Value

Imputed values according to cross-validation scheme.

References

Antoniadis, A. and Fan, J. (2001) Regularization of wavelet approximations. *Journal of the American Statistical Association*, **96**, 939–962.

Lee, Y. and Nelder, J.A. (1996) Hierarchical generalised linear models (with discussion). *Journal of the Royal Statistical Society Ser. B*, **58**, 619–678.

Lee, Y. and Nelder, J.A. (2001) Hierarchical generalised linear models: A synthesis of generalised linear models, random-effect models and structured dispersions. *Biometrika*, **88**, 987–1006.

See Also

`cvwavelet`, `cvtype`, `cvwavelet.after.impute`.

Examples

```
# 8-fold cross-validation scheme with block size 2
set.seed(1)
cv.index <- cvtype(n=1024, cv.bsize=2, cv.kfold=8, cv.random=TRUE)$cv.index

# Generate 1024 observation from Heavisine function
snr <- 5
testdata <- heav(1024)
x <- testdata$x
y <- testdata$meanf + rnorm(1024, 0, testdata$sdf / snr)

# Impute by wavelet
yimpute <- cvimpute.by.wavelet(y=y, impute.index=cv.index)$yimpute

# Compare imputed values and observations
par(mar=0.1+c(4,4,2,1))
plot(y, yimpute, xlab="Observations", ylab="Imputed Values",
     main="Piecewise Polynomial", cex=0.5);abline(0,1)
```

`cvimpute.image.by.wavelet` *Imputation for two-dimensional data by wavelet*

Description

This function performs imputation for two-dimensional test dataset of cross-validation given test dataset index and initial values.

Usage

```
cvimpute.image.by.wavelet(images, impute.index1, impute.index2,
   impute.tol=0.1^3, impute.maxiter=100, filter.number=2, ll=3)
```

Arguments

images	noisy image
impute.index1	test dataset row index according to cross-validation scheme
impute.index2	test dataset column index according to cross-validation scheme
impute.tol	tolerance for imputation
impute.maxiter	maximum iteration for imputation
filter.number	specifies the smoothness of wavelet in the decomposition (argument of WaveThresh)
ll	specifies the lowest level to be thresholded

Details

In wavelet context, test dataset of cross-validation is missing values. Based on h-likelihood concept and penalized least squares, this function performs imputation by wavelet for missing dataset, given the missing dataset. Lee and Nelder (1996, 2001) introduced the hierarchical likelihood as an extended likelihood for general models that include unobserved random variables such as missing. Following Lee and Nelder (1996, 2001), treat the missing values as random parameters and it has been known that a wavelet shrinkage estimator can be formulated by penalized least squares problem (Antoniadis and Fan, 2001). This arguments lead to the iterative algorithm for imputing the missing values based on wavelet shrinkage.

Value

Imputed values according to cross-validation scheme.

References

Antoniadis, A. and Fan, J. (2001) Regularization of wavelet approximations. *Journal of the American Statistical Association*, **96**, 939–962.

Lee, Y. and Nelder, J.A. (1996) Hierarchical generalised linear models (with discussion). *Journal of the Royal Statistical Society Ser. B*, **58**, 619–678.

Lee, Y. and Nelder, J.A. (2001) Hierarchical generalised linear models: A synthesis of generalised linear models, random-effect models and structured dispersions. *Biometrika*, **88**, 987–1006.

See Also

cvtype.image, cvwavelet, cvimpute.by.wavelet,

cvwavelet.after.impute, cvwavelet.image,

cvwavelet.image.after.impute

CVThresh — *Level-Dependent Cross-Validation Approach for Wavelet Thresholding*

Description

This package carries out level-dependent cross-validation method for the selection of thresholding value in wavelet shrinkage. This procedure is implemented by coupling a conventional cross validation with an imputation method due to a limitation of data length, a power of 2. It can be easily applied to classical leave-one-out and k-fold cross validation. Since the procedure is computationally fast, a level-dependent cross validation can be performed for wavelet shrinkage of various data such as a data with correlated errors.

cvtype — *Generating test dataset index for cross-validation*

Description

This function generates test dataset index for cross-validation.

Usage

```
cvtype(n, cv.bsize=1, cv.kfold, cv.random=TRUE)
```

Arguments

n	the number of observation
cv.bsize	block size of cross-validation
cv.kfold	the number of fold of cross-validation
cv.random	whether or not random cross-validation scheme should be used. Set cv.random=TRUE for random cross-validation scheme

Details

This function provides index of test dataset according to various cross-validation scheme. One may construct K test datasets in a way that each testset consists of blocks of b consecutive data. Set `cv.bsize` = b for this. To select each fold at random, set `cv.random` = `TRUE`.

Value

matrix of which row is test dataset index for cross-validation.

See Also

cvwavelet, cvimpute.by.wavelet, cvwavelet.after.impute.

Examples

```
# Traditional 4-fold cross-validation for 100 observations
cvtype(n=100, cv.bsize=1, cv.kfold=4, cv.random=FALSE)
# Random 4-fold cross-validation with block size 2 for 100 observations
cvtype(n=100, cv.bsize=2, cv.kfold=4, cv.random=TRUE)
```

`cvtype.image` *Generating test dataset index of two-dimensional data for cross-validation*

Description

This function generates test dataset index of two-dimensional data for cross-validation

Usage

```
cvtype.image(n, cv.bsize=c(1,1), cv.kfold)
```

Arguments

`n`	the size of image
`cv.bsize`	two-dimensional block size of cross-validation
`cv.kfold`	the number of fold of cross-validation

Details

This function provides indexes of two-dimensional cross-validation scheme. Only random cross-validation scheme is provided.

Value

Two matrix representing test dataset index of each dimension for cross-validation.

`cv.index1`	each row is test dataset index of one dimension
`cv.index2`	each row is test dataset index of the other dimension

See Also

`cvtype`, `cvwavelet`, `cvimpute.by.wavelet`,

`cvwavelet.after.impute`, `cvwavelet.image`,

`cvimpute.image.by.wavelet`, `cvwavelet.image.after.impute`

Examples

```
# Two-dimensional 4-fold cross-validation with block size 2*2
out <- cvtype.image(n=c(256,256), cv.bsize=c(2,2), cv.kfold=4)
```

`cvwavelet` *Wavelet reconstruction by level-dependent Cross-Validation*

Description

This function reconstructs the noise data by level-dependent cross-validation wavelet shrinkage.

Usage

```
cvwavelet(y=y, ywd=ywd, cv.optlevel, cv.bsize=1, cv.kfold,
    cv.random=TRUE, cv.tol=0.1^3, cv.maxiter=100,
    impute.vscale="independent", impute.tol=0.1^3, impute.maxiter=100,
    filter.number=10, family="DaubLeAsymm", thresh.type ="soft", ll=3)
```

Arguments

`y`	observation
`ywd`	DWT object
`cv.optlevel`	thresholding levels
`cv.bsize`	block size of cross-validation
`cv.kfold`	the number of fold of cross-validation
`cv.random`	whether or not random cross-validation scheme should be used. Set cv.random=TRUE for random cross-validation scheme
`cv.tol`	tolerance for cross-validation
`cv.maxiter`	maximum iteration for cross-validation
`impute.vscale`	specifies whether variance is adjusted level-by-level or not. “level" or “independent"
`impute.tol`	tolerance for imputation
`impute.maxiter`	maximum iteration for imputation
`filter.number`	specifies the smoothness of wavelet in the decomposition (argument of WaveThresh)
`family`	specifies the family of wavelets “DaubExPhase" or “DaubLeAsymm" (argument of WaveThresh)
`thresh.type`	specifies the type of thresholding “hard" or “soft" (argument of WaveThresh)
`ll`	specifies the lowest level to be thresholded

Details

This function performs level-dependent cross-validation wavelet shrinkage.

Value

`y`	observations
`yimpute`	imputed values by provided cross-validation scheme
`yc`	reconstruction by level-dependent cross-validation wavelet shrinkage
`cvthresh`	threshold values by level-dependent cross-validation

See Also

cvtype, cvimpute.by.wavelet, cvwavelet.after.impute.

Examples

```
data(ipd)
y <- as.numeric(ipd); n <- length(y); nlevel <- log2(n)
ywd <- wd(y)
#out <- cvwavelet(y=y, ywd=ywd, cv.optlevel=c(3:(nlevel-1)),
#                     cv.bsize=2, cv.kfold=4)

#ts.plot(ts(out$yc, start=1229.98, deltat=0.02, frequency=50),
#   main="Level-dependent Cross Validation", xlab = "Seconds", ylab="")
```

---

cvwavelet.after.impute

Cross-Validation Wavelet Shrinkage after imputation

Description

This function performs level-dependent cross-validation wavelet shrinkage given the cross-validation scheme and imputation values.

Usage

```
cvwavelet.after.impute(y, ywd, yimpute,
    cv.index, cv.optlevel, cv.tol=0.1^3, cv.maxiter=100,
    filter.number=10, family="DaubLeAsymm", thresh.type="soft", ll=3)
```

Arguments

y	observation
ywd	DWT object
yimpute	imputed values according to cross-validation scheme
cv.index	test dataset index according to cross-validation scheme
cv.optlevel	thresholding levels
cv.tol	tolerance for cross-validation
cv.maxiter	maximum iteration for cross-validation
filter.number	specifies the smoothness of wavelet in the decomposition (argument of WaveThresh)
family	specifies the family of wavelets “DaubExPhase" or “DaubLeAsymm" (argument of WaveThresh)
thresh.type	specifies the type of thresholding “hard" or “soft" (argument of WaveThresh)
ll	specifies the lowest level to be thresholded

Details

Calculating the threshold values and reconstructing noisy data y, given the index of each testdata, imputed values according to cross-validation scheme and discrete wavelet transform of y.

Value

Reconstruction and thresholding values by level-dependent cross-validation

yc	reconstruction
cvthresh	thresholding values by level-dependent cross-validation

See Also

cvwavelet, cvtype, cvimpute.by.wavelet.

Examples

```
data(ipd)
y <- as.numeric(ipd); n <- length(y); nlevel <- log2(n)

set.seed(1)
cv.index <- cvtype(n=n, cv.bsize=2, cv.kfold=4, cv.random=TRUE)$cv.index
yimpute <- cvimpute.by.wavelet(y=y, impute.index=cv.index)$yimpute

ywd <- wd(y)

#out <- cvwavelet.after.impute(y=y, ywd=ywd, yimpute=yimpute,
#cv.index=cv.index, cv.optlevel=c(3:(nlevel-1)))

#ts.plot(ts(out$yc, start=1229.98, deltat=0.02, frequency=50),
#   main="Level-dependent Cross Validation", xlab = "Seconds", ylab="")

##### Specifying thresholding structure
# cv.optlevel <- c(3) # Threshold (level 3 to finest level) at the same time.
# cv.optlevel <- c(3, 5) # Threshold two groups of resolution levels,
                         # (level 3, 4) and  (level 5 to finest level).
# cv.optlevel <- c(3,4,5,6,7,8) # Threshold each resolution level 3 to 8.
```

---

cvwavelet.image *Wavelet reconstruction of image by level-dependent Cross-Validation*

Description

This function reconstructs image by level-dependent cross-validation wavelet shrinkage.

Usage

```
cvwavelet.image(images, imagewd,
    cv.optlevel, cv.bsize=c(1,1), cv.kfold, cv.tol=0.1^3, cv.maxiter=100,
    impute.tol=0.1^3, impute.maxiter=100, filter.number=2, ll=3)
```

Arguments

images	noisy image
imagewd	two-dimensional wavelet transform
cv.optlevel	thresholding level
cv.bsize	block size of cross-validation
cv.kfold	the number of fold of cross-validation
cv.tol	tolerance for cross-validation
cv.maxiter	maximum iteration for cross-validation
impute.tol	tolerance for imputation
impute.maxiter	maximum iteration for imputation
filter.number	specifies the smoothness of wavelet in the decomposition (argument of WaveThresh)
ll	specifies the lowest level to be thresholded

Details

This function performs level-dependent cross-validation wavelet shrinkage for two-dimensional data.

Value

imagecv	reconstruction of image by level-dependent cross-validation wavelet shrinkage
cvthresh	threshold values by level-dependent cross-validation

See Also

cvtype.image, cvimpute.image.by.wavelet, cvwavelet.image.after.impute.

Examples

```
# Generate Noisy Lennon Image
data(lennon)
sdimage <- sd(as.numeric(lennon))
nlennon <- ncol(lennon); nlevel <- log2(ncol(lennon))
optlevel <- c(3:(nlevel-1))
set.seed(55)
lennonnoise <- lennon+matrix(rnorm(nlennon^2, 0, sdimage), nlennon, nlennon)

# Level-dependent Cross-validation Thresholding
lennonwd <- imwd(lennonnoise)
#lennoncv <- cvwavelet.image(images=lennonnoise, imagewd=lennonwd,
#      cv.optlevel=optlevel, cv.bsize=c(1,1), cv.kfold=10)$imagecv
#image(lennoncv, axes=FALSE, col=gray(100:0/100),
#   main="Level-dependent CV")
```

`cvwavelet.image.after.impute` *Cross-Validation Wavelet Shrinkage for two-dimensional data after imputation*

Description

This function performs level-dependent cross-validation wavelet shrinkage for two-dimensional data given the cross-validation scheme and imputation values.

Usage

```
cvwavelet.image.after.impute(images, imagewd, imageimpute,
   cv.index1=cv.index1, cv.index2=cv.index2,
   cv.optlevel=cv.optlevel, cv.tol=cv.tol, cv.maxiter=cv.maxiter,
   filter.number=2, ll=3)
```

Arguments

`images`	noisy image
`imagewd`	two-dimensional wavelet transform
`imageimpute`	two-dimensional imputed values according to cross-validation scheme
`cv.index1`	test dataset row index according to cross-validation scheme
`cv.index2`	test dataset column index according to cross-validation scheme
`cv.optlevel`	thresholding levels
`cv.tol`	tolerance for cross-validation
`cv.maxiter`	maximum iteration for cross-validation
`filter.number`	specifies the smoothness of wavelet in the decomposition (argument of WaveThresh)
`ll`	specifies the lowest level to be thresholded

Details

Calculating thresholding values and reconstructing noisy image given cross-validation scheme and imputation.

Value

Reconstruction of images and thresholding values by level-dependent cross-validation

`imagecv`	reconstruction of images
`cvthresh`	thresholding values by level-dependent cross-validation

See Also

`cvwavelet.image`, `cvtype.image`, `cvimpute.image.by.wavelet`.

dopp *Doppler function*

Description

This function generates Doppler function values for n equally spaced points in $[0, 1]$.

Usage

```
dopp(norx=1024)
```

Arguments

norx	the number of data or x values in [0, 1]

Details

Doppler function is introduced by Donoho and Johnstone (1994) and is useful test function evaluating a wavelet shrinkage method.

Value

Doppler function values $f(\frac{i}{n}), i = 1, \ldots, n$ and its variability $||f|| = \frac{\sum_{i=1}^{n}(f_i - \bar{f})^2}{n-1}$ where $\bar{f} = \frac{\sum_{i=1}^{n} f_i}{n}$.

References

Donoho, D.L. and Johnstone, I.M. (1994) Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, **81**, 425–455.

See Also

heav, ppoly, fg1.

Examples

```
testdopp <- dopp(1024)
plot(testdopp$x, testdopp$meanf, xlab="", ylab="",
     main="Plot of Doppler function", type="l")
```

`fg1` *fg1 function*

Description

This function generates fg1 function values for n equally spaced points in $[0, 1]$.

Usage

```
fg1(norx=1024)
```

Arguments

`norx`	the number of data or x values in [0, 1]

Details

A smooth function fg1 is introduced by Fan and Gijbels (1995) and is useful test function evaluating a wavelet shrinkage method.

Value

fg1 function values $f(\frac{i}{n}), i = 1, \ldots, n$ and its variability $||f|| = \frac{\sum_{i=1}^{n}(f_i - \bar{f})^2}{n-1}$ where $\bar{f} = \frac{\sum_{i=1}^{n} f_i}{n}$.

References

Fan, J. and Gijbels, I. (1995) Data-driven bandwidth selection in local polynomial fitting: Variable bandwidth and spatial adaptation. *Journal of the Royal Statistical Society Ser. B* **57**, 371–394.

See Also

`dopp`, `heav`, `ppoly`.

Examples

```
testfg1 <- fg1(1024)
plot(testfg1$x, testfg1$meanf, xlab="", ylab="",
     main="Plot of fg1 function", type="l")
```

heav *Heavisine function*

Description

This function generates Heavisine function values for n equally spaced points in $[0, 1]$.

Usage

```
heav(norx=1024)
```

Arguments

norx	the number of data or x values in [0, 1]

Details

Heavisine function is introduced by Donoho and Johnstone (1994) and is useful test function evaluating a wavelet shrinkage method.

Value

Heavisine function values $f(\frac{i}{n}), i = 1, \ldots, n$ and its variability $||f|| = \frac{\sum_{i=1}^{n}(f_i - \bar{f})^2}{n-1}$ where $\bar{f} = \frac{\sum_{i=1}^{n} f_i}{n}$.

References

Donoho, D.L. and Johnstone, I.M. (1994) Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, **81**, 425–455.

See Also

dopp, ppoly, fg1.

Examples

```
testheav <- heav(1024)
plot(testheav$x, testheav$meanf, xlab="", ylab="",
     main="Plot of Heavisine function", type="l")
```

ipd *Inductance plethysmography data*

Description

4,096 observations of inductance plethysmography data regularly sampled at 50Hz starting at 1229.98 seconds.

Usage

```
data(ipd)
```

Format

time series.

Source

This data set contains 4,096 observations of inductance plethysmography data regularly sampled at 50Hz starting at 1229.98 seconds. The data were collected in an investigation of the recovery of patients after general anesthesia.

The data set was used in Nason (1996) to illustrate cross-validation method for threshold selection. See the reference; Nason, G.P. (1996) Wavelet shrinkage by cross-validation. *Journal of the Royal Statistical Society Ser. B* **58**, 463–479.

ppoly *Piecewise polynomial function*

Description

This function generates Piecewise polynomial function values for n equally spaced points in $[0, 1]$.

Usage

```
ppoly(norx=1024)
```

Arguments

norx	the number of data or x values in [0, 1]

Details

Piecewise polynomial function with the discontinuity is introduced by Nason and Silverman (1994) and is useful test function evaluating a wavelet shrinkage method.

Value

Piecewise polynomial function values $f(\frac{i}{n}), i = 1, \ldots, n$ and its variability $||f|| = \frac{\sum_{i=1}^{n}(f_i - \bar{f})^2}{n-1}$ where $\bar{f} = \frac{\sum_{i=1}^{n} f_i}{n}$.

References

Nason, G.P. and Silverman, B.W. (1994) The discrete wavelet transform in S. *Journal of Computational and Graphical Statistics*, **3**, 163–191.

See Also

dopp, heav, fg1.

Examples

```
testpoly <- ppoly(1024)
plot(testpoly$x, testpoly$meanf, xlab="", ylab="",
     main="Plot of Piecewise polynomial function", type="l")
```

Index