Package ‘rSFA’

October 14, 2022

Maintainer Martin Zaefferer <martin.zaefferer@gmx.de>
License GPL (>= 2)
Title Slow Feature Analysis
Type Package
Author Wolfgang Konen <wolfgang.konen@fh-koeln.de>, Martin Zaefferer,
Patrick Koch; Bug hunting and testing by Ayodele Fasika, Ashwin
Kumar, Prawyn Jebakumar
Description Slow Feature Analysis (SFA), ported to R based on
‘matlab’ implementations of SFA: ‘SFA toolkit’ 1.0 by Pietro Berkes and ‘SFA toolkit’
2.8 by Wolfgang Konen.
Version 1.5
Date 2022-03-29
Depends R (>= 2.0.0)
Imports stats, MASS, graphics, grDevices
NeedsCompilation no
Repository CRAN
RoxygenNote 7.1.2
Date/Publication 2022-03-29 10:00:07 UTC

R topics documented:

rSFA-package ...................................................... 2
addNoisyCopies ............................................. 2
etaval ............................................................. 3
gaussClassifier ............................................... 4
gaussCreate .................................................. 5
sfa1 .............................................................. 5
sfa1Create .................................................... 6
sfa2 .............................................................. 6
sfa2Create .................................................... 8
sfaClassify .................................................... 9
addNoisyCopies

Description

Add noisy copies for parametric bootstrap

Description

Given training data X with true labels REALCLASS, add new records to X and REALCLASS, which are noisy copies of the training data.
etaval

Usage
addNoisyCopies(realclass, x, pars)

Arguments
realclass       true class of training data (can be vector, numerics, integers, factors)
x              a matrix containing the training data
pars            list of parameters:
pars$ncopies: Number of new records to add
pars$ncsort: Defines if training data should be sorted by class. Default is FALSE
pars$ncsigma: The noise in each column of x has the std.dev. pars$ncsigma*(standard
deviation of column). Default Value: 0.8
pars$ncmethod: =1: each 'old' record from X in turn is the centroid for a new
                pattern;
                =2: the centroid is the average of all records from the same class, the std.dev. is
                    the same for all classes;
                =3: centroid as in '2', the std.dev. is the std.dev. of all records from the same
                    class (*recommended*)

Value
- res contains two list entries: realclass and x (including added copies)

References
sfaPBootstrap

etaval
Computes the eta value of a signal (slowness)

Description
Computes the eta value of a signal (slowness)

Usage
etaval(x, T = length(x))

Arguments
x             The columns of signal correspond to different input components. Must be nor-
              malized (zero mean, unit variance)
T             Time interval
gaussClassifier

Returns the eta value of the signal in a time interval T time units long.

---

**gaussClassifier**  
*Classifier for SFA demos*

---

**Description**

Train or apply a Gaussian classifier.

**Usage**

```r
gaussClassifier(gauss, y, realC, method = "train")
```

**Arguments**

- `gauss`: List created by gaussCreate. Contains also the elements:
  - `aligned` = 0: do not align the Gaussian classifiers with axes, use full covariance matrix
  - = 1 (default): set the off-diagonals in covariance matrix to 0, i.e. the Gaussian classifier is forced to be aligned with the axes. This is more robust in the case where the data deviate largely from a multivariate normal distribution.
  - `epsD` [defaults to 0.04] replace diagonal elements of COV smaller than epsD with epsD to avoid too small Gaussians
- `y`: K x M matrix where K is the total number of patterns and M is the number of variables used for classification. I.e. each row of y contains the data for one pattern.
- `realC`: 1 x K matrix with NCLASS distinct real class labels needed only for method='train'. In case of method="apply" realC is not used and can have any value
- `method`: either "train" (default) or "apply"

**Value**

List gauss containing

- `predC`: 1 x K matrix: the predicted class
- `prob`: K x NCLASS matrix: prob(k,n) is the estimated probability that pattern k belongs to class m

**See Also**

`gaussCreate`
**gaussCreate**

Create an Gaussian classifier object

**Usage**

```r
gaussCreate(nclass, dimY)
```

**Arguments**

- `nclass` number of classes
- `dimY` dimension

**Value**

list of defaults for gauss classifier

**See Also**

- `gaussClassifier`

---

**sfa1**

The SFA1 algorithm, linear SFA.

**Description**

Y = sfa1(X) performs linear Slow Feature Analysis on the input data X and returns the output signals Y ordered by increasing temporal variation, i.e. the first signal Y[.1] is the slowest varying one, Y[.2] the next slowest and so on. The input data have to be organized with each variable in a column and each data (time) point in a row, i.e. X(t,i) is the value of variable nr. i at time t.

**Usage**

```r
sfa1(x)
```

**Arguments**

- `x` Input data, each column a different variable

**Value**

list sfaList with all learned information, where sfaList$y contains the outputs
See Also

  sfaStep sfa1Create sfaExecute

---

**sfa1Create**

Create structured list for linear SFA

---

**Description**

Create structured list for linear SFA

**Usage**

```r
sfa1Create(sfaRange, axType = "ORD1", regCt = 0)
```

**Arguments**

- `sfaRange`: number of slowly-varying functions to be kept
- `axType`: is the type of derivative approximation to be used, see `sfaTimediff`
- `regCt`: regularization constant, currently not used

**Value**

- `list sfaList` contains all arguments passed into `sfa1create` plus
- `deg` 2

This list will be expanded by other SFA functions with further SFA results

**See Also**

  sfa sfaStep sfa2Create

---

**sfa2**

The SFA2 algorithm, SFA with degree 2 expansion.

---

**Description**

`Y = sfa2(X)` performs expanded Slow Feature Analysis on the input data `X` and returns the output signals `Y` ordered by increasing temporal variation, i.e. the first signal `Y[,1]` is the slowest varying one, `Y[,2]` the next slowest varying one and so on. The input data have to be organized with each variable in a column and each data (time) point in a row, i.e. `X(t,i)` is the value of variable `i` at time `t`. By default an expansion to the space of 2nd degree polynomials is done, this can be changed by using different functions for `xpDimFun` and `sfaExpandFun`.


sfa2

Usage

sfa2(
  x,
  method = "SVDSFA",
  ppType = "PCA",
  xpDimFun = xpDim,
  sfaExpandFun = sfaExpand
)

Arguments

x input data
method eigenvector calculation method: ="SVDSFA" for singular value decomposition (recommended) or ="GENEIG" for generalized eigenvalues (unstable!). GENEIG is not implemented in the current version, since R lacks an easy option to calculate generalized eigenvalues.
ppType preprocessing type: ="PCA" (principal component analysis) or ="SFA1" (linear sfa)
xpDimFun function to calculate dimension of expanded data
sfaExpandFun function to expand data

Value

list sfaList with all SFA information, among them are

y a matrix containing the output Y (as described above)
- all input parameters to sfa2Create
- all elements of sfaList as specified in sfa2Step

See Also

sfa2Step sfa2Create sfaExecute sfa1

Examples

## prepare input data for simple demo
t=seq.int(from=0,by=0.011,to=2*pi)
x1=sin(t)+cos(11*t)^2
x2=cos(11*t)
x=data.frame(x1,x2)
## perform sfa2 algorithm with data
res = sfa2(x)
## plot slowest varying function of result
plot(t, res$y[,1],type="l",main="output of the slowest varying function")
## see http://www.scholarpedia.org/article/Slow_feature_analysis#The_algorithm
## for detailed description of this example
sfa2Create

Create structured list for expanded SFA

Description

'Expanded' SFA means that the input data are expanded into a higher-dimensional space with the function sfaExpandFun. See sfaExpand for the default expansion function.

Usage

sfa2Create(
  ppRange,
  sfaRange,
  ppType = "SFA1",
  axType = "ORD1",
  regCt = 0,
  opts = NULL,
  xpDimFun = xpDim,
  sfaExpandFun = sfaExpand
)

Arguments

ppRange  number of dimensions to be kept after preprocessing step - or - a two-number vector with lower and upper dimension number
sfaRange  number of slowly-varying functions to be kept
ppType  preprocessing type:="PCA", "PCA2" (principal component analysis) or="SFA1" (linear sfa)
axType  is the type of derivative approximation to be used, see sfaTimediff
regCt  regularization constant, currently not used
opts  optional list of additional options
xpDimFun  Function to calculate dimension of expanded data
sfaExpandFun  Function to expand data

Value

list sfaList contains all arguments passed into sfa2create plus
xpRange evaluates to xpDimFun(ppRange)
deg 2

This list will be expanded by other SFA functions with further SFA results

See Also

sfa2 sfaStep sfa1Create
sfaClassify

Predict Class for SFA classification

Description

Create a SFA classification mode, predict & evaluate on new data (xtst,realc_tst).
Author of orig. matlab version: Wolfgang Konen, May 2009 - Jan 2010

Usage

sfaClassify(x, realclass, xtst = 0, realcTst = 0, opts)

Arguments

x NREC x IDIM, training input data
realclass 1 x NREC, training class labels
xtst NTST x IDIM, test input data
realcTst 1 x NTST, test class labels
opts list with several parameter settings:

  gaussdim
  ...
*Filename [* = s,g,x] from where to load the models (see sfaClassify)

Value

list res containing

  res$errtrn 1 x 2 matrix: error rate with / w/o SFA on training set
  res$errtst 1 x 2 matrix: error rate with / w/o SFA on test set
  res$y output from SFA when applied to training data
  res$ytst output from SFA when applied to test data
  res$predT predictions with SFA + GaussClassifier on test set
  res$predX predictions w/o SFA (only GaussClassifier) on test set (only if opts.xFilename exists)

See Also

sfaClassPredict sfaExecute
sfaClassPredict  
*Predict Class for SFA classification*

**Description**

Use a SFA classification model (stored in opts$*Filename), predict & evaluate on new data (xtst,realc_tst).


**Usage**

\[ \text{sfaClassPredict(xtst, realcTst, opts)} \]

**Arguments**

- **xtst**: NTST x IDIM, test input data
- **realcTst**: 1 x NTST, test class labels
- **opts**: list with several parameter settings:
  - **gaussdim**
  - ...
  - *Filename* [* = s,g,x] from where to load the models (see sfaClassify)

**Value**

list res containing

- **res$errtst**: 1 x 2 matrix: error rate with / w/o SFA on test set
- **res$ytst**: output from SFA when applied to test data
- **res$predT**: predictions with SFA + GaussClassifier on test set
- **res$predX**: predictions w/o SFA (only GaussClassifier) on test set (only if opts.xFilename exists)

**See Also**

sfaClassify sfaExecute
sfaExecute

*Execute learned function for input data*

**Description**

After completion of the learning phase (step="sfa") this function can be used to apply the learned function to the input data.

The execution is completed in 4 steps:
1. projection on the input principal components (dimensionality reduction)
2. expansion (if necessary)
3. projection on the whitened (expanded) space
4. projection on the slow functions

**Usage**

`sfaExecute(sfaList, DATA, prj = NULL, ncomp = NULL)`

**Arguments**

- `sfaList` A list that contains all information about the handled sfa-structure
- `DATA` Input data, each column a different variable
- `prj` If not NULL, the preprocessing step 1 is skipped for SFA2
- `ncomp` number of learned functions to be used

**Value**

matrix DATA containing the calculated output

**See Also**

`sfa2 sfa1 sfaStep`

sfaExpand

*Degree 2 Expansion*

**Description**

Expand a signal in the space of polynomials of degree 2. This is the default expansion function used by rSFA.

**Usage**

`sfaExpand(sfaList, DATA)`
sfaNlRegress

Perform non-linear regression

Description

Given the data in arg, expand them nonlinearly in the same way as it was done in the SFA-object sfaList (expanded dimension M) and search the vector RCOEF of M constant coefficients, such that the sum of squared residuals between a given function in time FUNC and the function

\[ R(t) = (v(t) - v_0)' \times RCOEF, \quad t=1,...,T, \]

is minimal.

Usage

sfaNlRegress(sfaList, arg, func)

Arguments

sfaList    A list that contains all information about the handled sfa-structure
arg        Input data, each column a different variable
func       (T x 1) the function to be fitted nonlinearly

Value

returns a list res with elements

res$R       (T x 1) the function fitted by NL-regression
res$rcoef   (M x 1) the coefficients for the NL-expanded dimensions
sfaPBootstrap  

Parametric Bootstrap

Description
If training set too small, augment it with parametric bootstrap

Usage
sfaPBootstrap(realclass, x, sfaList)

Arguments
- realclass: true class of training data (can be vector, numerics, integers, factors)
- x: matrix containing the training data
- sfaList: list with several parameter settings, e.g. as created by sfa2Create
  - sfaList$xpDimFun (default): calculated dimension of expanded SFA space
  - sfaList$deg: degree of expansion (should not be 1, not implemented)
  - sfaList$ppRange: ppRange for SFA algorithm
  - sfaList$nclass: number of unique classes
  - sfaList$doPB: do (1) or do no (0) param. bootstrap.

Value
a list containing:
- x: training set extended to minimum number of records 1.5*(xpdim+nclass), if necessary
- realclass: training class labels, extended analogously

See Also
- addNoisyCopies

sfaStep  

Update a step of the SFA algorithm.

Description
sfaStep() updates the current step of the SFA algorithm. Depending on sfaList$deg it calls either sfa1Step or sfa2Step to do the main work. See further documentation there

Usage
sfaStep(sfaList, arg, step = NULL, method = NULL)
Arguments

sfaList A list that contains all information about the handled sfa-structure
arg Input data, each column a different variable
step Specifies the current SFA step. Must be given in the right sequence: for SFA1 objects: "preprocessing", "sfa"
for SFA2 objects: "preprocessing", "expansion", "sfa" Each time a new step is invoked, the previous one is closed, which might take some time.
method Method to be used: For sfaList$step="expansion" the choices are "TIME-SERIES" or "CLASSIF".
For sfaList$step="sfa" (sfa2Step only) the choices are "SVDSFA" (recommended) or "GENEIG" (unstable).

Value

list sfaList taken from the input, with new information added to this list. See sfa1Step or sfa2Step for details.

See Also

sfa1Step sfa2Step sfa1Create sfa2Create sfaExecute

Examples

## Suppose you have divided your training data into two chunks, ## DATA1 and DATA2. Let the number of input dimensions be N. To apply ## SFA on them write:
## Not run:
sfaList = sfa2Create(N,xpDim(N))
sfaList = sfaStep(sfaList, DATA1, "preprocessing")
sfaList = sfaStep(sfaList, DATA2)
sfaList = sfaStep(sfaList, DATA1, "expansion")
sfaList = sfaStep(sfaList, DATA2)
sfaList = sfaStep(sfaList, NULL, "sfa")
output1 = sfaExecute(sfaList, DATA1)
output2 = sfaExecute(sfaList, DATA2)
## End(Not run)

sfaTimediff Calculates the first derivative of signal data

Description

Calculates the first derivative of signal data
Usage

sfaTimediff(DATA, axType = "ORD1")

Arguments

DATA The matrix of signals for which the derivative is calculated (one column per signal)
axType Type of interpolation: "ORD1" (default) first order, "SCD" second, "TRD" third, "ORD3a" cubic polynomial

Value

matrix DATA
- DATA contains the derivative signals, with the same structure as the input data.

Note

setting axType to invalid values will lead to first order interpolation.

xpDim

Degree 2 Dimension Calculation

Description

Compute the dimension of a vector expanded in the space of polynomials of 2nd degree.

Usage

xpDim(n)

Arguments

n Dimension of input vector

Value

Dimension of expanded vector

See Also

sfa2 sfaExpand
# Index

* **analysis**  
  rSFA-package, 2

* **classification**  
  rSFA-package, 2

* **feature**  
  rSFA-package, 2

* **slow**  
  rSFA-package, 2

* **timeseries**  
  rSFA-package, 2

addNoisyCopies, 2, 13

etaval, 3

gaussClassifier, 4, 5  
gaussCreate, 4, 5

nlExpand, 12

rSFA (rSFA-package), 2  
rSFA-package, 2

sfa1, 5, 6, 7, 11  
sfa1Create, 6, 6, 8, 14  
sfa1Step, 13, 14

sfa2, 6, 8, 11, 12, 15  
sfa2Create, 6, 7, 8, 13, 14  
sfa2Step, 7, 13, 14

sfaClassify, 9, 9, 10  
sfaClassPredict, 9, 10  
sfaExecute, 6, 7, 9, 10, 11, 14  
sfaExpand, 8, 11, 15  
sfaNlRegress, 12  
sfaPBootstrap, 3, 13  
sfaStep, 6, 8, 11, 13  
sfaTimediff, 6, 8, 14

xpDim, 12, 15