Package ‘fsMTS’

April 6, 2020

Type Package

Title Feature Selection for Multivariate Time Series

Version 0.1.5

Description Implements feature selection routines for multivariate time series (MTS).
The list of implemented algorithms includes:
- own lags (independent MTS components),
- cross-correlation (see Schelter et al. (2006, ISBN:9783527406234)),
- graphical LASSO (see Hawthorn and Cheng (2014) <https://www.gla.ac.uk/media/Media_401739_smxx.pdf>),
- random forest (see Pavlyuk (2020) "Random Forest Variable Selection for Sparse Vector Autoregressive Models" in Contributions to Statistics, in production),
- mutual information (see Schelter et al. (2006, ISBN:9783527406234), Liu et al. (2016) <doi:10.1109/ChiCC.2016.7554480>),

In addition, the package implements functions for ensemble feature selection (using feature ranking and majority voting).

The package is implemented within Dmitry Pavlyuk's research project No. 1.1.1.2/VIAA/1/16/112 "Spatiotemporal urban traffic modelling using big data".

License GPL-3

Depends R (>= 3.6)

Imports glasso,lars,mpmi,freqdom,randomForestSRC

Suggests knitr, rmarkdown, sparsevar, plot.matrix, svMisc, MTS

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Author Dmitry Pavlyuk [aut, cre] (<https://orcid.org/0000-0003-3710-9678>)

Maintainer Dmitry Pavlyuk <Dmitry.Pavlyuk@tsi.lv>
Repository CRAN
Repository/R-Forge/Project fsmts
Repository/R-Forge/Revision 17
Repository/R-Forge/DateTimeStamp 2020-04-06 12:28:39
Date/Publication 2020-04-06 14:42:13 UTC
NeedsCompilation no

R topics documented:

fsMTS-package .................................................... 2
cutoff .............................................................. 3
fsEnsemble .......................................................... 4
fsMTS ................................................................. 5
fsSimilarity ......................................................... 8
fsSimilarityMatrix ................................................. 10
fsSparsity .......................................................... 11
traffic ............................................................... 12
traffic.mini ......................................................... 12

Index 13

fsMTS-package  Feature selection for Multivariate Time Series

Description

Feature selection for Multivariate Time Series

Details

Implementation of feature selection methods for multivariate time series

Author(s)

Dmitry Pavlyuk <Dmitry.V.Pavlyuk@gmail.com>
cutoff

Choosing most important features

description
cutoff chooses features of highest importance to reach the required percent of sparsity

usage
cutoff(feature.set, threshold)

arguments
feature.set a matrix that contains feature weights.
threshold the required sparsity of the resulting feature set

value
returns a binary feature matrix. Columns correspond to components of the time series; rows correspond to lags.

examples
# Load traffic data
data(traffic.mini)

# Scaling is sometimes useful for feature selection
# Exclude the first column - it contains timestamps
data <- scale(traffic.mini$data[, -1])

mCCF <- fsMTS(data, max.lag = 3, method = "CCF")
cutoff(mCCF, 0.3)
cutoff(mCCF, 0.1)

mIndependent <- fsMTS(data, max.lag = 3, method = "ownlags")
cutoff(mIndependent, 0.3)
cutoff(mIndependent, 0.1)
fsEnsemble

Description

fsEnsemble implements methods for ensemble learning of features for multivariate time series

Usage

fsEnsemble(feature.sets, threshold, method = c("ranking", "majority"))

Arguments

feature.sets
  a list of matrixes that contains weights for features, estimated by several feature selection algorithms (base learners)

threshold
  the required sparsity of the resulting feature set

method
  a ensemble learning algorithm. Implemented algorithms:
  • "ranking" - individual feature sets are ranked according to their weights and further the sum of ranks is used for feature selection (threshold share of features is selected). The algorithm uses ranking of feature with a minor priority to earlier lags and even smaller priority to order of MTS components. So, if features of 1st and 2nd lags have identical weights, the feature of the 1st lag will be preferred; if features of the same lag have identical weights, the order of features is used as a priority.
  • "majority" - base feature sets are for feature selection (threshold share of features is selected) and further the resulting feature set is estimated using majority voting (50 or more percent of base learners)

Value

returns a binary feature matrix. Columns correspond to components of the time series; rows correspond to lags.

References


Examples

# Load traffic data
data(traffic.mini)

# Scaling is sometimes useful for feature selection
# Exclude the first column - it contains timestamps
data <- scale(traffic.mini$data[, -1])
fsMTS <- fsMTS(data, max.lag=3, method="ownlags")

mCCF <- fsMTS(data, max.lag=3, method="CCF")

mDistance <- fsMTS(data, max.lag=3, method="distance", shortest = traffic.mini$shortest, step = 5)

mGLASSO <- fsMTS(data, max.lag=3, method="GLASSO", rho = 0.05)

mLARS <- fsMTS(data, max.lag=3, method="LARS")

mRF <- fsMTS(data, max.lag=3, method="RF")

mMI <- fsMTS(data, max.lag=3, method="MI")

mlist <- list(Independent = mIndep,
              Distance = mDistance,
              CCF = mCCF,
              GLASSO = mGLASSO,
              LARS = mLARS,
              RF = mRF,
              MI = mMI)

th <- 0.30

mlist["EnsembleRank"] <- fsEnsemble(mlist, threshold = th, method="ranking")

mlist["EnsembleMajV"] <- fsEnsemble(mlist, threshold = th, method="majority")

(msimilarity <- fsSimilarityMatrix(mlist, threshold = th, method="Kuncheva"))

---

**fsMTS**  
*Feature selection for multivariate time series*

**Description**

fsMTS implements algorithms for feature selection in multivariate time series.

**Usage**

```r
fsMTS(
  mts,
  max.lag,
  method = c("ownlags", "distance", "CCF", "MI", "RF", "GLASSO", "LARS", "PSC"),
  show.progress = F,
  localized = F,
  ...
)
```

**Arguments**

- **mts**  
an matrix object with values of the multivariate time series (MTS) MTS components are by k columns, observations are by rows
- **max.lag**  
the maximal lag value
- **method**  
a feature selection algorithm. Implemented algorithms:
- "ownlags" - only own (autoregressive) lags. The method constructs the matrix of features that represents independent AR(max.lag) processes for every MTS component. "distance" - distance-based feature selection. The method uses directed distances shortest between every pair of time series components (origin and destination). The lag l is selected as a potential relationship (feature) if the destination component is reachable from the origin component within \((l*step)\) time steps, rounded to integer value. All previous and next lags are not included into the resulting structure. "CCF" - cross-correlation-based. The method returns values of Pearson’s correlation coefficient between every MTS component and all other MTS components and their lags. See Yang et al.,(2005) as an example of application. Only own lags of every MTS component are included as selected features.

- "MI" - mutual information-based. The method returns values of mutual information between every component of the multivariate time series and all other components and their lags. The method is localized - mutual information is independently estimated for every MTS component and lags \((1:max.lag)\) of all MTS components. See Liu et al., (2016) as an example of application.

- "RF" - random forest estimation of \(k\) linear regression models. The method returns increase of mean square error of the multivariate time series and all other components and their lags. The method is localized - the linear regression is independently estimated by the random forest algorithm for every MTS component as a dependent variable and lags \((1:max.lag)\) of all MTS components as explanatory variables. See Pavlyuk (2020) for more details.

- "GLASSO" - feature selection using graphical LASSSO regularisation of the inverse covariance matrix. The method returns values from inverse correlation matrix between every MTS component and all other components and their lags. The method is localized - the sparse inverse correlation matrix is independently estimated for every time series component and lags \((1:_max.lag_)\) of all other components.

- "LARS" - feature selection using least angle regression. The method returns values of beta proportions from the least angle regression, estimated for every MTS component and all other components and their lags. The method is localized - the least angle regression is independently estimated for every MTS component and lags \((1:_max.lag_)\) of all other components.

- "PSC" - feature selection using partial spectral coherence of MTS components. The method returns maximal values of the partial spectral coherence function for all MTS lags.

- show.progress the logical parameter to print progress of calculation. By default is FALSE.

- localized the logical parameter to executed localized (component-wise) feature selection if the selected method supports this ("MI", "GLASSO", "RF"). Localized versions of algorithms are based on selection of features for independently for every MTS component from all lagged components. Non-localised versions include simultaneous feature selection for all components, including potential instantaneous effects (relationships between feature within the same lag). Later, non-localised algorithms ignore instantaneous effects and return only lagged features.
By default is TRUE

method-specific parameters:

- "shortest" ("distance" algorithm) matrix of externally provided shortest distances between every pair of time series' components.
- "step" ("distance" algorithm) distance that covered by the process during one time step of the time series. By default is 1.
- "rho" ("GLASSO" algorithm) non-negative regularization parameter for lasso. rho=0 means no regularization.

Details

The function implements selection of potential relationships between multivariate time series' components and their lags.

Value

returns a real-valued or binary (depends on the algorithm) feature matrix of $k*max.lag$ rows and $k$ columns, where $k$ is number of time series components (number of columns in the mts parameter). Columns correspond to components of the time series; rows correspond to lags (from 1 to max.lag).

References

Distance-based feature selection for MTS

Cross-corelation-based feature selection for MTS

Mutual information-based feature selection for MTS

Random forest-based feature selection for MTS

Graphical LASSO-based feature selection for MTS

Least angle regression for feature selection for MTS

Partial spectral coherence for feature selection for MTS

Examples

```r
# Load traffic data
data(traffic.mini)

# Scaling is sometimes useful for feature selection
# Exclude the first column - it contains timestamps
data <- scale(traffic.mini$data[, -1])

mIndep <- fsMTS(data, max.lag = 3, method = "ownlags")
mCCF <- fsMTS(data, max.lag = 3, method = "CCF")
mDistance <- fsMTS(data, max.lag = 3, method = "distance", shortest = traffic.mini$shortest, step = 5)
mGLASSO <- fsMTS(data, max.lag = 3, method = "GLASSO", rho = 0.05)
mLARS <- fsMTS(data, max.lag = 3, method = "LARS")
mRF <- fsMTS(data, max.lag = 3, method = "RF")
mMI <- fsMTS(data, max.lag = 3, method = "MI")

mlist <- list(Independent = mIndep,
               Distance = mDistance,
               CCF = mCCF,
               GLASSO = mGLASSO,
               LARS = mLARS,
               RF = mRF,
               MI = mMI)

th <- 0.30
(msimilarity <- fsSimilarityMatrix(mlist, threshold = th, method = "Kuncheva"))
```

---

**fsSimilarity**

Calculating similarity of two feature sets

**Description**

fsSimilarity implements different methods for calculation similarity of two feature sets.

**Usage**

```r
fsSimilarity(
  feature.set1,
  feature.set2,
  cutoff = FALSE,
  threshold = 1,
  method = c("Kuncheva", "Jaccard", "Hamming")
)
```
Arguments

- `feature.set1` a matrix that contains feature weights.
- `feature.set2` a matrix that contains feature weights.
- `cutoff` logical. If true, the input features sets are cut-off using the `cutoff` function with a specified threshold. By default is FALSE.
- `threshold` the threshold for feature selection using the `cutoff` function. By default is 1 (no cut-off)
- `method` a similarity metric. Implemented metrics:
  - "Jaccard" - a share of matching features to maximal possible number of matching features (Jaccard similarity)
  - "Kuncheva" - Kuncheva-like correction to the expected number of features matched by chance. See Kuncheva (2007)
  - "Hamming" - Hamming distance, normalised to [0,1], where 1 is for identical matrices

Value

returns a value from the [-1, 1] interval for Kuncheva and from the [0,1] interval for other algorithms, where 1 is for absolutely identical feature sets.

References


Examples

```r
# Load traffic data
data(traffic.mini)

# Scaling is sometimes useful for feature selection
# Exclude the first column - it contains timestamps
data <- scale(traffic.mini$data[, -1])

mCCF <- fsMTS(data, max.lag=3, method="CCF")
mLARS <- fsMTS(data, max.lag=3, method="LARS")
fsSimilarity(mCCF, mLARS, cutoff=TRUE, threshold=0.2, method="Kuncheva")
fsSimilarity(mCCF, mLARS, cutoff=TRUE, threshold=0.2, method="Jaccard")
fsSimilarity(mCCF, mLARS, cutoff=TRUE, threshold=0.2, method="Hamming")
```
**fsSimilarityMatrix**

**Constructing the similarity matrix**

**Description**

`fsSimilarityMatrix` constructs a square matrix of similarity metric values between MTS feature sets. Metrics are calculated using `fsSimilarity` function with cutting-off feature sets.

**Usage**

```r
fsSimilarityMatrix(feature.sets, threshold, method)
```

**Arguments**

- `feature.sets` a list of matrixes that contains weights for features, estimated by several feature selection algorithms.
- `threshold` the required sparsity of the resulting feature set
- `method` a similarity metric. Directly passed to `fsSimilarity` function

**Value**

returns a real-valued square matrix with pairwise similarity metric values of feature sets

**See Also**

`fsSimilarity`

**Examples**

```r
# Load traffic data
data(traffic.mini)

# Scaling is sometimes useful for feature selection
# Exclude the first column - it contains timestamps
data <- scale(traffic.mini$data[,-1])

mIndep <- fsMTS(data, max.lag=3, method="ownlags")
mCCF <- fsMTS(data, max.lag=3, method="CCF")
mDistance <- fsMTS(data, max.lag=3, method="distance", shortest = traffic.mini$shortest, step = 5)
mGLASSO <- fsMTS(data, max.lag=3, method="GLASSO", rho = 0.05)
mLARS <- fsMTS(data, max.lag=3, method="LARS")
mRF <- fsMTS(data, max.lag=3, method="RF")
mMI <- fsMTS(data, max.lag=3, method="MI")

mlist <- list(Independent = mIndep,
               Distance = mDistance,
               CCF = mCCF,
               GLASSO = mGLASSO,
               LARS = mLARS,
               RF = mRF,
               MI = mMI)
```
**fsSparsity**

Calculating sparsity of a feature set

**Description**

fsSparsity calculates the sparsity (share of non-zero components) of the feature set.

**Usage**

```r
fsSparsity(feature.set)
```

**Arguments**

- `feature.set`: a matrix that contains feature weights.

**Value**

returns a share of non-zero components in the feature set.

**Examples**

```r
# Load traffic data
data(traffic.mini)

# Scaling is sometimes useful for feature selection
# Exclude the first column - it contains timestamps
data <- scale(traffic.mini$data[, -1])

mCCF <- fsMTS(data, max.lag = 3, method = "CCF")
fsSparsity(cutoff(mCCF, 0.3))
```
**traffic**

*Urban traffic (pre-processed)*

**Description**

The fsMTS package includes the dataset `traffic` that contains information from 30 sensors deployed on arterial roads for one day with 5-minute temporal aggregation (288 observations).

**Usage**

```r
data(traffic)
```

**Format**

A dataframe with 288 observations of a 30-dimensional time series

---

**traffic.mini**

*Urban traffic (preprocessed and reduced)*

**Description**

The dataset `traffic.mini` is a reduced data set from 3 sensors deployed on arterial roads for 12 hours with 5-minute temporal aggregation (144 observations).

**Usage**

```r
data(traffic.mini)
```

**Format**

A dataframe with 144 observations of a 3-dimensional time series
Index

cutoff, 3

fsEnsemble, 4
fsMTS, 5
fsMTS-package, 2
fsSimilarity, 8, 10
fsSimilarityMatrix, 10
fsSparsity, 11

traffic, 12
traffic mini, 12