Package ‘mrf’

October 13, 2022

**Type** Package

**Title** Multiresolution Forecasting

**Version** 0.1.6

**Date** 2022-02-23

**Maintainer** Quirin Stier <research@quirin-stier.de>

**Description**
Forecasting of univariate time series using feature extraction with variable prediction methods is provided. Feature extraction is done with a redundant Haar wavelet transform with filter \( h = (0.5, 0.5) \). The advantage of the approach compared to typical Fourier based methods is an dynamic adaptation to varying seasonalties. Currently implemented prediction methods based on the selected wavelets levels and scales are a regression and a multi-layer perceptron. Forecasts can be computed for horizon 1 or higher. Model selection is performed with an evolutionary optimization. Selection criteria are currently the AIC criterion, the Mean Absolute Error or the Mean Root Error. The data is split into three parts for model selection: Training, test, and evaluation dataset. The training data is for computing the weights of a parameter set. The test data is for choosing the best parameter set. The evaluation data is for assessing the forecast performance of the best parameter set on new data unknown to the model. This work is published in Stier, Q.; Gehlert, T.; Thrun, M.C. Multiresolution Forecasting for Industrial Applications. Processes 2021, 9, 1697. <doi:10.3390/pr9101697>.

**Imports** limSolve, DEoptim, stats, forecast, monmlp, nnfor

**Suggests** knitr, rmarkdown

**Depends** R (>= 3.5.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**URL** https://www.deepbionics.org

**BugReports** https://github.com/Quirinms/MRFR/issues

**NeedsCompilation** no
mrf-package

Author Quirin Stier [aut, cre, ctr],
Michael Thrun [ths, cph, rev, fnd, ctb]
(<https://orcid.org/0000-0001-9542-5543>)

Repository CRAN

Date/Publication 2022-02-23 13:50:02 UTC

R topics documented:

- mrf-package ................................................................. 2
- entsoe ................................................................. 3
- mrf_elm_forecast .......................................................... 4
- mrf_forecast ............................................................. 5
- mrf_model_selection ....................................................... 6
- mrf_multi_step_forecast ................................................. 8
- mrf_neuralnet_one_step_forecast ..................................... 10
- mrf_mnetar_forecast .................................................... 11
- mrf_one_step_forecast ................................................... 13
- mrf_regression_lsm_optimization ..................................... 14
- mrf_regression_one_step_forecast .................................... 15
- mrf_requirement ......................................................... 17
- mrf_rolling_forecasting_origin ...................................... 18
- mrf_train ............................................................... 20
- wavelet_decomposition .................................................. 22
- wavelet_prediction_equation .......................................... 24
- wavelet_training_equations ............................................. 25

Index 28

<table>
<thead>
<tr>
<th>mrf-package</th>
<th>Multiresolution Forecasting</th>
</tr>
</thead>
</table>

Description

Forecasting of univariate time series using feature extraction with variable prediction methods is provided. Feature extraction is done with a redundant Haar wavelet transform with filter \( h = (0.5, 0.5) \). The advantage of the approach compared to typical Fourier based methods is a dynamic adaptation to varying seasonality. Currently implemented prediction methods based on the selected wavelet levels and scales are a regression and a multi-layer perceptron. Forecasts can be computed for horizon 1 or higher. Model selection is performed with an evolutionary optimization. Selection criteria are currently the AIC criterion, the Mean Absolute Error or the Mean Root Error. The data is split into three parts for model selection: Training, test, and evaluation dataset. The training data is for computing the weights of a parameter set. The test data is for choosing the best parameter set. The evaluation data is for assessing the forecast performance of the best parameter set on new data unknown to the model. This work is published in Stier, Q.; Gehlert, T.; Thrun, M.C. Multiresolution Forecasting for Industrial Applications. Processes 2021, 9, 1697. <doi:10.3390/pr9101697>. The package consists of a multiresolution forecasting method using a
redundant Haar wavelet transform based on the manuscript [Stier et al., 2021] which is currently in press. One-step and multi-step forecasts are computable with this method. Nested and non-nested cross validation is possible.

**Details**

Forecasting of univariate time series using feature extraction with variable prediction methods is provided. Feature extraction is done with a redundant Haar wavelet transform with filter \( h = (0.5, 0.5) \). The advantage of the approach compared to typical Fourier based methods is an dynamic adaptation to varying seasonalties. Currently implemented prediction methods based on the selected wavelets levels and scales are a regression and a multi-layer perceptron. Forecasts can be computed for horizon 1 or higher. Model selection is performed with an evolutionary optimization. Selection criterias are currently the AIC criterion, the Mean Absolute Error or the Mean Root Error. The data is split into three parts for model selection: Training, test, and evaluation dataset. The training data is for computing the weights of a parameter set. The test data is for choosing the best parameter set. The evaluation data is for assessing the forecast performance of the best parameter set on new data unknown to the model.

- **Package**: mrf
- **Type**: Package
- **Version**: 0.1.4
- **Date**: 2021-09-20
- **License**: CC BY-NC-SA 4.0

**Author(s)**

Quirin Stier

**References**


---

**entsoe**

*Entsoe DataFrame containing Time Series*

**Description**


**Usage**

data(entsoe)
mrf_elm_forecast

Format
A DataFrame with 3652 rows and 2 columns

Source
Archive

Examples

data(entsoe)
data = entsoe$value

Description
This function creates a one step forecast using a multi layer perceptron with one hidden Layer. The number of input is the sum of all coefficients chosen with the parameter CoefficientCombination. The CoefficientCombination parameter controls the number of coefficients chosen for each wavelet and smooth part level individually.

Usage
mrf_elm_forecast(UnivariateData, Horizon, Aggregation, Threshold="hard", Lambda=0.05)

Arguments

UnivariateData     [1:n] Numerical vector with n values.
Horizon            Number indicating horizon for forecast from 1 to horizon.
Aggregation        [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
Threshold          Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.
Lambda             Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Value
forecast          Numerical value with one step forecast
mrf_forecast

Author(s)
Quirin Stier

References

Examples
data(AirPassengers)
len_data = length(as.vector(array(AirPassengers)))
UnivariateData = as.vector(AirPassengers)[1:(len_data-1)]
Aggregation = c(2,4)
if(requireNamespace('nnfor', quietly = TRUE)){
  forecast = mrf_elm_forecast(UnivariateData, Horizon=1, Aggregation)
  true_value = array(AirPassengers)[len_data]
  error = true_value - forecast
}

mrf_forecast

Multiresolution Forecast

Description
Creates a multiresolution forecast for a given multiresolution model based on [Stier et al., 2021] which is currently in press. (mrf_train).

Usage
mrf_forecast(Model, Horizon=1)

Arguments
Model List containing model specifications from mrf_train().
Horizon Number indicating horizon for forecast from 1 to horizon.
mrf_model_selection

**Value**

- **List of**
  - **Forecast**: [1:Horizon] Numerical vector with forecast of horizon according to its index.
  - **Model**: List containing model specifications from mrf_train().

**Author(s)**

Quirin Stier

**References**


**Examples**

```r
data(AirPassengers)
Data = as.vector(AirPassengers)
len_data = length(Data)
Train = Data[1:(len_data-2)]
Test = Data[(len_data-1):len_data]
# One-step forecast (Multiresolution Forecast)
model = mrf_train(Train)
one_step = mrf_forecast(model, Horizon=1)
Error = one_step$Forecast - Test[1]
# Multi-step forecast (Multiresolution Forecast)
# Horizon = 2 => Forecast with Horizon 1 and 2 as vector
model = mrf_train(Train, Horizon=2)
multi_step = mrf_forecast(model, Horizon=2)
Error = multi_step$Forecast - Test
```

---

**mrf_model_selection**  
*Model selection for Multiresolution Forecasts*

**Description**

Evaluates the best coefficient combination for a given aggregation scheme based on a rolling forecasting origin based on the manuscript [Stier et al., 2021] which is currently in press.

**Usage**

```r
mrf_model_selection(UnivariateData, Aggregation, Horizon = 1, Window = 2, Method = "r", crit = "AIC", itermax = 1, lower_limit = 1, upper_limit = 2, NumClusters = 1, Threshold="hard", Lambda=0.05)
```
Arguments

- **UnivariateData**: [1:n] Numerical vector with n values.
- **Aggregation**: [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
- **Horizon**: Number indicating horizon for forecast from 1 to horizon.
- **Window**: Number indicating how many points are used for cross validation.
- **Method**: String indicating which method to use. Available methods: 'r' = Autoregression. 'nn' = Neural Network. 'elm' = Extreme Learning Machine. 'nnetar' = forecast::nnetar. Default: Method="r".
- **itermax**: Number of iterations used in the differential evolutionary optimization algorithm. Default: itermax = 1.
- **lower_limit**: [1:Scales+1] Numeric vector: Lower limit for coefficients selected for each level.
- **upper_limit**: [1:Scales+1] Numeric vector: Higher limit for coefficients selected for each level.
- **NumClusters**: Number of clusters used for parallel computing. Default: NumClusters = 1.
- **Threshold**: Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.
- **Lambda**: Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Details

The evaluation function (optimization function) is built with a rolling forecasting origin (rolling_window function), which computes a h-step ahead forecast (for h = 1, ..., horizon) for 'Window' many steps. The input space is searched with an evolutionary optimization method. The search is restricted to one fixed aggregation scheme (parameter: 'Aggregation'). The deployed forecast method can be an autoregression or a neural network (multilayer perceptron with one hidden layer).

Value

- **CoefficientCombination**: [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme. Best combination of coefficients found by the model selection algorithm.
- **Aggregation**: [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series. Best Aggregation scheme found by the model selection algorithm.
Author(s)
Quirin Stier

References

Examples

data(entsoe)
UnivariateData = entsoe$value
Aggregation = c(2,4)
res = mrf_model_selection(UnivariateData, Aggregation, Horizon = 1, Window = 2,
Method = "r", crit = "AIC", itermax = 1, lower_limit = 1, upper_limit = 2,
NumClusters = 1)
BestCoefficientCombination = res$CoefficientCombination

mrf_multi_step_forecast
Multiresolution Forecast

Description
This function creates a multi step forecast for all horizons from 1 to steps based on the manuscript [Stier et al., 2021] which is currently in press. The deployed forecast method can be an autoregression or a neural network (multilayer perceptron with one hidden layer). Multi step forecasts are computed recursively.

Usage
mrf_multi_step_forecast(UnivariateData, Horizon, Aggregation,
CoefficientCombination=NULL, Method = "r", Threshold="hard", Lambda=0.05)

Arguments
UnivariateData [1:n] Numerical vector with n values.
Horizon Number indicating horizon for forecast from 1 to horizon.
CoefficientCombination [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.
Aggregation [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
Method
String indicating which method to use. Available methods: 'r' = Autoregression. 'nn' = Neural Network. 'elm' = Extreme Learning Machine. 'nnetar' = forecast::nnetar. Default: Method="r".

Threshold
Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.

Lambda
Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Value
List of

multistep [1:Horizon] Numerical vector with forecast of horizon according to its index.

Author(s)
Quirin Stier

References


Examples

data(AirPassengers)
len_data = length(array(AirPassengers))
UnivariateData = as.vector(AirPassengers)[1:(len_data-1)]
# One-step forecast (Multiresolution Forecast)
one_step = mrf_multi_step_forecast(UnivariateData = UnivariateData,
                                  Horizon = 2,
                                  CoefficientCombination = c(1,1,1),
                                  Aggregation = c(2,4),
                                  Method="r")

# Multi-step forecast (Multiresolution Forecast)
# Horizon = 2 => Forecast with Horizon 1 and 2 as vector
multi_step = mrf_multi_step_forecast(UnivariateData = UnivariateData,
                                    Horizon = 2,
                                    CoefficientCombination = c(1,1,1),
                                    Aggregation = c(2,4),
                                    Method="r")
mrf_neuralnet_one_step_forecast

One Step Forecast with Neural Network

Description

This function creates a one step forecast using a multi layer perceptron with one hidden Layer. The number of input is the sum of all coefficients chosen with the parameter CoefficientCombination. The CoefficientCombination parameter controls the number of coefficients chosen for each wavelet and smooth part level individually.

Usage

mrf_neuralnet_one_step_forecast(UnivariateData, CoefficientCombination, Aggregation, Threshold="hard", Lambda=0.05)

Arguments

UnivariateData [1:n] Numerical vector with n values.
CoefficientCombination [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.
Aggregation [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
Threshold Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.
Lambda Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Value

forecast Numerical value with one step forecast

Author(s)

Quirin Stier
References


Examples

data(AirPassengers)
len_data = length(as.vector(array(AirPassengers)))
UnivariateData = as.vector(AirPassengers)[1:(len_data-1)]
CoefficientCombination = c(1,1,1)
Aggregation = c(2,4)
if(requireNamespace('monmlp', quietly = TRUE)){
  forecast = mrf_neuralnet_one_step_forecast(UnivariateData,
                                           CoefficientCombination,
                                           Aggregation)

  true_value = array(AirPassengers)[len_data]
  error = true_value - forecast
}

mrf_nnetar_forecast  Forecast with nnetar

Description

This function creates a one step forecast using a multi layer perceptron with one hidden Layer. The number of input is the sum of all coefficients chosen with the parameter CoefficientCombination. The CoefficientCombination parameter controls the number of coefficients chosen for each wavelet and smooth part level individually.

Usage

mrf_nnetar_forecast(UnivariateData, Horizon, Aggregation, Threshold="hard", Lambda=0.05)
Arguments

UnivariateData [1:n] Numerical vector with n values.
Horizon Number indicating horizon for forecast from 1 to horizon.
Aggregation [1:Scales] Numerical vector carrying numbers whose index is associated with
the wavelet level. The numbers indicate the number of time in points used for
aggregation from the original time series.
Threshold Character indicating if Thresholding is done on the wavelet decomposition or
not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard
thresholding. Threshold="soft" for soft thresholding. Any other input indicates
no thresholding.
Lambda Numeric value indicating the threshold for computing a hard or soft threshold
on the wavelet decomposition.

Value

forecast Numerical value with one step forecast

Author(s)

Quirin Stier

References

Aussem, A., Campbell, J., and Murtagh, F. Wavelet-based Feature Extraction and Decomposition
Strategies for Financial Forecasting. International Journal of Computational Intelligence in Finance,
6,5-12, 1998.

Renaud, O., Starck, J.-L., and Murtagh, F. Prediction based on a Multiscale De-
2003.


Renaud, O., Starck, J.-L., and Murtagh, F. Wavelet-based combined Signal Filter-

Examples

data(AirPassengers)
len_data = length(as.vector(array(AirPassengers))
UnivariateData = as.vector(AirPassengers)[1:(len_data-1)]
Aggregation = c(2,4)
if(requireNamespace('nnfor', quietly = TRUE)){
  forecast = mrf_nnetar_forecast(UnivariateData, Horizon=1, Aggregation)
  true_value = array(AirPassengers)[len_data]
  error = true_value - forecast
}
mrf_one_step_forecast

Description

This function creates a one step forecast using the multiresolution forecasting framework based on the manuscript [Stier et al., 2021] which is currently in press.

Usage

mrf_one_step_forecast(UnivariateData, Aggregation,
CoefficientCombination=NULL,
Method="r", Threshold="hard", Lambda=0.05)

Arguments

UnivariateData [1:n] Numerical vector with n values.
CoefficientCombination [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.
Aggregation [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
Method String indicating which method to use. Available methods: 'r' = Autoregression. 'nn' = Neural Network. 'elm' = Extreme Learning Machine. 'nnetar' = forecast::nnetar. Default: Method="r".
Threshold Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.
Lambda Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Value

forecast Numerical value with one step forecast

Author(s)

Quirin Stier

References

Examples

```r
data(AirPassengers)
len_data = length(array(AirPassengers))
UnivariateData = as.vector(AirPassengers)[1:(len_data-1)]
forecast = mrf_one_step_forecast(UnivariateData=UnivariateData,
CoefficientCombination=c(1,1,1), Aggregation=c(2,4))
true_value = array(AirPassengers)[len_data]
error = true_value - forecast
```

**mrf_regression_lsm_optimization**

*Least Square Method for Regression*

**Description**

This function computes the weights for the autoregression depending on the given wavelet decomposition. It uses ordinary least square method for optimizing a linear equation system.

**Usage**

```r
mrf_regression_lsm_optimization(points_in_future, lsmatrix)
```

**Arguments**

- `points_in_future`
  - `n` many values of the time series, for which there is an equation from a prediction scheme.
- `lsmatrix`
  - Matrix carrying predictive equations associated with a specific value of the time series.

**Value**

- `weights`
  - Array of weights carrying the solution for a matrix problem, which was solved with ordinary least squares.

**Author(s)**

Quirin Stier
References


Examples

data(AirPassengers)
len_data = length(array(AirPassengers))
CoefficientCombination = c(1,1,1)
Aggregation = c(2,4)
UnivariateData = as.vector(AirPassengers)
# Decomposition
dec_res <- wavelet_decomposition(UnivariateData, Aggregation)
# Training
trs_res <- wavelet_training_equations(UnivariateData,
dec_res$WaveletCoefficients,
dec_res$SmoothCoefficients,
dec_res$Scales,
CoefficientCombination, Aggregation)
arr_future_points = trs_res$points_in_future
matrix = trs_res$ls_matrix
# Optimization method
weights = mrf_regression_lsm_optimization(arr_future_points, matrix)
# Forecast
scheme = wavelet_prediction_equation(dec_res$WaveletCoefficients,
dec_res$SmoothCoefficients, CoefficientCombination, Aggregation)
forecast = weights

mrf_regression_one_step_forecast

One Step Forecast with Regression

Description

This function creates a one step forecast using an autoregression method. The ccps parameter controls the number of coefficients chosen for each wavelet and smooth part level individually.
Usage

mrf_regression_one_step_forecast(UnivariateData, CoefficientCombination, Aggregation, Threshold="hard", Lambda=0.05)

Arguments

UnivariateData [1:n] Numerical vector with n values.
CoefficientCombination [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.
Aggregation [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
Threshold Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.
Lambda Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Value

forecast Numerical value with one step forecast

Author(s)

Quirin Stier

References


Examples

data(AirPassengers)
len_data = length(as.vector(array(AirPassengers)))
UnivariateData = as.vector(AirPassengers)[1:(len_data-1)]
CoefficientCombination = c(1,1,1)
Aggregation = c(2,4)
forecast = mrf_regression_one_step_forecast(UnivariateData,
CoefficientCombination,
Aggregation)

true_value = array(AirPassengers)[len_data]
error = true_value - forecast

mrf_requirement  
Multiresolution Forecast Requirements

Description

Computes requirements for given model using insights of various published papers and the manuscript
[Stier et al., 2021] which is currently in press.

Usage

mrf_requirement(UnivariateData, CoefficientCombination, Aggregation)

Arguments

UnivariateData  [1:n] Numerical vector with n values.
CoefficientCombination
[1:Scales+1] Numerical vector with numbers which are associated with wavelet
levels. The last number is associated with the smooth level. Each number deter-
mines the number of coefficient used per level. The selection follows a specific
scheme.
Aggregation  [1:Scales] Numerical vector carrying numbers whose index is associated with
the wavelet level. The numbers indicate the number of time in points used for
aggregation from the original time series.

Value

List of

MinLen  Integer minimum required length for model.
StartTraining  Integer indicating the index of time series at which the training equations can be
built up.
NumberWeights  Number of weights required for building model.
NumberEquations  Number of equations which can be built with given data.
Author(s)

Quirin Stier

References


Examples

data(entsoe)
UnivariateData = entsoe$value
mrf_requirement(UnivariateData, c(2,3,4), c(2,4))

mrf_rolling_forecasting_origin

Rolling forecasting origin for Multiresolution Forecasts

Description

This function computes a rolling forecasting origin for one- or multi-step forecasts with a specific method based on the manuscript [Stier et al., 2021] which is currently in press. Multi-step forecasts are computed recursively with the one step forecast method.

Usage

mrf_rolling_forecasting_origin(UnivariateData, Aggregation, CoefficientCombination=NULL, Horizon = 2, Window = 3, Method = "r", NumClusters = 1, Threshold="hard", Lambda=0.05)
Arguments

**UnivariateData**  [1:n] Numerical vector with n values.

**CoefficientCombination**  [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.

**Aggregation**  [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.

**Horizon**  Number indicating horizon for forecast from 1 to horizon.

**Window**  Number indicating how many points are used for cross validation.

**Method**  String indicating which method to use. Available methods: 'r' = Autoregression. 'nn' = Neural Network. 'elm' = Extreme Learning Machine. 'nnetar' = forecast::nnetar. Default: Method="r".

**NumClusters**  Number of clusters used for parallel computing.

**Threshold**  Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.

**Lambda**  Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.

Details

Thus, h-step forecast for h = 1,..., horizon for window_size many steps can be computed. The forecasting method can be an autoregression or a neural network (multilayer perceptron). The CoefficientCombination parameter controls the number of coefficients chosen for each wavelet and smooth part level individually. The NumClusters parameter determines the number of cluster used for parallel computation. NumClusters = 1 performs a non parallel version. NumClusters is constrained by the maximum number of clusters available minus one to prevent the machine to be overchallenged.

Value

**List of**

**Error**  [1:Window,1:Horizon] Numerical Matrix with 'Window' many row entries indicating one time point with 'Horizon' many forecast errors.

**Forecast**  [1:Window,1:Horizon] Numerical Matrix with 'Window' many row entries indicating one time point with 'Horizon' many forecasts.

Author(s)

Quirin Stier
References


Examples

```r
data(AirPassengers)
UnivariateData = as.vector(array(AirPassengers))
res = mrf_rolling_forecasting_origin(UnivariateData,
  CoefficientCombination = c(10,10,10),
  Aggregation = c(2,4),
  Horizon = 2, Window = 3, Method = "r",
  NumClusters = 1)
Error = res$Error
Forecast = res$Forecast
```

---

**mrf_train**

**Multiresolution Forecast**

**Description**

Creates a multiresolution forecast model which can be used for forecasting with method `mrf_forecast` based on the manuscript [Stier et al., 2021] which is currently in press.

**Usage**

```r
mrf_train(Data, Horizon=1, Aggregation="auto", Method = "r",
  TimeSteps4ModelSelection=2, crit="AIC", InSample=FALSE, Threshold="hard",
  Lambda=0.05, NumClusters=1, itermax=1)
```

**Arguments**

- **Data**
  
  [1:n] Numerical vector with n values from the training data.

- **Horizon**
  
  Number indicating forecast horizon. Horizon = 1 means one-step forecast and Horizon > 1 means a one-step forecast and all multi-step forecasts from horizon 2 to 'Horizon'. Default: Horizon = 1.

- **Aggregation**
  
  [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series. Default: Aggregation = "auto".

- **Method**
  
  String indicating which method to use. Available methods: 'r' = Autoregression. 'nn' = Neural Network. 'elm' = Extreme Learning Machine. 'nnetar' = forecast::nnetar. Default: Method="r".

- **TimeSteps4ModelSelection**
  
  Number of time steps of data (newest part) on which a model selection is performed. Default: TimeSteps4ModelSelection = 2.

- **crit**
  
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InSample</td>
<td>Boolean, deciding if in-sample-forecast based on rolling forecasting origin is computed or not. TRUE = Computation of in-sample-forecast. FALSE = No computation. Default: InSample = FALSE</td>
</tr>
<tr>
<td>Threshold</td>
<td>Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold=&quot;hard&quot;. Possible entries: Threshold=&quot;hard&quot; for hard thresholding. Threshold=&quot;soft&quot; for soft thresholding. Any other input indicates no thresholding.</td>
</tr>
<tr>
<td>Lambda</td>
<td>Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.</td>
</tr>
<tr>
<td>NumClusters</td>
<td>Number of clusters used for parallel computing. Default: NumClusters = 1.</td>
</tr>
<tr>
<td>itermax</td>
<td>Number of iterations used in the differential evolutionary optimization algorithm. Default: itermax = 1.</td>
</tr>
</tbody>
</table>

**Value**

- **List with**
  - **Data** [1:n] Numerical vector with n values from the training data.
  - **Method** String indicating which method to use.
  - **Aggregation** [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.
  - **CoefficientCombination** [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.
  - **Horizon** Number indicating forecast horizon. Horizon = 1 means one-step forecast and Horizon > 1 means a one-step forecast and all multi-step forecasts from horizon 2 to 'Horizon'.
  - **ModelError** [1:TimeSteps4ModelSelection, 1:Horizon] Numerical matrix with one-/multi-steps in columns and the time steps rowwise. The error is according to the scheme of a rolling forecasting origin. The length depends on the minimum required length for constructing the wavelet model and the length of data. The newer part of the data is used for the model fit truncating the oldest data according to the minimum required length for constructing the model.
  - **ModelMAE** Integer: Mean Absolute Error (MAE) computed for the in-sample-forecast resulting from a rolling forecasting origin.

**Author(s)**

Quirin Stier

**References**

Examples

```r
data(AirPassengers)
Data = as.vector(AirPassengers)
len_data = length(Data)
Train = Data[1:(len_data-2)]
Test = Data[(len_data-1):len_data]
# One-step forecast (Multiresolution Forecast)
model = mrf_train(Train)
one_step = mrf_forecast(model, Horizon=1)
Error = one_step$Forecast - Test[1]
# Multi-step forecast (Multiresolution Forecast)
# Horizon = 2 => Forecast with Horizon 1 and 2 as vector
model = mrf_train(Train, Horizon=2)
multi_step = mrf_forecast(model, Horizon=2)
Error = multi_step$Forecast - Test
```

---

wavelet_decomposition  *Redundant Haar Wavelet Decomposition*

Description

This function decomposes a time series in its wavelet and smooth coefficients using the redundant Haar wavelet transform.

Usage

```r
wavelet_decomposition(UnivariateData, Aggregation = c(2, 4, 8, 16, 32),
Threshold="hard", Lambda=0.05)
```

Arguments

- **UnivariateData** [1:n] Numerical vector with n time series values
- **Aggregation** [1:Scales] Numerical vector of length 'Scales' carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of values used for aggregation from the original time series.
- **Threshold** Character indicating if Thresholding is done on the wavelet decomposition or not. Default: Threshold="hard". Possible entries: Threshold="hard" for hard thresholding. Threshold="soft" for soft thresholding. Any other input indicates no thresholding.
- **Lambda** Numeric value indicating the threshold for computing a hard or soft threshold on the wavelet decomposition.
Details

The resulting wavelet and smooth coefficients are stored in so called wavelet and smooth part levels. The smooth part level is created from the original times series by aggregation (average). This makes the times series in some sense smoother, hence the naming. Each individual smooth part level can be created from the original time series by aggregating over different number of values. The different smooth part levels are ordered, so that the number of values used for aggregation are ascending. A dyadic scheme is recommended (increasing sequences of the power of two). The dyadic scheme for 5 levels would require \( \text{agg\_per\_lvl} = c(2, 4, 8, 16, 32) \). So the first smooth part level would be the average of two points of the original time series, the second smooth part level would be the average of four points, and so on. This averaging is applied asymmetrical. That means, that the result of the average of a sequence of points is obtained for the last point in time of that sequence. So each smooth part level starts with a certain offset, since no average can be obtained for the first particular points in time. The wavelet levels are the differences between the original time series and the smooth levels. The first wavelet level is the difference of the original time series and the first smooth part level. The second wavelet level is the difference of the first and second smooth part level and so on.

Value

List of

- \text{UnivariateData} [1:n] Numerical vector with \( n \) time series values.
- \text{WaveletCoefficients} [\text{Scales}, n] Matrix with ‘\text{Scales}’ many wavelet scales row-wise with \( n \) columns corresponding to the time domain of a time series.
- \text{SmoothCoefficients} [\text{Scales}, n] Matrix with ‘\text{Scales}’ many smooth approximation scales row-wise with \( n \) columns corresponding to the time domain of a time series.
- \text{Scales} Number of wavelet levels.

Author(s)

Quirin Stier

References


Examples

data(AirPassengers)
plot(AirPassengers, type = "l", col = "black")
UnivariateData = as.vector(array(AirPassengers))
dec_res = wavelet_decomposition(UnivariateData, Aggregation = c(2,4))
plot(dec_res$SmoothCoefficients[2,4:length(dec_res$SmoothCoefficients[2,])],
    type = "l", col = "blue")
lines(array(AirPassengers)[4:length(dec_res$SmoothCoefficients[2,])],
    col = "black")

---

wavelet_prediction_equation

*One Step Forecast with Regression*

Description

This function delivers the required wavelet and smooth coefficients from the decomposition based on a prediction scheme.

Usage

```r
wavelet_prediction_equation(WaveletCoefficients, SmoothCoefficients,
    CoefficientCombination, Aggregation)
```

Arguments

- **WaveletCoefficients**: 
  
  [Scales, n] Matrix with 'Scales' many wavelet scales row-wise with n columns corresponding to the time domain of a time series.

- **SmoothCoefficients**: 
  
  [Scales, n] Matrix with 'Scales' many smooth approximation scales row-wise with n columns corresponding to the time domain of a time series.

- **CoefficientCombination**: 
  
  [1:Scales+1] Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.

- **Aggregation**: 
  
  [1:Scales] Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.

Value

- **future_point**: Numerical value carrying one step forecast.

Author(s)

Quirin Stier
wavelet_training_equations

References


Examples

data(AirPassengers)
len_data = length(array(AirPassengers))
CoefficientCombination = c(1,1,1)
Aggregation = c(2,4)
UnivariateData = as.vector(AirPassengers)
# Decomposition
dec_res <- wavelet_decomposition(UnivariateData, Aggregation)
# Training
trs_res <- wavelet_training_equations(UnivariateData,
dec_res$WaveletCoefficients,
dec_res$SmoothCoefficients,
dec_res$Scales,
CoefficientCombination, Aggregation)
arr_future_points = trs_res$points_in_future
matrix = trs_res$lsmatrix
# Optimization method
weights = mrf_regression_lsm_optimization(arr_future_points, matrix)
# Forecast
scheme = wavelet_prediction_equation(dec_res$WaveletCoefficients,
dec_res$SmoothCoefficients, CoefficientCombination, Aggregation)
forecast = weights

wavelet_training_equations

Generic Training Scheme for wavelet framework

Description

This function computes the input for the training phase required for one step forecasts. This computational step is required for all one step forecast procedures contained in this package.
Usage

\texttt{wavelet\_training\_equations(UnivariateData, WaveletCoefficients, SmoothCoefficients, Scales, CoefficientCombination, Aggregation)}

Arguments

- **UnivariateData** \([1:n]\) Numerical vector with \(n\) values.
- **WaveletCoefficients** \([Scales, n]\) Matrix with \('Scales' many wavelet scales row-wise with \(n\) columns corresponding to the time domain of a time series.
- **SmoothCoefficients** \([Scales, n]\) Matrix with \('Scales' many smooth approximation scales row-wise with \(n\) columns corresponding to the time domain of a time series.
- **Scales** Number of wavelet levels.
- **CoefficientCombination** \([1:Scales+1]\) Numerical vector with numbers which are associated with wavelet levels. The last number is associated with the smooth level. Each number determines the number of coefficient used per level. The selection follows a specific scheme.
- **Aggregation** \([1:Scales]\) Numerical vector carrying numbers whose index is associated with the wavelet level. The numbers indicate the number of time in points used for aggregation from the original time series.

Value

- **points\_in\_future** \(n\) many values of the time series, for which there is an equation from a prediction scheme.
- **lsmatrix** Matrix carrying predictive equations associated with a specific value of the time series.

Author(s)

Quirin Stier

References


Examples

data(AirPassengers)
len_data = length(array(AirPassengers))
CoefficientCombination = c(1,1,1)
Aggregation = c(2,4)
UnivariateData = as.vector(AirPassengers)
# Decomposition
dec_res <- wavelet_decomposition(UnivariateData, Aggregation)
# Training
trs_res <- wavelet_training_equations(UnivariateData,
    dec_res$WaveletCoefficients,
    dec_res$SmoothCoefficients,
    dec_res$Scales,
    CoefficientCombination, Aggregation)

arr_future_points = trs_res$points_in_future
matrix = trs_res$lsmatrix
# Optimization method
weights = mrf_regression_lsm_optimization(arr_future_points, matrix)
# Forecast
scheme = wavelet_prediction_equation(dec_res$WaveletCoefficients,
    dec_res$SmoothCoefficients, CoefficientCombination, Aggregation)
forecast = weights
Index

* Cross Validation
  mrf-package, 2
* Differential Evolution
  mrf_forecast, 5
  mrf_train, 20
* Evolutionary Optimization
  mrf_forecast, 5
  mrf_train, 20
* Multi step forecasts
  mrf_multi_step_forecast, 8
* Multi-step forecasts
  mrf_forecast, 5
  mrf_train, 20
* Multilayer Perceptron
  mrf_elm_forecast, 4
  mrf_neuralnet_one_step_forecast, 10
  mrf_nnetar_forecast, 11
  mrf_one_step_forecast, 13
* Multiresolution Forecasting
  mrf_forecast, 5
  mrf_requirement, 17
  mrf_train, 20
* Multiresolution
  mrf-package, 2
* Nested Cross Validation
  mrf-package, 2
* Neural Networks
  mrf_elm_forecast, 4
  mrf_neuralnet_one_step_forecast, 10
  mrf_nnetar_forecast, 11
  mrf_one_step_forecast, 13
* One Step Forecasts
  mrf_one_step_forecast, 13
* One-step forecasts
  mrf_forecast, 5
  mrf_train, 20
* One-step forecast
  mrf_elm_forecast, 4
  mrf_neuralnet_one_step_forecast, 10
  mrf_nnetar_forecast, 11
  mrf_regression_one_step_forecast, 15
* Ordinary least squares
  mrf_regression_lsm_optimization, 14
* Regression
  mrf_one_step_forecast, 13
  mrf_regression_one_step_forecast, 15
* Rolling Forecasting Origin
  mrf-package, 2
* Rolling forecasting origin
  mrf_rolling_forecasting_origin, 18
* Seasonal Univariate Time Series Forecasting
* Time Series Forecasting
  mrf_forecast, 5
  mrf_train, 20
* Univariate Time Series Forecasting
  mrf-package, 2
  mrf_forecast, 5
  mrf_train, 20
* Wavelets
  mrf-package, 2
  wavelet_decomposition, 22
  wavelet_prediction_equation, 24
  wavelet_training_equations, 25
* datasets
  entsoe, 3
  entsoe, 3
  mrf-package, 2
INDEX

mrf_elm_forecast, 4
mrf_forecast, 5
mrf_model_selection, 6
mrf_multi_step_forecast, 8
mrf_neuralnet_one_step_forecast, 10
mrf_nnetar_forecast, 11
mrf_one_step_forecast, 13
mrf_regression_lsm_optimization, 14
mrf_regression_one_step_forecast, 15
mrf_requirement, 17
mrf_rolling_forecasting_origin, 18
mrf_train, 20

wavelet_decomposition, 22
wavelet_prediction_equation, 24
wavelet_training_equations, 25