Package 'TSLSTMplus'

February 3, 2025

Title Long-Short Term Memory for Time-Series Forecasting, Enhanced

Version 1.0.6

Author Jaime Pizarroso Gonzalo [aut, ctb, cre],
Antonio Muñoz San Roque [aut]

Maintainer Jaime Pizarroso Gonzalo <jpizarroso@comillas.edu>

Description The LSTM (Long Short-Term Memory) model is a Recurrent Neural Network (RNN) based architecture that is widely used for time series forecasting. Customizable configurations for the model are allowed, improving the capabilities and usability of this model compared to other packages. This package is based on 'keras' and 'tensorflow' modules and the algorithm of Paul and Garai (2021) <doi:10.1007/s00500-021-06087-4>.

License GPL-3

Encoding UTF-8

Imports keras, tensorflow, stats, abind

NeedsCompilation no

RoxygenNote 7.3.2

Date 2025-02-03

Repository CRAN

Date/Publication 2025-02-03 14:40:02 UTC

Contents

lagmatrix 2
LSTMModel 3
minmax_scale 4
predict.LSTMModel 5
summary.LSTMModel 6
ts.lstm 7
ts.prepare.data 9

Index **11**

`lagmatrix` *Create lead/lags of a variable*

Description

Create an array with lead/lags of an input variable.

Usage

```
lagmatrix(x, lag)
```

Arguments

`x`	input variable.
`lag`	vector of leads and lags. Positive numbers are lags, negative are leads. O is the original `x`.

Value

An array with the resulting leads and lags (columns).

Note

This code was copied from the `ts.utils` package to avoid the archive operations of the `smooth` package in 16-02-2025. This function might be deprecated in future releases to use the one from `ts.utils` again.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>

Examples

```
x <- rnorm(10)
lagmatrix(x,c(0,1,-1))
```

LSTMModel *LSTMModel class*

Description

LSTMModel class for further use in predict function

Usage

```
LSTMModel(
  lstm_model,
  scale_output,
  scaler_output,
  scale_input,
  scaler_input,
  tsLag,
  xregLag,
  model_structure,
  batch_size,
  lags_as_sequences,
  stateful
)
```

Arguments

`lstm_model`	LSTM 'keras' model
`scale_output`	indicate which type of scaler is used in the output
`scaler_output`	Scaler of output variable (and lags)
`scale_input`	indicate which type of scaler is used in the input(s)
`scaler_input`	Scaler of input variable(s) (and lags)
`tsLag`	Lag of time series data
`xregLag`	Lag of exogenous variables
`model_structure`	Summary of the LSTM model previous to training
`batch_size`	Batch size used during training of the model
`lags_as_sequences`	Flag to indicate the model has been trained statefully
`stateful`	Flag to indicate if LSTM layers shall retain its state between batches.

Value

LSTMModel object

References

Paul, R.K. and Garai, S. (2021). Performance comparison of wavelets-based machine learning technique for forecasting agricultural commodity prices, Soft Computing, 25(20), 12857-12873

Examples

```
if (keras::is_keras_available()){
  y<-rnorm(100,mean=100,sd=50)
  x1<-rnorm(100,mean=50,sd=50)
  x2<-rnorm(100, mean=50, sd=25)
  x<-cbind(x1,x2)
  TSLSTM<-ts.lstm(ts=y,
                  xreg = x,
                  tsLag=2,
                  xregLag = 0,
                  LSTMUnits=5,
                  ScaleInput = 'scale',
                  ScaleOutput = 'scale',
                  Epochs=2)
}
```

---

minmax_scale *Min-Max Scaling of a Matrix*

Description

This function applies min-max scaling to a matrix. Each column of the matrix is scaled independently. The scaling process transforms the values in each column to a specified range, typically [0, 1]. The function subtracts the minimum value of each column (if ‘min‘ is ‘TRUE‘ or a numeric vector) and then divides by the range of each column (if ‘range‘ is ‘TRUE‘ or a numeric vector).

Usage

```
minmax_scale(x, min = TRUE, range = TRUE)
```

Arguments

`x`	A numeric matrix whose columns are to be scaled.
`min`	Logical or numeric vector. If ‘TRUE‘, the minimum value of each column is subtracted. If a numeric vector is provided, it must have a length equal to the number of columns in ‘x‘, and these values are subtracted from each corresponding column.
`range`	Logical or numeric vector. If ‘TRUE‘, each column is divided by its range. If a numeric vector is provided, it must have a length equal to the number of columns in ‘x‘, and each column is divided by the corresponding value in this vector.

Value

A matrix with the same dimensions as ‘x‘, where each column has been scaled according to the min-max scaling process.

Examples

```
data <- matrix(rnorm(100), ncol = 10)
scaled_data <- minmax_scale(data)
```

---

predict.LSTMModel *Predict using a Trained LSTM Model*

Description

This function makes predictions using a trained LSTM model for time series forecasting. It performs iterative predictions where each step uses the prediction from the previous step. The function takes into account the lags in both the time series data and the exogenous variables.

Usage

```
## S3 method for class 'LSTMModel'
predict(
  object,
  ts,
  xreg = NULL,
  xreg.new = NULL,
  horizon = NULL,
  BatchSize = NULL,
  ...
)
```

Arguments

`object`	An LSTMModel object containing a trained LSTM model along with normalization parameters and lag values.
`ts`	A vector or time series object containing the historical time series data. It should have a number of observations at least equal to the lag of the time series data.
`xreg`	(Optional) A matrix or data frame of exogenous variables to be used for prediction. It should have a number of rows at least equal to the lag of the exogenous variables.
`xreg.new`	(Optional) A matrix or data frame of exogenous variables to be used for prediction. It should have a number of rows at least equal to the lag of the exogenous variables.
`horizon`	The number of future time steps to predict.
`BatchSize`	(Optional) Batch size to use during prediction
`...`	Optional arguments, no use is contemplated right now

Value

A vector containing the forecasted values for the specified horizon.

Examples

```
if (keras::is_keras_available()){
    y<-rnorm(100,mean=100,sd=50)
    x1<-rnorm(150,mean=50,sd=50)
    x2<-rnorm(150, mean=50, sd=25)
    x<-cbind(x1,x2)
    x.tr <- x[1:100,]
    x.ts <- x[101:150,]
    TSLSTM<-ts.lstm(ts=y,
                    xreg = x.tr,
                    tsLag=2,
                    xregLag = 0,
                    LSTMUnits=5,
                    ScaleInput = 'scale',
                    ScaleOutput = 'scale',
                    Epochs=2)
    current_values <- predict(TSLSTM, xreg = x.tr, ts = y)
    future_values <- predict(TSLSTM, horizon=50, xreg = x, ts = y, xreg.new = x.ts)
 }
```

`summary.LSTMModel` *Summary of a Trained LSTM Model*

Description

This function generates the summary of the LSTM model.

Usage

```
## S3 method for class 'LSTMModel'
summary(object, ...)
```

Arguments

`object`	An LSTMModel object containing a trained LSTM model along with normalization parameters and lag values.
`...`	Optional arguments, no use is contemplated right now

Value

A vector containing the forecasted values for the specified horizon.

Examples

```
if (keras::is_keras_available()){
    y<-rnorm(100,mean=100,sd=50)
    x1<-rnorm(100,mean=50,sd=50)
    x2<-rnorm(100, mean=50, sd=25)
    x<-cbind(x1,x2)
    TSLSTM<-ts.lstm(ts=y,
                    xreg = x,
                    tsLag=2,
                    xregLag = 0,
                    LSTMUnits=5,
                    ScaleInput = 'scale',
                    ScaleOutput = 'scale',
                    Epochs=2)
    # Assuming TSLSTM is an LSTMModel object created using ts.lstm function
    summary(TSLSTM)
}
```

---

ts.lstm *Long Short Term Memory (LSTM) Model for Time Series Forecasting*

Description

The LSTM (Long Short-Term Memory) model is a Recurrent Neural Network (RNN) based architecture that is widely used for time series forecasting. Min-Max transformation has been used for data preparation. Here, we have used one LSTM layer as a simple LSTM model and a Dense layer is used as the output layer. Then, compile the model using the loss function, optimizer and metrics. This package is based on 'keras' and TensorFlow modules.

Usage

```
ts.lstm(
  ts,
  xreg = NULL,
  tsLag = NULL,
  xregLag = 0,
  LSTMUnits,
  DenseUnits = NULL,
  DropoutRate = 0,
  Epochs = 10,
  CompLoss = "mse",
  CompMetrics = "mae",
  Optimizer = optimizer_rmsprop,
  ScaleOutput = c(NULL, "scale", "minmax"),
  ScaleInput = c(NULL, "scale", "minmax"),
  BatchSize = 1,
```

```
  LSTMActivationFn = "tanh",
  LSTMRecurrentActivationFn = "sigmoid",
  DenseActivationFn = "relu",
  ValidationSplit = 0.1,
  verbose = 2,
  RandomState = NULL,
 EarlyStopping = callback_early_stopping(monitor = "val_loss", min_delta = 0, patience =
    3, verbose = 0, mode = "auto"),
  LagsAsSequences = TRUE,
  Stateful = FALSE,
  ...
)
```

Arguments

`ts`	Time series data
`xreg`	Exogenous variables
`tsLag`	Lag of time series data. If NULL, no lags of the output are used.
`xregLag`	Lag of exogenous variables
`LSTMUnits`	Number of unit in LSTM layers
`DenseUnits`	Number of unit in Extra Dense layers. A Dense layer with a single neuron is always added at the end.
`DropoutRate`	Dropout rate
`Epochs`	Number of epochs
`CompLoss`	Loss function
`CompMetrics`	Metrics
`Optimizer`	'keras' optimizer
`ScaleOutput`	Flag to indicate if ts shall be scaled before training
`ScaleInput`	Flag to indicate if xreg shall be scaled before training
`BatchSize`	Batch size to use during training
`LSTMActivationFn`	Activation function for LSTM layers
`LSTMRecurrentActivationFn`	Recurrent activation function for LSTM layers
`DenseActivationFn`	Activation function for Extra Dense layers
`ValidationSplit`	Validation split ration
`verbose`	Indicate how much information is given during training. Accepted values, 0, 1 or 2.
`RandomState`	seed for replication
`EarlyStopping`	EarlyStopping according to 'keras'

`LagsAsSequences`
: Use lags as previous timesteps of features, otherwise use them as "extra" features.

`Stateful`
: Flag to indicate if LSTM layers shall retain its state between batches.

`...`
: Extra arguments passed to keras::layer_lstm

Value

LSTMmodel object

References

Paul, R.K. and Garai, S. (2021). Performance comparison of wavelets-based machine learning technique for forecasting agricultural commodity prices, Soft Computing, 25(20), 12857-12873

Examples

```
if (keras::is_keras_available()){
    y<-rnorm(100,mean=100,sd=50)
    x1<-rnorm(100,mean=50,sd=50)
    x2<-rnorm(100, mean=50, sd=25)
    x<-cbind(x1,x2)
    TSLSTM<-ts.lstm(ts=y,
                    xreg = x,
                    tsLag=2,
                    xregLag = 0,
                    LSTMUnits=5,
                    ScaleInput = 'scale',
                    ScaleOutput = 'scale',
                    Epochs=2)
}
```

`ts.prepare.data` *Prepare data for Long Short Term Memory (LSTM) Model for Time Series Forecasting*

Description

The LSTM (Long Short-Term Memory) model is a Recurrent Neural Network (RNN) based architecture that is widely used for time series forecasting. Min-Max transformation has been used for data preparation. Here, we have used one LSTM layer as a simple LSTM model and a Dense layer is used as the output layer. Then, compile the model using the loss function, optimizer and metrics. This package is based on 'keras' and TensorFlow modules.

Usage

```
ts.prepare.data(ts, xreg = NULL, tsLag, xregLag = 0)
```

Arguments

ts	Time series data
xreg	Exogenous variables
tsLag	Lag of time series data
xregLag	Lag of exogenous variables

Value

dataset with all lags created from exogenous and time series data.

Examples

```
y <- rnorm(100,mean=100,sd=50)
x1 <- rnorm(100,mean=50,sd=50)
x2 <- rnorm(100, mean=50, sd=25)
x <- cbind(x1,x2)
ts.prepare.data(y, x, 2, 4)
```

Index

lagmatrix, 2
LSTMModel, 3

minmax_scale, 4

predict.LSTMModel, 5

summary.LSTMModel, 6

ts.lstm, 7
ts.prepare.data, 9