Package 'IFLSTM'

November 25, 2026

Type Package

Title Intuitionistic Fuzzy Long Short-Term Memory Network for Time Series

Version 0.1.0

Description Time series forecasting with an intuitionistic fuzzy long short-term memory (LSTM) network. Lagged values of the series and their intuitionistic fuzzy membership and non-membership degrees, obtained by intuitionistic fuzzy c-means clustering, are fed as a sequence to an LSTM network (Hochreiter and Schmidhuber, 1997) <doi:10.1162/neco.1997.9.8.1735> trained with the Adam optimiser. The network is implemented in plain R and needs no external deep-learning library. The lag order and the network hyper-parameters (number of units and epochs) can be set manually or tuned automatically by grid search over user-defined minimum, maximum and step ranges, with time-ordered validation and a progress display. The intuitionistic fuzzy approach follows Yolcu and Yolcu (2023) <doi:10.1016/j.eswa.2022.119336>.

License GPL-3

Encoding UTF-8

Imports stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Anita Sarkar [aut],
Pushkar Bora [aut],
Md Yeasin [aut, cre],
Ranjit Kumar Paul [aut],
Ankit Kumar Singh [aut],
Amrit Kumar Paul [aut]

Maintainer Md Yeasin <yeasin.iasri@gmail.com>

Repository CRAN

Date/Publication 2026-11-25 12:30:02 UTC

Contents

IFLSTM 2

Index **7**

IFLSTM *Intuitionistic Fuzzy Long Short-Term Memory Network for Time Series*

Description

Fits an intuitionistic fuzzy long short-term memory (IFLSTM) network to a univariate time series. The lagged values of the series, together with their intuitionistic fuzzy membership and non-membership degrees (from intuitionistic fuzzy c-means clustering), are fed to an LSTM network as a sequence. The lag order (sequence length) and the network hyper-parameters can each be set manually or tuned automatically by grid search. The LSTM is implemented in plain R, so no Python, TensorFlow or other deep-learning library is needed.

Usage

```
IFLSTM(
  Data,
  Class = 3,
  Split = 0.8,
  Lag_Tuning = c("auto", "manual"),
  Lag = 1,
  Lag_Range = c(min = 1, max = 12, step = 1),
  LSTM_Tuning = c("auto", "manual"),
  Units = 8,
  Epochs = 100,
  Units_Range = c(min = 4, max = 12, step = 4),
  Epochs_Range = c(min = 50, max = 150, step = 50),
  Learn_Rate = 0.01,
  Batch_Size = 32,
  Repeats = 1,
  Validation = c("holdout", "tscv"),
  Val_Split = 0.2,
  Folds = 3,
  Metric = c("RMSE", "MAE", "MAPE"),
  Forecast_Type = c("one-step", "recursive"),
  m = 2,
  Alpha = 0.85,
  Max_Iter = 100,
  Tol = 1e-05,
  Verbose = TRUE
)
```

Arguments

`Data`	Numeric time series (vector or `ts`) without missing values.
`Class`	Number of intuitionistic fuzzy clusters (at least 2).
`Split`	Proportion of the data used for training, strictly between 0 and 1.
`Lag_Tuning`	`"auto"` (default) to tune the lag order over `Lag_Range`, or `"manual"` to use `Lag`.
`Lag`	Lag order (sequence length) used when `Lag_Tuning = "manual"` (a single positive integer).
`Lag_Range`	Lag search range `c(min, max, step)` for `Lag_Tuning = "auto"` (positive integers). Lags too large for the data are dropped with a warning.
`LSTM_Tuning`	`"auto"` (default) to tune `Units` and `Epochs`, or `"manual"` to use the single values given.
`Units`, `Epochs`	Number of LSTM units (memory cells) and number of training epochs, used when `LSTM_Tuning = "manual"` (positive integers).
`Units_Range`, `Epochs_Range`	Search ranges `c(min, max, step)` for `LSTM_Tuning = "auto"` (positive integers).
`Learn_Rate`	Learning rate of the Adam optimiser.
`Batch_Size`	Number of sequences per mini-batch.
`Repeats`	Number of networks trained from different random starting weights; their predictions are averaged.
`Validation`	`"holdout"` (default) or `"tscv"` (expanding-window time-series cross-validation).
`Val_Split`	Share of the training set used for validation, strictly between 0 and 1.
`Folds`	Number of folds for `Validation = "tscv"`.
`Metric`	Criterion minimised during tuning: `"RMSE"` (default), `"MAE"` or `"MAPE"`.
`Forecast_Type`	`"one-step"` (default): every test value is predicted from the observed previous values. `"recursive"`: the test period is forecast multi-step ahead, feeding predictions back as lags. The same type is used during validation.
`m`	Fuzzifier of the clustering (greater than 1).
`Alpha`	Exponent of the Yager non-membership generator, in (0, 1].
`Max_Iter`, `Tol`	Maximum number of iterations and convergence tolerance of the clustering.
`Verbose`	Logical; show the grid-search progress and the result summary as messages (default `TRUE`).

Details

Model. The first `round(length(Data) * Split)` observations form the training set and the rest the test set. Intuitionistic fuzzy c-means clustering (Chaira, 2011) with `Class` clusters is fitted to the training observations only. Non-membership degrees come from the Yager-type generator $\nu = (1 - \mu^{\alpha})^{1/\alpha}$, so the hesitation degree $\pi = 1 - \mu - \nu$ is non-negative for $0 < \alpha \leq 1$; `Alpha = 1` gives ordinary fuzzy sets. To predict y_t with lag order L, the network reads the L time steps $y_{t-L}, \ldots, y_{t-1}$ (oldest first); at each step the input is the value (min-max scaled with training statistics) and its membership and non-membership degrees in every cluster, i.e. $1 + 2\,Class$ inputs

per step. Only past values enter the inputs, so the test-set forecasts use no information from the values being forecast.

Network. One LSTM layer (Hochreiter and Schmidhuber, 1997) with `Units` memory cells (input, forget and output gates, tanh candidate) is followed by a linear output on the last hidden state. The network is trained on the target scaled to [0, 1] by back-propagation through time with the Adam optimiser (Kingma and Ba, 2015), learning rate `Learn_Rate`, mini-batches of `Batch_Size` sequences, mean squared error loss and gradient-norm clipping at 5, for `Epochs` passes over the data. Weights are initialised as in Keras (Glorot-uniform input weights, orthogonal recurrent weights, forget-gate bias 1). Because the result depends on the random initial weights, `Repeats` networks can be trained from different starts and their predictions averaged. `Learn_Rate`, `Batch_Size` and `Repeats` are fixed settings used for every model, including during tuning.

Tuning. `Lag_Tuning` and `LSTM_Tuning` are set independently:

- `"manual"`: the single values `Lag` and / or `Units`, `Epochs` are used as given.
- `"auto"`: the parameters are tuned by grid search. Each search range is given as `c(min, max, step)` and expands to `seq(min, max, by = step)`; for example `Lag_Range = c(1, 12, 1)` gives the lags 1 to 12 and `Units_Range = c(4, 12, 4)` gives 4, 8, 12 units. If `max` is not reached exactly by the steps, the last value is the largest one below `max`. Use `min = max` for a single value.

When both are `"auto"` every combination of lag and network parameters is evaluated. The `Epochs` values are scored within one training run per lag and number of units (the validation error is recorded after each requested epoch), so tuning `Epochs` adds almost no computing time. With `Verbose = TRUE` (default) the search prints the values of every parameter, the number of combinations, and a progress bar with the best score so far, the elapsed time and the estimated time remaining.

Validation. Candidates are compared on the last `Val_Split` share of the training set. With `"holdout"` the model is fitted on the observations before that block and evaluated on the block. With `"tscv"` the block is divided into `Folds` consecutive parts and each part is predicted from a model fitted on all earlier observations (expanding window). The clustering and scaling are re-estimated on each fitting window. After tuning, the model is refitted on the whole training set with the best parameters.

Output. With `Verbose = TRUE` a summary of the selected parameters and the accuracy is shown as a message at the end (silence it with `Verbose = FALSE` or `suppressMessages()`), and the result is returned invisibly; assign it (e.g. `Model <- IFLSTM(...)`) to keep it.

Reproducibility. The clustering, the initial weights and the order of the mini-batches are random, so call `set.seed()` before `IFLSTM()` to obtain reproducible results.

Value

A list (returned invisibly) with components

`Accuracy` Matrix of RMSE, MAE and MAPE (%) for the training and test sets.

`Train_Fitted` Fitted values for the training targets (the first `Lag` training observations have no fitted value).

`Test_Prediction` Predictions for the test set.

`Train_Actual`, `Test_Actual` The corresponding observed values.

`Best_Params` List with `Lag`, `Units` and `Epochs` of the final model.

`Architecture` Time steps, inputs per step, units, number of trainable parameters per network, learning rate, batch size and number of repeats.

`Loss` Training loss (mean squared error on the scaled target) after each epoch of the final model, averaged over the repeats.

`Tuning` `NULL` when nothing was tuned; otherwise a list with the validation scheme, metric, number of combinations, best validation score, elapsed time and `History` (a data frame with the score of every combination).

`Networks` List of the `Repeats` trained networks (weight matrices `W`, biases `b`, output weights `V` and output bias `c0`).

`Scaling` Minimum and maximum of the training data used for the [0, 1] scaling.

`IFCM` The intuitionistic fuzzy clustering of the training set (centres, membership, non-membership and hesitation degrees).

`Settings` Other settings used.

References

Yolcu, O.C. and Yolcu, U. (2023). A novel intuitionistic fuzzy time series prediction model with cascaded structure for financial time series. *Expert Systems with Applications*, 215, 119336. doi:10.1016/j.eswa.2022.119336

Chaira, T. (2011). A novel intuitionistic fuzzy C means clustering algorithm and its application to medical images. *Applied Soft Computing*, 11(2), 1711–1717.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735

Kingma, D.P. and Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Examples

```
set.seed(123)
dat <- 50 + 10 * sin(2 * pi * (1:120) / 12) + rnorm(120)

# Everything manual: lag 12, 8 units, 50 epochs
Model1 <- IFLSTM(dat, Class = 3, Split = 0.8,
                 Lag_Tuning = "manual", Lag = 12,
                 LSTM_Tuning = "manual", Units = 8, Epochs = 50)
Model1$Accuracy

# Manual lag, units and epochs tuned on a very small grid
Model2 <- IFLSTM(dat, Class = 3, Split = 0.8,
                 Lag_Tuning = "manual", Lag = 6,
                 Units_Range = c(4, 8, 4),        # 4, 8
                 Epochs_Range = c(20, 40, 20),    # 20, 40
                 Verbose = FALSE)
Model2$Best_Params


# Lag, units and epochs tuned jointly (small grid)
Model3 <- IFLSTM(dat, Class = 3, Split = 0.8,
```

```
                  Lag_Range = c(3, 12, 3),         # 3, 6, 9, 12
                  Units_Range = c(4, 8, 4),        # 4, 8
                  Epochs_Range = c(25, 50, 25),    # 25, 50
                  Verbose = FALSE)
head(Model3$Tuning$History)
plot(Model3$Loss, type = "l", xlab = "Epoch", ylab = "Training loss")
```

Index

IFLSTM, 2