Package 'ARInfoLSTM'

September 16, 2026

Type Package

Title ARIMA-Informed LSTM for Time Series Forecasting

Version 0.1.0

Description Implements an ARIMA-Informed Long Short-Term Memory (LSTM) framework for univariate time series forecasting. The package integrates statistical information extracted from AutoRegressive Integrated Moving Average (ARIMA) models with deep learning-based LSTM architectures to improve forecasting accuracy, stability, and interpretability. Inspired by the philosophy of Physics-Informed Machine Learning (PIML), the proposed framework incorporates information from classical statistical models into neural network learning, creating a hybrid forecasting approach that combines domain knowledge with data-driven intelligence. The methodology is motivated by hybrid forecasting framework proposed by Yeasin and Paul (2024) <doi:10.1007/s11227-023-05542-3>.

License GPL-3

Encoding UTF-8

Imports torch (>= 0.11.0), forecast (>= 8.21), ggplot2 (>= 3.4.0), cli (>= 3.6.0), coro, stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Md Yeasin [aut],
Ranjit Kumar Paul [aut, cre],
Pushkar Bora [aut]

Maintainer Ranjit Kumar Paul <ranjitstat@gmail.com>

Repository CRAN

Date/Publication 2026-09-16 11:40:02 UTC

Contents

`accuracy.AiL` *Accuracy Table for AiL Model*

Description

Accuracy Table for AiL Model

Usage

```
accuracy.AiL(x)
```

Arguments

`x`	An AiL object returned by AiL().

Value

Data frame with columns: Model, Lambda, RMSE, MAE, MAPE, SMAPE.

`AccuracyAiL` *Out-of-Sample Accuracy for AiL Forecasts*

Description

Compares AiL out-of-sample forecasts against actual values that arrived after training. Returns RMSE, MAE, SMAPE, and MAPE.

Usage

```
AccuracyAiL(fit, actual, lambda = NULL, verbose = TRUE)
```

Arguments

fit	An AiL object returned by AiL().
actual	Numeric vector of actual values observed after training. The function forecasts length(actual) steps and compares.
lambda	Lambda value to use. NULL = best lambda.
verbose	Logical. If TRUE (default), prints a formatted accuracy report to the console. Set to FALSE to suppress console output and only receive the returned data frame.

Value

Data frame with columns: Lambda, h, RMSE, MAE, SMAPE, MAPE. Also prints a formatted accuracy table to the console. The data frame has a "comparison" attribute with step-by-step Actual vs Forecast vs Error columns.

Examples

```
y   <- cumsum(rnorm(60, 0.5, 2))
fit <- AiL(y, mode = "manual", lag = 3, arima_order = c(1,1,1),
           hidden_size = 4, num_layers = 1, lr = 0.01,
           epochs = 20, patience = 5,
           lambda_list = c(0.5), verbose = FALSE)

# Simulate actual future values
new_data <- cumsum(rnorm(12, 0.5, 2)) + tail(y, 1)

acc <- AccuracyAiL(fit, actual = new_data)
print(acc)

# Step-by-step comparison table
attr(acc, "comparison")
```

AiL *ARIMA-Informed LSTM (AiL)*

Description

Trains an LSTM on the FULL dataset using the ARIMA-informed composite loss:

```
Loss = (1-lambda) * MSE(y_hat, y) + lambda * MSE(y_hat, y_ARIMA)
```

Trains on ALL data — no train/test split. Use predictAiL() for out-of-sample forecasting. An internal 20 percent validation split handles early stopping.

Usage

```
AiL(
  data,
  mode = c("auto", "manual"),
  arima_order  = NULL,      # NULL = auto.arima | c(p,d,q) = manual
  lag,                      # COMPULSORY
  hidden_size = 32,
  num_layers = 1,
  lr = 0.001,
  dropout = 0.0,
  epochs = 500,
  patience = 30,
  tune_epochs = 50,
  batch_size = 16,
  lambda_list = seq(0.1, 0.9, 0.1),
  hidden_grid = c(8, 16, 24, 32, 40),
  layers_grid = c(1, 2),
  lr_grid = c(0.001, 0.003, 0.005),
  dropout_grid = c(0.0, 0.1, 0.2),
  verbose = TRUE,
  seed = 42L
)
```

Arguments

`data`	Numeric vector. Full time series — ALL observations used for training.
`mode`	`"auto"` runs auto.arima + grid-search LSTM tuning. `"manual"` uses user-supplied parameters.
`arima_order`	Integer vector `c(p,d,q)`. NULL = auto.arima.
`lag`	Lag window (look-back). COMPULSORY — user must specify. No default value. Example: lag = 3 means the model uses the last 3 observations to predict the next value. arima_order: NULL = auto.arima; c(p,d,q) = user-specified order.
`hidden_size`	LSTM hidden units (manual mode). Default 32.
`num_layers`	Stacked LSTM layers (manual mode). Default 1.
`lr`	Learning rate (manual mode). Default 0.001.
`dropout`	Dropout rate (manual mode). Default 0.0.
`epochs`	Maximum training epochs. Default 500.
`patience`	Early-stopping patience. Default 30.
`tune_epochs`	Epochs per grid-search combo (auto). Default 50.
`batch_size`	Mini-batch size. Default 16.
`lambda_list`	Lambda values to test. Default seq(0.1, 0.9, 0.1).
`hidden_grid`	Hidden-unit grid (auto mode).
`layers_grid`	Layers grid (auto mode).

lr_grid	Learning-rate grid (auto mode).
dropout_grid	Dropout grid (auto mode).
verbose	Print progress. Default TRUE.
seed	Random seed. Default 42.

Value

S3 object of class "AiL" with elements:

results Named list per lambda. Each contains: tr_rmse, tr_mae, tr_mape, tr_smape, tr_preds, tr_actuals, val_rmse, loss_hist, time, lambda, model. Best lambda selected by Training RMSE.

best_model Complete summary of best lambda model containing: lambda, hyperparams, arima order, metrics (RMSE/MAE/MAPE/SMAPE), predictions (Actual vs Predicted data frame), loss_history, model object.

arima ARIMA order and fitted values on full data.

best_hp Best LSTM hyperparameters found.

tuning_log Grid-search log (auto mode only).

meta Settings: lag, lambda_list, mode, best_lam_nm, selection="Train RMSE".

data_info data, N, scaler, last_seq_scaled, lag.

Examples

```
y <- cumsum(rnorm(60, 0.5, 2))

# Auto mode -- train on full data (small grid for a fast example)
fit <- AiL(y, mode = "auto", lag = 3,
           hidden_grid = c(4, 8), layers_grid = c(1L),
           lr_grid = c(0.01), dropout_grid = c(0.0),
           tune_epochs = 5, epochs = 20, patience = 5,
           lambda_list = c(0.5), verbose = FALSE)
print(fit)
summary(fit)

# Forecast 12 steps ahead (out-of-sample)
fc <- predictAiL(fit, h = 12)
print(fc)

# Plot: historical + forecast (single point, no CI)
plotAiL(fit, h = 12)

# Lambda RMSE bar chart
plotlambda(fit)

# Manual mode (skips grid search -- fastest option)
fit2 <- AiL(y, mode = "manual",
            lag         = 3,            # compulsory
            arima_order = c(1,1,1),     # manual ARIMA order
            hidden_size = 4, num_layers = 1,
            lr = 0.01, epochs = 20, patience = 5,
```

```
            verbose = FALSE)
predictAiL(fit2, h = 6)

# If actual new data arrives -- accuracy check
new_data <- rnorm(12)
AccuracyAiL(fit, actual = new_data)
plotAiL(fit, h = 12, actual = new_data)
```

`plotAiL` *Plot AiL Forecast*

Description

Plots three things in one chart:

1. Actual full training data (dark line).
2. Model in-sample predictions on training data (blue line).
3. Out-of-sample h-step forecast (red solid line).

If actual future values are provided via `actual`, they are overlaid as a green line for direct comparison with the red forecast.

Usage

```
plotAiL(fit, h = 12, lambda = NULL, actual = NULL,
        newdata = NULL,n_hist = NULL)
```

Arguments

`fit`	An AiL object returned by AiL().
`h`	Forecast horizon (steps ahead). Default 12.
`lambda`	Lambda value to use. NULL = best lambda.
`actual`	Optional numeric vector of actual future values. If provided, overlaid as a green line for visual comparison.
`newdata`	Optional new observations to update starting sequence.
`n_hist`	Number of historical points to display. NULL = all.

Value

A ggplot2 object (invisibly).

Examples

```
y   <- cumsum(rnorm(60, 0.5, 2))
fit <- AiL(y, mode = "manual", lag = 3, arima_order = c(1,1,1),
           hidden_size = 4, num_layers = 1, lr = 0.01,
           epochs = 20, patience = 5,
           lambda_list = c(0.5), verbose = FALSE)

# Forecast only
plotAiL(fit, h = 12)

# Show only last 50 historical points
plotAiL(fit, h = 12, n_hist = 50)

# With actual future values overlaid (green)
new_data <- cumsum(rnorm(12, 0.5, 2)) + tail(y, 1)
plotAiL(fit, h = 12, actual = new_data)
```

`plotlambda` *Plot Training RMSE Across All Lambda Values*

Description

Bar chart showing Training RMSE for each lambda value tested. The best lambda (lowest Training RMSE) is highlighted in gold. A red dashed horizontal line marks the minimum Training RMSE. Best lambda is selected by Training RMSE (consistent with AiL()).

Usage

```
plotlambda(fit)
```

Arguments

`fit`	An AiL object returned by AiL().

Value

A ggplot2 object (invisibly).

Examples

```
y   <- cumsum(rnorm(60, 0.5, 2))
fit <- AiL(y, mode = "manual", lag = 3, arima_order = c(1,1,1),
           hidden_size = 4, num_layers = 1, lr = 0.01,
           epochs = 20, patience = 5,
           lambda_list = c(0.3, 0.7), verbose = FALSE)

plotlambda(fit)
# Gold bar  = best lambda (lowest Training RMSE)
```

```
# Red dashed = minimum Training RMSE line
```

---

predictAiL *Out-of-Sample Forecast from a Fitted AiL Model*

Description

Forecasts h steps ahead using recursive one-step prediction.

Usage

```
predictAiL(fit, h = 1, lambda = NULL, newdata = NULL)
```

Arguments

fit	An AiL object returned by AiL().
h	Number of steps to forecast ahead. Default 1.
lambda	Lambda value to use. NULL = best auto-selected lambda.
newdata	Optional numeric vector of new observations that arrived after training. If supplied, the forecast starting sequence is updated.

Value

Data frame with columns:

h Forecast horizon step (1, 2, ..., h).

time_idx Time index (N + 1, N + 2, ..., N + h).

forecast Point forecast (mean).

Examples

```
y   <- cumsum(rnorm(60, 0.5, 2))
fit <- AiL(y, mode = "manual", lag = 3, arima_order = c(1,1,1),
           hidden_size = 4, num_layers = 1, lr = 0.01,
           epochs = 20, patience = 5,
           lambda_list = c(0.3, 0.7), verbose = FALSE)

# Forecast 12 steps ahead with best lambda
fc <- predictAiL(fit, h = 12)
print(fc)

# Forecast with specific lambda
fc3 <- predictAiL(fit, h = 6, lambda = 0.3)

# Update starting sequence with new observations, then forecast
new_obs <- rnorm(5, mean = tail(y,1), sd = 2)
fc_new  <- predictAiL(fit, h = 12, newdata = new_obs)
```

`print.AiL_forecast` *Print method for AiL_forecast*

Description

Print method for AiL_forecast

Usage

```
## S3 method for class 'AiL_forecast'
print(x, ...)
```

Arguments

`x`	An AiL_forecast object returned by predictAiL().
`...`	Additional arguments passed to print.data.frame().

Value

Invisibly returns `x`.

Index