Package 'temper'

July 22, 2025

Type Package

Title Temporal Encoder-Masked Probabilistic Ensemble Regressor

Version 1.0.0

Maintainer Giancarlo Vercellino <giancarlo.vercellino@gmail.com>

Description Implements a probabilistic ensemble time-series forecaster that combines an auto-encoder with a neural decision forest whose split variables are learned through a differentiable feature-mask layer. Functions are written with 'torch' tensors and provide CRPS (Continuous Ranked Probability Scores) training plus mixture-distribution post-processing.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports torch (>= 0.11.0), purrr (>= 1.0.1), imputeTS (>= 3.3), lubridate (>= 1.9.2), ggplot2 (>= 3.5.1), scales (>= 1.3.0)

URL https://rpubs.com/giancarlo_vercellino/temper

Suggests knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

NeedsCompilation no

Author Giancarlo Vercellino [aut, cre, cph]

Repository CRAN

Date/Publication 2025-07-15 11:40:02 UTC

Contents

dummy_set *Tech Stock Time Series Dataset*

Description

A multivariate dataset for closing prices for several major tech stocks over time. Source: YahooFinance.

Usage

```
data(dummy_set)
```

Format

A data frame with 2133 observations of 4 variables:

dates Character vector of dates in "YYYY-MM-DD" format.

TSLA.Close Numeric. Closing prices for Tesla.

MSFT.Close Numeric. Closing prices for Microsoft.

MARA.Close Numeric. Closing prices for MARA Holdings.

Examples

```
data(dummy_set)
plot(as.Date(dummy_set$dates), dummy_set$TSLA.Close, type = "l")
```

temper *Temporal Encoder–Masked Probabilistic Ensemble Regressor*

Description

Temper trains and deploys a hybrid forecasting model that couples a temporal auto-encoder (shrinks a sliding window of length ‘past‘ into a latent representation of size ‘latent_dim‘) and a masked neural decision forest (an ensemble of ‘n_trees‘ soft decision trees of depth ‘depth‘; feature-level dropout is governed by ‘init_prob‘ and annealed by a Gumbel–Softmax with parameter ‘temperature‘) and a CRPS loss (Continuous Ranked Probability Score) that blends the probabilistic forecasting error with a reconstruction term (‘lambda_rec × MSE‘), to yield multi-step probabilistic forecasts and their fan chart. Model weights are optimized with ADAM or other options, optional early stopping.

Implements a probabilistic ensemble time-series forecaster that combines an auto-encoder with a neural decision forest whose split variables are learned through a differentiable feature-mask layer. Functions are written with 'torch' tensors and provide CRPS (Continuous Ranked Probability Scores) training plus mixture-distribution post-processing.

Usage

```
temper(
  ts,
  future,
  past,
  latent_dim,
  n_trees = 30,
  depth = 6,
  init_prob = 0.8,
  temperature = 0.5,
  n_bases = 10,
  train_rate = 0.7,
  epochs = 30,
  optimizer = "adam",
  lr = 0.005,
  batch = 32,
  lambda_rec = 0.3,
  patience = 15,
  verbose = TRUE,
  alpha = 0.1,
  dates = NULL,
  seed = 42
)
```

Arguments

ts	Numeric vector of length at least past + future. Represents the input time series in levels (not log-returns). Missing values are automatically imputed using na_kalman.
future	Integer ≥ 1. Forecast horizon: the number of steps ahead to predict.
past	Integer ≥ 1. Length of the sliding window used to feed the encoder.
latent_dim	Integer ≥ 1. Dimensionality of the autoencoder's latent bottleneck.
n_trees	Integer ≥ 1. Number of trees in the neural decision forest ensemble. Usually in the range of 30 to 200. Default: 30.
depth	Integer ≥ 1. Depth of each decision tree (i.e., number of binary splits). Usually in the range of 4 to 12. Default: 6.
init_prob	Numeric in $(0, 1)$. Initial probability that each input feature is kept by the feature mask (used for stochastic feature selection). A value of 0 means always dropped; 1 means always included. Default: 0.8.
temperature	Positive numeric. Temperature parameter for the Gumbel–Softmax distribution used during feature masking. Lower values lead to harder (closer to binary) masks; higher values encourage smoother gradients. Default: 0.5.
n_bases	Integer ≥ 1. Max numbers of bases for the Gaussian mixture. Default: 10.
train_rate	Numeric in $(0, 1)$. Proportion of samples allocated to the training set. The remaining samples form the validation set used for early stopping. Default: 0.7.

`epochs`	Positive integer. Maximum number of training epochs. Have a look at the loss plot to decide the right number of epochs. Default: 30.
`optimizer`	Character string. Optimizer to use for training (adam, adamw, sgd, rprop, rmsprop, adagrad, asgd, adadelta). Default: adam.
`lr`	Positive numeric. Learning rate for the optimizer. Default: 0.005.
`batch`	Positive integer. Mini-batch size used during training. Default: 32.
`lambda_rec`	Non-negative numeric. Weight applied to the reconstruction loss relative to the probabilistic CRPS forecasting loss. Default: 0.3.
`patience`	Positive integer. Number of consecutive epochs without improvement on the validation CRPS before early stopping is triggered. Default: 15.
`verbose`	Logical. If `TRUE`, prints CRPS values for each epoch during training. Default: TRUE.
`alpha`	Numeric in $(0, 1)$. Confidence level used to define the predictive interval band width in the output fan chart. Default: 0.1.
`dates`	Optional `Date` vector of the same length as ts. If supplied, fan chart x-axes use calendar dates; otherwise, integer time indices are used. Default: NULL.
`seed`	Optional integer. Used to seed both R and Torch random number generators for reproducibility. Default: 42.

Value

A named list with four components

‘loss‘ A ggplot in which training and validation CRPS are plotted against epoch number, useful for diagnosing over-/under-fitting.

‘pred_funs‘ A length-‘future‘ list. Each element contains four empirical distribution functions (pdf, cdf, icdf, sampler) created by empfun

‘plot‘ A ggplot object showing the historical series, median forecast and predictive interval. A print-ready fan chart.

‘time_log‘ An object measuring the wall-clock training time.

Author(s)

Maintainer: Giancarlo Vercellino <giancarlo.vercellino@gmail.com> [copyright holder]

See Also

Useful links:

- https://rpubs.com/giancarlo_vercellino/temper

Examples

```
set.seed(2025)
ts <- cumsum(rnorm(250))          # synthetic price series
fit <- temper(ts, future = 3, past = 20, latent_dim = 5, epochs = 2)

# 80 % predictive interval for the 3-step-ahead forecast
pfun <- fit$pred_funs$t3$pfun
pred_interval_80 <- c(pfun(0.1), pfun(0.9))

# Visual diagnostics
print(fit$plot)
print(fit$loss)
```

Index