Package ‘modeltime.gluonts’

November 30, 2020

Type Package

Title 'GluonTS' Deep Learning

Version 0.1.0

Description Use the 'GluonTS' deep learning library inside of 'modeltime'. Available models include 'DeepAR', 'N-BEATS', and 'N-BEATS' Ensemble. Refer to "GluonTS - Probabilistic Time Series Modeling" (<https://ts.gluon.ai/index.html>).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends modeltime (>= 0.3.1)

Imports parsnip, timetk, magrittr, rlang (>= 0.1.2), reticulate, tibble, forcats, dplyr, tidyr, purrr, stringr, glue, fs

Suggests tidyverse, tidymodels, knitr, rmarkdown, roxygen2, testthat

VignetteBuilder knitr

URL https://github.com/business-science/modeltime.gluonts

BugReports https://github.com/business-science/modeltime.gluonts/issues

NeedsCompilation no

Author Matt Dancho [aut, cre], Business Science [cph]

Maintainer Matt Dancho <mdancho@business-science.io>

Repository CRAN

Date/Publication 2020-11-30 09:40:02 UTC

R topics documented:

  as_pandas_timestamp .................................................. 2
  deepar_fit_impl ..................................................... 3
as_pandas_timestamp

Convert R Date or POSIXt to Pandas Timestamp

Description

Convert R Date or POSIXt to Pandas Timestamp

Usage

as_pandas_timestamp(x, ..., pass_time_zone = FALSE)

Arguments

x A Date or Date Time

... Additional parameters passed to Pandas Timestamp

pass_time_zone Whether or not to include the time zone in the conversion to Pandas. GluonTS does not work with Pandas Time Zones. Default: FALSE.

Examples

dt <- as.Date("2011-01-01")
as_pandas_timestamp(dt)

dt_time <- as.POSIXct("2011-01-01 12:43:01", tz = "CMT")
as_pandas_timestamp(dt_time, pass_time_zone = TRUE)
GluonTS DeepAR Modeling Function (Bridge)

Usage

deepar_fit_impl(
x, y, freq, prediction_length, id, epochs = 5, batch_size = 32, num_batches_per_epoch = 50, learning_rate = 0.001, learning_rate_decay_factor = 0.5, patience = 10, minimum_learning_rate = 5e-05, clip_gradient = 10, weight_decay = 1e-08, init = "xavier", ctx = NULL, hybridize = TRUE, context_length = NULL, num_layers = 2, num_cells = 40, cell_type = "lstm", dropout_rate = 0.1, use_feat_dynamic_real = FALSE, use_feat_static_cat = FALSE, use_feat_static_real = FALSE, cardinality = NULL, embedding_dimension = NULL, distr_output = "default", scaling = TRUE, lags_seq = NULL, time_features = NULL, num_parallel_samples = 100
)

Arguments

x A dataframe of xreg (exogenous regressors)
**deepar_fit_impl**

- `y`: A numeric vector of values to fit
- `freq`: A pandas timeseries frequency such as "5min" for 5-minutes or "D" for daily. Refer to Pandas Offset Aliases.
- `prediction_length`: Numeric value indicating the length of the prediction horizon
- `id`: A quoted column name that tracks the GluonTS FieldName "item_id"
- `epochs`: Number of epochs that the network will train (default: 5).
- `batch_size`: Number of examples in each batch (default: 32).
- `num_batches_per_epoch`: Number of batches at each epoch (default: 50).
- `learning_rate`: Initial learning rate (default: 10^-3).
- `learning_rate_decay_factor`: Factor (between 0 and 1) by which to decrease the learning rate (default: 0.5).
- `patience`: The patience to observe before reducing the learning rate, nonnegative integer (default: 10).
- `minimum_learning_rate`: Lower bound for the learning rate (default: 5x10^-5).
- `clip_gradient`: Maximum value of gradient. The gradient is clipped if it is too large (default: 10).
- `weight_decay`: The weight decay (or L2 regularization) coefficient. Modifies objective by adding a penalty for having large weights (default 10^-8).
- `init`: Initializer of the weights of the network (default: “xavier”).
- `ctx`: The mxnet CPU/GPU context. Refer to using CPU/GPU in the mxnet documentation. (default: NULL, uses CPU)
- `hybridize`: Increases efficiency by using symbolic programming. (default: TRUE)
- `context_length`: Number of steps to unroll the RNN for before computing predictions (default: NULL, in which case context_length = prediction_length)
- `num_layers`: Number of RNN layers (default: 2)
- `num_cells`: Number of RNN cells for each layer (default: 40)
- `cell_type`: Type of recurrent cells to use (available: ‘lstm’ or ‘gru’; default: ‘lstm’)
- `dropout_rate`: Dropout regularization parameter (default: 0.1)
- `use_feat_dynamic_real`: Whether to use the ‘feat_dynamic_real’ field from the data (default: FALSE)
- `use_feat_static_cat`: Whether to use the feat_static_cat field from the data (default: FALSE)
- `use_feat_static_real`: Whether to use the feat_static_real field from the data (default: FALSE)
- `cardinality`: Number of values of each categorical feature. This must be set if use_feat_static_cat == TRUE (default: NULL)
- `embedding_dimension`: Dimension of the embeddings for categorical features (default: min(50, (cat+1)//2) for cat in cardinality)
### deepar_predict_impl

**Bridge prediction Function for DeepAR Models**

**Description**

Bridge prediction Function for DeepAR Models

**Usage**

```r
deeper_predict_impl(object, new_data)
```

**Arguments**

- `object`: An object of class `model_fit`
- `new_data`: A rectangular data object, such as a data frame.

### deep_ar

**General Interface for DeepAR Time Series Models**

**Description**

deep_ar() is a way to generate a specification of a DeepAR model before fitting and allows the model to be created using different packages. Currently the only package is gluonts.
Usage

depth_ar(
    mode = "regression",
    id,
    freq,
    prediction_length,
    lookback_length = NULL,
    cell_type = NULL,
    num_layers = NULL,
    num_cells = NULL,
    dropout = NULL,
    epochs = NULL,
    batch_size = NULL,
    num_batches_per_epoch = NULL,
    learn_rate = NULL,
    learn_rate_decay_factor = NULL,
    learn_rate_min = NULL,
    patience = NULL,
    clip_gradient = NULL,
    penalty = NULL
)

Arguments

mode A single character string for the type of model. The only possible value for this model is "regression".
id A quoted column name that tracks the GluonTS FieldName "item_id"
freq A pandas timeseries frequency such as "5min" for 5-minutes or "D" for daily. Refer to Pandas Offset Aliases.
prediction_length Numeric value indicating the length of the prediction horizon
lookback_length Number of steps to unroll the RNN for before computing predictions (default: NULL, in which case context_length = prediction_length)
cell_type Type of recurrent cells to use (available: 'lstm' or 'gru'; default: 'lstm')
num_layers Number of RNN layers (default: 2)
num_cells Number of RNN cells for each layer (default: 40)
dropout Dropout regularization parameter (default: 0.1)
epochs Number of epochs that the network will train (default: 5).
batch_size Number of examples in each batch (default: 32).
num_batches_per_epoch Number of batches at each epoch (default: 50).
learn_rate Initial learning rate (default: 10-3).
learn_rate_decay_factor Factor (between 0 and 1) by which to decrease the learning rate (default: 0.5).
Details

These arguments are converted to their specific names at the time that the model is fit. Other options and arguments can be set using set_engine(). If left to their defaults here (see above), the values are taken from the underlying model functions. If parameters need to be modified, update() can be used in lieu of recreating the object from scratch.

The model can be created using the fit() function using the following engines:

- GluonTS DeepAR: "gluonts_deepar" (the default)

Engine Details

The standardized parameter names in modeltime can be mapped to their original names in each engine:

<table>
<thead>
<tr>
<th>modeltime</th>
<th>DeepAREstimater</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>NA</td>
</tr>
<tr>
<td>freq</td>
<td>freq</td>
</tr>
<tr>
<td>prediction_length</td>
<td>prediction_length</td>
</tr>
<tr>
<td>lookback_length</td>
<td>context_length (= prediction_length)</td>
</tr>
<tr>
<td>epochs</td>
<td>epochs (5)</td>
</tr>
<tr>
<td>batch_size</td>
<td>batch_size (32)</td>
</tr>
<tr>
<td>num_batches_per_epoch</td>
<td>num_batches_per_epoch (50)</td>
</tr>
<tr>
<td>learn_rate</td>
<td>learning_rate (0.001)</td>
</tr>
<tr>
<td>learn_rate_decay_factor</td>
<td>learning_rate_decay_factor (0.5)</td>
</tr>
<tr>
<td>learn_rate_min</td>
<td>minimum_learning_rate (5e-5)</td>
</tr>
<tr>
<td>patience</td>
<td>patience (10)</td>
</tr>
<tr>
<td>clip_gradient</td>
<td>clip_gradient (10)</td>
</tr>
<tr>
<td>penalty</td>
<td>weight_decay (1e-8)</td>
</tr>
<tr>
<td>cell_type</td>
<td>cell_type ('lstm')</td>
</tr>
<tr>
<td>num_layers</td>
<td>num_layers (2)</td>
</tr>
<tr>
<td>num_cells</td>
<td>num_cells (40)</td>
</tr>
<tr>
<td>dropout</td>
<td>dropout_rate (0.1)</td>
</tr>
</tbody>
</table>

Other options can be set using set_engine().

Engine

gluonts_deepar
The engine uses `gluonts.model.deepar.DeepAREstimator()`. Default values that have been changed to prevent long-running computations:

- `epochs = 5`: GluonTS uses 100 by default.

**Required Parameters**

The `gluonts` implementation has several *Required Parameters*, which are user-defined.

1. **ID Variable (Required):**

   An important difference between other parsnip models is that each time series (even single time series) must be uniquely identified by an ID variable.

   - The ID feature must be of class `character` or `factor`.
   - This ID feature is provided as a quoted expression during the model specification process (e.g. `deep_ar(id = "ID")` assuming you have a column in your data named "ID").

2. **Frequency (Required):**

   The GluonTS models use a Pandas Timestamp Frequency `freq` to generate features internally. Examples:

   - `freq = "5min"` for timestamps that are 5-minutes apart
   - `freq = "D"` for Daily Timestamps

   The Pandas Timestamps are quite flexible. Refer to Pandas Offset Aliases.

3. **Prediction Length (Required):**

   Unlike other parsnip models, a `prediction_length` is required during the model specification and fitting process.

**Fit Details**

The following features are REQUIRED to be available in the incoming data for the fitting process.

- **Fit**: `fit(y ~ date + id, data)` includes a target feature that is a function of a "date" and "id" feature. The ID feature must be pre-specified in the model specification.

- **Predict**: `predict(model, new_data)` where `new_data` contains both a column named "date" and "id".

**ID Variable**

An ID feature must be included in the recipe or formula fitting process. This assists with cataloging the time series inside GluonTS ListDataset. The column name must match the quoted feature name specified in the `deep_ar(id = "id")` expects a column inside your data named "id".

**Date and Date-Time Variable**

It's a requirement to have a date or date-time variable as a predictor. The `fit()` interface accepts date and date-time features and handles them internally.

**See Also**

`fit.model_spec()`, `set_engine()`
Examples

library(tidymodels)
library(tidyverse)
library(timetk)

# ---- MODEL SPEC ----
# - Important: Make sure *required* parameters are provided
model_spec <- deep_ar(
    # User Defined (Required) Parameters
    id = "id",
    freq = "M",
    prediction_length = 24,

    # Hyper Parameters
    epochs = 1,
    num_batches_per_epoch = 4
) %>%
  set_engine("gluonts_deepar")

model_spec

# ---- TRAINING ----
# Important: Make sure the date and id features are included as regressors
# and do NOT dummy the id feature.
model_fitted <- model_spec %>%
  fit(value ~ date + id, m750)

model_fitted

# ---- PREDICT ----
# - IMPORTANT: New Data must have id and date features
new_data <- tibble(
    id = factor("M750"),
    date = as.Date("2015-07-01")
)

predict(model_fitted, new_data)

install_gluonts

Install Gluonts

Description

Installs Gluonts Probabilistic Deep Learning Time Series Forecasting Software using reticulate::py_install().
- A Python Environment will be created named `r-gluonts`.
- The Modletime GluonTS R package will connect to the `r-gluonts` Python environment to use GluonTS

### Usage

```r
install_gluonts()
```

### Examples

```r
## Not run:
install_gluonts()
## End(Not run)
```

---

**nbeats**

*General Interface for N-BEATS Time Series Models*

### Description

`nbeats()` is a way to generate a *specification* of a N-BEATS model before fitting and allows the model to be created using different packages. Currently the only package is `gluonts`. There are 2 N-Beats implementations: (1) Standard N-Beats, and (2) Ensemble N-Beats.

### Usage

```r
nbeats(
  mode = "regression",
  id,
  freq,
  prediction_length,
  lookback_length = NULL,
  loss_function = NULL,
  bagging_size = NULL,
  num_stacks = NULL,
  num_blocks = NULL,
  epochs = NULL,
  batch_size = NULL,
  num_batches_per_epoch = NULL,
  learn_rate = NULL,
  learn_rate_decay_factor = NULL,
  learn_rate_min = NULL,
  patience = NULL,
  clip_gradient = NULL,
  penalty = NULL
)
```
Arguments

mode
- A single character string for the type of model. The only possible value for this model is "regression".

id
- A quoted column name that tracks the GluonTS FieldName "item_id"

freq
- A pandas timeseries frequency such as "5min" for 5-minutes or "D" for daily. Refer to Pandas Offset Aliases.

prediction_length
- Numeric value indicating the length of the prediction horizon

lookback_length
- Number of time units that condition the predictions. Also known as 'lookback period'. Default is 2 * prediction_length.

loss_function
- The loss function (also known as metric) to use for training the network. Unlike other models in GluonTS this network does not use a distribution. One of the following: "sMAPE", "MASE" or "MAPE". The default value is "MAPE".

bagging_size
- (Applicable to Ensemble N-Beats). The number of models that share the parameter combination of 'context_length' and 'loss_function'. Each of these models gets a different initialization random initialization. Default and recommended value: 10.

num_stacks
- The number of stacks the network should contain. Default and recommended value for generic mode: 30 Recommended value for interpretable mode: 2

num_blocks
- The number of blocks per stack. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: 1. Recommended value for interpretable mode: 3.

epochs
- Number of epochs that the network will train (default: 5).

batch_size
- Number of examples in each batch (default: 32).

num_batches_per_epoch
- Number of batches at each epoch (default: 50).

learn_rate
- Initial learning rate (default: 10^-3).

learn_rate_decay_factor
- Factor (between 0 and 1) by which to decrease the learning rate (default: 0.5).

learn_rate_min
- Lower bound for the learning rate (default: 5x10^-5).

patience
- The patience to observe before reducing the learning rate, nonnegative integer (default: 10).

clip_gradient
- Maximum value of gradient. The gradient is clipped if it is too large (default: 10).

penalty
- The weight decay (or L2 regularization) coefficient. Modifies objective by adding a penalty for having large weights (default: 10^-8).

Details

These arguments are converted to their specific names at the time that the model is fit. Other options and arguments can be set using set_engine(). If left to their defaults here (see above), the values are taken from the underlying model functions. If parameters need to be modified, update() can be used in lieu of recreating the object from scratch.

The model can be created using the fit() function using the following engines:
- **GluonTS N-BEATS**: "gluonts_nbeats" (the default)
- **GluonTS N-BEATS Ensemble**: "gluonts_nbeats_ensemble"

**Engine Details**

The standardized parameter names in `modeltime` can be mapped to their original names in each engine:

<table>
<thead>
<tr>
<th>modeltime</th>
<th>NBEATSEstimator</th>
<th>NBEATSEnsembleEstimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ListDataset('item_id')</td>
<td>ListDataset('item_id')</td>
</tr>
<tr>
<td>freq</td>
<td>freq</td>
<td>freq</td>
</tr>
<tr>
<td>prediction_length</td>
<td>prediction_length</td>
<td>prediction_length</td>
</tr>
<tr>
<td>lookahead_length</td>
<td>context_length (= 2 x prediction_length)</td>
<td>meta_context_length (= prediction_length x c(2,4))</td>
</tr>
<tr>
<td>bagging_size</td>
<td>NA</td>
<td>meta_bagging_size (3)</td>
</tr>
<tr>
<td>loss_function</td>
<td>loss_function (‘sMAPE’)</td>
<td>meta_loss_function (list(‘sMAPE’))</td>
</tr>
<tr>
<td>num_stacks</td>
<td>num_stacks (30)</td>
<td>num_stacks (30)</td>
</tr>
<tr>
<td>num_blocks</td>
<td>num_blocks (list(1))</td>
<td>num_blocks (list(1))</td>
</tr>
<tr>
<td>epochs</td>
<td>epochs (5)</td>
<td>epochs (5)</td>
</tr>
<tr>
<td>batch_size</td>
<td>batch_size (32)</td>
<td>batch_size (32)</td>
</tr>
<tr>
<td>num_batches_per_epoch</td>
<td>num_batches_per_epoch (50)</td>
<td>num_batches_per_epoch (50)</td>
</tr>
<tr>
<td>learn_rate</td>
<td>learning_rate (0.001)</td>
<td>learning_rate (0.001)</td>
</tr>
<tr>
<td>learn_rate_decay_factor</td>
<td>learning_rate_decay_factor (0.5)</td>
<td>learning_rate_decay_factor (0.5)</td>
</tr>
<tr>
<td>learn_rate_min</td>
<td>minimum_learning_rate (5e-5)</td>
<td>minimum_learning_rate (5e-5)</td>
</tr>
<tr>
<td>patience</td>
<td>patience (10)</td>
<td>patience (10)</td>
</tr>
<tr>
<td>clip_gradient</td>
<td>clip_gradient (10)</td>
<td>clip_gradient (10)</td>
</tr>
<tr>
<td>penalty</td>
<td>weight_decay (1e-8)</td>
<td>weight_decay (1e-8)</td>
</tr>
</tbody>
</table>

Other options can be set using `set_engine()`.

**Engine**

gluonts_nbeats

The engine uses `gluonts.model.n_beats.NBEATSEstimator()`. Default values that have been changed to prevent long-running computations:

- **epochs = 5**: GluonTS uses 100 by default.
- **loss_function = 'sMAPE'**: GluonTS by default uses MAPE. MAPE can suffer from issues with small values.

**Required Parameters**

The `gluonts_nbeats` implementation has several **Required Parameters**, which are user-defined.

1. **ID Variable (Required):**

An important difference between other parsnip models is that each time series (even single time series) must be uniquely identified by an ID variable.

- The ID feature must be of class character or factor.
• This ID feature is provided as a quoted expression during the model specification process (e.g. `nbeats(id = "ID")` assuming you have a column in your data named "ID").

2. Frequency (Required):
The GluonTS models use a Pandas Timestamp Frequency `freq` to generate features internally. Examples:
- `freq = "5min"` for timestamps that are 5-minutes apart
- `freq = "D"` for Daily Timestamps

The Pandas Timestamps are quite flexible. Refer to Pandas Offset Aliases.

3. Prediction Length (Required):
Unlike other parsnip models, a `prediction_length` is required during the model specification and fitting process.

`gluonts_nbeats_ensemble`

The engine uses `gluonts.model.n_beats.NBEATSEnsembleEstimator()`.

Number of Models Created
This model is very good, but can be expensive (long-running) due to the number of models that are being created. The number of models follows the formula: length(lookback_length) x length(loss_function) x meta_bagging_size

The default values that have been changed from GluonTS implementation to prevent long-running computations:
- `epochs = 5`: GluonTS uses 100 by default.
- `lookback_length = prediction_length * c(2,4)`: GluonTS uses range of 2:7, which doubles the number of models created.
- `bagging_size = 3`: Averages 5 like models together. GluonTS uses 10, which doubles the number of models created.
- `loss_function = 'sMAPE'`: GluonTS uses 3 `meta_loss_function = list('sMAPE', 'MASE', 'MAPE')`, which 3X’s (triples) the number of models created.

The result is: 2 x 1 x 3 = 6 models. Each model will have 5 epochs by default.

Required Parameters
The `gluonts_nbeats_ensemble` implementation has several Required Parameters, which are user-defined.

1. ID Variable (Required):
An important difference between other parsnip models is that each time series (even single time series) must be uniquely identified by an ID variable.

• The ID feature must be of class character or factor.
• This ID feature is provided as a quoted expression during the model specification process (e.g. `nbeats(id = "ID")` assuming you have a column in your data named "ID").

2. Frequency (Required):
The GluonTS models use a Pandas Timestamp Frequency `freq` to generate features internally. Examples:
• freq = "5min" for timestamps that are 5-minutes apart
• freq = "D" for Daily Timestamps

The Pandas Timestamps are quite flexible. Refer to Pandas Offset Aliases.

3. Prediction Length (Required):

Unlike other parsnip models, a prediction_length is required during the model specification and fitting process.

Fit Details

The following features are REQUIRED to be available in the incoming data for the fitting process.

• Fit: fit(y ~ date + id, data): Includes a target feature that is a function of a "date" and "id" feature. The ID feature must be pre-specified in the model_specification.

• Predict: predict(model, new_data) where new_data contains both a column named "date" and "id".

ID Variable

An ID feature must be included in the recipe or formula fitting process. This assists with cataloging the time series inside GluonTS ListDataset. The column name must match the quoted feature name specified in the nbeats(id = "id") expects a column inside your data named "id".

Date and Date-Time Variable

It's a requirement to have a date or date-time variable as a predictor. The fit() interface accepts date and date-time features and handles them internally.

See Also

fit.model_spec(), set_engine()

Examples

library(tidymodels)
library(tidyverse)
library(timetk)

# ---- MODEL SPEC ----
# - Important: Make sure *required* parameters are provided
model_spec <- nbeats(

  # User Defined (Required) Parameters
  id = "id",
  freq = "M",
  prediction_length = 24,

  # Hyper Parameters
  epochs = 1,
  num_batches_per_epoch = 4
)

%>%
```r
set_engine("gluonts_nbeats")

model_spec

# ---- TRAINING ----
# Important: Make sure the date and id features are included as regressors
# and do NOT dummy the id feature.
model_fitted <- model_spec %>%
  fit(value ~ date + id, m750)

model_fitted

# ---- PREDICT ----
# - IMPORTANT: New Data must have id and date features
new_data <- tibble(
  id = factor("M750"),
  date = as.Date("2015-07-01")
)

predict(model_fitted, new_data)
```

---

### `nbeats_ensemble_fit_impl`

**GluonTS N-BEATS ENSEMBLE Modeling Function (Bridge)**

**Description**

GluonTS N-BEATS ENSEMBLE Modeling Function (Bridge)

**Usage**

```r
nbeats_ensemble_fit_impl(
  x,
  y,
  freq,
  prediction_length,
  id,
  epochs = 5,
  batch_size = 32,
  num_batches_per_epoch = 50,
  learning_rate = 0.001,
  learning_rate_decay_factor = 0.5,
  patience = 10,
  minimum_learning_rate = 5e-05,
  clip_gradient = 10,
  weight_decay = 1e-08,
  init = "xavier",
)```
ctx = NULL,
hybridize = TRUE,
meta_context_length = prediction_length * c(2, 4),
meta_loss_function = list("sMAPE"),
meta_bagging_size = 3,
num_stacks = 30,
num_blocks = list(1),
widths = list(512),
sharing = list(FALSE),
expansion_coefficient_lengths = list(32),
stack_types = list("G")
)

**Arguments**

- **x**: A dataframe of xreg (exogenous regressors)
- **y**: A numeric vector of values to fit
- **freq**: A `pandas` timeseries frequency such as "5min" for 5-minutes or "D" for daily. Refer to Pandas Offset Aliases.
- **prediction_length**: Numeric value indicating the length of the prediction horizon
- **id**: A quoted column name that tracks the GluonTS FieldName "item_id"
- **epochs**: Number of epochs that the network will train (default: 5).
- **batch_size**: Number of examples in each batch (default: 32).
- **num_batches_per_epoch**: Number of batches at each epoch (default: 50).
- **learning_rate**: Initial learning rate (default: 10^-3).
- **learning_rate_decay_factor**: Factor (between 0 and 1) by which to decrease the learning rate (default: 0.5).
- **patience**: The patience to observe before reducing the learning rate, nonnegative integer (default: 10).
- **minimum_learning_rate**: Lower bound for the learning rate (default: 5x10^-5).
- **clip_gradient**: Maximum value of gradient. The gradient is clipped if it is too large (default: 10).
- **weight_decay**: The weight decay (or L2 regularization) coefficient. Modifies objective by adding a penalty for having large weights (default 10^-8).
- **init**: Initializer of the weights of the network (default: “xavier”).
- **ctx**: The mxnet CPU/GPU context. Refer to using CPU/GPU in the mxnet documentation. (default: NULL, uses CPU)
- **hybridize**: Increases efficiency by using symbolic programming. (default: TRUE)
- **meta_context_length**: The different 'context_length' (also known as 'lookback period') to use for training the models. The 'context_length' is the number of time units that condition the predictions. Default and recommended value: list(multiplier * prediction_length for multiplier in range(2, 7))
**meta_loss_function**

The different 'loss_function' (also known as metric) to use for training the models. Unlike other models in GluonTS this network does not use a distribution. Default and recommended value: list("sMAPE","MASE","MAPE")

**meta_bagging_size**

The number of models that share the parameter combination of 'context_length' and 'loss_function'. Each of these models gets a different initialization random initialization. Default (3). Recommended value: 10

**num_stacks**

The number of stacks the network should contain. Default and recommended value for generic mode: 30 Recommended value for interpretable mode: 2

**num_blocks**

The number of blocks per stack. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: 1. Recommended value for interpretable mode: 3.

**widths**

Widths of the fully connected layers with ReLu activation in the blocks. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: list(512) Recommended value for interpretable mode: list(256,2048)

**sharing**

Whether the weights are shared with the other blocks per stack. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: list(FALSE) Recommended value for interpretable mode: list(TRUE)

**expansion_coefficient_lengths**

If the type is "G" (generic), then the length of the expansion coefficient. If type is "T" (trend), then it corresponds to the degree of the polynomial. If the type is "S" (seasonal) then its not used. A list of ints of length 1 or 'num_stacks'. Default value for generic mode: list(32) Recommended value for interpretable mode: list(3)

**stack_types**

One of the following values: "G" (generic), "S" (seasonal) or "T" (trend). A list of strings of length 1 or 'num_stacks'. Default and recommended value for generic mode: list("G") Recommended value for interpretable mode: list("T","S")

**Details**

The total number of models used is:

meta_context_length x meta_loss_function x meta_bagging_size

---

nbeats_ensemble_predict_impl

*Bridge prediction Function for N-BEATS ENSEMBLE Models*

**Description**

Bridge prediction Function for N-BEATS ENSEMBLE Models

**Usage**

nbeats_ensemble_predict_impl(object, new_data)
**Arguments**

- `object` An object of class `model_fit`
- `new_data` A rectangular data object, such as a data frame.

**Description**

GluonTS N-BEATS Modeling Function (Bridge)

**Usage**

```r
nbeats_fit_impl(
  x, y, freq, prediction_length, id, epochs = 5, batch_size = 32, num_batches_per_epoch = 50, learning_rate = 0.001, learning_rate_decay_factor = 0.5, patience = 10, minimum_learning_rate = 5e-05, clip_gradient = 10, weight_decay = 1e-08, init = "xavier", ctx = NULL, hybridize = TRUE, context_length = NULL, loss_function = "sMAPE", num_stacks = 30, num_blocks = list(1), widths = list(512), sharing = list(FALSE), expansion_coefficient_lengths = list(32), stack_types = list("G")
)
```

**Arguments**

- `x` A dataframe of xreg (exogenous regressors)
- `y` A numeric vector of values to fit
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>A pandas timeseries frequency such as &quot;5min&quot; for 5-minutes or &quot;D&quot; for daily. Refer to Pandas Offset Aliases.</td>
</tr>
<tr>
<td>prediction_length</td>
<td>Numeric value indicating the length of the prediction horizon</td>
</tr>
<tr>
<td>id</td>
<td>A quoted column name that tracks the GluonTS FieldName &quot;item_id&quot;</td>
</tr>
<tr>
<td>epochs</td>
<td>Number of epochs that the network will train (default: 5).</td>
</tr>
<tr>
<td>batch_size</td>
<td>Number of examples in each batch (default: 32).</td>
</tr>
<tr>
<td>num_batches_per_epoch</td>
<td>Number of batches at each epoch (default: 50).</td>
</tr>
<tr>
<td>learning_rate</td>
<td>Initial learning rate (default: 10^-3 ).</td>
</tr>
<tr>
<td>learning_rate_decay_factor</td>
<td>Factor (between 0 and 1) by which to decrease the learning rate (default: 0.5).</td>
</tr>
<tr>
<td>patience</td>
<td>The patience to observe before reducing the learning rate, nonnegative integer (default: 10).</td>
</tr>
<tr>
<td>minimum_learning_rate</td>
<td>Lower bound for the learning rate (default: 5x10^-5 ).</td>
</tr>
<tr>
<td>clip_gradient</td>
<td>Maximum value of gradient. The gradient is clipped if it is too large (default: 10).</td>
</tr>
<tr>
<td>weight_decay</td>
<td>The weight decay (or L2 regularization) coefficient. Modifies objective by adding a penalty for having large weights (default 10^-8).</td>
</tr>
<tr>
<td>init</td>
<td>Initializer of the weights of the network (default: “xavier”).</td>
</tr>
<tr>
<td>ctx</td>
<td>The mxnet CPU/GPU context. Refer to using CPU/GPU in the mxnet documentation. (default: NULL, uses CPU)</td>
</tr>
<tr>
<td>hybridize</td>
<td>Increases efficiency by using symbolic programming. (default: TRUE)</td>
</tr>
<tr>
<td>context_length</td>
<td>Number of time units that condition the predictions Also known as 'lookback period'. Default is 2 * prediction_length</td>
</tr>
<tr>
<td>loss_function</td>
<td>The loss function (also known as metric) to use for training the network. Unlike other models in GluonTS this network does not use a distribution. One of the following: &quot;sMAPE&quot;, &quot;MASE&quot; or &quot;MAPE&quot;. The default value is &quot;MAPE&quot;.</td>
</tr>
<tr>
<td>num_stacks</td>
<td>The number of stacks the network should contain. Default and recommended value for generic mode: 30 Recommended value for interpretable mode: 2</td>
</tr>
<tr>
<td>num_blocks</td>
<td>The number of blocks per stack. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: 1. Recommended value for interpretable mode: 3.</td>
</tr>
<tr>
<td>widths</td>
<td>Widths of the fully connected layers with ReLu activation in the blocks. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: list(512) Recommended value for interpretable mode: list(256, 2048)</td>
</tr>
<tr>
<td>sharing</td>
<td>Whether the weights are shared with the other blocks per stack. A list of ints of length 1 or 'num_stacks'. Default and recommended value for generic mode: list(FALSE) Recommended value for interpretable mode: list(TRUE)</td>
</tr>
<tr>
<td>expansion_coefficient_lengths</td>
<td>If the type is &quot;G&quot; (generic), then the length of the expansion coefficient. If type is &quot;T&quot; (trend), then it corresponds to the degree of the polynomial. If the type is &quot;S&quot;</td>
</tr>
</tbody>
</table>
**save_gluonts_model**

Saving and Loading GluonTS Models

Description

GluonTS models require a special storage process that saves / loads the recipe used to recreate a model to / from a directory that the user defines.

Usage

```r
save_gluonts_model(object, path, overwrite = FALSE)
load_gluonts_model(path)
```

Arguments

- **object**: A fitted model object
- **path**: A directory to store the GluonTS model files
- **overwrite**: Whether or not to allow overwriting a GluonTS model’s directory. Default: FALSE.

**nbeats_predict_impl**

Bridge prediction Function for N-BEATS Models

Description

Bridge prediction Function for N-BEATS Models

Usage

```r
nbeats_predict_impl(object, new_data)
```

Arguments

- **object**: An object of class `model_fit`
- **new_data**: A rectangular data object, such as a data frame.

**stack_types**

One of the following values: "G" (generic), "S" (seasonal) or "T" (trend). A list of ints of length 1 or 'num_stacks'. Default value for generic mode: list(32) Recommended value for interpretable mode: list(3)

**nbeats_predict_impl**

Bridge prediction Function for N-BEATS Models

Description

Bridge prediction Function for N-BEATS Models

Usage

```r
nbeats_predict_impl(object, new_data)
```

Arguments

- **object**: An object of class `model_fit`
- **new_data**: A rectangular data object, such as a data frame.
to_gluon_list_dataset

## Examples

```r
## Not run:
library(tidymodels)
library(tidyverse)
library(timetk)

model_fit <- nbeats(
  # User Defined (Required) Parameters
  id = "id",
  freq = "M",
  prediction_length = 24,

  # Hyper Parameters
  epochs = 1,
  num_batches_per_epoch = 4
)

set_engine("gluonts_nbeats") %>%
  fit(value ~ date + id, m750)

# Saves the related files needed to recreate the model
model_fit %>% save_gluonts_model(path = "/dir_nbeats_model/"

# Loads the model
load_gluonts_model(path = "/dir_nbeats_model/"

## End(Not run)
```

---

to_gluon_list_dataset  Convert a data frame to a GluonTS ListDataset

### Description

A ListDataset is the format required by GluonTS. This function simplifies creating a GluonTS ListDataset.

### Usage

```r
to_gluon_list_dataset(data, date_var, value_var, id_var = NULL, freq = "D")
```

### Arguments

- **data**  
  A data.frame
- **date_var**  
  The date column (Timestamps)
- **value_var**  
  The value column (Target)
- **id_var**  
  The Time Series ID column for tracking time series in GluonTS
- **freq**  
  the Pandas Timestamp Frequency.
Examples

```r
library(timetk)

m4_daily %>%
  to_gluon_list_dataset(
    date_var = date,
    value_var = value,
    id_var = id,
    freq = "D"
  )
```
Index

as_pandas_timestamp, 2

deep_ar, 5
deepar_fit_impl, 3
deepar_predict_impl, 5

fit.model_spec(), 8, 14

install_gluonts, 9

load_gluonts_model
  (save_gluonts_model), 20

nbeats, 10
nbeats_ensemble_fit_impl, 15
nbeats_ensemble_predict_impl, 17
nbeats_fit_impl, 18
nbeats_predict_impl, 20

save_gluonts_model, 20
set_engine(), 8, 14

to_gluon_list_dataset, 21