# tigerforecast Documentation

*Release 0.0.1*

**alexjyu**

**Nov 14, 2019**

# Contents

For an introduction to TigerForecast, start at the TigerForecast GitHub page.

CHAPTER 1

# tigerforecast package

## 1.1 Subpackages

### 1.1.1 tigerforecast.utils package

#### dataset_registry

| | |
|---|---|
| *unemployment*([verbose]) | Description: Checks if unemployment data exists, downloads if not. |
| *uci_indoor*([verbose]) | Description: Checks if uci_indoor data exists, downloads if not. |
| *sp500*([verbose]) | Description: Checks if S&P500 data exists, downloads if not. |
| *crypto*([verbose]) | Description: Checks if cryptocurrency data exists, downloads if not. |
| *enso*(input_signals, include_month, ...) | Description: Transforms the ctrl_indices dataset into a format suitable for online learning. |

#### tigerforecast.utils.unemployment

tigerforecast.utils.**unemployment**(*verbose=True*)

**Description: Checks if unemployment data exists, downloads if not.** Dataset credits: https://fred.stlouisfed.org/series/UNRATE, Federal Reserve Bank of St. Louis.

**Parameters** **verbose** (*boolean*) – Specifies if download progress should be printed

**Returns** Dataframe containing Unemployment data

### tigerforecast.utils.uci_indoor

tigerforecast.utils.**uci_indoor**(*verbose=True*)

**Description: Checks if uci_indoor data exists, downloads if not.** Dataset credits: F. Zamora-Martínez, P. Romeu, P. Botella-Rocamora, J. Pardo, On-line learning of indoor temperature forecasting methods towards energy efficiency, Energy and Buildings, Volume 83, November 2014, Pages 162-172, ISSN 0378-7788

**Parameters** **verbose** (*boolean*) – Specifies if download progress should be printed

**Returns** Dataframe containing uci_indoor data

### tigerforecast.utils.sp500

tigerforecast.utils.**sp500**(*verbose=True*)

Description: Checks if S&P500 data exists, downloads if not.

**Parameters** **verbose** (*boolean*) – Specifies if download progress should be printed

**Returns** Dataframe containing S&P500 data

### tigerforecast.utils.crypto

tigerforecast.utils.**crypto**(*verbose=True*)

Description: Checks if cryptocurrency data exists, downloads if not.

**Parameters** **None** –

**Returns** Dataframe containing cryptocurrency data

### tigerforecast.utils.enso

tigerforecast.utils.**enso**(*input_signals*, *include_month*, *output_signals*, *history*, *timeline*)

Description: Transforms the ctrl_indices dataset into a format suitable for online learning.

**Parameters**

- **input_signals** (*list of strings*) – signals used for prediction
- **include_month** (*boolean*) – True if the month should be used as a feature, False otherwise
- **output_signals** (*list of strings*) – signals we are trying to predict
- **history** (*int*) – number of past observations used for prediction
- **timeline** (*int/list of ints*) – the forecasting timeline(s)

**Returns** Input Observations y (numpy.ndarray): Labels

**Return type** X (numpy.ndarray)

### random

| | |
|---|---|
| *set_key*([key]) | Descripton: Fix global random key to ensure reproducibility of results. |
| *generate_key*() | Descripton: Generate random key. |
| *get_global_key*() | Descripton: Get current global random key. |

### tigerforecast.utils.set_key

tigerforecast.utils.**set_key**(*key=None*)

Descripton: Fix global random key to ensure reproducibility of results.

**Parameters** **key** (*int*) – key that determines reproducible output

### tigerforecast.utils.generate_key

tigerforecast.utils.**generate_key**()

Descripton: Generate random key.

**Returns** Random random key

### tigerforecast.utils.get_global_key

tigerforecast.utils.**get_global_key**()

Descripton: Get current global random key.

**Returns** Current global random key

### optimizers

| | |
|---|---|
| *optimizers.Optimizer*([pred, loss, ...]) | Description: Core class for method optimizers |
| *optimizers.Adagrad*([pred, loss, ...]) | Description: Ordinary Gradient Descent optimizer. |
| *optimizers.Adam*([pred, loss, learning_rate, ...]) | Description: Ordinary Gradient Descent optimizer. |
| *optimizers.ONS*([pred, loss, learning_rate, ...]) | Online newton step algorithm. |
| *optimizers.SGD*([pred, loss, learning_rate, ...]) | Description: Stochastic Gradient Descent optimizer. |
| *optimizers.OGD*([pred, loss, learning_rate, ...]) | Description: Ordinary Gradient Descent optimizer. |
| *optimizers.mse*(y_pred, y_true) | Description: mean-square-error loss :param y_pred: value predicted by method :param y_true: ground truth value :param eps: some scalar |
| *optimizers.cross_entropy*(y_pred, y_true[, eps]) | Description: cross entropy loss, y_pred is equivalent to logits and y_true to labels :param y_pred: value predicted by method :param y_true: ground truth value :param eps: some scalar |

### tigerforecast.utils.optimizers.Optimizer

**class** tigerforecast.utils.optimizers.**Optimizer**(*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)

Description: Core class for method optimizers

**Parameters**

- **pred** (*function*) – a prediction function implemented with jax.numpy
- **loss** (*function*) – specifies loss function to be used; defaults to MSE
- **learning_rate** (*float*) – learning rate. Default value 0.01
- **hyperparameters** (*dict*) – additional optimizer hyperparameters

**Returns** None

**__init__** (*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Initialize self. See help(type(self)) for accurate signature.

### Methods

| | |
|---|---|
| *__init__*([pred, loss, learning_rate, ...]) | Initialize self. |
| gradient(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |
| reset() | |
| set_loss(new_loss) | Description: updates internal loss |
| set_predict(pred[, loss]) | Description: Updates internally stored pred and loss functions, resets internal parameters |

## tigerforecast.utils.optimizers.Adagrad

**class** tigerforecast.utils.optimizers.**Adagrad**(*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Description: Ordinary Gradient Descent optimizer. :param pred: a prediction function implemented with jax.numpy :type pred: function :param loss: specifies loss function to be used; defaults to MSE :type loss: function :param learning_rate: learning rate :type learning_rate: float

**Returns** None

**__init__** (*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Initialize self. See help(type(self)) for accurate signature.

### Methods

| | |
|---|---|
| *__init__*([pred, loss, learning_rate, ...]) | Initialize self. |
| gradient(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |
| reset() | |
| set_loss(new_loss) | Description: updates internal loss |
| set_predict(pred[, loss]) | Description: Updates internally stored pred and loss functions, resets internal parameters |
| update(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |

## tigerforecast.utils.optimizers.Adam

**class** tigerforecast.utils.optimizers.**Adam**(*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Description: Ordinary Gradient Descent optimizer. :param pred: a prediction function implemented with

jax.numpy :type pred: function :param loss: specifies loss function to be used; defaults to MSE :type loss: function :param learning_rate: learning rate :type learning_rate: float

**Returns** None

**__init__** (*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Initialize self. See help(type(self)) for accurate signature.

### Methods

| | |
|---|---|
| *__init__*([pred, loss, learning_rate, ...]) | Initialize self. |
| gradient(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |
| reset() | |
| set_loss(new_loss) | Description: updates internal loss |
| set_predict(pred[, loss]) | Description: Updates internally stored pred and loss functions, resets internal parameters |
| update(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |

## tigerforecast.utils.optimizers.ONS

**class** tigerforecast.utils.optimizers.**ONS**(*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Online newton step algorithm.

**__init__** (*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Initialize self. See help(type(self)) for accurate signature.

### Methods

| | |
|---|---|
| *__init__*([pred, loss, learning_rate, ...]) | Initialize self. |
| general_norm(x) | |
| gradient(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |
| norm_project(y, A, c) | Project y using norm A on the convex set bounded by c. |
| reset() | |
| set_loss(new_loss) | Description: updates internal loss |
| set_predict(pred[, loss]) | Description: Updates internally stored pred and loss functions, resets internal parameters |
| update(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |

## tigerforecast.utils.optimizers.SGD

**class** tigerforecast.utils.optimizers.**SGD**(*pred=None*, *loss=<function mse>*, *learning_rate=0.0001*, *hyperparameters={}*)
Description: Stochastic Gradient Descent optimizer. :param pred: a prediction function implemented with jax.numpy :type pred: function :param loss: specifies loss function to be used; defaults to MSE :type loss: function :param learning_rate: learning rate :type learning_rate: float

**Returns** None

**__init__** (*pred=None*, *loss=<function mse>*, *learning_rate=0.0001*, *hyperparameters={}*)
Initialize self. See help(type(self)) for accurate signature.

### Methods

| | |
|---|---|
| *__init__*([pred, loss, learning_rate, ...]) | Initialize self. |
| gradient(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |
| reset() | |
| set_loss(new_loss) | Description: updates internal loss |
| set_predict(pred[, loss]) | Description: Updates internally stored pred and loss functions, resets internal parameters |
| update(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |

## tigerforecast.utils.optimizers.OGD

**class** tigerforecast.utils.optimizers.**OGD** (*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Description: Ordinary Gradient Descent optimizer. :param pred: a prediction function implemented with jax.numpy :type pred: function :param loss: specifies loss function to be used; defaults to MSE :type loss: function :param learning_rate: learning rate :type learning_rate: float

**Returns** None

**__init__** (*pred=None*, *loss=<function mse>*, *learning_rate=1.0*, *hyperparameters={}*)
Initialize self. See help(type(self)) for accurate signature.

### Methods

| | |
|---|---|
| *__init__*([pred, loss, learning_rate, ...]) | Initialize self. |
| gradient(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |
| reset() | |
| set_loss(new_loss) | Description: updates internal loss |
| set_predict(pred[, loss]) | Description: Updates internally stored pred and loss functions, resets internal parameters |
| update(params, x, y[, loss]) | Description: Updates parameters based on correct value, loss and learning rate. |

## tigerforecast.utils.optimizers.mse

tigerforecast.utils.optimizers.**mse** (*y_pred*, *y_true*)
Description: mean-square-error loss :param y_pred: value predicted by method :param y_true: ground truth value :param eps: some scalar

### tigerforecast.utils.optimizers.cross_entropy

tigerforecast.utils.optimizers.**cross_entropy**(*y_pred*, *y_true*, *eps=1e-09*)
Description: cross entropy loss, y_pred is equivalent to logits and y_true to labels :param y_pred: value predicted by method :param y_true: ground truth value :param eps: some scalar

### boosting

| | |
|---|---|
| *boosting.SimpleBoost*() | Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series |

### tigerforecast.utils.boosting.SimpleBoost

**class** tigerforecast.utils.boosting.**SimpleBoost**
Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series

**__init__**()
Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| help() | Description: Prints information about this class and its methods. |
| initialize(method_id, method_params[, n, m, ...]) | Description: Initializes autoregressive method parameters :param method_id: id of weak learner method :type method_id: string :param method_params: dict of params to pass method :type method_params: dict :param N: default 3. |
| predict(x) | |
| to_ndarray(x) | Description: If x is a scalar, transform it to a (1, 1) numpy.ndarray; otherwise, leave it unchanged. |
| update(y) | |

#### Attributes

| |
|---|
| compatibles |

### autotuning

| | |
|---|---|
| *autotuning.grid_search.GridSearch*() | Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series |

#### tigerforecast.utils.autotuning.grid_search.GridSearch

**class** tigerforecast.utils.autotuning.grid_search.**GridSearch**

Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

##### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| search(method_id, method_params, problem_id, ...) | Description: Search for optimal method parameters :param method_id: id of method :type method_id: string :param method_params: initial method parameters dict (updated by search space) :type method_params: dict :param problem_id: id of problem to try on :type problem_id: string :param problem_params: problem parameters dict :type problem_params: dict :param loss: a function mapping y_pred, y_true -> scalar loss :type loss: function :param search_space: dict mapping parameter names to a finite set of options :type search_space: dict :param trials: number of random trials to sample from search space / try all parameters :type trials: int, None :param smoothing: loss computed over smoothing number of steps to decrease variance :type smoothing: int :param min_steps: minimum number of steps that the method gets to run for :type min_steps: int :param verbose: if 1, print progress and current parameters :type verbose: int |

### 1.1.2 tigerforecast.problems package

#### core

This is a core

| |
|---|
| *Problem*() |

#### tigerforecast.problems.Problem

**class** tigerforecast.problems.**Problem**

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Description: closes the problem and returns used memory |
| help() | Description: prints information about this class and its methods |
| initialize(**kwargs) | Description: resets problem to time 0 |
| step([action]) | Description: run one timestep of the problem's dynamics. |

#### Attributes

| |
|---|
| spec |

### custom

| | |
|---|---|
| *tigerforecast.problems.CustomProblem*() | Description: class for implementing algorithms with enforced modularity |
| *tigerforecast.problems.register_custom_problem*(...) | Description: global custom problem method |
| *tigerforecast.problems.MyProblem*() | Description: Make dataset into tigerforecast problem class. |

### tigerforecast.problems.CustomProblem

**class** tigerforecast.problems.**CustomProblem**

Description: class for implementing algorithms with enforced modularity

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |

### tigerforecast.problems.register_custom_problem

tigerforecast.problems.**register_custom_problem**(*custom_problem_class*, *custom_problem_id*)

Description: global custom problem method

### tigerforecast.problems.MyProblem

**class** tigerforecast.problems.**MyProblem**

Description: Make dataset into tigerforecast problem class.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Not implemented |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the timestep corresponding to the last (observation, label) pair returned. |
| initialize(file[, X, y, T]) | Description: Initialize the problem class. |
| step() | Description: Moves time forward by one month and returns the corresponding observation and label. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### time_series

| | |
|---|---|
| *tigerforecast.problems.Problem*() | |
| *tigerforecast.problems.SP500*() | Description: Outputs the daily opening price of the S&P 500 stock market index from January 3, 1986 to June 29, 2018. |
| *tigerforecast.problems.UCI_Indoor*() | Description: Outputs various weather metrics from a UCI dataset from 13/3/2012 to 11/4/2012 |
| *tigerforecast.problems.ENSO*() | Description: Collection of monthly values of control indices useful for predicting La Nina/El Nino. |
| *tigerforecast.problems.Crypto*() | Description: Outputs the daily price of bitcoin from 2013-04-28 to 2018-02-10 |
| *tigerforecast.problems.Random*() | Description: A random sequence of scalar values taken from an i.i.d. |
| *tigerforecast.problems.ARMA*() | Description: Simulates an autoregressive moving-average time-series. |
| *tigerforecast.problems.Unemployment*() | Description: Monthly unemployment rate since 1948. |
| *tigerforecast.problems.LDS_TimeSeries*() | Description: Simulates a linear dynamical system. |
| *tigerforecast.problems.LSTM_TimeSeries*() | Description: Produces outputs from a randomly initialized recurrent neural network. |
| *tigerforecast.problems.RNN_TimeSeries*() | Description: Produces outputs from a randomly initialized recurrent neural network. |

### tigerforecast.problems.SP500

**class** tigerforecast.problems.**SP500**

Description: Outputs the daily opening price of the S&P 500 stock market index from January 3, 1986 to June 29, 2018.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Description: closes the problem and returns used memory |
| help() | Description: prints information about this class and its methods |
| initialize([normalization]) | Description: Check if data exists, else download, clean, and setup. |
| step() | Description: Moves time forward by one day and returns value of the stock index :param None: |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.UCI_Indoor

**class** tigerforecast.problems.**UCI_Indoor**

Description: Outputs various weather metrics from a UCI dataset from 13/3/2012 to 11/4/2012

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Not implemented |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the date corresponding to the last value of the uci_indoor that was returned :param None: |
| initialize([pred_indices]) | Description: Check if data exists, else download, clean, and setup. |
| step() | Description: Moves time forward by fifteen minutes and returns weather metrics :param None: |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.ENSO

**class** tigerforecast.problems.**ENSO**

Description: Collection of monthly values of control indices useful for predicting La Nina/El Nino. More specifically, the user can choose any of pna, ea, wa, wp, eu, soi, esoi, nino12, nino34, nino4, oni of nino34 (useful for La Nino/El Nino identification) to be used as input and/or output in the problem instance.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Not implemented |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the timestep corresponding to the last (observation, label) pair returned. |
| initialize([input_signals, include_month, ...]) | Description: Initializes the ctrl_indices dataset to a format suited to the online learning setting. |
| step() | Description: Moves time forward by one month and returns the corresponding observation and label. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.Crypto

**class** tigerforecast.problems.**Crypto**

Description: Outputs the daily price of bitcoin from 2013-04-28 to 2018-02-10

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Not implemented |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the date corresponding to the last price of bitcoin that was returned :param None: |
| initialize() | Description: Check if data exists, else download, clean, and setup. |
| step() | Description: Moves time forward by one day and returns price of the bitcoin :param None: |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.Random

**class** tigerforecast.problems.**Random**

Description: A random sequence of scalar values taken from an i.i.d. normal distribution.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Not implemented |
| help() | Description: prints information about this class and its methods |
| hidden() | Not implemented |
| initialize() | Description: Randomly initialize the hidden dynamics of the system. |
| step() | Description: Moves the system dynamics one time-step forward. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.ARMA

**class** tigerforecast.problems.**ARMA**

Description: Simulates an autoregressive moving-average time-series.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Description: closes the problem and returns used memory |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the hidden state of the system. |
| initialize([p, q, n, noise_list, c, ...]) | Description: Randomly initialize the hidden dynamics of the system. |

Continued on next page

Table 33 – continued from previous page

| | |
|---|---|
| step() | Description: Moves the system dynamics one time-step forward. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.Unemployment

**class** tigerforecast.problems.**Unemployment**

Description: Monthly unemployment rate since 1948.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Not implemented |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the date corresponding to the last unemployment rate value :param None: |
| initialize() | Description: Check if data exists, else download, clean, and setup. |
| step() | Description: Moves time forward by one day and returns price of the bitcoin :param None: |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.LDS_TimeSeries

**class** tigerforecast.problems.**LDS_TimeSeries**

Description: Simulates a linear dynamical system.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |

Continued on next page

Table 37 – continued from previous page

| | |
|---|---|
| close() | Description: closes the problem and returns used memory |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the hidden state of the system. |
| initialize(n, m, d[, noise]) | Description: Randomly initialize the hidden dynamics of the system. |
| step() | Description: Moves the system dynamics one time-step forward. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.LSTM_TimeSeries

**class** tigerforecast.problems.**LSTM_TimeSeries**

Description: Produces outputs from a randomly initialized recurrent neural network.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Description: closes the problem and returns used memory |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the hidden state of the RNN when computed on the last l inputs. |
| initialize(n, m[, h]) | Description: Randomly initialize the RNN. |
| step() | Description: Takes an input and produces the next output of the RNN. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### tigerforecast.problems.RNN_TimeSeries

**class** tigerforecast.problems.**RNN_TimeSeries**

Description: Produces outputs from a randomly initialized recurrent neural network.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| close() | Description: closes the problem and returns used memory |
| help() | Description: prints information about this class and its methods |
| hidden() | Description: Return the hidden state of the RNN when computed on the last l inputs. |
| initialize(n, m[, h]) | Description: Randomly initialize the RNN. |
| step() | Description: Takes an input and produces the next output of the RNN. |

#### Attributes

| |
|---|
| compatibles |
| spec |

### 1.1.3 tigerforecast.methods package

#### core

| |
|---|
| *Method* |

#### tigerforecast.methods.Method

**class** tigerforecast.methods.**Method**

**__init__**()
Initialize self. See help(type(self)) for accurate signature.

#### Methods

| |
|---|
| initialize(**kwargs) |
| predict([x]) |
| update(**kwargs) |

#### time_series

| | |
|---|---|
| *tigerforecast.methods.AutoRegressor*() | Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series |
| *tigerforecast.methods.LastValue*() | Description: Predicts the last value in the time series, i.e. |
| *tigerforecast.methods.PredictZero*() | Description: Predicts the next value in the time series to be 0, i.e. |

Continued on next page

Table 45 – continued from previous page

| | |
|---|---|
| *tigerforecast.methods.RNN*() | Description: Produces outputs from a randomly initialized recurrent neural network. |
| *tigerforecast.methods.LSTM*() | Description: Produces outputs from a randomly initialized LSTM neural network. |
| *tigerforecast.methods.LeastSquares*() | Description: Implements online least squares. |
| *tigerforecast.methods.WaveFiltering*() | Description: Predicts the last value in the time series, i.e. |

### tigerforecast.methods.AutoRegressor

**class** tigerforecast.methods.**AutoRegressor**

Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| forecast(x[, timeline]) | Description: Forecast values 'timeline' timesteps in the future :param x: Value at current time-step :type x: int/numpy.ndarray :param timeline: timeline for forecast :type timeline: int |
| initialize([n, m, p, optimizer]) | Description: Initializes autoregressive method parameters |
| predict(x) | Description: Predict next value given observation :param x: Observation :type x: int/numpy.ndarray |
| update(y) | Description: Updates parameters using the specified optimizer :param y: True value at current time-step :type y: int/numpy.ndarray |

#### Attributes

| |
|---|
| compatibles |

### tigerforecast.methods.LastValue

**class** tigerforecast.methods.**LastValue**

Description: Predicts the last value in the time series, i.e. x(t) = x(t-1)

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |

Continued on next page

Table 48 – continued from previous page

| | |
|---|---|
| forecast(x[, timeline]) | Description: Forecast values 'timeline' timesteps in the future :param x: Value at current time-step :type x: int/numpy.ndarray :param timeline: timeline for forecast :type timeline: int |
| initialize([n, m]) | Description: Initialize the (non-existent) hidden dynamics of the method :param None: |
| predict(x) | Description: Takes input observation and returns next prediction value :param x: value at current time-step :type x: float/numpy.ndarray |
| update(y) | Description: Takes update rule and adjusts internal parameters :param y: true value :type y: float/np.ndarray |

#### Attributes

| |
|---|
| compatibles |

### tigerforecast.methods.PredictZero

**class** tigerforecast.methods.**PredictZero**

Description: Predicts the next value in the time series to be 0, i.e. x(t) = 0

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| forecast(x[, timeline]) | Description: Forecast values 'timeline' timesteps in the future :param x: Value at current time-step :type x: int/numpy.ndarray :param timeline: timeline for forecast :type timeline: int |
| initialize([n, m]) | Description: Initialize the (non-existent) hidden dynamics of the method :param None: |
| predict(x) | Description: Takes input observation and returns next prediction value :param x: value at current time-step :type x: float/numpy.ndarray |
| update([rule]) | Description: Takes update rule and adjusts internal parameters :param rule: rule with which to alter parameters :type rule: function |

#### Attributes

| |
|---|
| compatibles |

### tigerforecast.methods.RNN

**class** tigerforecast.methods.**RNN**

Description: Produces outputs from a randomly initialized recurrent neural network.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| forecast(x[, timeline]) | Description: Forecast values 'timeline' timesteps in the future :param x: Value at current time-step :type x: float/numpy.ndarray :param timeline: timeline for forecast :type timeline: int |
| initialize([n, m, l, h, optimizer, loss, lr]) | Description: Randomly initialize the RNN. |
| predict(x[, timeline]) | Description: Predict next value given observation :param x: Observation :type x: float/numpy.ndarray |
| to_ndarray(x) | Description: If x is a scalar, transform it to a (1, 1) numpy.ndarray; otherwise, leave it unchanged. |
| update(y) | Description: Updates parameters :param y: True value at current time-step :type y: int/numpy.ndarray |

#### Attributes

| |
|---|
| compatibles |

### tigerforecast.methods.LSTM

**class** tigerforecast.methods.**LSTM**

Description: Produces outputs from a randomly initialized LSTM neural network.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| forecast(x[, timeline]) | Description: Forecast values 'timeline' timesteps in the future :param x: Value at current time-step :type x: int/numpy.ndarray :param timeline: timeline for forecast :type timeline: int |
| initialize([n, m, l, h, optimizer]) | Description: Randomly initialize the LSTM. |
| predict(x) | Description: Predict next value given observation :param x: Observation :type x: int/numpy.ndarray |
| to_ndarray(x) | Description: If x is a scalar, transform it to a (1, 1) numpy.ndarray; otherwise, leave it unchanged. |
| update(y) | Description: Updates parameters :param y: True value at current time-step :type y: int/numpy.ndarray |

#### Attributes

| |
|---|
| compatibles |

### tigerforecast.methods.LeastSquares

**class** tigerforecast.methods.**LeastSquares**

Description: Implements online least squares.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| initialize(x, y[, reg]) | Description: Initializes method parameters |
| predict([x]) | |
| step(x, y) | Description: Predict next value given observation and update internal parameters based |
| update(**kwargs) | |

#### Attributes

| |
|---|
| compatibles |

### tigerforecast.methods.WaveFiltering

**class** tigerforecast.methods.**WaveFiltering**

Description: Predicts the last value in the time series, i.e. x(t) = x(t-1)

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| eigen_pairs(T, k) | |
| forecast(x[, timeline]) | Description: Forecast values 'timeline' timesteps in the future :param x: Value at current time-step :type x: int/numpy.ndarray :param timeline: timeline for forecast :type timeline: int |
| initialize(n, m, k, T, eta, R_M) | Description: Initialize the (non-existent) hidden dynamics of the method :param None: |
| predict(x) | Description: Takes input observation and returns next prediction value :param x: value at current time-step :type x: float/numpy.ndarray |

Continued on next page

Table 58 – continued from previous page

| | |
|---|---|
| update(y) | Description: Takes update rule and adjusts internal parameters :param y: true value :type y: float/np.ndarray |

#### Attributes

| |
|---|
| compatibles |

### boosting

| | |
|---|---|
| *tigerforecast.methods.boosting.SimpleBoost*() | Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series |

### tigerforecast.methods.boosting.SimpleBoost

**class** tigerforecast.methods.boosting.**SimpleBoost**

Description: Implements the equivalent of an AR(p) method - predicts a linear combination of the previous p observed values in a time-series

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| help() | Description: Prints information about this class and its methods. |
| initialize(method_id, method_params[, n, m, ...]) | Description: Initializes autoregressive method parameters :param method_id: id of weak learner method :type method_id: string :param method_params: dict of params to pass method :type method_params: dict :param N: default 3. |
| predict(x) | |
| to_ndarray(x) | Description: If x is a scalar, transform it to a (1, 1) numpy.ndarray; otherwise, leave it unchanged. |
| update(y) | |

#### Attributes

| |
|---|
| compatibles |

## 1.1.4 tigerforecast.experiments package

### core

| | |
|---|---|
| *create_full_problem_to_methods*(problems_ids, ...) | Description: Associate all given problems to all given methods. |
| *run_experiment*(problem, method[, metric, ...]) | Description: Initializes the experiment instance. |
| *run_experiments*(problem, method[, metric, ...]) | Description: Initializes the experiment instance. |

### tigerforecast.experiments.create_full_problem_to_methods

tigerforecast.experiments.**create_full_problem_to_methods**(*problems_ids*, *method_ids*)

Description: Associate all given problems to all given methods.

**Parameters**

- **problem_ids** (*list*) – list of problem names
- **method_ids** (*list*) – list of method names

**Returns** association problem -> method

**Return type** full_problem_to_methods (dict)

### tigerforecast.experiments.run_experiment

tigerforecast.experiments.**run_experiment**(*problem*, *method*, *metric='mse'*, *lr_tuning=True*, *key=0*, *timesteps=None*, *verbose=0*)

Description: Initializes the experiment instance.

**Parameters**

- **problem** (*tuple*) – problem id and parameters to initialize the specific problem instance with
- **method** (*tuple*) – method id and parameters to initialize the specific method instance with
- **metric** (*string*) – metric we are interesting in computing for current experiment
- **key** (*int*) – for reproducibility
- **timesteps** (*int*) – number of time steps to run experiment for

**Returns** loss series for the specified metric over the entirety of the experiment time (float): time elapsed memory (float): memory used

**Return type** loss (list)

### tigerforecast.experiments.run_experiments

tigerforecast.experiments.**run_experiments**(*problem*, *method*, *metric='mse'*, *lr_tuning=True*, *n_runs=1*, *timesteps=None*, *verbose=0*)

Description: Initializes the experiment instance.

**Parameters**

- **problem** (*tuple*) – problem id and parameters to initialize the specific problem instance with

- **method** (*tuple*) – method id and parameters to initialize the specific method instance with
- **metric** (*string*) – metric we are interesting in computing for current experiment
- **key** (*int*) – for reproducibility
- **timesteps** (*int*) – number of time steps to run experiment for

**Returns** loss series for the specified metric over the entirety of the experiment time (float): time elapsed memory (float): memory used

**Return type** loss (list)

### metrics

| | |
|---|---|
| *mse*(y_pred, y_true) | Description: mean-square-error loss |
| *cross_entropy*(y_pred, y_true[, eps]) | Description: cross entropy loss, y_pred is equivalent to logits and y_true to labels |

### tigerforecast.experiments.mse

tigerforecast.experiments.**mse**(*y_pred*, *y_true*)

Description: mean-square-error loss

**Parameters**

- **y_pred** – value predicted by method
- **y_true** – ground truth value
- **eps** – some scalar

### tigerforecast.experiments.cross_entropy

tigerforecast.experiments.**cross_entropy**(*y_pred*, *y_true*, *eps=1e-09*)

Description: cross entropy loss, y_pred is equivalent to logits and y_true to labels

**Parameters**

- **y_pred** – value predicted by method
- **y_true** – ground truth value
- **eps** – some scalar

### experiment

| | |
|---|---|
| *Experiment*() | Description: Streamlines the process of performing experiments and comparing results of methods across a range of problems. |

### tigerforecast.experiments.Experiment

**class** tigerforecast.experiments.**Experiment**

Description: Streamlines the process of performing experiments and comparing results of methods across a range of problems.

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| add_all_method_variants(method_id[, ...]) | |
| add_method(method_id[, method_params, ...]) | Description: Add a new method to the experiment instance. |
| add_problem(problem_id[, problem_params, name]) | Description: Add a new problem to the experiment instance. |
| avg_regret(loss) | |
| graph([problem_ids, metric, avg_regret, ...]) | Description: Show a graph for the results of the experiments for specified metric. |
| initialize([problems, methods, ...]) | Description: Initializes the experiment instance. |
| scoreboard([metric, start_time, n_digits, ...]) | Description: Show a scoreboard for the results of the experiments for specified metric. |
| to_csv(table_dict, save_as) | Save to csv file |

### new_experiment

| | |
|---|---|
| *NewExperiment*() | Description: class for implementing algorithms with enforced modularity |

### tigerforecast.experiments.NewExperiment

**class** tigerforecast.experiments.**NewExperiment**

Description: class for implementing algorithms with enforced modularity

**__init__**()

Initialize self. See help(type(self)) for accurate signature.

#### Methods

| | |
|---|---|
| *__init__*() | Initialize self. |
| initialize(problems, methods[, ...]) | Description: Initializes the new experiment instance. |
| run_all_experiments() | Descripton: Runs all experiments and returns results |

### precomputed

| | |
|---|---|
| *recompute*([verbose, load_bar]) | Description: Recomputes all the results. |
| *load_prob_method_to_result*([problem_ids, ...]) | Description: Initializes the experiment instance. |

### tigerforecast.experiments.recompute

tigerforecast.experiments.**recompute**(*verbose=False*, *load_bar=False*)

Description: Recomputes all the results.

**Parameters**

- **verbose** (*boolean*) – Specifies whether to print what experiment is currently running.
- **load_bar** (*boolean*) – Specifies whether to show a loading bar while the experiments are running.

### tigerforecast.experiments.load_prob_method_to_result

tigerforecast.experiments.**load_prob_method_to_result**(*problem_ids=['ARMA-v0', 'Crypto-v0', 'SP500-v0']*, *method_ids=['LastValue', 'AutoRegressor', 'RNN', 'LSTM']*, *problem_to_methods=None*, *metrics='mse'*)

Description: Initializes the experiment instance.

**Parameters**

- **problem_ids** (*list*) – ids of problems to evaluate on
- **method_ids** (*list*) – ids of methods to use
- **problem_to_methods** (*dict*) – map of the form problem_id -> list of method_id. If None, then we assume that the user wants to test every method in method_to_params against every problem in problem_to_params
- **metrics** – metrics to load

CHAPTER 2

# help

email alexjyu@google.com

## 2.1 Need further help

Please join the IRC channel

CHAPTER 3

# License

Some license

CHAPTER 4

# Contact

alexjyu@google.com

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## t

# Index

## Symbols



## A



## C