Foxhound

Release 0.1.2

João Gabriel Segato Kruse

Oct 18, 2021

CONTENTS

CHAPTER

ONE

CAUSATIONS MODULE

class causations.**Causations**(*options*, *seed=111*, *cuda=False*)

Class responsible for causation finding with TCDF

get_causation(*datafiles*)

Finds causations between all variables in datafiles

Parameters **alldelays** (*pandas.DataFrame*) – DataFrame containing all time series to be considered

Returns All delays and variable names

Return type *(List[List[int]], List[str])*

getextendeddelays(*gtfile*, *columns*)

Collects the total delay of indirect causal relationships.

static plotgraph(*alldelays*, *columns*)

Plots a temporal causal graph showing all discovered causal relationships annotated with the time delay between cause and effect.

Parameters

- **alldelays** (*List[List[int]]*) – delays between each two variables
- **columns** (*List[str]*) – List with all variable names

runTCDF(*df_data*)

Loops through all variables in a dataset and return the discovered causes, time delays, losses, attention scores and variable names.

CHAPTER

TWO

CORRELATOR MODULE

correlator.**correlate**(*x*, *y*, *margin*, *method='pearson'*)

Find delay and correlation between x and each column o y

Parameters

- **x** (*pandas.Series*) – Main signal
- **y** (*pandas.DataFrame*) – Secondary signals
- **method** (*str*, optional) – Correlation method. Defaults to *pearson*. Options: *pearson*,`robust`,`kendall`,`spearman`

Returns List of correlation coefficients and delays in samples in the same order as y's columns

Return type *(List[float], List[int])*

Notes

Uses the pandas method corrwith (which can return pearson, kendall or spearman coefficients) to correlate. If robust correlation is used, the mapping presented in[1] is used and then Pearson correlation is used. To speedup the lag finding, the delays are calculated in log intervals and then interpolated by splines, as shown in[2], and the lag with maximum correlation found in this interpolated function is then used as the delay.

References

correlator.**find_delays**(*x*, *y*)

Find delay between x and each column o y

Parameters

- **x** (*pandas.Series*) – Main signal
- **y** (*pandas.DataFrame*) – Secondary signals

Returns Dataframe with the delay value for each column of y

Return type *pandas.DataFrame*

correlator.**interpolate**(*x*, *idx*, *margin*)

Interpolate data to match idx+-margin

Parameters

[1] Raymaekers, J., Rousseeuw, P. "Fast Robust Correlation for High-Dimensional Data", Technometrics, vol. 63, Pages 184-198, 2021

[2] Sakurai, Yasushi & Papadimitriou, Spiros & Faloutsos, Christos. (2005). BRAID: Stream mining through group lag correlations. Proceedings of the ACM SIGMOD International Conference on Management of Data. 599-610.

- **x** (*pandas.Dataframe*) – Signal
- **idx** (*pandas.DatetimeIndex*) – Index to match
- **margin** (*float*) – Percentage of values to add to each side of index

Returns Dataframe with the same columns as x interpolated to match idx+-margin

Return type *pandas.DataFrame*

Notes

It infers the frequency for the given DatetimeIndex and extends it to margin times prior and after. This new DatetimeIndex is then combined with the given DataFrame and the NaN values are completed with linear interpolation then. In the end, only the new index values are kept, so that it matches exactly the given idx dates (except for the margin values).

correlator.**lagged_corr**(*x*, *y*, *lag*, *method='pearson'*)

Find correlation between x and each column o y for a specific time lag

Parameters

- **x** (*pandas.Series*) – Main signal
- **y** (*pandas.DataFrame*) – Secondary signals
- **lag** (*int*) – Number of samples to apply as lag before computing the correlation
- **method** (*str*, optional) – Correlation method. Defaults to *pearson*. Options: *pearson*,`kendall`,`spearman`

Returns Dataframe with the correlation value for each column of y

Return type *pandas.DataFrame*

CHAPTER

THREE

DATASET MODULE

class dataset.**Dataset**(*filename=None*, *date_name='datetime'*)

Class for data retrieval and correlation/causation finding.

EPICS

Flag to indicate whether it is using EPICS or a local csv

Type *bool*

dataset

Dataset being used when using local csv

Type *pandas.DataFrame*

loop

Loop for async requests

Type *asyncio.loop*

last_searches

Dictionary with regex and PV names pairs for all regex already searched

Type *dict*

last_dataset_metadata

Dictionary containing the dates and PVs that were last requested

Type *dict*

last_dataset

DataFrame with the last requested time series

Type *pandas.DataFrame*

causation(*x_label*, *begin=None*, *end=None*, *margin=0.2*, *options=('Adam', 1, 4, 0.8, 4, 500, 0.01, 1000)*)

Finds causation graph between x_lab and the PVs in the dataset with TCDF [1]__

Parameters

- **x_label** (*str*) – Name of the main PV
- **begin** (*Datetime.datetime*, optional) – Beginning date. Default: None (uses ARCHIVER's default)
- **end** (*Datetime.datetime*, optional) – End date. Default: None (uses ARCHIVER's default)
- **margin** (*float*, optional) – Percentage of time to consider before and after the defined interval. Default: 0.2 (20%)

- **options** (*tuple(str,int,int,float,int,int,float,int)*, optional) – Options for the optimizer, depth, kernel size, significance, stride, log interval, training rate and number of epochs. Default: ('Adam',1,4,0.8,4,500,0.01,1000)

Returns Lists containing delays in samples and PV names

Return type *(List[int],List[str])*

See also:

Causation.get_causation performs TCDF

References

causation_EPICS(*x_label*, *regex*, *begin=None*, *end=None*, *margin=0.2*, *options=('Adam', 1, 4, 0.8, 4, 500, 0.01, 1000)*)

Finds causation graph between x_lab and the PVs in the dataset with TCDF [2]__

Parameters

- **x_label** (*str*) – Name of the main PV
- **regex** (*str*) – Java regex to match for other PVs
- **begin** (*Datetime.datetime*, optional) – Beginning date. Default: None (uses ARCHIVER's default)
- **end** (*Datetime.datetime*, optional) – End date. Default: None (uses ARCHIVER's default)
- **margin** (*float*, optional) – Percentage of time to consider before and after the defined interval. Default: 0.2 (20%)
- **options** (*tuple(str,int,int,float,int,int,float,int)*, optional) – Options for the optimizer, depth, kernel size, significance, stride, log interval, training rate and number of epochs. Default: ('Adam',1,4,0.8,4,500,0.01,1000)

Returns Lists containing delays in samples and PV names

Return type *(List[int],List[str])*

See also:

Causation.get_causation performs TCDF

References

correlate(*x_label*, *begin=None*, *end=None*, *margin=0.2*, *method='Pearson'*)

Computes the maximum correlation and delay between x_label and each PV in the csv

Parameters

- **x_label** (*str*) – Name of the main PV
- **begin** (*Datetime.datetime*, optional) – Beginning date. Default: None (uses ARCHIVER's default)
- **end** (*Datetime.datetime*, optional) – End date. Default: None (uses ARCHIVER's default)
- **margin** (*float*, optional) – Percentage of time to consider before and after the defined interval. Default: 0.2 (20%)

- **method** (*str*, optional) – Method to be used. Default: pearson (other options are spearman, kendall and robust)

Returns Lists containing the correlation coefficients (rounded to 2 decimals), delays in samples and PV names

Return type *(List[float],List[int],List[str])*

correlate_EPICS(*x_label*, *regex*, *begin=None*, *end=None*, *margin=0.2*, *method='Pearson'*)

Computes the maximum correlation and delay between x_label and each PV matching regex

Parameters

- **x_label** (*str*) – Name of the main PV
- **regex** (*str*) – Java regex to match
- **begin** (*Datetime.datetime*, optional) – Beginning date. Default: None (uses ARCHIVER's default)
- **end** (*Datetime.datetime*, optional) – End date. Default: None (uses ARCHIVER's default)
- **margin** (*float*, optional) – Percentage of time to consider before and after the defined interval. Default: 0.2 (20%)
- **method** (*str*, optional) – Method to be used. Default: pearson (other options are spearman, kendall and robust)

Returns Lists containing the correlation coefficients (rounded to 2 decimals), delays in samples and PV names

Return type *(List[float],List[int],List[str])*

get_EPICS_pv(*name*, *start_time=None*, *end_time=None*)

Gets DataFrame with the PVs requested

Parameters **name** (*List[str]*) – List with PV names to be requested

Returns DataFrame where each column is one PVs timeseries

Return type *pandas.DataFrame*

Notes

If the same names have already been requested, reuses the old result, else it makes another request

get_columns(*regex='.*'*)

Gets PV names in opened dataset (for csv datasets only) according to regex

Parameters **regex** (*str*) – Java regex to match

Returns List containing the names

Return type *List[str]*

get_fs(*names*)

Finds the sample rate of the time series being names

Parameters **names** (*List[str]*) – Name of the time series being considered

Returns List with Timedeltas corresponding to the sampling rate for each name in names

Return type *List[Datetime.Timedelta]*

number_of_vars(*regex*)

Gets number of elements that match the regex in Archiver

Parameters **regex** (*str*) – Java regex for the PVs being considered

Returns Number of PVs matching the regex

Return type int

Notes

If the same expression has already been searched, reuses the old result, else it makes another request

to_date(*delays*, *names*)

Converts delays from samples to times

Parameters

- **delays** (*List[int]*) – Delays in samples
- **names** (*List[str]*) – Name of the time series being considered

Returns List with Timedeltas corresponding to the delays for each name in names

Return type *List[Datetime.Timedelta]*

CHAPTER

FOUR

EPICS REQUESTS MODULE

async epics_requests.**call_fetch**(*pv_list*, *dt_init*, *dt_end*)
Get PVs time series

Parameters

- **pv_list** (*List[str]*) – List with the name of all PVs to request
- **dt_init** (aware *Datetime.datetime*) – Date and time indicating where to begin the timeseries
- **dt_end** (aware *Datetime.datetime*) – Date and time indicating where to end the timeseries

Returns DataFrame where the columns correspond to each PV name in pv_list and the index is a Index with the date as a *str*

Return type *pandas.DataFrame*

epics_requests.**correct_datetime**(*x*)
Turn Dataframes Index to DatetimeIndex

Parameters **x** (*pandas.DataFrame*) – DataFrame with index corresponding to strings of datetimes

Returns DataFrame with the new DatetimeIndex

Return type *pandas.DataFrame*

epics_requests.**get_names**(*regex=None*, *limit=100*)
Get all pv names from Archiver that correspond to the given regex

Parameters

- **regex** (*str*, optional) – Regex that will be used. Default: None, which matches all pvs (in the order of millions)
- **limit** (*int*, optional) – Maximum number of names to get (the larger the number, the longer it takes)

Returns List containing all the names

Return type *List[str]*

CHAPTER

FIVE

LAYOUT MODULE

layout.**get_error_layout**()
Get layout for error message windows

Returns Layout for error message window

Return type *List[List[Element]]*

layout.**get_fig_size**()
Get size of image to fit current window

Returns Width and Height of the image in pixels

Return type *(int, int)*

layout.**get_layout**()
Get layout for main application window

Returns Layout for main application window

Return type *List[List[Element]]*

layout.**get_param_layout**()
Get layout for causality finding parameters definition window

Returns Layout for causality finding parameters window

Return type *List[List[Element]]*

layout.**window_dimension**(*monospaced_font*)
Get window dimensions with respect to a font.

Parameters **monospaced_font** (*tkinter.font.Font*) – Font which will be used as unit of measure

Returns Width and Height of the screen in respective font units

Return type (int, int)

CHAPTER

SIX

TCDF MODULE

tcdf.**findcauses**(*target*, *cuda*, *epochs*, *kernel_size*, *layers*, *log_interval*, *lr*, *optimizername*, *seed*, *dilation_c*, *significance*, *data*)

Discovers potential causes of one target time series, validates these potential causes with PIVM and discovers the corresponding time delays

tcdf.**preparedata**(*df_data*, *target*)

Reads data from csv file and transforms it to two PyTorch tensors: dataset x and target time series y that has to be predicted.

tcdf.**train**(*epoch*, *traindata*, *traintarget*, *modelname*, *optimizer*, *log_interval*, *epochs*)

Trains model by performing one epoch and returns attention scores and loss.

CHAPTER

SEVEN

PLOTS MODULE

class plots.**Plots**(*canvas*, *FIGSIZE_X=800*, *FIGSIZE_Y=800*)

Class responsible for plotting and displaying the figures on the Canvas. Also controls interactions with the plots.

clear()

Removes the markers from the plot

get_times()

Times of the markers

on_click(*event*)

Action to be performed when the plot is clicked. Manages the markers in the plot

twinx_canvas(*x*, *x_label*, *y*, *y_label*, *colors='r'*, *t=None*, *t_label='Time'*)

Plot two variables in different y axis

Parameters

- **x** (*pandas.Series*) – First time series
- **x_label** (*str*) – Name of the first time series (will appear on axis)
- **y** (*List[pandas.Series]*) – Secondary time series
- **y_label** (*List[str]*) – Name of the secondary time series (will appear on axis)
- **colors** (`List[str], optional) – Names of the colors to use for each variable (same as matplotlib names). Default: 'r'
- **t** (*iterable, optional*) – Values for the x axis of the plot. Default: None (uses series index as x)
- **t_label** (*str*) – Name of the x axis. Default: 'Time'

update_canvas(*x*, *x_label*, *t=None*, *t_label='Time'*)

Plot variable

Parameters

- **x** (*pandas.Series*) – Time series
- **x_label** (*str*) – Name of the first time series (will appear on axis)
- **t** (*iterable, optional*) – Values for the x axis of the plot. Default: None (uses series index as x)
- **t_label** (*str*) – Name of the x axis. Default: 'Time'

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

T

U

W