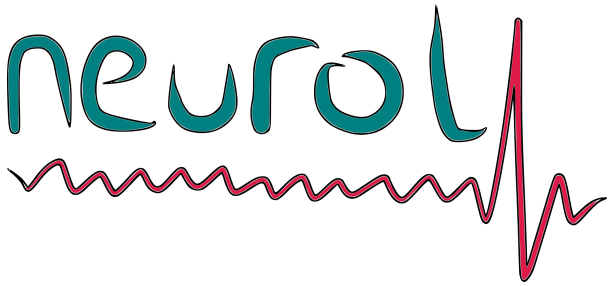
neurol

Release 0.0.5

Feb 10, 2022

Contents:



Welcome! `neurol` is a python package for implementing Brain-Computer Interfaces in a modular manner. With the help of tools in the package, you will be able define the behavior of your intended BCI and easily implement it. A `neurol` BCI is defined by a number of components:

- A `classifier` which decodes brain data into some kind of 'brain-state'
- An `action` which provides feedback depending on the decoded 'brain-state'
- An optional `calibrator` which runs at startup and modifies the operation of the BCI
- An optional `transformer` which transforms the current `buffer` of data into the form expected by the `classifier`

The `neurol` BCI manages an incoming stream of brain data and uses the above user-defined functions to run a brain-computer interface.

The package includes generic utility functions to aid in creating the `classifier`'s, `transfromer`'s, and `calibrator`'s for common BCI use-cases. It also comes prepackaged with a growing list of trained machine learning models for common BCI classification tasks.

CHAPTER 1

neurol package

1.1 neurol.models subpackage

1.1.1 neurol.models.classification_tools module

Module containing functions for performing classification, via machine learning models or otherwise, related to Brain-Computer Interface applications.

neurol.models.classification_tools.**get_channels**(*signal*, *channels*, *device=None*)
Returns a signal with only the desired channels.

Parameters

- **signal** (*np.ndarray*) – a signal of shape [n_samples, n_channels]
- **channels** (*array*) – str names or int indices of the desired channels. returned in given order.
- **device** (*str*) – name of the device. Optional.

Returns numpy array of signal with shape [n_channels, n_desired_channels]. Includes only the selected channels in the order given.

neurol.models.classification_tools.**softmax_predict**(*input_*, *predictor*, *thresh=0.5*)
Consolidates a softmax prediction to a one-hot encoded prediction.

Parameters

- **input** – the input taken by the predictor
- **predictor** – function which returns a softmax prediction given an **input_**
- **thresh** – the threshold for a positive prediction for a particular class.

neurol.models.classification_tools.**encode_ohe_prediction**(*prediction*)
Returns the index number of the positive class in a one-hot encoded prediction.

neurol.models.classification_tools.**decode_prediction**(*prediction*, *decode_dict*)
Returns a more intelligible reading of the prediction based on the given decode_dict

neurol.models.classification_tools.**threshold_clf**(*features*, *threshold*, *clf_consolidator='any'*)
Classifies given features based on a given threshold.

Parameters

- **features** – an array of numerical features to classify
- **threshold** – threshold for classification. A single number, or an array corresponding to *features* for element-wise comparison.
- **clf_consolidator** – method of consolidating element-wise comparisons with threshold into a single classification.

'any': positive class if any features passes the threshold 'all': positive class only if all features pass threshold 'sum': a count of the number of features which pass the threshold function: a custom function which takes in an array of booleans

and returns a consolidated classification

Returns classification for the given features. Return type *clf_consolidator*.

1.1.2 neurol.models.data_exploration module

Module containing functions to study and analyze neural signals, especially to provide insights for building machine learning models to perform classification relevant to Brain-Computer Interface applications.

neurol.models.data_exploration.**plot_signal**(*signal*, *sampling_rate*, *signal_type=None*, *ch_names=None*, *event_timestamps=None*, ***plt_kwargs*)
Plots signal.

Parameters

- **signal** – signal as array of shape [n_samples, n_channels].
- **sr** (*float*) – sampling rate in samples per second.
- **signal_type** – (optional) gives a title for the y-axis.
- **ch_names** – (optional) array of names for each channel (used for legend).
- **event_timestamps** – (optional) 1-D array of times at which an event/stimulus occured.
- ****plt_kwargs** – matplotlib keyword args

neurol.models.data_exploration.**plot_grid**(*signals*, *num_signals=None*, *sampling_rate=1*, *cols=4*, *fig_size=(10*, *6)*, *sharey=True*, *sharex=True*, *random=True*, *fig_axes=None*, *show=True*)
Plot an (optionally random) set of signals [epochs] in a grid from a larger array of given signals.

Parameters

- **signals** – array of signals to plot from (num_signals, num_samples).
- **num_signals** (*int*) – the number of siganls to plot.
- **sampling_rate** (*float*) – sampling rate of signals.
- **cols** (*int*) – the number of columns in the grid.

- **fig_size** – tuple (x,y) of figure size in inches.
- **sharey** (*bool*) – whether to share scale on y-axis (see matplotlib).
- **sharex** (*bool*) – whether to share scale on x-axis (see matplotlib).
- **random** (*bool*) – whether to choose signals randomly or just use the first num_signals.
- **fig_axes** – optionally, an existing tuple of (fig,axes) to plot on (see matplotlib) rather creating new one.
- **show** (*bool*) – whether to show plot inline.

Returns

matplotlib figure and axes with sample of signals plotted in a grid

Return type fig, axes

neurol.models.data_exploration.**stim_triggered_average**(*signal*, *sampling_rate*, *timestamps*, *duration_before*, *duration_after*, *plot=False*)

Inspired by the computational neuroscience concept of the spike-triggered average, this function computes the average signal characteristic around known events.

Parameters

- **signal** – signal as an array of shape [samples, channels].
- **sr** (*float*) – sampling rate of the signal.
- **timestamps** – array of floats containing the timestamps for each event.
- **duration_before** – the duration to be considered before each event.
- **duration_after** – the duration to be considered after each event.
- **plot** (*optional*) – whether or not to plot the stim_triggered_average.

Returns

average signal characteristic around event. relative_time: relative time of each sample in stim_triggered_average

with respect to event.

Return type stim_triggered_average

neurol.models.data_exploration.**plot_PCA**(*epochs*, *sampling_rate=1*, *n_components=None*, *return_PCA=False*, *PCA_kwargs=None*, *plot_grid_kwargs=None*)

performs principal component analysis and plots principal components of epochs of a signal.

Parameters

- **epochs** – array of epochs (n_epochs, n_samples).
- **sr** (*float*) – sampling rate.
- **num_components** (*int*) – number of components to use. If none is passed, all are used.
- **return_PCA** (*bool*) – whether to return the independent components.
- **PCA_kwargs** (*dict*) – dictionary containing kwargs for PCA function (see scikit-learn).
- **plot_grid_kwargs** (*dict*) – dictionary containing kwargs for plot_grid function.

neurol.models.data_exploration.**plot_ICA**(*epochs*, *sampling_rate=1*, *n_components=None*, *return_ICA=False*, *FastICA_kwargs=None*, *plot_grid_kwargs=None*)

performs independent component analysis and plots independent components of epochs of a signal.

Parameters

- **epochs** – array of epochs (n_epochs, n_samples).
- **sr** (*float*) – sampling rate.
- **num_components** (*int*) – number of components to use. If none is passed, all are used.
- **return_ICA** (*bool*) – whether to return the independent components.
- **FastICA_kwargs** (*dict*) – dictionary containing kwargs for FastICA function (see scikit-learn).
- **plot_grid_kwargs** (*dict*) – dictionary containing kwargs for plot_grid function.

1.1.3 neurol.models.model_tools module

Module for managing the models which come pre-packaged with neurol. Includes functionality for importing and using the models.

neurol.models.model_tools.**get_model**(*model_name*)

gets the specified trained model.

Parameters **model_name** (*str*) – name of model. See documentation for list of available models.

Returns trained model.of

Return type model

neurol.models.model_tools.**get_predictor**(*model_name*)

gets the predictor for the specified model.

Parameters **model_name** (*str*) – name of model. See documentation for list of available models.

Returns predictor of trained model.

Return type predictor

1.1.4 neurol.models.preprocessing module

Module containing functions for the preparation of neural data for use with with BCI-related models.

neurol.models.preprocessing.**epoch**(*signal*, *window_size*, *inter_window_interval*)

Creates overlapping windows/epochs of EEG data from a single recording.

Parameters

- **signal** – array of timeseries EEG data of shape [n_samples, n_channels]
- **window_size** (*int*) – desired size of each window in number of samples
- **inter_window_interval** (*int*) – interval between each window in number of samples (measured start to start)

Returns numpy array object with the epochs along its first dimension

neurol.models.preprocessing.**labels_from_timestamps**(*timestamps*, *sampling_rate*, *length*)

takes an array containing timestamps (as floats) and returns a labels array of size 'length' where each index corresponding to a timestamp via the 'samplingRate'.

Parameters

- **timestamps** – an array of floats containing the timestamps for each
- **event** (*units matching sampling_rate*) –
- **sampling_rate** (*float*) – the sampling rate of the EEG data.
- **length** (*int*) – the number of samples of the corresponing EEG recording.

Returns an integer array of size 'length' with a '1' at each time index where a corresponding timestamp exists, and a '0' otherwise.

neurol.models.preprocessing.**label_epochs**(*labels*, *window_size*, *inter_window_interval*, *label_method*)

create labels for individual eoicgs of EEG data based on the label_method.

Parameters

- **labels** – an integer array indicating a class for each sample measurement
- **window_size** (*int*) – size of each window in number of samples (matching window_size in epoched data)
- **inter_window_interval** (*int*) – interval between each window in number of samples (matching inter_window_interval in epoched data)
- **label_method** (*str/func*) – method of consolidating labels contained in epoch into a single label.

  'containment': whether a positive label occurs in the epoch, 'count': the count of positive labels in the epoch, 'mode': the most common label in the epoch func: func_name(epoched_labels) outputs label of epoched_labels

Returns a numpy array with a label correponding to each epoch

neurol.models.preprocessing.**label_epochs_from_timestamps**(*timestamps*, *sampling_rate*, *length*, *window_size*, *inter_window_interval*, *label_method='containment'*)

Directly creates labels for individual windows of EEG data from timestamps of events.

Parameters

- **timestamps** – an array of floats containing the timestamps for each event (units matching sampling_rate).
- **sampling_rate** (*float*) – sampling rate of the recording.
- **length** (*int*) – the number of samples of the corresponing EEG recording.
- **window_size** (*int*) – size of each window in number of samples (matches window_size in epoched data)
- **inter_window_interval** (*int*) – interval between each window in number of samples (matches inter_window_interval in epoched data)
- **label_method** (*str/func*) – method of consolidating labels contained in epoch into a single label.

'containment': whether a positive label occurs in the epoch, 'count': the count of positive labels in the epoch, 'mode': the most common label in the epoch func: func_name(epoched_labels) outputs label of epoched_labels

Returns an array with a label correponding to each window

neurol.models.preprocessing.**epoch_and_label**(*data*, *sampling_rate*, *timestamps*, *window_size*, *inter_window_interval*, *label_method='containment'*)

Epochs a signal (single EEG recording) and labels each epoch using timestamps of events and a chosen labelling method.

Parameters

- **data** – array of timeseries EEG data of shape [n_samples, n_channels]
- **timestamps** – an array of floats containing the timestamps for each event in units of time.
- **sampling_rate** (*float*) – the sampling rate of the EEG data.
- **window_size** (*float*) – desired size of each window in units of time.
- **inter_window_interval** (*float*) – interval between each window in units of time (measured start to start)
- **label_method** (*str/func*) – method of consolidating labels contained in epoch into a single label.

  'containment': whether a positive label occurs in the epoch, 'count': the count of positive labels in the epoch, 'mode': the most common label in the epoch func: func_name(epoched_labels) outputs label of epoched_labels

Returns array of epochs with shape [n_epochs, n_channels] labels: array of labels corresponding to each epoch of shape [n_epochs,]

Return type epochs

neurol.models.preprocessing.**compute_signal_std**(*signal*, *corrupt_intervals=None*, *sampling_rate=1*)

Computes and returns the standard deviation of a signal channel-wise while avoiding corrupt intervals

Parameters

- **signal** – signal of shape [n_samples, n_channels]
- **corrupt_intervals** – an array of 2-tuples indicating the start and end time of the corrupt interval (units of time)
- **sampling_rate** – the sampling rate in units of samples/unit of time

Returns standard deviation of signal channel-wise of shape [1, n_channels]

neurol.models.preprocessing.**split_corrupt_signal**(*signal*, *corrupt_intervals*, *sampling_rate=1*)

Splits a signal with corrupt intervals and returns array of signals with the corrupt intervals filtered out. This is useful for treating each non_corrupt segment as a seperate signal to ensure continuity within a single signal.

Parameters

- **signal** – signal of shape [n_samples, n_channels]
- **corrupt_intervals** – an array of 2-tuples indicating the start and end time of the corrupt interval (units of time)
- **sampling_rate** – the sampling rate in units of samples/unit of time

Returns array of non_corrupt signals of shape [n_signal, n_samples, n_channels]

neurol.models.preprocessing.**epoch_band_features**(*epoch_*, *sampling_rate*, *bands='all'*, *return_dict=True*)

Computes power features of EEG frequency bands over the epoch channel-wise.

Parameters

- **epoch** – a single epoch of shape [n_samples, n_channels]
- **sampling_rate** – the sampling rate of the signal in units of samples/sec
- **bands** – the requested frequency bands to get power features for. 'all': all of ['theta', 'alpha_low', 'alpha_high', 'beta', 'gamma'] otherwise an array of strings of the desired bands.
- **return_dict** (*bool*) – returns *band_features* in the form of a dictionary if True, else returns as numpy array in order of *bands*

Returns a dictionary of arrays of shape [1, n_channels] containing the power features over each frequency band per channel.

1.2 neurol.BCI module

Module implementing a general Brain-Computer Interface which manages and incoming stream of neural data and responds to it in real-time.

class neurol.BCI.**generic_BCI**(*classifier*, *transformer=None*, *action=<built-in function print>*, *calibrator=None*)

Bases: object

Implements a generic Brain-Computer Interface.

Internally manages a buffer of the signal given in *stream* and continuously performs classification on appropriately transformed real-time neural data. At each classification, a corresponding *action* is performed.

Variables

- **classifier** (*function*) – a function which performs classification on the most recent data (transformed as needed). returns classification.
- **transformer** (*function*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects.
- **action** (*function*) – a function which takes in the classification, and performs some action.
- **calibrator** (*function*) – a function which is run on startup to perform calibration using *stream*; returns *calibration_info* which is used by *classifier* and *transformer*.
- **calibration_info** – the result of the calibrator, if applicable.
- **buffer_length** (*int*) – the length of the *buffer*; specifies the number of samples of the signal to keep for classification.
- **brain_state** – the most recent brain state classification.

__init__(*classifier*, *transformer=None*, *action=<built-in function print>*, *calibrator=None*)

Initialize a generic BCI object.

See class documentation for infromation about the class itself.

Parameters

- **classifier** (*function*) – a function which performs classification on the most recent data (transformed as needed). returns class.
- **transformer** (*function*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects.
- **action** (*function*) – a function which takes in the classification, and performs some action.
- **calibrator** (*function*) – a function which is run on startup to perform calibration using *stream*; returns *calibration_info* which is used by *classifier* and *transformer*.

calibrate(*stream*)
runs the *calibrator*.

return value of *calibrator* is stored in the object's *calibration_info* which the *transformer* and *classifier* can use at run-time of BCI.

Parameters **stream** (*neurol.streams object*) – neurol stream for brain data.

run(*stream*)
Runs the defined Brain-Computer Interface.

Internally manages a buffer of the signal given in *stream* and continuously performs classification on appropriately transformed real-time neural data. At each classification, a corresponding *action* is performed.

Parameters **stream** (*neurol.streams object*) – neurol stream for brain data.

test_update_rate(*stream*, *test_length=10*, *perform_action=True*)
Returns the rate at which the BCI is able to make a classification and perform its action.

Parameters

- **stream** (*neurol.streams object*) – neurol stream for brain data.
- **test_length** (*float*) – how long to run the test for in seconds.
- **perform_action** (*bool*) – whether to perform the action or skip it.

class neurol.BCI.**fsm_BCI**(*classifier*, *transformer=None*, *action=<built-in function print>*, *calibrator=None*)
Bases: neurol.BCI.generic_BCI

Implements a Finite-State-Machine-inspired Brain-Computer Interface.

Classification of brain-state is not only dependent on the transformed real-time brain signal, but also the previous brain state.

Internally manages a buffer of the signal given in *stream* and continuously performs classification on appropriately transformed real-time neural data. At each classification, a corresponding *action* is performed.

Variables

- **classifier** (*function*) – a function which performs classification on the most recent data (transformed as needed). returns classification.
- **transformer** (*function*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects.
- **action** (*function*) – a function which takes in the classification, and performs some action.
- **calibrator** (*function*) – a function which is run on startup to perform calibration using *stream*; returns *calibration_info* which is used by *classifier* and *transformer*.
- **calibration_info** – the result of the calibrator, if applicable.

- **buffer_length** (*int*) – the length of the *buffer*; specifies the number of samples of the signal to keep for classification.
- **brain_state** – the most recent brain state classification.

class neurol.BCI.**retentive_BCI**(*classifier*, *transformer=None*, *action=<built-in function print>*, *calibrator=None*, *memory_length=10*)

Bases: neurol.BCI.generic_BCI

Implements a Brain-Computer Interface with memory of past brain states.

Classification of brain-state is not only dependent on the transformed real-time brain signal, but also the finite list of previous brain states.

Internally manages a buffer of the signal given in *stream* and continuously performs classification on appropriately transformed real-time neural data. At each classification, a corresponding *action* is performed.

Variables

- **classifier** (*function*) – a function which performs classification on the most recent data (transformed as needed). returns classification.
- **transformer** (*function*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects.
- **action** (*function*) – a function which takes in the classification, and performs some action.
- **calibrator** (*function*) – a function which is run on startup to perform calibration using *stream*; returns *calibration_info* which is used by *classifier* and *transformer*.
- **calibration_info** – the result of the calibrator, if applicable.
- **brain_state** – the most recent brain state classification.
- **memory_length** (*int*) – number of brain states into the past to remember.
- **past_states** – a list of the past classifications of brain states. used in next classification. length is memory_length.

__init__(*classifier*, *transformer=None*, *action=<built-in function print>*, *calibrator=None*, *memory_length=10*)

Initialize a retentive BCI object.

See class documentation for infromation about the class itself.

Parameters

- **classifier** (*function*) – a function which performs classification on the most recent data (transformed as needed). returns class.
- **transformer** (*function*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects.
- **action** (*function*) – a function which takes in the classification, and performs some action.
- **calibrator** (*function*) – a function which is run on startup to perform calibration using *stream*; returns *calibration_info* which is used by *classifier* and *transformer*.
- **memory_length** (*int*) – number of brain states to remember into past.

class neurol.BCI.**automl_BCI**(*model*, *epoch_len*, *n_states*, *transformer=None*, *action=<built-in function print>*)

Bases: neurol.BCI.generic_BCI

Implements a Brain-Computer Interface which builds its own classifier by training a machine learning model in the calibration stage.

At calibration, data is recorded for some number of brain-states then a machine-learning classifier is trained on the transformed data.

Internally manages a buffer of the signal given in *stream* and continuously performs classification on appropriately transformed real-time neural data. At each classification, a corresponding *action* is performed.

Variables

- **model** – a model object which has fit(X, y) and predict(X) methods.
- **classifier** (*function*) – the model's predictor after training. accepts transformed data and returns classification.
- **transformer** (*function*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects.
- **action** (*function*) – a function which takes in the classification, and performs some action.
- **brain_state** – the most recent brain state classification.

__init__ (*model*, *epoch_len*, *n_states*, *transformer=None*, *action=<built-in function print>*)
Initialize an autoML BCI object.

See class documentation for infromation about the class itself.

Parameters

- **model** – a model object which has fit(X, y) and predict(X) methods.
- **epoch_len** (*int*) – the length of the epochs (in # of samples) used in training and prediction by the model.
- **n_states** (*int*) – the number of brain states being classified.
- **transformer** (*function, optional*) – function which takes in the most recent data (*buffer*) and returns the transformed input the classifer expects. Defaults to None.
- **action** (*function, optional*) – a function which takes in the classification, and performs some action. Defaults to print.

build_model (*stream*, *recording_length*)
records brain signal

Parameters

- **stream** (*neurol.streams object*) – neurol stream for brain data.
- **recording_length** (*float*) – length in seconds for the recording of each brain state to be used for training the model.

run (*stream*)
Runs the defined Brain-Computer Interface.

Internally manages a buffer of the signal given in *stream* and continuously performs classification on appropriately transformed real-time neural data. At each classification, a corresponding *action* is performed.

Parameters **stream** (*neurol.streams object*) – neurol stream for brain data.

1.3 neurol.BCI_tools module

Module including utility functions for creating *classifier*'s, *transfromer*'s, and *calibrator*'s for use in the *BCI* module.

neurol.BCI_tools.**ensemble_transform**(*signal*, *epoch_len=None*, *channels=None*, *device=None*, *transformers=None*, *filter_=False*, *sampling_rate=None*, *filter_kwargs=None*)

Ensemble transform function. Takes in buffer as input. Extracts the appropriate channels and samples, performs filtering, and transforms.

Parameters

- **signal** (*np.ndarray*) – signal of shape: [n_samples, n_channels]
- **epoch_len** (*int*) – length of epoch expected by classifier (# of samples). optional.
- **channels** (*list of str or int*) – list of channels expected by classifier. See get_channels. optional.
- **device** (*str*) – device name. used to get channels and sampling_rate.
- **filter** (*boolean*) – whether to perform filtering
- **filter_kwargs** (*dict*) – dictionary of kwargs passed to filtering function. See biosppy.signals.tools.filter_signal. by default, an order 8 bandpass butter filter is performed between 2Hz and 40Hz.

neurol.BCI_tools.**filter_signal**(*signal*, *sampling_rate*, *ftype='butter'*, *band='bandpass'*, *frequency=(2, 40)*, *order=8*, ***filter_kwargs*)

applies frequency-based filters to a given signal.

Parameters

- **signal** (*np.ndarray*) – signal of shape [n_samples, n_channels]
- **sampling_rate** (*float*) – sampling rate of signal.
- **ftype** (*str, optional*) – type of filter. one of 'FIR', 'butter', 'chebby1', 'chebby2', 'ellip', or 'bessel'. Defaults to 'butter'.
- **band** (*str, optional*) – band type. one of 'lowpass', 'highpass', 'bandpass', or 'bandstop'. Defaults to 'bandpass'.
- **frequency** (*float or tuple of floats, optional*) – cutoff frequencies. single if 'lowpass'/'highpass', tuple if 'bandpass'/'bandstop'. Defaults to (2,40).
- **order** (*int, optional*) – order of filter. Defaults to 8.
- ****filter_kwargs** – keyword args for biosppy.signals.tools.filter_signal

Returns filtered signal

Return type [np.ndarray]

neurol.BCI_tools.**band_power_calibrator**(*stream*, *channels*, *device*, *bands*, *percentile=50*, *recording_length=10*, *epoch_len=1*, *inter_window_interval=0.2*)

Calibrator for *generic_BCI.BCI* which computes a given *percentile* for the power of each frequency band across epochs channel-wise. Useful for calibrating a concentration-based BCI.

Parameters

- **stream** (*neurol.streams object*) – neurol stream for brain data.
- **channels** – array of strings with the names of channels to use.

- **device** (*str*) – device name for use by *classification_tools*
- **bands** – the frequency bands to get power features for. 'all': all of ['theta', 'alpha_low', 'alpha_high', 'beta', 'gamma'] otherwise an array of strings of the desired bands.
- **percentile** – the percentile of power distribution across epochs to return for each band.
- **recording_length** (*float*) – length of recording to use for calibration in seconds.
- **epoch_len** (*float*) – the length of each epoch in seconds.
- **inter_window_interval** (*float*) – interval between each window/epoch in seconds (measured start to start)

Returns array of shape [n_bands, n_channels] of the *percentile* of the power of each band

Return type clb_info

neurol.BCI_tools.**band_power_transformer**(*signal*, *sampling_rate*, *bands*)

Transformer for *generic_BCI.BCI* which chooses channels, epochs, and gets power features on some choice of bands.

Parameters

- **signal** (*np.ndarray*) – most recent stream data. shape: [n_samples, n_channels]
- **sampling_rate** (*float*) – sampling_rate of signal.
- **bands** – the frequency bands to get power features for. 'all': all of ['theta', 'alpha_low', 'alpha_high', 'beta', 'gamma'] otherwise a list of strings of the desired bands.

Returns array of shape [n_bands, n_channels] of the channel-wise power of each band over the epoch.

Return type transformed_signal

1.4 neurol.plot module

Module for plotting stream of neural data. Includes time domain, fourrier transform, and spectrogram live plots.

neurol.plot.**plot**(*stream*, *channels=None*, *w_size=(1920*, *1080)*)

plots data stream. one row per channel.

Parameters

- **stream** (*neurol.streams object*) – neurol stream for a data source.
- **channels** – channels to plot. list/tuple of channel indices, or dict with indices as keys and names as values. Defaults to None (plots all channels w/o names).
- **w_size** (*tuple, optional*) – initial size of window in pixels. Defaults to (1920, 1080).

neurol.plot.**plot_fft**(*stream*, *channels=None*, *w_size=(1920*, *1080)*)

plots fourrier transform of data stream from inlet. one row per channel.

Parameters

- **stream** (*neurol.streams object*) – neurol stream for a data source.
- **channels** – channels to plot. list/tuple of channel indices, or dict with indices as keys and names as values. Defaults to None (plots all channels w/o names).

- **w_size** (*tuple, optional*) – initial size of window in pixels. Defaults to (1920, 1080).

neurol.plot.**plot_spectrogram**(*stream*, *channels=None*, *w_size=(1920, 1080)*)
plots spectrogram of data stream from inlet. one row per channel.

Parameters

- **stream** (*neurol.streams object*) – neurol stream for a data source.
- **channels** – channels to plot. list/tuple of channel indices, or dict with indices as keys and names as values. Defaults to None (plots all channels w/o names).
- **w_size** (*tuple, optional*) – initial size of window in pixels. Defaults to (1920, 1080).

1.5 neurol.connect_device module

Module containing functions for quickly connecting to BCI-related streaming devices.

neurol.connect_device.**connect_muse**()
connects to any available muse headset. returns ble2lsl.ble2lsl.Streamer object.

neurol.connect_device.**get_lsl_EEG_inlets**()
resolves all EEG lsl streams and returns their inlets in an array.

1.6 neurol.streams

module for handling streams of data from different sources

class neurol.streams.**lsl_stream**(*pylsl_inlet*, *buffer_length=2048*)
Bases: object

A generalized stream object for an lsl data source.

Manages a buffer of data and makes it available. Used by neurol.BCI and neurol.plot.

__init__(*pylsl_inlet*, *buffer_length=2048*)
initialize an lsl_stream object.

Parameters

- **pylsl_inlet** (*pylsl.pylsl.StreamInlet*) – inlet of connected lsl device
- **buffer_length** (*int, optional*) – length of data buffer. Defaults to 2048.

get_data(*max_samples=2048*)
gets latest data.

record_data(*duration*)
records from stream for some duration of time.

Parameters **duration** (*float*) – length of recording in seconds.

update_buffer()
updates buffer with most recent available data.

Returns True if new data available, False if not.

Return type [bool]

close()
closes the pylsl inlet stream

CHAPTER 2

Installation

`neurol` can be easily installed using `pip`:

```
$ pip install neurol
```

CHAPTER 3

Indices

- genindex
- modindex
- search

Index

Symbols

A

B

C

D

E

F

G

L

N

P

R

S

T

U