Timeflux rASR

Release 0.1.2.dev1+ga2f343f

Timeflux Team

Jun 08, 2021

CONTENTS

This plugin performs real-time artifact correction on EEG data.

CHAPTER

ONE

INSTALLATION

First, make sure that Timeflux is installed.

You can then install this plugin in the `timeflux` environment:

```
$ conda activate timeflux
$ pip install timeflux_rasr
```

If you encounter issues with `pyRiemann`, try installing the latest version directly from the repository:

```
$ pip install git+https://github.com/pyRiemann/pyRiemann
```

CHAPTER

TWO

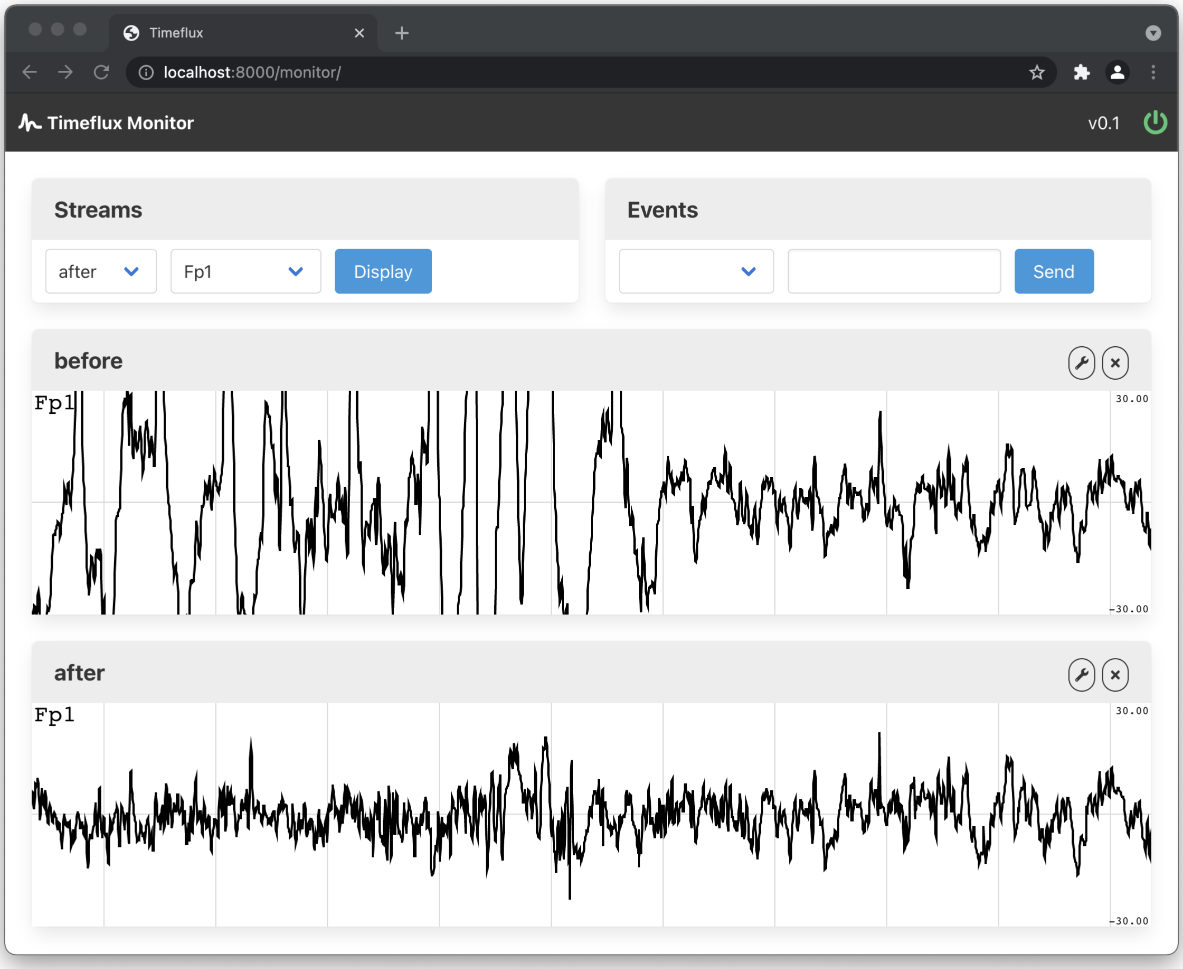
WITNESS THE MAGIC!

```
$ timeflux -d examples/replay.yaml
```

CHAPTER

THREE

SCREENSHOT



CHAPTER

FOUR

REFERENCES

- Blum-Jacobsen paper
- ASR EEGLAB implementation (code, documentation)
- rASR Matlab implementation

4.1 API Reference

This page contains auto-generated API reference documentation.

timeflux_rasr

4.1.1 timeflux_rasr

timeflux_rasr.estimators

estimators

timeflux_rasr.estimators.blending

blending

class `timeflux_rasr.estimators.blending.`**`Blending`**(*window_overlap=1*, *merge=False*, *windowing=None*)

Bases: `sklearn.base.BaseEstimator`, `sklearn.base.TransformerMixin`

Smooths features temporally to avoid discontinuities by sine-cosine blending interpolation.

This estimator smooths features to avoid discontinuities by blending the redundant samples that has been transformed by different operations. Only samples with the exact same timestamp will be blended. Features with an unique timestamp will be left untouched. Therefore this function is useful only if the two following criteria are met: - there is an overlapping between the features of each consecutive observation - the features have been transformed by different operators or by an adaptive operator.

Parameters

- **window_overlap** (*int (default=1)*) – Number of overlapping features that will be used for blending.
- **merge** (*bool (default=False)*) – If set to True, the output will be a unique 2D matrix when all windows will be collapsed and overlapping samples merged. By default, each window will be blended with the previous window.
- **windowing** (*bool | None (default=None)*) – If None or False, there is no interpolation for the first trial. If True, the first trial will be interpolate to zeros (sinus windowing).

Variables

- **n_channels** (*int*) – The dimension managed by the fitted Blending, e.g. number of electrodes.`
- **last_window** (*ndarray, shape (Nt, Ne)*) – The last window observed during the last call of transform. It will be blended with the first window.

fit(*self*, *X*, *y=None*)

Parameters

- **X** (*ndarray, shape (n_trials, n_samples, n_channels)*) – Training data.
- **y** (*ndarray, shape (n_trials,) | None, optional*) – labels corresponding to each trial, not used (mentioned for sklearn comp)

Returns **self** – the fitted Blending estimator.

Return type Blending instance.

transform(*self*, *X*)

Blend signals :param X: Data to clean, already filtered :type X: ndarray, shape (n_trials, n_samples, n_channels)

Returns **Xblended** – Cleaned data

Return type ndarray, shape (n_trials, n_samples, n_channels)

fit_transform(*self*, *X*, *y=None*)

Parameters

- **X** (*ndarray, shape (n_trials, n_samples, n_channels)*) – Training data.
- **y** (*ndarray, shape (n_trials,) | None, optional*) – labels corresponding to each trial, not used (mentioned for sklearn comp)

Returns **X** – blended data

Return type ndarray, shape (n_trials, n_samples, n_channels)

timeflux_rasr.estimators.rasr

rasr

`timeflux_rasr.estimators.rasr.`**`logger`**

`class` `timeflux_rasr.estimators.rasr.`**`RASR`**(*estimator='scm'*, *rejection_cutoff=3.0*, *max_dimension=0.66*, ***kwargs*)

Bases: `sklearn.base.BaseEstimator`, `sklearn.base.TransformerMixin`

RASR Implements this (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6499032/) paper. Matlab code from the author here: https://github.com/s4rify/rASRMatlab

Parameters

- **estimator** (*string (default: 'scm')*) – covariance matrix estimator. For regularization consider 'lwf' or 'oas' For a complete list of estimator, see *pyriemann.utils.covariance*
- **rejection_cutoff** (*float (default: 3.0)*) – Standard deviation cutoff for rejection. Data portions whose variance is larger than this threshold relative to the calibration data are considered missing data and will be removed. The most aggressive value that can be used without losing too much EEG is 2.5. A quite conservative value would be 5.
- **max_dimension** (*float (default: 0.66)*) – Maximum dimensionality of artifacts to remove. Up to this many dimensions (or up to this fraction of dimensions) can be removed for a given data segment. If the algorithm needs to tolerate extreme artifacts a higher value than the default may be used (the maximum fraction is 1.0).
- **max_dropout_fraction** (*float (default: 0.1)*) – Maximum fraction of windows that can be subject to signal dropouts (e.g., sensor unplugged), used for threshold estimation in _fit_eeg_distribution.
- **min_clean_fraction** (*float (default: 0.25)*) – Minimum fraction of windows that need to be clean, used for threshold estimation in _fit_eeg_distribution.
- **quantile_range** – additional parameters passed to _fit_eeg_distribution (should be kept as default in general).
- **step_sizes** – additional parameters passed to _fit_eeg_distribution (should be kept as default in general).
- **beta_range** – additional parameters passed to _fit_eeg_distribution (should be kept as default in general).

Variables

- **Ne** (*int*) – The dimension managed by the fitted RASR, e.g. number of electrodes.
- **mixing** (*ndarray, shape(n_chan, n_chan)*) – Mixing matrix computed from geometric median covariance matrix U such as: $mixing = M : M * M = U$
- **threshold** (*ndarray, shape(n_chan,)*) – Threshold operator used to find the subspace dimension such as: $threshold = T : X_{clean} = m(V_{clean}^T M)^+ V^T X$

Init.

`fit`(*self*, *X*, *y=None*)

Parameters

- **X** (*ndarray, shape (n_trials, n_samples, n_channels)*) – Training data, already filtered.
- **y** (*ndarray, shape (n_trials,) | None, optional*) – labels corresponding to each trial, not used (mentioned for sklearn comp)

Returns **self** – the fitted RASR estimator.

Return type RASR instance.

`transform`(*self*, *X*)

Clean signal :param X: Data to clean, already filtered :type X: ndarray, shape (n_trials, n_samples, n_channels)

Returns **Xclean** – Cleaned data

Return type ndarray, shape (n_trials, n_samples, n_channels)

`fit_transform`(*self*, *X*, *y=None*)

Parameters

- **X** (*ndarray, shape (n_trials, n_samples, n_channels)*) – Training data.
- **y** (*ndarray, shape (n_trials,) | None, optional*) – labels corresponding to each trial, not used (mentioned for sklearn comp)

Returns **X** – Cleaned data

Return type ndarray, shape (n_trials, n_samples, n_channels)

PYTHON MODULE INDEX

INDEX