Available online at www.sciencedirect.com

ScienceDirect

SoftwareX ▮ (▮▮▮▮) ▮▮▮–▮▮▮

SoftwareX

www.elsevier.com/locate/softx

PySpike—A Python library for analyzing spike train synchrony

Mario Mulansky*, Thomas Kreuz

Institute for Complex Systems, CNR, Via Madonna del Piano 10 – 50019 Sesto Fiorentino, Italy

Received 10 March 2016; received in revised form 14 July 2016; accepted 15 July 2016

Abstract

Understanding how the brain functions is one of the biggest challenges of our time. The analysis of experimentally recorded neural firing patterns (spike trains) plays a crucial role in addressing this problem. Here, the PySpike library is introduced, a Python package for spike train analysis providing parameter-free and time-scale independent measures of spike train synchrony. It allows to compute similarity and dissimilarity profiles, averaged values and distance matrices. Although mainly focusing on neuroscience, PySpike can also be applied in other contexts like climate research or social sciences. The package is available as Open Source on Github and PyPI.

Keywords: Synchrony; Spike train analysis; Spike train distance; Python

Code metadata

Current code version	v0.5.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-16-00032
Legal Code License	BSD License
Code versioning system used	git
Software code languages, tools, and services used	Python, Cython
Compilation requirements, operating environments & dependencies	`numpy`, `cython`, `matplotlib`, `nosetests`
If available Link to developer documentation/manual	http://www.pyspike.de
Support email for questions	mario.mulansky@gmx.net

1. Introduction

Gaining insight into the inner workings of the brain remains a largely unsolved challenge that requires combined efforts of biophysics, medicine, experimental as well as computational neuroscience [1]. The basis for scientific advancement in this field are experimental recordings of neural activity usually represented in terms of spike trains, i.e. lists of spike times for each recorded neuron. With sophisticated modern recording techniques, it is now possible to perform highly parallel measurements of neural activity, typically resulting in very large sets of spike trains [2,3]. This generates an increased demand for powerful and high quality data analysis tools that are capable of processing large datasets as produced by parallel recordings.

There exist numerous methods to analyze spike train data, e.g. based on spike count distributions, interspike intervals or exact spike times. One very important approach is to quantify the synchrony between spike trains. In the past decades several synchrony measures have been proposed [4–6] which have already been used, among others, to quantify the reliability of neuronal responses [7], to analyze the role of spike synchronization in feature binding [8], and to distinguish different stimuli in the context of neuronal coding [1].

The PySpike library[1] introduced here (logo shown in Fig. 1) is a Python package that allows one to compute two

* Corresponding author.
E-mail addresses: mario.mulansky@isc.cnr.it (M. Mulansky), thomas.kreuz@cnr.it (T. Kreuz).

[1] www.pyspike.de.

http://dx.doi.org/10.1016/j.softx.2016.07.006
2352-7110/

Fig. 1. Logo of the PySpike library.

different dissimilarity measures, the ISI-distance [9], the SPIKE-distance [10] and additionally the similarity measure SPIKE-Synchronization [11,12]. Each of these three methods is *time-resolved*, *parameter-free* and *time-scale independent* and therefore highly versatile. Being time-resolved, for example, means these measure can detect changes in synchrony over time, while being parameter-free makes them readily applicable with unambiguous results, as no parameter optimization is required. These measures have already been applied in many experimental studies in the past, for example [13–16].

Although developed with a neuroscientific context in mind, the synchrony measures discussed here can be applied to any form of discrete time series consisting of event sequences of any kind. In fact, such measures have already been utilized in several other research areas, such as climate research [17] or social sciences [18,19].

PySpike is a library aimed to perform automatized data analysis with Python scripts. It is therefore a complementary approach to the SPIKY software package,[2] a Matlab framework for spike train analysis providing a similar functionality but additionally offering a sophisticated GUI [11,20]. Several other software packages for spike train analysis have been developed in the recent past, notably SyncPy[3] [21], a Python based GUI for quantifying synchrony in time series. However, it currently does not include the synchrony measures implemented in PySpike. A C++ implementation of the spike train distance measures[4] was presented in [22], but it is based on sampled data and therefore of substantially inferior performance [11]. Finally, a comprehensive collection of scientific software for spike train analysis is also provided as part of [23], aiming specifically at multivariate recordings.[5]

2. Spike train distances

Here, discrete time series are represented by *spike trains*, sequences of time points denoting the occurrence of an event (spike) at those time points: $s = \{t_1, t_2, t_3, \ldots\}$. Generally, a time-resolved distance measure maps a pair of spike trains s_1, s_2 onto a profile $\{s_1, s_2\} \rightarrow S(t)$ with $0 \leq S(t) \leq 1$. The overall distance value can easily be obtained by integration: $D_S = \int S(t)\,\mathrm{d}t$.

PySpike provides three such distance measures: ISI-distance, SPIKE-distance and SPIKE-Synchronization. These methods are sensitive to different aspects of spike train synchrony (interspike intervals, exact spike timings, spike matching, respectively). Hence, the choice of method should be informed by assumptions on how information is encoded in the spike trains. In the following, we give a brief introduction to the measures provided in the PySpike library. For a detailed discussion of the methods and their properties see Appendix A and [12].

The **ISI-distance** profile $I(t)$, introduced in [9], quantifies dissimilarity in terms of the relative differences of the concurrent interspike intervals of the two spike trains. Essentially, it measures the relative differences of the instantaneous rates of the two spike trains, but it is not sensitive to exact spike timings. The ISI-distance profile is a bivariate piecewise constant function.

The **SPIKE-distance** profile $S(t)$, first introduced in [24] and refined in [10], represents a dissimilarity profile based on exact spike timings. Thus, the SPIKE-distance quantifies spike train dissimilarity in terms of deviations from exact coincidences of spikes in the two spike trains. This results in a bivariate piecewise linear profile.

While the fundamental definition of both the ISI- and the SPIKE-distance profile is bivariate (distance profile of two spike trains), the generalization to a multivariate context is a straightforward average over all spike train pairs [25].

SPIKE-Synchronization [11,12] is a straight-forward, normalized coincidence counter with an adaptive coincidence window. It quantifies similarity in terms of the fraction of coincidences between two spike trains and hence is a very intuitive measure. The generalization of SPIKE-Synchronization for many spike trains can be defined based on all spike train pairs leading to a consistent multivariate framework with similarity again quantified as the overall fraction of coincidences in all spike trains [11,12].

3. Package structure

3.1. The spike train

The central data structure of the PySpike library is the `SpikeTrain`, a Python class representing an individual spike train. This class contains the (sorted) spike times as a `numpy.array` as well as the start and end time of the spike train. Such `SpikeTrain` objects can either be created directly by providing the spike times, generated randomly from a Poisson process using the `generate_poisson_spikes` function, imported from text files via `load_spike_trains_from_txt` or imported from time series via `import_spike_trains_from_time_series`. These objects then serve as input to calculate the distance measures, cf. Fig. 2.

3.2. Computing profiles

Being time-resolved is a main advantage of the three spike train synchrony measures discussed here. Hence, PySpike contains functionality to compute synchrony profiles: `isi_profile` computes the piecewise constant ISI-profile $I(t)$, `spike_profile` returns the piecewise linear SPIKE-profile $S(t)$ and `spike_sync_profile` yields the discrete

[2] http://www.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/SPIKY.html.
[3] https://github.com/syncpy/SyncPy.
[4] https://github.com/modulus-metric.
[5] http://spiketrain-analysis.org/software.