Journal of Machine Learning Research 21 (2020) 1-6 Submitted 9/19; Revised 12/19; Published 2/20

pyts: A Python Package for Time Series Classification

Johann Faouzi JOHANN.FAOUZI@ICM-INSTITUTE.ORG
Aramis Lab, INRIA Paris
Brain and Spine Institute
75013 Paris – France

Hicham Janati HICHAM.JANATI@INRIA.FR
Parietal team, INRIA Saclay
Neurospin, Bât 145 CEA Saclay
91191 Gif sur Yvette – France

Editor: Andreas Mueller

Abstract

`pyts` is an open-source Python package for time series classification. This versatile toolbox provides implementations of many algorithms published in the literature, preprocessing functionalities, and data set loading utilities. `pyts` relies on the standard scientific Python packages `numpy`, `scipy`, `scikit-learn`, `joblib`, and `numba`, and is distributed under the BSD-3-Clause license. Documentation contains installation instructions, a detailed user guide, a full API description, and concrete self-contained examples. Source code and documentation can be downloaded from `https://github.com/johannfaouzi/pyts`.

Keywords: time series, classification, machine learning, python

1. Introduction

A time series is a sequence of values indexed in chronological order. Given their structured nature, they are very common in many real-world applications. With the increasing availability of sensors and the development of *Internet of things* devices, the amount of time series data and the number of applications is continuously growing. Applications can be found in a virtually unlimited number of fields ranging from finance and econometrics (Perron, 1989) to geology with earthquake prediction (Amei et al., 2012), and climate science with weather forecasting (Agrawal et al., 2012).

In machine learning, exploiting the structural patterns of the data in order to assign a certain label to a new observation is known as classification. Classifying time series data can solve several real-world problems including disease detection using electrocardiogram data (Olszewski, 2001), household device classification to reduce carbon footprint, and image classification (Ratanamahatana and Keogh, 2005). However, standard machine learning classification is not always well suited for time series because of the possibly high correlation between back-to-back time points. One typical example is the Naive Bayes algorithm, which assumes a conditional independence between each feature given the class the time series belongs to. For this reason, algorithms dedicated to time series classification have been developed such as bag-of-words models, shapelet-based approaches and modifications to standard tree-based algorithms.

In this paper we present `pyts`, a Python package for time series classification. To the best of our knowledge, among several Python packages for time series analysis, `pyts` is the only Python package entirely dedicated to time series classification. The following sections describe the project development, the dependencies, the API design, a comparison to related Python packages and future works.

2. Project Development

Code quality. The package comes up with a lot of unit tests, leading to 97% code coverage for the release of version 0.10. Moreover, analysis tools such as `PEP8` and `Flake8` are used and LGTM[1] gives the package an $A+$ grade for code quality.

Continuous integration. In order to ensure new code integration and backward compatibility, continuous integration is performed on all supported platforms.

Collaborative development. The package is hosted on GitHub, which allows for collaboration between the developers and any contributor, discussion on eventual issues, and feature requests.

Documentation. A detailed documentation is hosted on Read the Docs with installation instructions, a comprehensive user guide, the whole API documentation and numerous examples.

3. Dependencies

NumPy: the base N-dimensional array package. `numpy` (Walt et al., 2011) provides among other things a powerful N-dimensional array object, linear algebra operations, Fourier transform, and random number capabilities.

SciPy: the fundamental library for scientific computing. `scipy` (Jones et al., 2001–) contains mathematical algorithms, statistical tools, sparse matrices, and convenience functions built on top of `numpy`.

Scikit-learn: machine learning in Python. `scikit-learn` (Pedregosa et al., 2011) is a versatile toolbox for data mining and data analysis consisting of many machine learning algorithms and utility tools.

Joblib: running Python functions as pipeline jobs. `joblib` is a set of tools to provide lightweight pipelining in Python, such as transparent disk-caching of functions and simple parallel computing.

Numba: a high performance Python compiler. `numba` (Lam et al., 2015) translates Python functions to optimized machine code at run time using the LLVM compiler library. Numba-compiled numerical algorithms in Python can reach the speeds of C or FORTRAN.

4. API Design

As illustrated in Listing 1, the `pyts` API is inspired by the `scikit-learn` API and is designed to be compatible with `scikit-learn` tools such as cross validation, model selection,

1. `https://lgtm.com/projects/g/johannfaouzi/pyts?mode=list`

```python
from pyts.classification import BOSSVS
from pyts.datasets import load_gunpoint

# Load the GunPoint data set
X_train, X_test, y_train, y_test = load_gunpoint(return_X_y=True)

# Create the classifier
clf = BOSSVS(window_size=28)

# Fit the classifier on the training set
clf.fit(X_train, y_train)

# Compute the accuracy on the test set
accuracy = clf.score(X_test, y_test)

# Print accuracy
print(accuracy)
0.98
```

Listing 1: Code snippet illustrating `pyts`'s intuitive API on a classification example.

and pipelines. All transformers have three main methods: `fit` derives statistics on the data set if applicable, `transform` computes the transformation with the learned statistics, and `fit_transform` is an optimized version of `fit` followed by `transform` when possible. All classifiers have two main methods: `fit` learns statistics from the data set and `predict` derives predicted classes.

5. Assumptions on Input Data

One important challenge with time series is their number of time points. A data set of equal-length time series consists of time series that all have the same number of time points. A data set of varying-length time series consists of time series that may have different numbers of time points. For computational efficiency, most algorithms implemented in `pyts` can only deal with data sets of equal-length time series. One exception is our implementation of Dynamic Time Warping (Sakoe and Chiba, 1978) and its variants.

Data sets of univariate and multivariate are both supported, being represented as two-dimensional arrays and three-dimensional arrays respectively.

6. Comparison to Related Software

`pyts` is not the only Python package providing tools for time series classification. `sktime` (Lning et al., 2019) and `tslearn` (Tavenard, 2017) are dedicated to time series analysis in general, while `tsfresh` (Christ et al., 2018), `cesium` (Naul et al., 2016) and `seglearn` (Burns and Whyne, 2018) are focused on extracting statistics-based features from time series. Table 1 provides an extensive comparison of available algorithms for time series classification in these packages. `sktime` and `pyts` stand out with their numbers of implementations of

Algorithms	pyts	sktime	tslearn	seglearn	tsfresh	cesium
Feature Extraction	✘	✘	✘	✔	✔	✔
Dynamic Time Warping	✔	✔	✔	✘	✘	✘
Support Vector Machine	✘	✘	✔	✘	✘	✘
Learning Shapelet	✘	✘	✔	✘	✘	✘
Shapelet Transform	✔	✔	✘	✘	✘	✘
SAX-VSM	✔	✘	✘	✘	✘	✘
BOSS	✔	✔	✘	✘	✘	✘
BOSSVS	✔	✘	✘	✘	✘	✘
WEASEL	✔	✘	✘	✘	✘	✘
WEASEL+MUSE	✔	✘	✘	✘	✘	✘
Recurrence Plot	✔	✘	✘	✘	✘	✘
Gramian Angular Field	✔	✘	✘	✘	✘	✘
Markov Transition Field	✔	✘	✘	✘	✘	✘
Time Series Forest	✘	✔	✘	✘	✘	✘
Proximity Forest	✘	✔	✘	✘	✘	✘
Elastic Ensemble	✘	✔	✘	✘	✘	✘
RISE	✘	✔	✘	✘	✘	✘

Table 1: Availability of several algorithms for time series classification published in the literature. Packages (versions): `pyts` (0.10.0), `sktime` (0.3.1), `tslearn` (0.2.5), `seglearn` (1.1.0), `tsfresh` (0.13.0), `cesium` (0.9.10).

	Adiac	ECG200	GunPoint	MiddlePhalanxTW	Plane
(Schäfer, 2015)	0.765	0.870	1.000	0.526	1.000
pyts	0.752	0.870	1.000	0.526	1.000

Table 2: Accuracy scores on the test set for the BOSS transformer followed by a one-nearest neighbor classifier with the BOSS metric on several data sets.

algorithms published in the literature. `sktime` provides more tree-based algorithms, while `pyts` is more focused on dictionary-based and image-based models.

In order to provide users confidence in our implementations, we also track the predictive performance of our implementations on some data sets and compare them to the results published in the literature. Table 2 highlights the accuracy scores on the test set for the BOSS transformer (Schäfer, 2015) followed by a one-nearest neighbor classifier with the BOSS metric (Schäfer, 2015) on several data sets.

7. Conclusion

The `pyts` Python package is a versatile toolbox for time series classification, providing implementations of several algorithms published in the literature, preprocessing tools, and data set loading utilities. A detailed documentation and concrete examples illustrate the

use of the functionalities made available. Future works include better support for data sets of unequal-length time series and multivariate time series.

References

A. Agrawal, V. Kumar, A. Pandey, and I. Khan. An application of time series analysis for weather forecasting. *International Journal of Engineering Research and Applications*, 2: 974–980, 2012.

A. Amei, W. Fu, and C. H. Ho. Time series analysis for predicting the occurrences of large scale earthquakes. *International Journal of Applied Science and Technology*, 2, 2012.

D. M. Burns and C. M. Whyne. Seglearn: A python package for learning sequences and time series. *Journal of Machine Learning Research*, 19(83):1–7, 2018.

M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307: 72–77, 2018.

E. Jones, T. Oliphant, Peterson P., et al. Scipy: Open source scientific tools for python, 2001–. URL `http://www.scipy.org/`.

S. K. Lam, A. Pitrou, and S. Seibert. Numba: A LLVM-based python JIT compiler. In *Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 7:1–7:6. ACM, 2015.

Markus Lning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Kirly. sktime: A unified interface for machine learning with time series, 2019.

B. Naul, S. van der Walt, A. Crellin-Quick, J. Bloom, and F. Prez. cesium: Open-source platform for time-series inference. In *Python in Science Conference*, pages 27–35, 2016.

R. Olszewski. *Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data*. PhD thesis, Carnegie Mellon University, 2001.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

P. Perron. The great crash, the oil price shock, and the unit root hypothesis. *Econometrica*, 57(6):1361–1401, 1989.

C. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *International Conference on Data Mining*, pages 506–510. Society for Industrial and Applied Mathematics, 2005.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

Patrick Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, November 2015. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-014-0377-7.

R. Tavenard. tslearn: A machine learning toolkit dedicated to time-series data, 2017. URL `https://github.com/rtavenar/tslearn`.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, 2011.