Tslearn, A Machine Learning Toolkit for Time Series Data

Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Russwurm, Kushal Kolar, et al.

Journal of Machine Learning Research 21 (2020) 1-6 Submitted 1/20; Revised 4/20; Published 4/20

Tslearn, A Machine Learning Toolkit for Time Series Data

Romain Tavenard ROMAIN.TAVENARD@UNIV-RENNES2.FR
Université de Rennes, CNRS, LETG-Rennes, IRISA-Obelix, Rennes, France

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

Gilles Vandewiele GILLES.VANDEWIELE@UGENT.BE
IDLab, Ghent University – imec, Ghent, Belgium

Felix Divo FELIX.DIVO@STUD.TU-DARMSTADT.DE
Technische Universität Darmstadt, Darmstadt, Germany

Guillaume Androz GUILLAUME.ANDROZ@ICENTIA.COM
Icentia Inc., Québec, Canada

Chester Holtz CHHOLTZ@ENG.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, USA

Marie Payne MARIE.PAYNE@MAIL.MCGILL.CA
McGill University, Montreal, Québec, Canada

Roman Yurchak ROMAN.YURCHAK@SYMERIO.COM
Symerio, Paris, France

Marc Rußwurm MARC.RUSSWURM@TUM.DE
Technical University of Munich, Chair of Remote Sensing Technology, Munich, Germany

Kushal Kolar KUSHALKOLAR@GMAIL.COM
Sars International Centre for Marine Molecular Biology, University of Bergen, Norway

Eli Woods ELI@EAZE.COM
Eaze Technologies, Inc., San Francisco, CA, USA

Editor: Alexandre Gramfort

Abstract

`tslearn` is a general-purpose Python machine learning library for time series that offers tools for pre-processing and feature extraction as well as dedicated models for clustering, classification and regression. It follows `scikit-learn`'s Application Programming Interface for transformers and estimators, allowing the use of standard pipelines and model selection tools on top of `tslearn` objects. It is distributed under the BSD-2-Clause license, and its source code is available at `https://github.com/tslearn-team/tslearn`.

Keywords: time series, clustering, classification, pre-processing, data mining

1. Introduction

Temporal data are ubiquitous in many application domains, such as medicine, robotics, or videos. Dealing with such data requires dedicated methods that take into account the

high correlation between consecutive samples in a time series. Moreover, in many cases, one would like a time series approach to encode invariance to small time shifts, which once again implies using specific methodologies. `tslearn` is a Python package that provides tools to fit relevant models to time series data. In the following, time series data is understood as series of features collected over time. It includes pre-processing routines, feature extractors, and machine learning models for classification (Bagnall et al., 2017; Fawaz et al., 2019), regression and clustering (Aghabozorgi et al., 2015). Both univariate and multivariate time series can be handled in `tslearn`. Further, data sets can contain time series of variable-length, as discussed below. `tslearn` follows `scikit-learn`'s API for transformers and estimators, allowing the use of `scikit-learn`'s pipelines and model selection tools on `tslearn` objects. We use continuous integration tools to test each contribution to the library and target a coverage of at least 90% to detect bugs as early as possible. Online documentation is available at `https://tslearn.readthedocs.io/` and semantic versioning is used (this document describes version 0.3.1). `tslearn` is distributed under the BSD-2-Clause license. In the following, we present basic usage of the library, an overview of the implemented algorithms and a comparison to other related software.

2. Implementation Description

`tslearn` v0.3.1 is a cross-platform software package for Python 3.5+. It depends on `numpy` (Van Der Walt et al., 2011) & `scipy` (Virtanen et al., 2020) packages for basic array manipulations and standard linear algebra routines and on `scikit-learn` (Pedregosa et al., 2011) for its API and utilities. It also utilizes `Cython` (Behnel et al., 2011), `numba` (Lam et al., 2015) and `joblib` (Varoquaux et al., 2010) for efficient computation. Finally, `keras` (Chollet et al., 2015) with `tensorflow` (Abadi et al., 2016) backend is an optional dependency that is necessary to use the `shapelets` module in `tslearn` that provides an efficient implementation of the shapelet model by Grabocka et al. (2014).

In `tslearn`, a time series data set can be represented through a three-dimensional `numpy` array of shape (n, T, d) where n is the number of time series in the set, T their length, and d their dimensionality. If time series from the set are not equal-sized, `NaN` values are appended to the shorter ones and T is hence the maximum of sizes for time series in the set:

```
from tslearn.utils import to_time_series_dataset
my_first_time_series = [1, 3, 4, 2]
my_second_time_series = [1, 2, 4, 2, 1]
formatted_dataset = to_time_series_dataset([my_first_time_series,
                                            my_second_time_series])
# formatted_dataset.shape is then (2, 5, 1)
```

Then, data can be fed to transformers that are available in the `preprocessing` and `piecewise` modules and/or estimators that follow the `scikit-learn` API[1]. Note that `scikit-learn`'s pipelines and model-selection tools can be used in conjunction with `tslearn` transformers and estimators, as shown in the code snippet below:

1. One noticeable difference when compared with `scikit-learn` estimators and transformers, though, is that `tslearn` ones expect 3-dimensional arrays which can contain `NaN` values, as this is the basic format for `tslearn` data sets.

```
from sklearn.model_selection import KFold, GridSearchCV
from tslearn.neighbors import KNeighborsTimeSeriesClassifier

knn = KNeighborsTimeSeriesClassifier(metric="dtw")
p_grid = {"n_neighbors": [1, 5]}
cv = KFold(n_splits=2, shuffle=True, random_state=0)
clf = GridSearchCV(estimator=knn, param_grid=p_grid, cv=cv)
clf.fit(X, y)
```

3. Contents

Our package first offers basic utilities to format data sets for use with `tslearn` transformers and estimators, and to cast time series data sets from and to other Python time series toolkit formats. It also provides standard pre-processing techniques and feature extraction methods, implemented as `scikit-learn`-compatible transformers.

Many of our learning algorithms rely on the use of time series specific metrics that are themselves made available in a dedicated `metrics` module.

In terms of machine learning methods, we provide implementations of clustering algorithms (some of which rely on barycenter computation routines that are also part of our public API) specifically tailored for temporal data. Supervised learning methods (for regression and classification) are also provided. All these are implemented as `scikit-learn`-compatible estimators.

The importance of providing time-series specific methods for machine learning is illustrated in the example below and the corresponding Figure 1, where standard Euclidean k-means fails while DTW-based ones (Sakoe and Chiba, 1978; Petitjean et al., 2011; Cuturi and Blondel, 2017) can distinguish between different time series profiles:

```
from tslearn.clustering import TimeSeriesKMeans
from tslearn.datasets import CachedDatasets

# Load the 'Trace' data set
X_train = CachedDatasets().load_dataset('Trace')[0]

# Define parameters for each metric
euclidean_params = {'metric': 'euclidean'}
dba_params = {'metric': 'dtw'}
sdtw_params = {'metric': 'softdtw', 'metric_params': {'gamma': .01}}

# Perform clustering for each metric
y_preds = []
for params in (euclidean_params, dba_params, sdtw_params):
    km = TimeSeriesKMeans(n_clusters=3, random_state=0, **params)
    y_preds.append(km.fit_predict(X_train))
```

4. Comparison to Related Software

`tslearn` is not the only Python package to focus on machine learning for time series. Some packages, like `cesium-ml`[2] and `seglearn` (Burns and Whyne, 2018) focus on pre-

2. The `cesium-ml` package can be found here: `http://cesium-ml.org`.

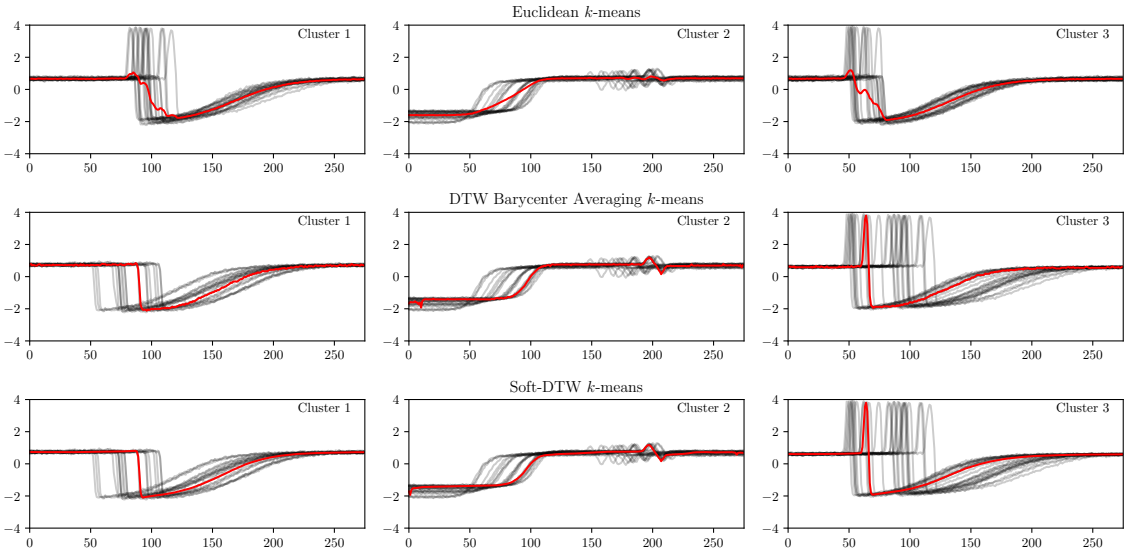

Figure 1: k-means clustering ($k = 3$) using different base metrics. Each graph represents a cluster (*i.e.* a different `y_preds` value), with its centroid plotted in bold red.

processing time series data to feed `scikit-learn` models. Similarly, `tsfresh` (Christ et al., 2018) specializes in feature extraction from time series. `pyts` (Faouzi and Janati, 2020) and `sktime` (Löning et al., 2019), on the other hand, focus on supervised learning. Other packages focus on some families of methods, such as `statsmodel` (Seabold and Perktold, 2010) that provides standard statistical models for time series analysis, `hmmlearn`[3] that focuses on Hidden Markov Models, `stumpy` (Law, 2019) that relies on the matrix profile data structure (Yeh et al., 2016), or `pyflux` (Taylor, 2016), which offers a large panel of probabilistic models aimed at forecasting.

Compared to these packages, `tslearn` is a general-purpose machine learning library for time series that offers pre-processing and feature extraction tools as well as dedicated models for clustering, classification and regression. Note however that, since the packages listed above are more focused than `tslearn`, they tend to incorporate a wider range of algorithms for their task of interest than our library. This is why we have included utilities to cast data sets between `tslearn` format and the ones used by these libraries, in order to help facilitate interoperability.

5. Conclusion

`tslearn` is a general-purpose Python machine learning library for time series. It implements several standard estimators for time series for problems such as clustering, classification and regression. It is under active development with aim at the integration of additional methods.

3. The `hmmlearn` package can be found here: `https://github.com/hmmlearn/hmmlearn`.

Specific attention will be paid to keep `tslearn`'s genericity rather than focus, for example, on supervised settings for which other tools exist.

Acknowledgments

Romain Tavenard would like to acknowledge support for this project from the Agence Nationale de la Recherche (ANR grant ANR-18-CE23-0006).

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

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.

F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series FeatuRe extraction on basis of scalable hypothesis tests. *Neurocomputing*, 307:72–77, 2018.

M. Cuturi and M. Blondel. Soft-DTW: a differentiable loss function for time-series. In *Proceedings of the International Conference on Machine Learning*, pages 894–903, 2017.

J. Faouzi and H. Janati. pyts: A python package for time series classification. *Journal of Machine Learning Research*, 21(46):1–6, 2020.

H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 392–401, 2014.

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

S. M. Law. STUMPY: A powerful and scalable Python library for time series data mining. *Journal of Open Source Software*, 4(39):1504–1505, 2019.

M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király. sktime: A Unified Interface for Machine Learning with Time Series. In *Proceedings of the NeurIPS Workshop on Systems for Machine Learning*, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

F. Petitjean, A. Ketterlin, and P. Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

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.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *Python in Science Conference*, 2010.

R. Taylor. PyFlux: An open source time series library for Python, 2016.

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.

G. Varoquaux et al. Joblib: running Python functions as pipeline jobs. `https://github.com/joblib/joblib`, 2010.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. Matrix Profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *Proceedings of the International Conference on Data Mining*, pages 1317–1322, Dec 2016.