

Seglearn: A Python Package for Learning Sequences and Time Series

David M. Burns
Sunnybrook Research Institute

D.BURNS@UTORONTO.CA

Cari M. Whyne
Sunnybrook Research Institute
2075 Bayview Ave. Room S620.
Toronto, ON, Canada. M4N 3M5.

CARI.WHYNE@SUNNYBROOK.CA

Editor: Alexandre Gramfort

Abstract

`seglearn` is an open-source Python package for performing machine learning on time series or sequences. The implementation provides a flexible pipeline for tackling classification, regression, and forecasting problems with multivariate sequence and contextual data. Sequences and series may be learned directly with deep learning models or via feature representation with classical machine learning estimators. This package is compatible with `scikit-learn` and is listed under `scikit-learn` "Related Projects". The package depends on `numpy`, `scipy`, and `scikit-learn`. `seglearn` is distributed under the BSD 3-Clause License. Documentation includes a detailed API description, user guide, and examples. Unit tests provide a high degree of code coverage. Source code and documentation can be downloaded from <https://github.com/dmbee/seglearn>.

Keywords: Machine-Learning, Time-Series, Sequences, Python

1. Introduction

Many real-world machine learning problems e.g. voice recognition, human activity recognition, power systems fault detection, stock price and temperature prediction, involve data that is captured as sequences over a period of time (Aha, 2018). Sequential data sets do not fit the standard supervised learning framework, where each sample (\mathbf{x}, y) within the data set is assumed to be independently and identically distributed (iid) from a joint distribution $P(\mathbf{x}, y)$ (Bishop, 2011). Instead, the data consist *sequences* of (\mathbf{x}, y) pairs, and nearby values of (\mathbf{x}, y) within a *sequence* are likely to be correlated to each other. Sequence learning exploits the sequential relationships in the data to improve algorithm performance.

2. Supported Problem Classes

Sequence data sets have a general formulation (Dietterich, 2002) as sequence pairs $\{(\mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^N$, where each \mathbf{X}_i is a multivariate sequence with T_i samples $\langle \mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \dots, \mathbf{x}_{i,T_i} \rangle$ and each \mathbf{y}_i target is a univariate sequence with T_i samples $\langle y_{i,1}, y_{i,2}, \dots, y_{i,T_i} \rangle$. The targets \mathbf{y}_i can ei-

ther be sequences of categorical class labels (for classification problems), or sequences of continuous data (for regression problems). The number of samples T_i varies between the sequence pairs in the data set. Time series with a regular sampling period may be treated equivalently to sequences. Irregularly sampled time series are formulated with an additional sequence variable \mathbf{t}_i that increases monotonically and indicates the timing of samples in the data set $\{(\mathbf{t}_i, \mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^N$.

Important sub-classes of the general sequence learning problem are sequence classification and sequence prediction. In sequence classification problems (eg song genre classification), the target for each sequence is a fixed class label y_i and the data takes the form $\{(\mathbf{X}_i, y_i)\}_{i=1}^N$. Sequence prediction involves predicting a future value of the target ($y_{i,t+f}$) or future values $\langle y_{i,t+1}, y_{i,t+2}, \dots, y_{i,t+f} \rangle$, given $\langle \mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \dots, \mathbf{x}_{i,t} \rangle$, $\langle y_{i,1}, y_{i,2}, \dots, y_{i,t} \rangle$, and sometimes also $\langle \mathbf{x}_{i,t+1}, \mathbf{x}_{i,t+2}, \dots, \mathbf{x}_{i,t+f} \rangle$.

A final important generalization is the case where contextual data associated with each sequence, but not varying within the sequence, exists to support the machine learning algorithm performance. Perhaps the algorithm for reading electrocardiograms will be given access to laboratory data, the patient’s age, or known medical diagnoses to assist with classifying the sequential data recovered from the leads.

`seglearn` provides a flexible, user-friendly framework for learning time series and sequences in all of the above contexts. Transforms for sequence padding, truncation, and sliding window segmentation are implemented to fix sample number across all sequences in the data set. This permits utilization of many classical and modern machine learning algorithms that require fixed length inputs. Sliding window segmentation transforms the sequence data into a piecewise representation (segments), which is particularly effective for learning periodized sequences (Bulling et al., 2014). An interpolation transform is implemented for resampling irregularly sampled time series. The sequence or time series data can be learned directly with various neural network architectures (Lipton et al., 2015), or via a feature representation which greatly enhances performance of classical algorithms (Bulling et al., 2014).

3. Installation

The `seglearn` source code is available at: <https://github.com/dmbee/seglearn>. It is operating system agnostic, and implemented purely in Python. The dependencies are `numpy`, `scipy`, and `scikit-learn`. The package can be installed using pip:

```
$ pip install seglearn
```

Alternatively, `seglearn` can be installed from the sources:

```
$ git clone https://github.com/dmbee/seglearn
$ cd seglearn
$ pip install .
```

Unit tests can be run from the root directory using `pytest`.

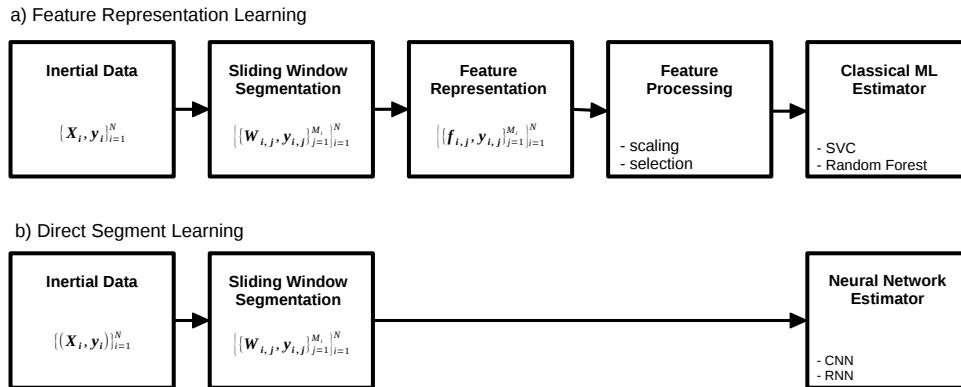


Figure 1: Example `seglearn` pipelines for a) learning segment feature representations, b) learning segments directly. \mathbf{X}_i : time series, \mathbf{y}_i : time series target, N : number of time series in the data set, $\mathbf{W}_{i,j}$: segment (derived from \mathbf{X}_i), $\mathbf{f}_{i,j}$: feature vector (calculated from $\mathbf{W}_{i,j}$), $\mathbf{y}_{i,j}$: segment target, M_i : number of segments derived from time series \mathbf{X}_i , SVC: Support Vector Classifier, CNN: Convolution Neural Network, RNN: Recurrent Neural Network.

4. Implementation

The `seglearn` API was implemented for compatibility with `scikit-learn` and its existing framework for model evaluation and selection. The `seglearn` package provides means for handling sequence data, segmenting it, computing feature representations, calculating train-test splits and cross-validation folds along the temporal axis.¹ An iterable, indexable data structure is implemented to represent sequence data with supporting contextual data.

The `seglearn` functionality is provided within a `scikit-learn` pipeline allowing the user to leverage `scikit-learn` transformer and estimator classes, which are particularly helpful in the feature representation approach to segment learning. Direct segment learning with neural networks is implemented in pipeline using the `keras` package, and its `scikit-learn` API. Examples of both approaches are provided in the documentation and example gallery. The integrated learning pipeline, from raw data to final estimator, can be optimized within the `scikit-learn` `model_selection` framework. This is important because segmentation parameters (eg window size, segment overlap) can have a significant impact on sequence learning performance (Burns et al., 2018; Bulling et al., 2014).

Sliding window segmentation transforms sequence data into a piecewise representation (segments), such that predictions are made and scored for all segments in the data set. Sliding window segmentation can be performed for data sets with a single target value per sequence, in which case that target value is mapped to all segments generated from the parent sequence. If the target for each is sequence is also a sequence, the target is segmented

1. Note splitting time series data along the temporal axis violates the assumption of independence between train and test samples. However, this is useful in some cases, such as the analysis of a single series.

as well and various methods may be used to select a single target value from the target segment (e.g. mean value, middle value, last value, etc.) or the target segment sequence can be predicted directly if an estimator implementing sequence to sequence prediction is utilized.

A human activity recognition data set (Burns et al., 2018) consisting of inertial sensor data recorded by a smartwatch worn during shoulder rehabilitation exercises is provided with the source code to demonstrate the features and usage of the `seglearn` package.

5. Basic Example

This example demonstrates the use of `seglearn` for performing sequence classification with our smartwatch human activity recognition data set.

```
>>> import seglearn as sgl
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.preprocessing import StandardScaler
>>>
>>> data = sgl.load_watch()
>>> X_train, X_test, y_train, y_test = train_test_split(data["X"], data["y"])
>>>
>>> clf = sgl.Pype([("seg", sgl.SegmentX(width=100, overlap=0.5)),
...                 ("features", sgl.FeatureRep()),
...                 ("scaler", StandardScaler()),
...                 ("rf", RandomForestClassifier())])
>>>
>>> clf.fit(X_train, y_train)
>>> score = clf.score(X_test, y_test)
>>> print("accuracy score:", score)
```

```
accuracy score: 0.7805084745762711
```

6. Comparison to other Software

Three other Python packages for performing machine learning on time series and sequences were identified: `tslearn` (Tavenard, 2017), `cesium-ml` (Naul et al., 2016), and `tsfresh` (Christ et al., 2018). These were compared to `seglearn` based on time series learning capabilities (Table 1), and performance (Table 2).

`cesium-ml` (v0.9.6) and `tsfresh` (v0.11.1) support feature representation learning of multi-variate time series, and currently implement more features than does `seglearn`. However, the feature representation transformers are implemented as a pre-processing step, independent to the otherwise sklearn compatible pipeline. This design choice precludes end-to-end model selection. There are no examples or apparent support for problems where the target is a sequence/time series or integration with deep learning models.

	tslearn	cesium-ml	ts-fresh	seglearn
Active development (2018)	✓	✓	✓	✓
Documentation	✓	✓	✓	✓
Unit Tests	✓	✓	✓	✓
Multivariate time series	✓	✓	✓	✓
Context data	X	✓	X	✓
Time series target	X	X	X	✓
Sliding window segmentation	X	X	X	✓
Temporal folds	X	X	X	✓
sklearn compatible model selection	X	X	X	✓
Feature representation learning	X	✓	✓	✓
Number of implemented features	N/A	58	64	20
Deep learning	X	X	X	✓
Classification	✓	✓	✓	✓
Clustering	✓	✓	✓	✓
Regression	✓	✓	✓	✓
Forecasting	X	✓	✓	✓

Table 1: Comparison of time series learning package features for **tslearn** v0.1.18.4, **cesium-ml** v0.9.6, **tsfresh** v0.11.1 and **seglearn** v1.0.2.

tslearn (v0.1.18.4) implements time-series specific classical algorithms for clustering, classification, and barycenter computation for time series with varying lengths. There is no support for feature representation learning, learning context data, or deep learning.

The performance comparison was conducted using our human activity recognition data set with 140 multivariate time series with 6 channels sampled uniformly at 50 Hz and 7 activity classes. The series' were all truncated to 4 seconds (200 samples). Classification accuracy was measured on 35 series' held out for testing, and 105 used for training. **seglearn**, **cesium-ml**, and **tsfresh** were tested using the sklearn implementation of the SVM classifier with a radial basis function (RBF) kernel on 5 features (median, minimum, maximum, standard deviation, and skewness) calculated on each channel (total 30 features). **tslearn** was evaluated with its own SVM classifier implementing a global alignment kernel (Cuturi et al., 2007). The testing was performed using an Intel Core i7-4770 testbed with 16 GB of installed memory, on Linux Mint 18.3 with Python 2.7.12.

Classification accuracy was identical between **cesium-ml**, **tsfresh**, and **seglearn** (as they used the same features and classifier in the evaluation) though **seglearn** significantly outperformed the other packages in terms of computation time. Classification performance of the global alignment kernel SVM (GAK-SVM) implemented in **tslearn** was poor on our data set, even following hyper-parameter optimization of gamma by grid search over the log space $[10^{-4}, 10^4]$. GAK-SVM is not typically applied to raw inertial data for human activity recognition in the literature, and better performance has been achieved with this algorithm in other time series applications (Cuturi and Doucet, 2011; Lorincz et al., 2013).

	tslearn	cesium-ml	ts-fresh	seglearn
Classification accuracy	0.057	0.714	0.714	0.714
Computation time (seconds)	0.79	62.9	0.40	0.088

Table 2: Comparison of time series learning package performance on our human activity recognition dataset.

References

- David Aha. UCI Machine Learning Repository, March 2018. URL <https://archive.ics.uci.edu/ml/index.php>.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2nd edition, April 2011. ISBN 978-0-387-31073-2.
- Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, January 2014. ISSN 03600300. doi: 10.1145/2499621.
- David Burns, Nathan Leung, Michael Hardisty, Cari Whyne, Patrick Henry, and Stewart McLachlin. Shoulder Physiotherapy Exercise Recognition: Machine Learning the Inertial Signals from a Smartwatch. *arXiv:1802.01489 [cs]*, February 2018. arXiv: 1802.01489.
- Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRE Extraction on basis of Scalable Hypothesis tests (tsfresh A Python package). *Neurocomputing*, 307:72–77, September 2018. ISSN 0925-2312. doi: 10.1016/j.neucom.2018.03.067.
- Marco Cuturi and Arnaud Doucet. Autoregressive Kernels For Time Series. *arXiv:1101.0673 [stat]*, January 2011. arXiv: 1101.0673.
- Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes, and Tomoko Matsui. A Kernel for Time Series Based on Global Alignments. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, pages II–413–II–416, Honolulu, HI, April 2007. IEEE. ISBN 978-1-4244-0727-9. doi: 10.1109/ICASSP.2007.366260.
- Thomas G. Dietterich. Machine Learning for Sequential Data: A Review. In *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, Berlin, Heidelberg, 2002. ISBN 978-3-540-44011-6 978-3-540-70659-5. doi: 10.1007/3-540-70659-3_2.
- Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.
- Andras Lorincz, Laszlo Jeni, Zoltan Szabo, Jeffrey Cohn, and Takeo Kanade. Emotional Expression Classification using Time-Series Kernels. *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 889–895, June 2013. doi: 10.1109/CVPRW.2013.131. arXiv: 1306.1913.

Brett Naul, Stfan van der Walt, Arien Crellin-Quick, Joshua S. Bloom, and Fernando Prez. cesium: Open-Source Platform for Time-Series Inference. *arXiv:1609.04504 [cs]*, September 2016. arXiv: 1609.04504.

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