A framework for subword-level dictionary-based time series classification

Bachelor's Thesis Exposé

Submitted by: Leonard Clauß
Submission Date: May 2, 2019

Supervisor: Dr. rer. nat. Patrick Schäfer
Supervisor: Prof. Dr. Ulf Leser
Institution: Department of Computer Science, Humboldt-Universität zu Berlin

Contents

1 Introduction

A time series is a sequence of values, e.g. from a sensor, ordered in time. Time series classification is the problem of classifying an unlabeled time series after seeing a set of labeled ones. This has various applications, such as classifying handwritten letters, gesture recognition (Figure 1) or detecting myocardial infarctions based on an electrocardiogram. Because of this broad range of applications it is of interest to find accurate and fast algorithms that solve this problem.

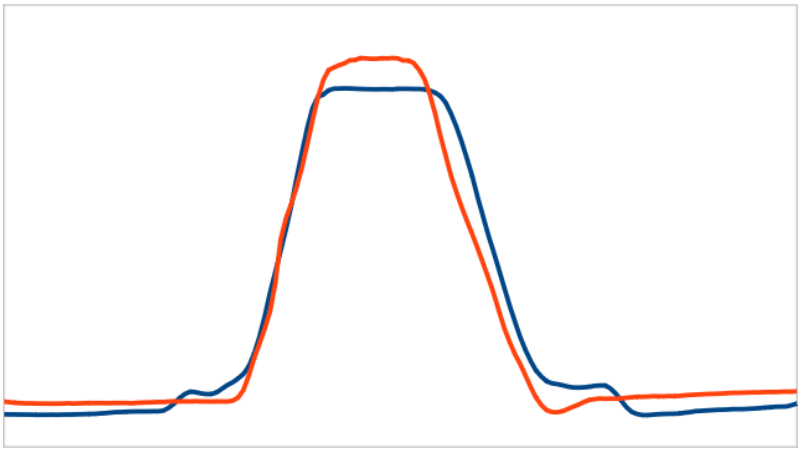

Figure 1: X-position of the right hand's centroid while drawing a gun, pointing it to a target and returning it (blue) and while doing the same without a gun but pointing with the index finger (red)[1]

In the past few years many new algorithms for time series classification have been presented. These can be mainly divided into the following groups [2, 3]:

1. Whole series: These algorithms try to compare two time series as a whole by defining a distance measure.
2. Intervals: Here discriminatory features are extracted over (random) time series intervals.
3. Shapelets: This family of algorithms tries to find short characteristic subsequences that signal a specific class if present.
4. Dictionary based: These methods transform a time series into words by (1) sliding a window over the time series and (2) discretizing each subsequence to a word. They then (3) count the occurrences of them and classify based on the resulting histogram, the so-called bag-of-patterns (Figure 2).
5. Deep Learning: Some more recent approaches show promising results using vanilla deep learning models such as Convolutional Neural Networks or Fully Connected Neural Networks.
6. Ensembles: These approaches combine multiple algorithms that classify a time series on their own. Afterwards the time series is classified by a vote of the members. This combination improves accuracy but comes with a massive loss of performance.

Approaches from the last three groups achieve state-of-the-art performance. In this work we look at the dictionary based approaches. As already stated, these types of algorithms first discretize a time series into words over a small alphabet and then compare two time series by comparing their bags-of-patterns. Two time series are similar in this representation if they share similar word frequencies. In the current state-of-the-art methods only entire words are counted and compared.

[1] `http://timeseriesclassification.com/description.php?Dataset=GunPoint`

This can cause the following problem. Consider a classifier that learned that the discretized time series from a specific class C frequently contain the word *abcd*. A recorded time series may be slightly different but does not contain that word and has multiple occurrences of *abbd* instead. This difference between the third letters might be just a result of discretization boundaries since two values that are near a discretization interval boundary have a small absolute difference but will get assigned different characters. As a result, this word will look completely different to the classifier and thus the time series probably will not get assigned its correct class C.

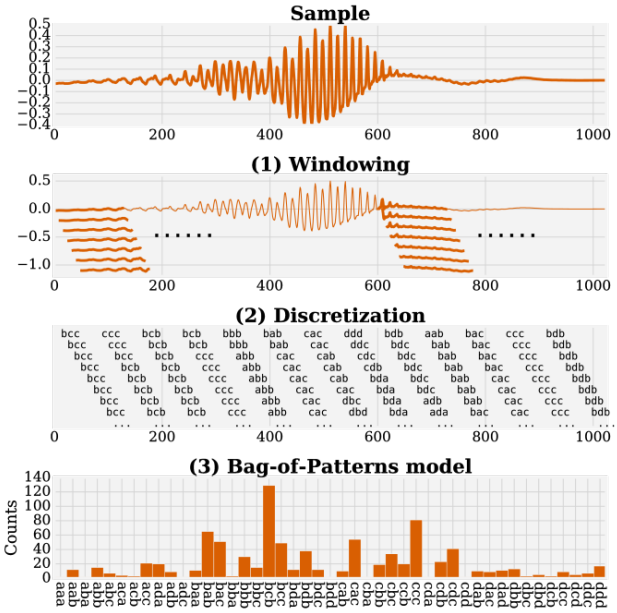

Figure 2: Transformation of a time series into the Bag-of-Patterns model[2]

To avoid this problem, one could not only select words as discriminative features but also subwords or character skip-grams, i.e., a subsequence of characters that does not need to be consecutive. In natural language processing several algorithms exist that find such frequent sequences but they have not been applied to time series classification yet. This thesis aims to investigate and evaluate how these can be used for the feature selection and what impact they have on the accuracy of the resulting algorithm.

2 Background and Related Work

Since we investigate dictionary based approaches in this work, their state-of-the-art methods will be explained next.

2.1 Symbolic representation

Two popular methods for discretizing a time series into a symbolic representation exist that reduce the dimensionality of the time series. The first one is Symbolic Aggregate approXimation (SAX) [4]. When using SAX, one first has to define the alphabet size a and the length w of the resulting word. The algorithm then approximates a given time series into such a word by dividing the time series in w intervals, calculating the mean over each interval and assigning a letter from the alphabet according to a predefined lookup table.

The other method is Symbolic Fourier Approximation (SFA) [7]. Again, alphabet size and the symbolic word length have to be predefined. Then, the algorithm applies a truncated Discrete Fourier transform that only keeps the first w Fourier coefficients. On the one hand this reduces the dimensionality, and as such word length, on the other it is used as a low pass filter to reduce noise. After that each coefficient gets discretized to an alphabet symbol using a lookup table.

Both methods are commonly used with a sliding window. That means contiguous subsequences of a defined length l are discretized into words, beginning at the start of the time series and then iteratively increasing the offset. The resulting bag-of-patterns is the basis for dictionary based time series classification algorithms.

[2] figure from [8], page 3

2.2 Recent research

The currently best dictionary based algorithm is WEASEL [8]. It competes with ensembles like COTE but has a much faster runtime in comparison. For discretization it uses SFA and a sliding window with multiple lengths. An ANOVA f-test is applied to find discriminative Fourier coefficients which then get discretized into words. Additionally bi-grams of words are considered. Because of the huge resulting feature space (in terms of the distinct number of words), a Chi-squared test is applied to find the most discriminative features. For classification it uses logistic regression that runs in linear time in the number of features.

2.3 Subword features

Many possibilities exist to find discriminative subwords. These can be split up into three approaches. The simplest, inspired by natural language processing, is to just search frequent character-n-grams, i.e., n consecutive characters in a word. The other more advanced methods are searching for discriminative subwords with gaps or searching (long) consecutive subsequences.

Subwords with gap constraints With the former method, we could for example find that occurrences of $ab*d$, where $*$ is an arbitrary character, are typical for the class C. To find such patterns we could model the problem as a search for frequent item set. In general, these describe common subsets in a given database of sets of elements. In our case a time series would be a database of words and each word is a set of items (characters). Since the order of the characters in a word is important, we need to use a different alphabet for each position in the words, so for instance a word could be $a_1b_2c_3a_4$ and a frequent item set $\{a_1, b_2, a_4\}$ which would stand for the pattern $ab*a$. An algorithm for this problem is Apriori [1]. It finds frequent item sets and then generates association rules which we do not need. For the first part the algorithm takes the database and a maximum number of item sets or the required minimal support as inputs. The latter describes the probability that a time series contains a specific item set and only if that probability is high enough it will be output.

Subwords with interval constraints This method only considers subwords with gaps but these might not be discriminative enough. As we have an ordered alphabet, instead of allowing any character for a specific position in the pattern we could restrict it to a character interval. In our example, the pattern $ab[b, c]d$ could be discriminative for the class C. This could be found by a modified Apriori algorithm.

Character skip-grams Another related approach is to search character skip-grams. [5] proposed an algorithm named Sqn2Vec for this. After learning on a training dataset it maps a sequence, e.g. a time series, to a low dimensional vector so that two vectors are close to each other if the respective sequences are similar. For that it finds frequent subsequences in the dataset that do not have to be consecutive but may have gaps of a maximum length between their characters. These so called sequential patterns are then represented by vectors. A sequence gets associated with a set of sequential patterns and its vector gets calculated from theirs.

Long representative subsequences Concerning the search for long consecutive subsequences, [9] presented an interesting idea. They used Byte Pair Encoding (BPE) for word segmentation to translate rare words (although they merge characters instead of bytes). The algorithm finds the most frequent sequences of characters in a vocabulary of words and their occurrences by merging the most frequent pair of characters into a new one. This is done iteratively until the required number of merges is found. The same method is applicable to time series classification because the merged characters represent frequent subwords.

A similar idea was presented in [6]. They also tried to classify time series by subwords but instead of BPE they used SEQL. This algorithm, that was designed for classifying DNA or text, finds discriminative subsequences by iteratively searching through the feature space with a branch-and-bound strategy. Therefor it greedily selects those subsequences that minimize a loss function.

When using any of these methods with SFA, it must be taken into account that characters at different positions have varying meanings due to the Fourier coefficients they represent, e.g. the subsequence *ab* in the words *cdab* and *abcd*. A simple workaround for this would be the same as for the frequent item sets, that is to have different characters for different positions in a word.

3 Motivation and Objectives

The motivation of this bachelor's thesis is to investigate how discriminatory features on subword-level can be obtained and evaluate whether they can be used for time series classification. Therefore, the main objectives are:

1. Explain current time series classification methods, specifically dictionary based, in order to gain a deeper understanding of them. Give an overview of possible approaches for feature selection at subword-level. Choose appropriate subword-level extraction algorithms for character-n-grams, subwords with gap or interval constraints and long consecutive subsequences and either implement them or look for available implementations. These should have a common interface, as they will be used interchangeably in the following pipeline.

2. Implement the toolbox that contains the different subword selection algorithms. Integrate it into the WEASEL pipeline[3] that first transforms a time series into a symbolic representation, then generates a feature vector (histogram) based on a picked subword-level extraction algorithm from above and finally trains a classifier. For the discretization and classification the existing WEASEL code will be used.

3. Train the different combinations on time series benchmark datasets to find the best fitting parameters for the subword feature selection algorithms. Evaluate accuracy and prediction runtime of the classifier on test datasets and compare them against the current state-of-the-art algorithms. Highlight datasets where this approach works well and where it does not. Discuss whether this topic should be followed up and what further steps could be taken.

By implementing the proposed toolbox we will show to what extent the subword-level approaches are useful for time series classification. Additionally, it will provide a tool for future usage.

4 Methods and Results

We implement the toolbox in Java. Hence for the three chosen algorithms that find subwords either available implementations in Java will be used or we will reimplement them. Afterwards we will integrate the toolbox into the WEASEL classifier, adding the following steps to the it:

- While training the classifier with the training data, the toolbox runs a selected subword-level extraction algorithm on it to find discriminatory features for each class. These subwords and the transformation rules get stored as the so-called dictionary.

- When transforming a time series into a bag-of-patterns an additional step in the pipeline is done after discretizing a sliding window into a word. The toolbox then tries to match the word with the saved subwords or patterns and returns the corresponding ones using the dictionary. If none are found, the word itself might be returned.

All datasets we use for the evaluation will be from the UEA & UCR Time Series Classification Repository[4]. We measure accuracy and classification runtime for each algorithm and compare them against each other and the state-of-the-art algorithms, namely WEASEL, HIVE-COTE (ensemble), and ResNet (deep learning).

The project and its different software versions will be tracked via a private HU GitLab repository. Afterwards we will upload the final code and results to a public GitHub repository in order for it to be available for further investigation.

[3] `https://github.com/patrickzib/SFA`

[4] `http://timeseriesclassification.com/dataset.php`

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.*, 31(3):606–660, May 2017.

[3] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *CoRR*, abs/1809.04356, 2018.

[4] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: A novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, October 2007.

[5] Dang Nguyen, Wei Luo, Tu Dinh Nguyen, Svetha Venkatesh, and Dinh Phung. Sqn2Vec: Learning sequence representation via sequential patterns with a gap constraint. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 569–584, Cham, 2019. Springer International Publishing.

[6] Thach Le Nguyen, Severin Gsponer, Iulia Ilie, and Georgiana Ifrim. Interpretable time series classification using all-subsequence learning and symbolic representations in time and frequency domains. *CoRR*, abs/1808.04022, 2018.

[7] Patrick Schäfer and Mikael Högqvist. SFA: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 516–527, New York, NY, USA, 2012. ACM.

[8] Patrick Schäfer and Ulf Leser. Fast and accurate time series classification with WEASEL. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 637–646, New York, NY, USA, 2017. ACM.

[9] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics, 2016.