Data Mining and Knowledge Discovery (2019) 33:1183–1222
https://doi.org/10.1007/s10618-019-00633-3

Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations

Thach Le Nguyen[1] · Severin Gsponer[1] · Iulia Ilie[1] · Martin O'Reilly[1] · Georgiana Ifrim[1]

Received: 17 May 2018 / Accepted: 12 April 2019 / Published online: 21 May 2019

Abstract

The time series classification literature has expanded rapidly over the last decade, with many new classification approaches published each year. Prior research has mostly focused on improving the accuracy and efficiency of classifiers, with interpretability being somewhat neglected. This aspect of classifiers has become critical for many application domains and the introduction of the EU GDPR legislation in 2018 is likely to further emphasize the importance of interpretable learning algorithms. Currently, state-of-the-art classification accuracy is achieved with very complex models based on large ensembles (COTE) or deep neural networks (FCN). These approaches are not efficient with regard to either time or space, are difficult to interpret and cannot be applied to variable-length time series, requiring pre-processing of the original series to a set fixed-length. In this paper we propose new time series classification algorithms to address these gaps. Our approach is based on symbolic representations of time series, efficient sequence mining algorithms and linear classification models. Our linear models are as accurate as deep learning models but are more efficient regarding running time and memory, can work with variable-length time series and can be interpreted by highlighting the discriminative symbolic features on the original time series. We advance the state-of-the-art in time series classification by proposing new algorithms built using the following three key ideas: (1) Multiple resolutions of symbolic representations: we combine symbolic representations obtained using different parameters, rather than one fixed representation (e.g., multiple SAX representations); (2) Multiple domain representations: we combine symbolic representations in time (e.g., SAX) and frequency (e.g., SFA) domains, to be more robust across problem types; (3) Efficient navigation in a huge symbolic-words space: we extend a symbolic sequence

Responsible editor: Dr. Panagiotis Papapetrou.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10618-019-00633-3) contains supplementary material, which is available to authorized users.

Extended author information available on the last page of the article

classifier (SEQL) to work with multiple symbolic representations and use its greedy feature selection strategy to effectively filter the best features for each representation. We show that our multi-resolution multi-domain linear classifier (mtSS-SEQL+LR) achieves a similar accuracy to the state-of-the-art COTE ensemble, and to recent deep learning methods (FCN, ResNet), but uses a fraction of the time and memory required by either COTE or deep models. To further analyse the interpretability of our classifier, we present a case study on a human motion dataset collected by the authors. We discuss the accuracy, efficiency and interpretability of our proposed algorithms and release all the results, source code and data to encourage reproducibility.

Keywords Time series classification · Multi-resolution multi-domain symbolic representations · SAX · SFA · SEQL · Linear models · Interpretable classifier

1 Introduction

State-of-the-art time series classification (TSC) accuracy is currently achieved with complex models such as large ensembles (COTE) (Bagnall et al. 2015, 2017; Lines et al. 2016) and deep neural networks (FCN, ResNet) (Wang et al. 2017; Ismail Fawaz et al. 2019). Although these algorithms achieve high accuracy, they are not efficient in either time or space (Bagnall et al. 2017), do not work on variable-length time series and are not interpretable. Simpler distance-based methods such as 1-Nearest Neighbour with Dynamic Time Warping (DTW) distance (Wang et al. 2013) provide strong baselines, but they lag significantly behind in accuracy, suffer from high running times and are negatively affected by noise (Schäfer 2016). It is also not possible to interpret the classification decision: we can compare the unlabelled time series to a labelled nearest neighbour, but we get no information about which parts of the time series are important for the classification decision.

Recent dictionary-based methods built on top of symbolic representations of time series deliver promising accuracy. The symbolic aggregate approximation (SAX) by Lin et al. (2007) has been the inspiration of numerous studies on time series analysis such as Castro and Azevedo (2010), Chen et al. (2005), Kasten et al. (2007), and Costa da Silva and Klusch (2007). SAX is a discretisation of time series in the time domain, and produces a sequence of symbols representing the raw numeric data. It allows compression and enables data mining algorithms to run efficiently. SAX-VSM (Senin and Malinchik 2013), FastShapelet (Rakthanmanon and Keogh 2013) and SAX-SEQL (Nguyen et al. 2017) are representative time series classification methods based on SAX. Existing SAX-based methods are highly dependent on SAX parameters and do not achieve state-of-the-art accuracy (Nguyen et al. 2017). A new symbolic representation introduced by Schäfer and Högqvist (2012), named Symbolic Fourier Approximation (SFA), and its classification frameworks (BOSS, WEASEL and MUSE) (Schäfer 2015, 2016; Schäfer and Leser 2017a, b), have further advanced the state-of-the-art regarding both accuracy and efficiency. SFA uses discretisation in the frequency domain, and by itself does not provide significantly enhanced accuracy as compared to SAX.[1] To achieve high accuracy, the proposed SFA-based algorithms BOSS and BOSS VS (Schäfer 2015, 2016) ensemble many models built on different SFA representations. The WEASEL approach (Schäfer and Leser 2017a) creates a very large feature space using multi-resolution SFA, applies feature selection and then learns a linear model. This method has high accuracy, but it is not memory-efficient. Furthermore, it only uses SFA features, which is not appropriate for some problem types.[2]

While symbolic representations hold promise, the process of finding the optimal symbolic representation is costly. The SAX-VSM algorithm (Senin and Malinchik 2013) attempts to find the optimal SAX transformation parameters with an optimization algorithm (DIRECT) and an evaluation method (cross-validation). However, it fails to outperform the accuracy of recent state-of-the-art classifiers, e.g., BOSS, WEASEL, COTE. State-of-the-art classifiers such as BOSS (Schäfer 2015) and WEASEL (Schäfer and Leser 2017a) generate symbolic representations at multiple resolutions by varying the symbolic representation parameters (e.g., working with multiple SFA representations obtained with different fixed parameters). This shotgun approach, despite its simplicity, showed to be more robust and efficient than globally optimising the parameters of one fixed representation. Another motivation for multiple representations of time series is the possibility of exploring different representation domains (e.g., time and frequency domain). Representations from different domains describe time series from different perspectives and by combining this knowledge it is possible to train a better model. An ensemble like COTE (Bagnall et al. 2015) takes advantage of knowledge on time series extracted from different domains. However, COTE gathers knowledge provided by its member classifiers, which have a combined time complexity that makes this approach impractical for many real-world problems. Symbolic representations are appealing as they can standardize the representations in a sequential structure, effectively unifying knowledge from different domains without the need of multiple classification algorithms. Existing approaches mostly combine the same symbolic representation obtained at multiple resolutions, but do not take advantage of distinct symbolic representations from multiple domains, possibly because this would generate a very large feature space. The key challenge is to develop a method that is able to efficiently work with a huge feature space of symbolic words obtained from distinct symbolic representations, and to select the best feature subset without having to explicitly evaluate each feature.

We build on these findings to propose two new time series classification algorithms based on multi-resolution multi-domain symbolic representations. The first algorithm achieves high accuracy by ensembling models built on multiple symbolic representations. The second algorithm achieves even better accuracy by (i) creating a large feature space using multiple symbolic representations, (ii) efficiently selecting important symbolic features and (iii) training a linear model (logistic regression). For efficient training using a massive symbolic feature space, we extend a symbolic sequence learning algorithm named SEQL (Ifrim and Wiuf 2011). SEQL was originally designed as a binary classifier for sequence data such as DNA or text. It works on

[1] According to our experiments comparing single SAX vs single SFA classifiers, shown in Sect. 4.

[2] Experiments and discussion backing these statements are available in Sects. 5 and 7.

either sequences of symbols (no white space) or sequences of words. The algorithm is able to explore the all-subsequence space by employing a branch-and-bound feature search strategy and it can efficiently train a linear model based on selected discriminative subsequences. In Nguyen et al. (2017) we proposed a time series classification algorithm that combines one fixed SAX representation with the SEQL learning algorithm. That approach works with variable-length SAX words which makes the SAX representation less dependent on parameters, but its accuracy is only comparable to SAX-VSM and lags behind the accuracy of more recent classifiers, e.g., COTE and FCN. Aiming to improve the accuracy of the combo of symbolic representations and simple linear models, we extend SEQL to work with both SAX and SFA symbolic representations, and to extract features from multiple resolutions and multiple domains. Our approach can also be extended to include other symbolic representations that turn the original time series into a sequence of symbols. We explain the requirements for the candidate symbolic representations in Sect. 4.

In our experiments, we explore different combinations for the input representation: a single symbolic representation with fixed parameters (e.g., SAX with fixed parameters), a single symbolic representation with multiple parameters (e.g., SAX with multiple resolutions) and distinct symbolic representations with multiple parameters (e.g., combining SAX and SFA representations and varying their parameters). We evaluate our algorithms on the well-known UCR time series classification benchmark (Chen et al. 2015) and discuss the impact of using different symbolic representations and learning algorithms on different types of problems (e.g., motion, image, sensor, ECG time series).

We also investigate another aspect of TSC which is the interpretability of the model. The ability to explain a classification decision is often excluded from the discussion in this field, as the community has focused mostly on classifier accuracy and efficiency. However, while accuracy and efficiency are very important, the model interpretability is an essential evaluation concern for any time series classifier. In many applications, it is important to know the key parts of the input data which are relevant for the analysis task or to understand the classification decision. For example, in sports science, an athlete executes a particular exercise and expects automated feedback on whether the exercise is executed correctly or not, and when it is incorrect, the athlete should receive feedback on which parts of the movement need correction. For the TSC problem, we want to highlight to the users the data examined by the model in order to make predictions. Our main interpretable classifier is a linear model (i.e., a list of weighted features), so we can use the weighted features learned by the model to highlight the parts of the time series that lead to a classification decision. We present a case study on TSC interpretability using a human motion dataset and a discussion by a domain expert in jump technique biomechanics (Sect. 7).

Our main contributions are as follows:

- We present new TSC algorithms which incorporate symbolic representations from multiple resolutions and multiple domains (SAX and SFA) by extending an efficient sequence learning algorithm (SEQL).
- We analyze the theoretical time and space complexity of all our proposed algorithms.

- We present an extensive experimental study of our approaches on the UCR Time Series Archive and compare to the state-of-the-art TSC methods.
- We demonstrate how to interpret our linear classification models in the context of time series analysis on known datasets.
- We conduct a case study on the interpretability of our TSC models for a human motion dataset. We also record, report and discuss the accuracy, running time and memory usage of all the classifiers evaluated on this real-world problem.
- All our code, data and detailed results are available from https://github.com/lnthach/Mr-SEQL.

The rest of this paper is structured as follows. In Sect. 2 we describe related prior work. In Sect. 3 we describe the requirements for the input symbolic representations and describe SAX and SFA in detail. In Sect. 4 we present our proposed TSC framework. In Sect. 5 we present experimental results. In Sect. 6 we discuss model interpretation in the context of TSC. In Sect. 7 we present a case study for the interpretability of our classifier on human motion data. In Sect. 8 we conclude and discuss future work directions.

2 Related work

Nearest neighbour classifiers with Euclidean or time warping distances are typical baselines for time series analysis. Wang et al. (2013) studied 8 different time series representations and 9 similarity measures and evaluated their performance on 38 datasets across various domains and tasks. In particular, for the TSC task, the study used 1-Nearest-Neighbor (1NN) classifiers to evaluate the accuracy of these measures. The conclusions provide interesting insights into the effectiveness of these measures and reaffirm the competitiveness of DTW in comparison to newer methods. A more recent comparison of elastic distance measures is presented in a study by Lines and Bagnall (2015) where an ensemble approach was shown to outperform a strong DTW baseline (where the warping window, a parameter important for DTW, was set through cross-validation). Nevertheless, elastic distance based approaches, even those based on ensembles, are outperformed in accuracy by more recent approaches.

Recently, there has been notable interest in shapelet-based classification algorithms after the first proposal by Ye and Keogh (2009). Shapelets are discriminative segments extracted from time series and can be used for classification. Moreover shapelets are interpretable, thus they can offer insight into the data. Since shapelet discovery is usually time-consuming, studies by Ye and Keogh (2011), Rakthanmanon and Keogh (2013), Gordon et al. (2012) focused on enhancing the efficiency of the process. FastShapelet (Rakthanmanon and Keogh 2013) finds k-best shapelets in the dimension-reduced space of SAX. It speeds up shapelet discovery, but its accuracy is comparable to Ye and Keogh (2009). On the other hand, Grabocka et al. (2014) formulated the problem as an optimization task and solved it with a stochastic gradient learning algorithm. The studies (Lines et al. 2012; Bostrom and Bagnall 2015) used the shapelets to create a transformed dataset, in which the distance between the time series and a shapelet is a feature. One challenge with shapelet-based algorithms, e.g., FastShapelet, is that repeated runs of the algorithm produces different shapelets, which also affects the interpretation.

COTE, first introduced as Flat-COTE (Bagnall et al. 2015), is an ensemble method which incorporates 35 different classifiers for TSC. HIVE-COTE (Lines et al. 2016), a recent extension of COTE, added 3 more classifiers to the collection. Overall, COTE is among the most accurate TSC algorithms that have been tested on the UCR benchmark (Chen et al. 2015). It is one of the few classifiers that incorporate descriptions of time series from different domains. However, its learning framework is based on a large ensemble. This demands substantial computation resources as COTE's time complexity is determined by the slowest algorithm. The work by Kate (2016) is another example of a multi-domain classifier: the algorithm combines SAX and DTW feature spaces for classification.

The excellent recent survey on TSC (Bagnall et al. 2017) has contributed a systematic framework to evaluate time series classifiers. In this study, the authors reproduced the experiments of 18 state-of-the-art classifiers in addition to two baseline classifiers (1-NN DTW and Rotation Forest) on the extended UCR benchmark that includes 85 datasets across a range of different TSC problem types (e.g., motion, image, ECG). The results were analysed based on algorithm type and problem type. The main claim of the survey was that benchmark classifiers are hard to beat and that COTE was by far the most accurate algorithm. Nevertheless, the authors mainly focused on the classifier accuracy for evaluation, and did not evaluate either the efficiency or interpretability of the methods compared.

Regarding symbolic representations of time series, SAX is perhaps the most studied representation (Lin et al. 2003, 2007, 2012; Rakthanmanon and Keogh 2013). The BOP (Bag-Of-Patterns) approach presented by Lin et al. (2012) builds a histogram of SAX words for each time series. A new sample is classified by comparing the histograms to find the nearest neighbour in the training set. SAX-VSM (Senin and Malinchik 2013) is another SAX-based classifier which also uses the BOP framework. It first builds a dictionary of distinct SAX words from training data (a vector space representation) and for efficiency reasons, instead of histograms it computes a single vector of tf-idf weights for each class. In addition, it employs an optimization algorithm to search for the optimal parameters of SAX. However the tuning cost is substantial due to the need of cross-validation. In addition, the SAX-VSM authors have analysed the interpretability of SAX-VSM models by mapping SAX words having high tf-idf scores back to the original time series. Our prior work (Nguyen et al. 2017) proposed a novel TSC algorithm using symbolic representations. Our approach was a combination of a fixed symbolic representation (SAX) and two adaptations of a sequence classifier (SEQL by Ifrim and Wiuf (2011)). One of the proposed classifiers, SAX-VFSEQL, learns approximate subsequences of symbolic words and can thus reduce the influence of SAX parameters and noise on the classification accuracy. Although more flexible than previous approaches that cannot learn symbolic sub-words, the accuracy of SAX-VFSEQL still suffers from being limited to only one fixed SAX representation and thus falls behind the accuracy of more recent methods.

Schäfer and Högqvist (2012) introduced a new symbolic representation to index time series, the Symbolic Fourier Approximation (SFA). This approach uses a Discrete Fourier Transform as the core approximation technique. Based on this work, the authors proposed several classification frameworks for time series, which includes 1NN-BOSS (Schäfer 2015) and BOSS VS (Schäfer 2016) (ensemble methods), WEASEL (Schäfer and Leser 2017a), and MUSE (Schäfer and Leser 2017b). BOSS uses ensembles of histograms of SFA-words and 1NN classifiers, while BOSS VS uses tf-idf class centroids and 1NN for classification. In the SFA-based TSC algorithms family, WEASEL is the most recent work on univariate time series, while MUSE was developed to classify multivariate data. Both methods employ heavy feature engineering and feature selection techniques to filter the huge feature space created by multiple resolutions of SFA transformations, before feeding the selected features to a linear model. WEASEL is more accurate than the BOSS algorithms, but suffers from memory efficiency issues since it does not include effective methods for pruning features early, and hence also needs to carefully restrict the feature space (e.g., by restricting the SFA parameters and the type of features). The authors of WEASEL and MUSE do not discuss the interpretability of these methods, arguably because of the non-linear characteristics of the SFA transformation. As we show in Sect. 7, our algorithms are more accurate than WEASEL and use one order of magnitude less memory for training models.

The popularity surge of deep learning has inspired various studies to exploit its power for TSC. A comprehensive review of state-of-the-art deep learning algorithms for TSC was published recently (Ismail Fawaz et al. 2019). The studied algorithms include: Multi Layer Perceptron (MLP), Fully Convolutional Neural Network (FCN), Residual Network (ResNet), Encoder, Multi-scale Convolutional Network (MCNN), Time Le-Net (t-LeNet) and a few others. Similarly, the study by Wang et al. (2017) examined the performance of FCN, ResNet, and MLP on the UCR Archive. We compare our proposed TSC algorithms to these deep learning approaches and discuss their accuracy as reported by Ismail Fawaz et al. (2019). Although Wang et al. (2017) emphasize the simplicity of their method and its effectiveness as a TSC baseline, there is no detailed discussion about efficiency. We downloaded the code but we could not reproduce the experiments on a regular PC in a reasonable amount of time, as we did with many of the other existing algorithms. On the other hand, Ismail Fawaz et al. (2019) outlined the huge amount of effort and computational resources (8,730 experiments, 60GPUs, 100 days of running time) required for their study of deep learning approaches. Additionally, these approaches do not work on variable-length time series and thus require pre-processing of the original dataset to create fixed length time series. In comparison, our algorithms are as accurate as large ensembles (Flat-COTE, HIVE-COTE) and deep learning approaches (FCN, ResNet), only need a few hours to train and predict for the entire UCR archive on a regular PC, and use orders of magnitude less memory. We provide a detailed discussion of the accuracy and time/memory efficiency of the TSC methods compared in Sects. 5 and 7.

3 Symbolic representation of time series

In this section, we discuss two notable symbolic representations of time series: the Symbolic Aggregate approXimation (SAX) (Lin et al. 2003) and the Symbolic Fourier Approximation (SFA) (Schäfer and Högqvist 2012). Although they are different tech-

Table 1 Notation for our time series classification (TSC) framework

Symbols	Description
V	Raw (normalized) numeric time series
N	Number of time series
L	Length of original time series
w	Length of a symbolic word
α	Size of alphabet
l	Size of sliding window

niques that generate descriptions of time series in different domains, SAX and SFA produce very similar output in terms of structure. This property is strongly desirable in our approach as our classifiers require that the symbolic representation, regardless of the method, has a certain standard structure. Moreover, both have been shown to be powerful representations for the TSC task.

Generally, the output of both techniques can be described as a sequence of symbols taken from an alphabet α, e.g., *aaba*. In practice, it is common to employ a sliding window of length l and repeatedly apply the transformation on the time series within this window. As a result, the output is a sequence of words, each of which is actually a symbolic sequence of length w, e.g., for $w = 4$, *abba abbc bacc aacc*.

Formally, a symbolic sequence S of length n has the following form:

$$S = s_1 s_2 \ldots s_n \text{ where } s_i \text{ is in } \{a_1, a_2, \ldots a_\alpha\} \cup \{\text{␣}\}$$

The space character means that S can be either a sequence of symbols or a sequence of words. We will show that the output of SAX and SFA can take the above form and therefore can work with our sequence classifier. Table 1 summarises the notations used in this paper for the symbolic representations.

3.1 Symbolic aggregate approximation

SAX was introduced by Lin et al. (2003) and is a transformation method to convert a numeric sequence (time series) to a symbolic representation, i.e., a sequence of symbols with a predefined length w and an alphabet of size α. Generally, the technique includes three steps:

1. Compute the Piecewise Aggregate Approximation (PAA) (Keogh et al. 2001) of the time series.
2. Compute the lookup table for the given alphabet.
3. Map the PAA to a symbolic sequence by using the lookup table.

In the first step, the time series is z-normalized and then divided into w equal-length segments and each segment is replaced with its mean value. The result is the PAA vector of length w. In the second step, a lookup table is built for the alphabet α. Each symbol in the alphabet is associated with an interval, i.e., a continuous range of values. The intervals are obtained by dividing the domain of the time series to α

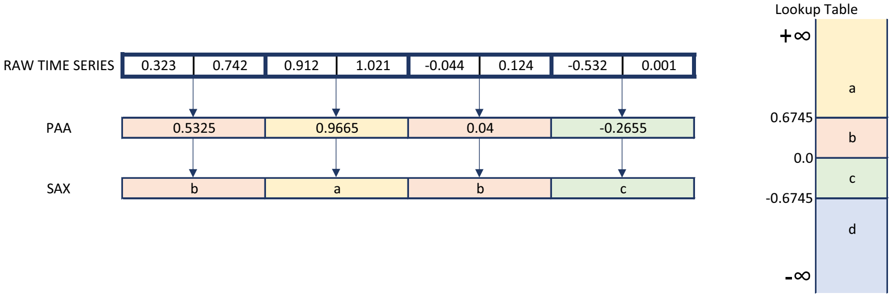

Fig. 1 An example of SAX transformation that includes: the raw time series, the Piecewise Aggregate Approximation, the lookup table and the final SAX output for ($w = 4, \alpha = 4$)

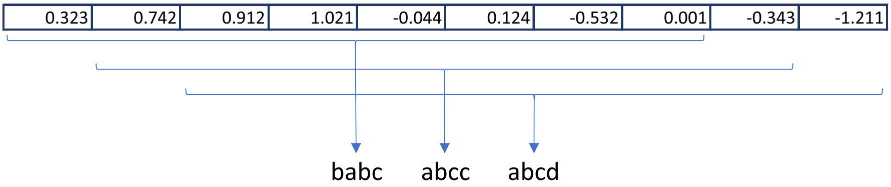

Fig. 2 Sliding window shifting to obtain a time series representation based on SAX-words ($l = 8, w = 4, \alpha = 4$)

disjoint areas with equal probability, assuming that the values of the time series are normally distributed (hence the z-normalization). Finally, each entry of the PAA vector is then replaced by a symbol taken from the alphabet by using this lookup table.

Figure 1 illustrates an example of SAX output with parameters set to $L = 8$, $w = 4$ and $\alpha = 4$. The lookup table divides the domain of the time series into 4 intervals by defining 3 breakpoints (−0.6745, 0.0, and 0.6745) and links each interval to a symbol from the alphabet $\{a, b, c, d\}$. Each entry in the PAA vector is the average of the corresponding segment in the raw time series. The SAX sequence is produced by looking up the PAA from the table. The first entry (0.5325) falls within the range [0.0, 0.6745) thus the first symbol taken is b.

SAX can also be combined with a sliding window of length l, usually done to process longer time series (Fig. 2). Our previous study (Nguyen et al. 2017) also found that the sliding window technique has a positive impact on the classification accuracy, arguably because it can capture a finer description of the time series.

The procedure to transform a time series to a SAX representation with a sliding window can be summarised in Algorithm 1. The sliding window starts from the first time-stamp, i.e., the beginning of the time series. The subsequence of length l within the window is then transformed to a symbolic sequence of length w with the previously described steps. This sequence is commonly referred to as a SAX word. The process is repeated until the window reaches the end of the time series. Hence the final output is a sequence of equal-length SAX words.

In our implementation of SAX, we also discard continuously repeated SAX words as often done by related works (i.e., numerosity reduction). We apply the same practice on SFA, which will be discussed next.

Algorithm 1 SAX with sliding window

```
Set window length $l$
Set word size $w$
Set alphabet size $\alpha$
Compute lookuptable
$L = length(timeseries)$
**for all** $t$ in $[0, L - l)$ **do**
    $normed\_ts = z\_normalize(timeseries[t : t + l])$
    $PAA = computePAA(normed\_ts)$
    $S =$ “"
    **for** $v$ in PAA **do**
        $S += lookup(v)$
    **end for**
    **if** $S$ is not a repeated word **then**
        Add $S$ to the final representation
    **end if**
**end for**
```

3.2 Symbolic Fourier approximation

SFA (Schäfer and Högqvist 2012) also transforms a time series to a symbolic representation. Similarly to SAX, SFA employs a sliding window to extract segments of time series before transformation (although Schäfer and Leser (2017a) also suggested non-overlapping window for their SFA-based WEASEL classification framework). Hence SFA's parameters also include the window size l, the word length w and the alphabet size α.

The core differences between SAX and SFA are the choices of approximation and discretisation techniques. SFA uses a Discreet Fourier Transform (DFT) method to approximate a time series. DFT is well known in the signal processing community and can act as a filter to remove noise from data. The same authors also introduced a multiple coefficient binning (MCB) method to discretise the approximation. The overall procedure consists of two major steps:

1. MCB discretisation: Compute the lookup table from the DFT approximations of the training data.
2. Map the DFT approximation of the input time series to its SFA representation with the lookup table.

Both steps employ the DFT technique to approximate an input time series with a vector of length w. Basically, DFT decomposes a time series into a series of sinusoid waves, each of which can be represented by a Fourier coefficient. The coefficient is a complex number, hence it can be defined by 2 real values: one for the imaginary part and one for the real part. Therefore, only the first $w/2$ coefficients of the series are used to create a sequence of length w.

The discretisation step (MCB) computes the lookup table from the training data. It first computes the DFT approximations of all l-length segments extracted from the entire training data to obtain a set of w-length vectors. Then from this set of vectors, it computes a set of $\alpha - 1$ breakpoints for each i-th position ($1 \leq i \leq w$) according to the distribution of the i-th values (i.e., equi-depth binning method). The areas divided

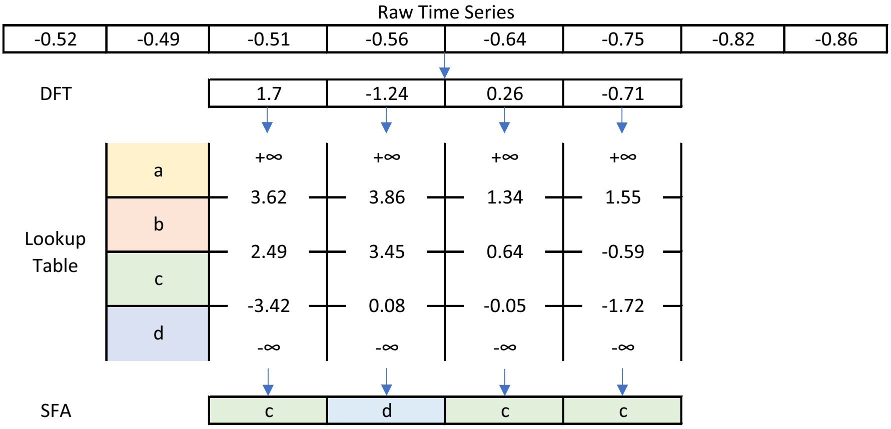

Fig. 3 An example of SFA transformation that includes: the raw time series, the DFT approximation, the lookup table and the final SFA output for $(w = 4, \alpha = 4)$. Note that a symbol has different meanings at each position, e.g., c at position 1 is different from c at position 3, as it represents a different interval for each position

Table 2 Comparison between SAX and SFA symbolic representations of time series

Method	Approximation	Discretisation	Complexity
SAX	PAA	equi-prob	$\mathcal{O}(NL \log L)$
SFA	DFT	MCB + equi-depth	$\mathcal{O}(NL)$

by the breakpoints are associated with the symbols from the alphabet. The result is a lookup table of w columns and α rows (Fig. 3).

The transformation step first computes the DFT approximation for each l-length segment of the input time series. For each i-th entry of the approximation vector, the corresponding symbol is looked up from the i-th column of the lookup table. Thus each segment of the time series is transformed to a sequence of length w. The final result is a sequence of SFA words extracted from the input time series.

It is important to note that, even at the same level of discretisation α (i.e., number of intervals across the domain), a symbol in SFA has more expressive power than a symbol of SAX. While a symbol of SFA can represent w different (usually overlapping) intervals ($w = 4$ as in Fig. 3), a symbol of SAX can only represent one single interval.

Figure 3 gives an example of SFA transformation. The raw time series is DFT approximated with a vector of length $w = 4$. The i-th entry of this vector is then replaced by a symbol according to the i-th column in the lookup table. Table 2 summarizes the difference between the SAX and SFA representations. We use the same SFA implementation as described by Schäfer and Högqvist (2012) and report the time complexity from that paper.

4 Sequence learner with multiple symbolic representations of time series

Typically, the SAX representation is susceptible to how we set parameters, i.e., the sliding window size l, the word size w or the size of the alphabet α. Each choice of parameters captures a different structure of the time series which is essential for the classification task. One solution for this issue is to search for the optimal parameters, either by a naive grid search or a more complex optimization algorithm (e.g., DIRECT as in SAX-VSM (Senin and Malinchik 2013)). In our previous work, we mitigate this issue by introducing a new algorithm that can learn discriminative sub-words from a SAX word-based representation (Nguyen et al. 2017). However, while it was still competitive at the time of publication, that algorithm has fallen behind most recent state-of-the-art (e.g., WEASEL) in terms of accuracy.

Here we introduce a new approach which uses multiple resolutions and multiple domain representations of time series. Fast Shapelets, BOSS and WEASEL are notable classifiers using multiple resolutions, although they only support representations from one domain (via SFA). Their competitive results demonstrates the potential of combining multiple symbolic resolutions. Our hypothesis is that a single representation of time series, even an optimal one, might be insufficient to capture the necessary structure for the classification task. By combining knowledge from multiple resolutions and multiple domains, a learning algorithm can deliver a more robust model.

This section first describes the core technique of our approach, which is a sequence classification framework for the (single) symbolic representation of time series. After that, two new methods based on the core technique are proposed, to make use of multiple representations. The first one is an ensemble method and the second one is a feature selection method combined with a linear model.

4.1 Sequence learner with symbolic representation of time series

Figure 4 sketches our approach which is composed of two components: a symbolic representation and an efficient sequence classifier. The symbolic representation can be either SAX or SFA.

Our core algorithm for classification is Sequence Learner (SEQL) (Ifrim and Wiuf 2011). SEQL was originally designed as a binary classifier for sequence data such as DNA or text. The algorithm is able to explore the all-subsequence space by employing a branch-and-bound feature search strategy. Thus it can select a set of discriminative subsequences in an effective manner. With the symbolic representation of time series, the symbolic features can easily be translated back to a set of time series' discriminative segments. TSC with symbolic representation is not a new idea, however the most common approach is to build a dictionary directly from the results of the transformation (usually a bag of symbolic words).

Nguyen et al. (2017) introduced two adaptations of SEQL for the TSC task. The first one (SAX-VSEQL) can learn subsequences from the SAX words while the second one (SAX-VFSEQL) can approximate the subsequences. The latter was proposed mainly to make the representation less dependent on the symbolic parameters (l, w, and α). As

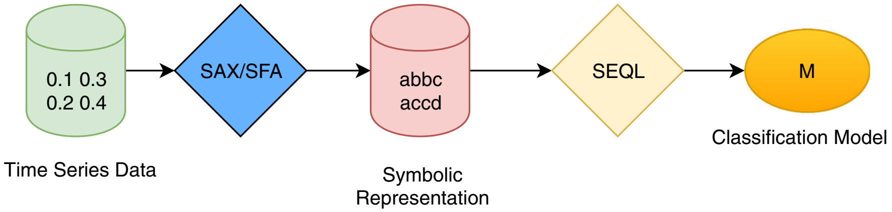

Fig. 4 SEQL classification algorithm adapted to symbolic representations (SAX or SFA) of time series

multiple resolutions of a given symbolic representation can create the same effect, we decided to adapt only the lightweight VSEQL version for our new classifiers. Hence from here on, we use the name SEQL to refer to the VSEQL TSC classifier by Nguyen et al. (2017).

The training input for SEQL is a set of sequences and their labels. The output is a linear mapping function $f : S \rightarrow \{-1, +1\}$ to predict the label of new sequences. The function f is represented by a parameter vector $\beta = (\beta_1, \ldots, \beta_j, \ldots, \beta_d)$ (where d is the number of features) which minimizes the loss function $L(\beta)$ using greedy coordinate descent:

$$\beta^* = \underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} L(\beta) \tag{1}$$

where

$$L(\beta) = \sum_{i=1}^{N} \xi(y_i, x_i, \beta) + C R_\alpha(\beta) \tag{2}$$

and $\xi(y_i, x_i, \beta)$ is the binomial log-likelihood loss:

$$\xi(y_i, x_i, \beta) = log(1 + e^{-y_i \beta^T x_i}) \tag{3}$$

The x_i in Eqs. (2) and (3) denotes the feature vector of the sequence S_i (in all-subsequence feature space) while y_i denotes its true label (either $+1$ or -1); i is the index of training examples, N is the total number of training examples. On the right hand side of Eq. (2), C denotes the regularization weight and $R_\alpha(\beta)$ denotes the elastic-net regularization. In SEQL the all-subsequence feature space is not explicitly generated; an iterative process is applied to efficiently generate and search the feature space for the best feature to be optimized next. The classification decision is computed as $f(x) = \beta^t x$. SEQL (binary classifier) was shown to work well for multi-class classification through the one-vs-all approach (Ifrim and Wiuf 2011).

The SEQL workflow is shown in Algorithm 2. At each iteration, it searches for the coordinate j of β which has maximum gradient $j = argmax_l \left| \frac{\partial L}{\partial \beta_l}(\beta) \right|$. Since each coordinate corresponds to a subsequence, this process is translated to finding the most discriminative subsequence in the context of SEQL. The search is done by navigating the sequence tree starting from the unigrams. The size of a complete tree is the total number of subsequences, which is normally impractical to traverse. However, SEQL employs a greedy branch-and-bound strategy to selectively generate and prune any

Table 3 An example model trained by SEQL: a linear model in the space of all symbolic sub-words

Subsequences	Coefficients
cccdbcdda	0.014
cccdcbcddaaa	0.013
ccdbc	0.012
cccdbc	0.007
ccccdbc	0.004
cccbbdd	−0.006
cccbbd	−0.010
ccbbddb	−0.011
cccbb	−0.012
ccccbb	−0.014

unpromising part of the tree. This strategy relies on the anti-monotonicity property of a sequence, i.e., a sequence is always equally or less frequent than all of its subsequences. This property allows the algorithm to calculate a gradient upper-bound for each subsequence, which enables the search-tree pruning decision. More details about the algorithm can be found in the original paper (Ifrim and Wiuf 2011).

SEQL takes sequences of characters or sequences of words as input. For the TSC task we take as input the sequence of symbolic words resulting from a symbolic transformation of the time series. To be able to learn symbolic subwords, we consider the characters (rather than the words) to be the unigrams of the representation, and thus each subsequence of symbols from the training data is a potential feature during learning. To achieve this we need to restrict the feature expansion and search to stay within a symbolic word, so the features learned are always contiguous subsequences or subwords. In our previous paper (Nguyen et al. 2017), we discuss more details about this adaptation in the context of the SAX-words symbolic representation. The output of SEQL training is a linear model which is essentially a set of subsequences and their coefficients (Table 3). The coefficients can be interpreted as the discriminative power of the subsequence.

We discuss next two algorithmic adaptations to be able to combine a single symbolic representation of time series (that produces a sequence of symbolic words from a numeric signal) and the sequence mining algorithm SEQL (to efficiently search the entire space of symbolic sub-words and select good features). The first adaptation,

Algorithm 2 SEQL workflow

```
Initialize the subsequence tree with all unigrams
Set $\beta = 0$
**while** !termination condition **do**
    Explore the subsequence tree to find the best subsequence $\hat{s}$ with maximum gradient value *//This step uses branch-and-bound search based on the structure of the feature space; only a small subset of features is generated and evaluated for selecting the best feature, which makes the search very fast. See Ifrim and Wiuf (2011) for mathematical details.*
    Update $\beta$
**end while**
```

SAX-SEQL, was presented by Nguyen et al. (2017). The second adaptation, SFA-SEQL, is proposed here for the first time. Based on these adaptations, we then propose two new TSC algorithms: (i) ensemble-based and (ii) linear model-based time series classifiers.

SAX-SEQL: To combine the SAX representation with the SEQL classifier, the time series is transformed to its SAX representation with a sliding window. The words are concatenated with a space delimiter to form a sequence of SAX words for each time series. Then SEQL learns a linear classification model from the new training data. The test data is also transformed to its SAX-words representation before being classified (Algorithm 3).

Algorithm 3 The workflow of SAX-SEQL

```
Set $l$, $w$ and $\alpha$
$train = SAXtransform(train\_time\_series, l, w, \alpha)$
$test = SAXtransform(test\_time\_series, l, w, \alpha)$
Train with SEQL $M = SEQLearner(train)$. // This step results in a linear model: a list of weighted symbolic features.
Test with SEQL $predictions = SEQLClassifier(M, test)$
```

SFA-SEQL: To combine the SFA representation with the SEQL classifier, the time series is transformed to its SFA-words representation with a sliding window. Nevertheless, to be able to use this representation with SEQL, we need to make some modifications to the SFA-words output as described below. As discussed in Sect. 3.2, the same symbol in an SFA representation can imply different intervals. For example, the lookup table in Fig. 3 can produce a sequence "*dabb*" in which the 3-rd and 4-th positions share the same symbol "*b*". However, while the former represents the interval [0.64, 1.34), the latter corresponds to interval [−0.59, 1.55). Moreover, one *b* symbol is linked to the real part, while the second *b* is linked to the imaginary part of the same Fourier coefficient. Unfortunately, this tricky feature is not compatible with SEQL, since that algorithm would be misled by treating both "*b*" as the same unigram. To fix this issue, we let each column in the lookup table have its own alphabet, e.g., the symbolic representation of the same example would be "$d_1a_2b_3b_4$". This way, SEQL can recognize two different unigrams "b_3" and "b_4". Technically, the size of the alphabet for SEQL has increased w times, but the level of SFA discretisation (the number of intervals across the domain) is still α. This means that SFA does not get more expressive power by this transformation, it is simply a trick on top of standard SFA, to let SEQL know that the semantics of the SFA symbols at each position is different. Hence, we still refer to the size of alphabet as α when we discuss the SFA representation. Besides the above issue, the procedure for training and testing SFA-SEQL is identical to that of SAX-SEQL.

4.2 Ensemble SEQL

Ensemble SEQL is a new algorithm we propose to combine SEQL with multiple resolutions and multiple domain symbolic representations. Each model is trained by the

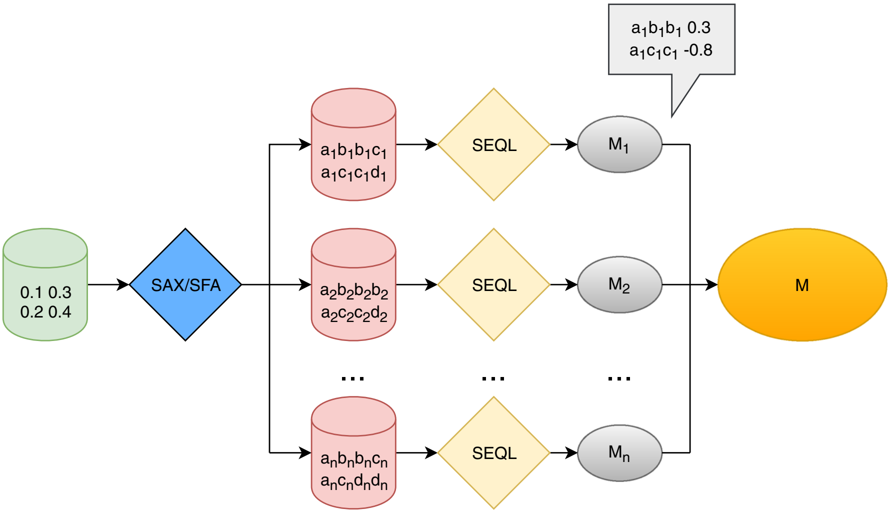

Fig. 5 An ensemble of SEQL models trained from different SAX (or SFA) representations. We call this a multi-resolution approach for a given symbolic representation

same classifier (SEQL) but with a different symbolic representation of the training data as input. Figure 5 illustrates the training procedure to produce n SEQL models from different SAX or SFA representations for the ensemble. Different representations of the time series can be generated simply by adjusting the transformation parameters (l, w, and α) to obtain multiple resolutions for a given symbolic representation. Algorithm 4 shows how we can adjust the window length l to train multiple models (Line 3). As a result, the number of representations is approximately $\sqrt{L}$, i.e., the longer the time series, the larger the number of representations it can produce. For each set of parameters, the raw training data is transformed to the SAX or the SFA representation (Line 4) and a new SEQL model M_i is trained upon this representation (Line 5). The output is an ensemble M of all M_i.

In our experiments, we fix the values for the word length ($w = 16$) and alphabet size ($\alpha = 4$), and only adjust the sliding window size (l) as we observed that there was little benefit in varying all three. The minimum window size starts from $minl = 16$, as we should not have a window shorter than the word size, and the step size for increasing the window length is $\sqrt{L}$. It is also worth noting that in the variable length time series scenario, L can be set with the maximum length in the training set. This setting for the window size follows similar settings investigated by Schäfer (2016).

For the prediction (Algorithm 5), the unlabelled time series is converted to a SAX representation with the same set of configurations chosen in the training step (Line 6). Each model makes a prediction based on the representation of the corresponding configuration (Line 6). The sign of the predicted score aggregation will determine the predicted class of the time series (Line 9).

Algorithm 4 Ensemble SEQL: Training

```
Set word size $w = 16$
Set alphabet size $\alpha = 4$
Set minimum window size $minl = 16$
for $l = minl, l <= L, l += sqrt(L)$ do
    $sax = SAXtransform(raw\_time\_series, l, w, \alpha)$
    Train the SEQL model from the symbolic representation $M_i = SEQL(sax)$
    $M[l, w, \alpha] = M_i$
end for
```

Algorithm 5 Ensemble SEQL: Testing

```
Set word size $w = 16$
Set alphabet size $\alpha = 4$
Set minimum window size $minl = 16$
$score = 0$
for $l = minl, l <= L, l += sqrt(L)$ do
    $sax = SAXtransform(raw\_time\_series, l, w, \alpha)$
    $score += M[l, w, \alpha].predict(sax)$
end for
$prediction = sign(score)$
```

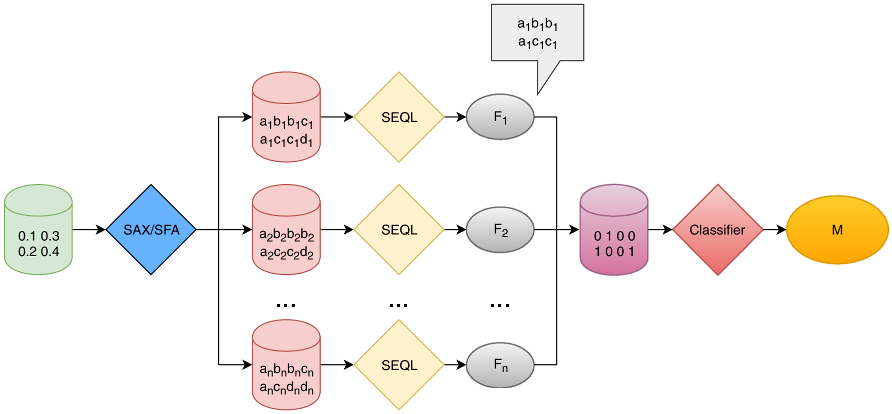

Fig. 6 SEQL as feature selection method. Features are selected from multiple-resolutions and/or multiple-domain symbolic representations and fed to a logistic regression algorithm

4.3 SEQL as feature selection

In this section we propose a second new algorithm for training a TSC with multiple symbolic representations and linear models. The learning output of SEQL is essentially a list of subsequences selected from the training data (Table 3), hence the method can be used for feature selection. The process diagram for this scheme is illustrated in Fig. 6.

As it can be seen from the diagram, we first transform time series data to multiple symbolic representations (either SAX or SFA). We then feed each representation to a SEQL trainer. Algorithm 6 explains how the features were extracted using SEQL. It is basically Algorithm 4 with an additional for-loop to collect the new features. A feature is identified by a subsequence learned by SEQL and the associated SAX (or SFA) configuration (Line 9). With the collected set of features F, it is possible to apply any traditional classification technique such as logistic regression, support vector machine or random forest. After experimenting with several learning algorithms, we chose logistic regression for its simplicity, accuracy and interpretability.

Algorithm 6 SEQL as Feature Selection

```
1: Set word size $w = 16$
2: Set alphabet size $\alpha = 4$
3: Set minimum window size $minl = 16$
4: Set of features $F = \{\}$
5: for $l = minl, l <= L, l += sqrt(L)$ do
6:     $sax = SAXtransform(raw\_time\_series, l, w, a)$
7:     Train the SEQL model $M_i = SEQL(sax)$
8:     for all $subsequence$ in $M_i$ do
9:         $F$.add(new Feature($subsequence, l, w, \alpha$))
10:    end for
11: end for
```

It is also worth to note that feature engineering methods are also applicable here. In fact, multiple representations can lead to strongly correlated features since they are essentially generated from the same time series (with different transformations). An extra step to filter unnecessary features might be useful in practice. However, we demonstrate in our experiments that, even with the absence of such filters, the model learnt upon the SEQL feature selection is very accurate on test data.

4.4 SEQL with multiple representations from multiple domains

So far we have discussed the algorithms in the scenario of single-type representations, i.e., either SAX or SFA. However, as SEQL can work with both representations, making use of both for classification is fairly straightforward. The ensemble M can contain both SAX and SFA models: $M = M_{SAX} \cup M_{SFA}$. The set of features F can also contain both SAX and SFA features: $F = F_{SAX} \cup F_{SFA}$.

Table 4 summarizes different combinations between the symbolic representations and all proposed variants of SEQL-based algorithms. We prefix the algorithms that use multiple representations with 'mt'.

4.5 Time and space complexity

Table 5 summarize the theoretical complexity of our methods in comparison with well-known time series classifiers. The time complexity of SEQL is proportional to the number of the subsequences it has to evaluate. In the worst case, SEQL has to explore the complete subsequence space, i.e., when it fails to prune any part of the

Table 4 Combinations of symbolic representations and variants of SEQL-based algorithms

Input Type	Symbolic Representation	SEQL variant	Name
Single	SAX	SEQL	SAX-SEQL
Single	SFA	SEQL	SFA-SEQL
Multiple	SAX	Ensemble SEQL	mtSAX-SEQL
Multiple	SFA	Ensemble SEQL	mtSFA-SEQL
Multiple	SAX and SFA	Ensemble SEQL	mtSS-SEQL
Multiple	SAX	SEQL as Feature Selection	mtSAX-SEQL+LR
Multiple	SFA	SEQL as Feature Selection	mtSFA-SEQL+LR
Multiple	SAX and SFA	SEQL as Feature Selection	mtSS-SEQL+LR

Table 5 Theoretical complexity of state-of-the-art time series classifiers

Algorithm	Theoretical Complexity
mtSAX-SEQL	$\mathcal{O}(NL^{\frac{3}{2}} \log L)$
mtSFA-SEQL	$\mathcal{O}(NL^{\frac{3}{2}})$
BOSS	$\mathcal{O}(N^2L^2)$
BOSS VS	$\mathcal{O}(NL^{\frac{3}{2}})$
WEASEL	$\mathcal{O}(NL^2)$
EE_PROP	$\mathcal{O}(N^2L^2)$
COTE	$\mathcal{O}(N^2L^4)$
Fast Shapelet	$\mathcal{O}(NL^2)$
Learning Shapelet	$\mathcal{O}(NL^2)$

tree. Let N_s denote the total number of sequences and l_s denote the length of the sequence. The time complexity of SEQL-based classifiers is then:

$$\begin{aligned} T(\text{SEQL}) &= \mathcal{O}(N_s(1 + \cdots + l_s)) \\ &= \mathcal{O}(N_s l_s^2) \end{aligned} \tag{4}$$

For the symbolic representation of time series, in our algorithm each word is counted as a sequence, accordingly $N_s \leq N(L-l)$ and $l_s = w$:

$$\begin{aligned} T(\text{SAX-SEQL}) &= T(\text{SAX}) + T(\text{SEQL}) \\ &= \mathcal{O}(NL \log L) + \mathcal{O}(N(L-l)w^2) \\ &= \mathcal{O}(NL \log L) + \mathcal{O}(NLw^2) \end{aligned} \tag{5}$$

In practice, w is often a constant, hence:

$$\begin{aligned} T(\text{SAX-SEQL}) &= \mathcal{O}(NL \log L) + \mathcal{O}(NL) \\ &= \mathcal{O}(NL \log L) \end{aligned} \tag{6}$$

If multiple representations are used, the complexity also depends on the number of representations, which is approximately $\sqrt{L}$:

$$T(\text{mtSAX-SEQL}) = \mathcal{O}(NL^{\frac{3}{2}} \log L) \tag{7}$$

Similarly, if the type of the representation is SFA:

$$T(\text{SFA-SEQL}) = \mathcal{O}(NL) \tag{8}$$

$$T(\text{mtSFA-SEQL}) = \mathcal{O}(NL^{\frac{3}{2}}) \tag{9}$$

In terms of memory, SEQL needs space to store the training data and the subsequence tree. Generally, the size of the tree is bounded by the alphabet and the length of the subsequences. The length of the subsequences in turn is bounded by the word length parameter w. As a result, the number of nodes in the worst case is $\alpha^w + \alpha^{w-1} + \cdots + \alpha = \dfrac{\alpha^{w+1} - \alpha}{\alpha - 1}$ which is nevertheless a constant since α and w are fixed. The tree requires an inverted index structure in which each node stores a list of indexes. This list can be as long as the size of the training data, i.e., $\mathcal{O}(N)$. Overall, the memory requirement grows linearly in accordance to the training data.

In practice, SEQL almost never reaches the worst scenario. When using symbolic transformations, repeated words can be discarded (often referred to as numerosity reduction in related literature (Lin et al. 2003)). The pruning technique in SEQL is effective in practice and drastically diminishes the number of subsequences to be evaluated. Finally, in the case of multiple representations parallelism is possible since the individual training of models in a representation are independent.

5 Evaluation

We study two symbolic representations (i.e., SAX and SFA) and evaluate new variants of SEQL-based algorithms for the TSC task. To test our approaches, we experiment in total with 8 different combinations between the two symbolic representations and the SEQL variants (Table 4). The parameter settings for the experiments are informed by existing literature on symbolic representations for TSC (Lin et al. 2007; Schäfer 2016; Schäfer and Leser 2017a; Nguyen et al. 2017) and are set as shown in Table 6. For multiple representations we use a minimum sliding window of size $l = 16$ and an increment step of $\sqrt{L}$. We set $w = 16$, $\alpha = 4$ for SAX as by Nguyen et al. (2017), and $w = 8$, $\alpha = 4$ for SFA, as by Schäfer (2016). For SEQL we use default parameters as by Ifrim and Wiuf (2011).

Our proposed algorithms were tested on all 85 datasets of the UCR Time Series Classification Archive (Chen et al. 2015). The archive has been incrementally extended by researchers working with time series and contains a vast collection of data from multiple domains and problem types. It is perhaps the most common used benchmark for recent studies on TSC. We performed all experiments with the default single split of training and test set given by the benchmark. Our test system is a Linux PC with

Table 6 Parameter settings for the experiments: l is the window size, w is the word size and α is the alphabet size

Input representation(s)	l	w	α
Single SAX	Fixed: $0.2 * L$	16	4
Single SFA	Fixed: $0.2 * L$	8	4
Multiple SAX	Varied: minl=16, increment $\sqrt{L}$	16	4
Multiple SFA	Varied: minl=16, increment $\sqrt{L}$	8	4

Intel Core i7-4790 Processor (Quad Core HT, 3.60GHz), 16GB 1600 MHz memory and 256 Gb SSD storage. All our code was developed in C++ and can be found at https://github.com/lnthach/Mr-SEQL.

Following are short recaps of all state-of-the-art algorithms included in the next sections. Footnotes indicate from where we obtained the results. In some cases, we were unable to obtain the full results on all UCR datasets from the original authors, hence we opted to use the reproduced results from more recent studies.

- **COTE** (Flat-COTE (Bagnall et al. 2015) and HIVE-COTE (Lines et al. 2016)) are large ensembles of different time series classifiers.[3]
- **BOSS and WEASEL** (Schäfer 2015; Schäfer and Leser 2017a) are multi-resolution SFA-based time series classifiers.[4]
- **Deep Learners** (Ismail Fawaz et al. 2019) include FCN, ResNet, MLP, Encoder, Time-CNN, MCDCNN, MCNN, t-LeNet and TWIESN.[5]
- **LS** (Learning Shapelet) (Grabocka et al. 2014) learns shapelets from time series by optimizing an objective function[4].
- **EE_PROP** (Lines and Bagnall 2015) is an ensemble of distance-based classifiers[3].
- **TSBF** (Baydogan et al. 2013) uses Random Forest on a generated bag of features[3].
- **DTW and DTW_CV** (Lines and Bagnall 2015) are 1NN classifiers with DTW as distance measurement. DTW_CV uses cross validation to set a constraint for the warping window[4].
- **ST** (Shapelet Transform) (Lines et al. 2012) transforms the data to shapelet space to improve classification accuracy[3].
- **FS** (Fast Shapelet) (Rakthanmanon and Keogh 2013) finds k-best shapelets in the dimension-reduced space of SAX[3].
- **SAX-VSM** (Senin and Malinchik 2013) generates a dictionary of SAX words for each class. It tunes SAX parameters with an optimization algorithm[3].

For comparison of multiple state-of-the-art classifiers, we follow the recommendations by Demšar (2006), Garcia and Herrera (2008), Benavoli et al. (2016): we first detect the significant differences in rankings with the Friedmann test and follow with the Wilcoxon signed rank test with Holm's correction and used the Critical Difference diagram (CD) for visualization (e.g., Fig. 7). This type of diagram is used very often

[3] http://www.timeseriesclassification.com/results.php.

[4] https://www2.informatik.hu-berlin.de/~schaefpa/weasel/.

[5] https://github.com/hfawaz/dl-4-tsc.

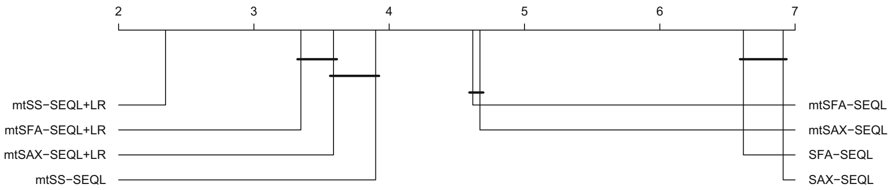

Fig. 7 Average error ranking of compared SEQL-based classifiers ordered (left-best) based on performance on the UCR Archive. The best model mtSS-SEQL+LR is a linear model that combines features from SAX and SFA representations

in the literature on TSC for visual comparison of different methods across different datasets (Bagnall et al. 2017; Schäfer 2015). The diagram shows the classifiers on a spectrum of average error ranking (the rank of the method with regard to classification error, averaged across 85 datasets), therefore classifiers to the left of the diagram (lower rank) perform better than classifiers to the right. The cliques (thick horizontal black lines) group methods that do not have a statistically significant difference in performance. All procedures were performed using the tool *scmamp* (Calvo and Santaf 2016).

5.1 Comparison of algorithms

To evaluate our approach, we first look at how the variants of SEQL-based classifiers fare against each other. Next we examine the group of SAX-based methods and then SFA-based methods. Finally, we compare our most accurate representative against the state-of-the-art.

5.1.1 SEQL-based methods

Figure 7 is a CD diagram that presents the ranking of our proposed algorithms, based on the error attained on the UCR Archive.

As it can be seen, there is no critical difference between SAX-based classifiers (SAX-SEQL) and SFA-based classifiers (SFA-SEQL) even though SFA-based classifiers seem to perform better on average. The figure suggests that the substantial improvement in accuracy arrives from the combination of multiple representations, either by the ensemble of models or by combining features: the more symbolic representations are used, the more accurate the resulting classifier is. From right to left in the critical diagram (Fig. 7), we start with a single representation (SAX/SFA), then add more representations of the same type (mtSAX/mtSFA), and finally combine representations of different types (mtSS). By combining representations from multiple-resolutions and multiple-domains to create features, we allow the classifier to select only those representations and features that represent the data well, and we do not have to decide in advance what are suitable symbolic parameters for the representations. We also note that the second algorithm we propose, which combines: (i) multiple symbolic representations, (ii) feature selection with SEQL and (iii) a logistic regression classifier (mtSS-SEQL+LR) performs better than the ensemble algorithm

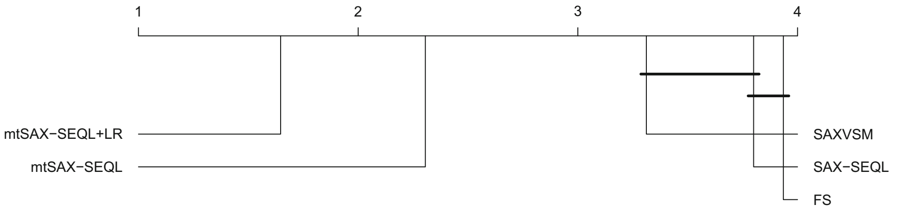

Fig. 8 Average error ranking of compared SAX-based classifiers ordered (left-best) based on performance on the UCR Archive

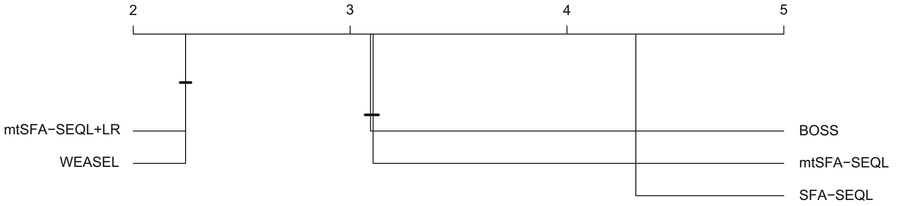

Fig. 9 Average error ranking of compared SFA-based classifiers ordered (left-best) based on performance on the UCR Archive

(mtSS-SEQL). This is an interesting finding: creating a single rich feature space that combines symbolic representations and training a linear model delivers better accuracy than ensembling models trained with different representations. The linear model also has the advantage that it is simple to interpret, as we discuss in Sect. 7.

5.1.2 SAX-based methods

SAX-VSM and Fast Shapelet are perhaps the most well-known time series classifiers which utilize the SAX transformation. However, both have been already shown to be inferior in accuracy to more recent state-of-the-art methods. Our multi-resolution SAX-based variants also easily outperform both methods as can be seen in Fig. 8.

From the diagram, we also notice that the difference between SAX-VSM and the single SAX-SEQL model is not critical, even though the former optimizes the parameters, while the latter uses a set of fixed parameter values. On the other hand, the low performance of Fast Shapelet may suggest the method fails to select the most discriminative features, despite the fact that it also employs multiple representations (multiple SAX resolutions).

5.1.3 SFA-based methods

The SFA-based algorithm family includes WEASEL, BOSS and our SEQL-based classifiers (SFA-SEQL, mtSFA-SEQL and mtSFA-SEQL+LR). We excluded the mtSS variants since they incorporate not only SFA but also SAX representations.

Figure 9 shows the comparison between SFA-based classifiers. Interestingly, the performance of our algorithms matches those of BOSS (for mtSFA-SEQL) and WEASEL (for mtSFA-SEQL+LR) with almost no difference in terms of average

Table 7 TSC algorithms grouped in comparison to DTW_CV

Significantly better	Not significantly different	Significantly worse
HIVE-COTE (+8.7%)	Encoder (−1.1%)	DTW (−2.2%)
mtSS-SEQL+LR (+7.9%)	SAX-VSM (−1.8%)	MLP (−3.4%)
Flat-COTE (+7.8%)		Time.CNN (−4%)
WEASEL (+7.3%)		FS (−5.3%)
ResNet (+6.5%)		MCDCNN (−7.8%)
ST (+6.2%)		TWIESN (−8%)
BOSS (+5.4%)		MCNN (−37.5%)
FCN (+4.8%)		t.LeNet (−39.8%)
EE_PROP (+3.3%)		
LearningShapelet (+2.7%)		
TSBF (+1.8%)		

The percentage in the brackets gives the difference in mean accuracy over the UCR Archive. Positive value means that this method is on average more accurate than the DTW_CV benchmark and vice versa

ranking. It is worth noting that the output of the SFA transformation used in our experiments is more similar to the transformation used in BOSS, rather than in WEASEL. For WEASEL, the authors applied new transformation techniques including non-overlapping windows, bi-gram sequences and full ranges of sliding window sizes. The diagram highlights the potential advantage of our SEQL-based algorithms in exploring the all-subsequence SFA word space, an ability which the BOSS and WEASEL classifiers lack.

5.1.4 Comparing to the state-of-the-art

In this section, we investigate 22 time series classifiers including our representative mtSS-SEQL+LR. By picking DTW_CV as the benchmark classifier (a common choice in literature (Bagnall et al. 2017; Lines and Bagnall 2015; Schäfer and Leser 2017a)), we divide the rest to three different groups based on the results of a Wilcoxon signed rank test at a cutoff of $p = 0.05$ (Table 7): significantly better, significantly worse or not significantly different to the benchmark.

Figure 10 shows the CD diagram for the classifiers which are found significantly better than the DTW_CV benchmark. Ranking wise, HIVE-COTE takes the lead in average ranking followed by our method mtSS-SEQL+LR. Note that the COTE ensembles consist of time-based and frequency-based classifiers, hence they also rely on knowledge extracted from multiple domains. Nevertheless, the diagram also suggests that there is no significant difference between COTE classifiers and mtSS-SEQL+LR regarding their accuracy on the UCR benchmark.

Regarding time and space efficiency, FCN, ResNet and COTE are very demanding of computation resources and did not run on our machine. We also found that WEASEL requires large amounts of memory to run, likely due to its enormous feature space. In our experiments on a regular Linux PC we got out-of-memory errors when we tried to run WEASEL even on moderately large datasets (e.g., a few hundred training time

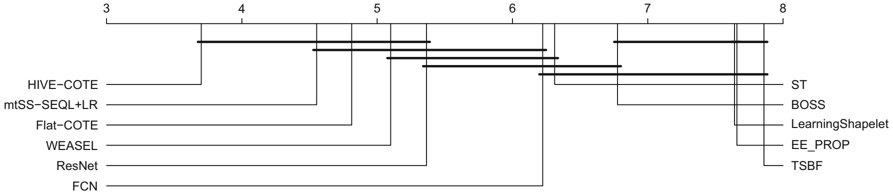

Fig. 10 Average error ranking of the most accurate classifiers ordered (left-best) based on performance on the UCR Archive

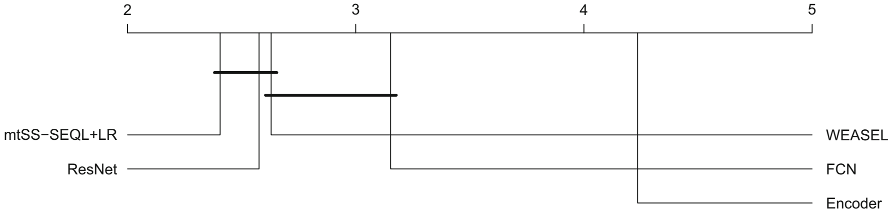

Fig. 11 Average error ranking of the most accurate linear classifiers and deep learning classifiers ordered (left-best) based on performance on the UCR Archive. Linear classifiers are much simpler but have comparable Accuracy to deep learners

series). Since these methods do not run on a regular PC (and many assume access to significant computation resources such as GPU clusters), the accuracy of the state-of-the-art methods is reported from published results. This also raises interesting research questions regarding the feasibility of running many of the state-of-the-art TSC methods under strict resource-constrained requirements (e.g., running TSC on a mobile phone).

In Fig. 11, we examine the difference between the most accurate linear algorithms (mtSS-SEQL+LR and WEASEL) and deep learning algorithms. For deep learning algorithms, we select Encoder which is comparable to the benchmark (DTW_CV) and FCN, ResNet (Ismail Fawaz et al. 2019) which are significantly better than the benchmark. The diagram shows no significant difference between mtSS-SEQL+LR, ResNet, and WEASEL while both mtSS-SEQL+LR and ResNet are significantly better than FCN.

5.1.5 Comparing TSC algorithms by problem type

Following the discussion by Bagnall et al. (2017), we also measured the performance of our proposed algorithms for each type of data in the UCR Archive. In summary, there are 7 types of data: Image Outline (29 datasets), Sensor Readings (16 datasets), Motion Capture (14 datasets), Spectrographs (7 datasets), Electric Devices (6 datasets), ECG measurements (7 datasets) and Simulated (6 datasets).

Table 8 reports the average error rank of each SEQL-based method on each type of data. SFA-based classifiers appear to not be suitable for time series data from Electric Devices. In addition, we can also see the difference between SAX and SFA from the table, by comparing mtSAX-SEQL+LR and mtSFA-SEQL+LR (same classifier but

Table 8 Average Error Rank by problem type for SEQL-based classifiers

Methods	IMG	SPECTR	SENSOR	SIMUL	ECG	DEV	MOTION
mtSS-SEQL+LR	**2.59**	**3.36**	**2. 22**	**1.75**	**1.71**	2.67	**1.93**
mtSFA-SEQL+LR	3.24	3.93	3.16	3.67	2.57	4.83	3.11
mtSAX-SEQL+LR	4.17	4.50	3.34	2.75	2.93	**1.50**	3.79
mtSS-SEQL	4.16	4.57	3.94	4.25	4.00	3.17	3.11
mtSFA-SEQL	4.17	5.43	4.66	4.67	4.57	6.25	4.39
mtSAX-SEQL	4.53	3.50	5.28	4.58	5.21	3.33	5.18
SFA-SEQL	6.38	4.93	6.56	7.33	7.36	7.67	6.89
SAX-SEQL	6.76	5.79	6.84	7.00	7.64	6.58	7.61

Bold values indicate best method

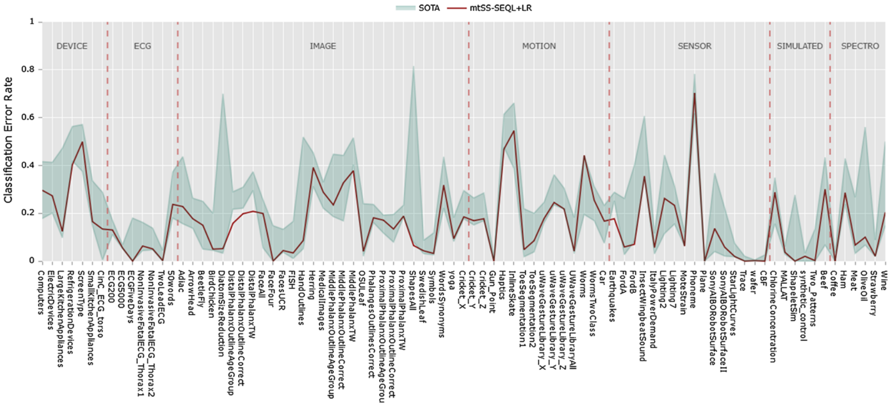

Fig. 12 Classification error rate of mtSS-SEQL+LR on 85 UCR Archive datasets. The light-colored area depicts the overall performance of the other ten state-of-the-art classifiers shown in the first column of Table 7

with different input representation). It seems SFA has an advantage in Motion and Image categories while SAX has an advantage in Electric Devices category.

Figure 12 provides an overview of mtSS-SEQL+LR (our best variant) performance in each group of problems compared to other state-of-the-art classifiers. To analyze the statistical differences, we performed the same test done in Sect. 5.1.4 but separately for each type of data. Furthermore, we chose our classifier (mtSS-SEQL+LR) to be the benchmark this time. We found that, with the exception of Motion data, where HIVE-COTE is found to be significantly better, state-of-the-art classifiers are either significantly worse or not different to ours (Table 9). The table also notes the difference in average error between mtSS-SEQL+LR and the corresponding method (negative difference means mtSS-SEQL+LR is more accurate). The test also suggests that it is harder to establish the difference between classifiers when the sample size is small, as in the case of problems with fewer datasets.

We also note that all the datasets in UCR are sampled to fixed length, an artifact which hides the fact that most state-of-the-art methods do not work on variable-length time series. We elaborate more on this issue in Sect. 7.

Table 9 TSC algorithms grouped in comparison to mtSS-SEQL+LR for each problem type

Data type	Significantly better	Not significantly different
Image (29 datasets)		HIVE-COTE (+0.3%), WEASEL (−0.6%), Flat-COTE (−1.1%) ResNet (−4%), FCN (−4.6%)
Sensor (16 datasets)		Flat-COTE (+0.8%), ResNet (+0.5%), HIVE-COTE (−0.4%), FCN (−0.9%), LearningShapelet (−0.9%), ST (−1.2%), WEASEL (−2.3%)
Motion (14 datasets)	HIVE-COTE (+1.5%)	Flat-COTE (+1%), WEASEL (+0.8%), ST (−0.3%), BOSS (−1.6%)
ECG (7 datasets)		Flat-COTE (+1.9%), HIVE-COTE (+1.4%), WEASEL (+1.3%), ST (0%), FCN (−0.2%), ResNet (−0.9%), Encoder (−1.4%), BOSS (−1.9%), EE_PROP (−3.2%), MLP (−4.5%)
Spectro (7 datasets)		HIVE-COTE (+2.3%), WEASEL (+1.1%), ST (+0.5%), ResNet (+0.2%), BOSS (−1%), Flat-COTE (−1.3%), SAXVSM (−1.9%), TSBF (−5.2%), Time.CNN (−11.1%)
Simulated (6 datasets)		WEASEL (+0.5%), HIVE-COTE (+0.3%), Flat-COTE (−0.4%), ResNet (−1.1%), ST (−1.3%), LearningShapelet (−1.9%), TSBF (−3.2%), EE_PROP (−4.2%), FCN (−5%), DTW (−7.2%), MLP (−10.6%), Encoder (−12.2%)
Device (6 datasets)		HIVE-COTE (+2.6%), ResNet (+2.3%), FCN (+1.7%), ST (−0.1%), Flat-COTE (−1.2%), BOSS (−3.8%), SAXVSM (−4.9%), WEASEL (−4.9%), TSBF (−10%)

The percentage in the brackets gives the difference in mean accuracy over all datasets of the respective type. Positive value means that this method is on average more accurate than the benchmark (mtSS-SEQL+LR) and vice versa

Table 10 Average running time (seconds) of SEQL-based time series classifiers across all UCR datasets

Methods	Transform	TotalLearn	MaxLearn	TotalTest	MaxTest	LogReg
SAX-based	83.219	131.530	3.704	67.748	2.554	5.829
SFA-based	4.210	66.127	2.233	18.036	0.729	5.941

5.2 Running time of our algorithms

Table 10 reports the average running time (in seconds) for each step in our experiments. As it was discussed previously, the independent training of each representation makes it possible to parallelise our algorithm. However, our current implementation is limited to sequential programming. Therefore, beside the total time for training (TotalLearn) and testing (TotalTest) we also report the longest time for training (MaxLearn) and testing (MaxTest) of a single representation per dataset, as the theoretical optimal running time for a parallel implementation. In summary, we show the runtime for the following steps in our algorithms:

- Transform: Average running time to transform raw data to symbolic representation. Note that we used the authors' implementation (Schäfer 2016) for SFA transformation.
- TotalLearn: Average training time in the case of multiple representations.
- MaxLearn: Maximum average training time in the case of single representation.
- TotalTest: Average testing time in the case of multiple representations.
- MaxLearn: Maximum average testing time in the case of single representation.
- LogReg: Average training and testing time with logistic regression.

We attempted to reproduce the experiments by other studies including SAX-VSM, BOSS, WEASEL, FCN and ResNet in order to have a fair comparison in terms of efficiency. However, except BOSS, none of these methods managed to complete the experiment on our machine (a regular PC). To the best of our knowledge the current implementation of BOSS is the most efficient implementation among the top accuracy classifiers. On the other hand, WEASEL demands a lot of memory which was simply not available on the machine we used. We suspect WEASEL's rich feature space and the absence of effective pruning techniques are the causes for this huge demand of computing resources. In Sect. 7 we report the time and memory used by these algorithms to train and predict on a moderate-size human motion dataset.

5.3 Comparing the impact of multiple representations on accuracy and speed

Previous experiments suggest that the strength of our models derives from multiple symbolic representations of time series. Adding representations seems to be the main factor for more accurate classification. However, it also raises computing cost as well as the risk of overfitting. We are also interested in the effect of combining representations from different domains.

In this experiment, we adjust the number of representations and observe the impact on accuracy and running time. This can be done by expanding or shrinking the step

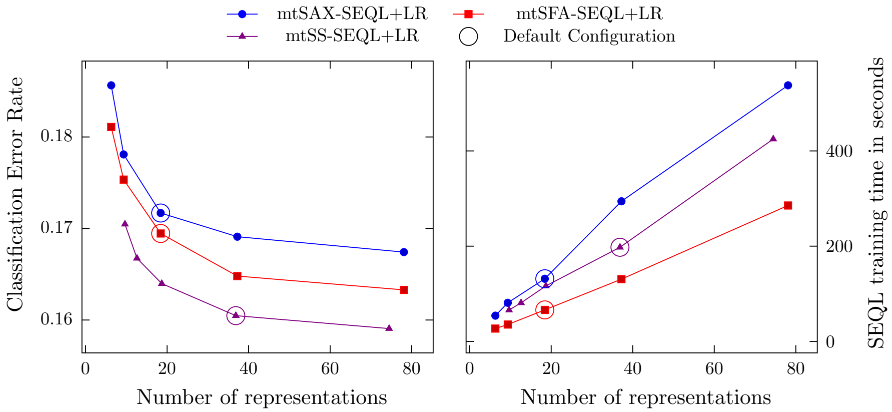

Fig. 13 Impact of multiple symbolic representations on TSC: trade-off between accuracy and run time when increasing the number of input symbolic representations. All measurements are averaged across the UCR Time Series Archive

when varying the window size l, e.g., instead of incrementing by $\sqrt{L}$, we can increment by $0.5\sqrt{L}$ to increase the number of representations. The default configuration is similar to that of BOSS (Schäfer 2015), i.e., varying different window sizes with a step of $\sqrt{L}$, as in the experiments for previous sections. The experiment was conducted with 3 variants of our SEQL-based classifier: mtSAX-SEQL+LR, mtSFA-SEQL+LR and mtSS-SEQL+LR.

Figure 13 visualizes the results of the experiment. Indeed, feeding more representations to the classifier results in more accurate models, with extra cost however. In fact, our method can even achieve a lower error rate than the default configuration we used in the previous experiements. This experiment also suggests that multi-domain models (mtSS) are more accurate than single-domain models (mtSAX and mtSFA) given the same number of input representations (e.g., 20 representations for mtSS achieves lower error than 20 mt-SAX or 20 mt-SFA representations).

6 Interpretability

As described in Sect. 4, the output of our best SEQL-based algorithm is a linear model (a weighted list of selected features) which makes interpretation possible. In this section, we focus on classifier interpretation in the context of TSC, i.e., how can we identify the time series segments that are important for the classification decision. Since we visualize the data in the time domain, we only discuss SAX-SEQL-based models here. Regarding SFA representations, there have so far been no results that report whether SFA classifiers are interpretable. We visualize SAX sequences by mapping each SAX word back to its corresponding segment in the original time series. Technically, the same mapping can be done with SFA sequences, but currently we are not aware of good ways to visualize the impact of such (frequency domain) features, in particular since it is more intuitive to a human to visualise data in the time domain.

Table 11 Top 10 features selected by mtSAX-SEQL+LR from the GunPoint time series dataset

l	w	a	Coefficients	Subsequences
42	16	4	0.066	cbaab
53	16	4	0.062	db
53	16	4	0.062	ddddb
42	16	4	0.062	da
31	16	4	0.060	bbbbbbbbbbcdddd
53	16	4	−0.054	aaaaaabbbb
20	16	4	−0.054	bbbbaaaaaa
53	16	4	−0.055	bbbcddddd
53	16	4	−0.056	bbbbbbbaaa
53	16	4	−0.061	bbbbbbaaa

6.1 Feature importance

For our TSC algorithm mtSAX-SEQL+LR we can evaluate the importance of the features selected in the final model by studying the coefficients learned by logistic regression. Basically, the coefficient of a feature implies which class the feature represents (based on the sign) and how decisive the feature is in the classification decision (based on the absolute value). For multiple representations, a feature is defined not only by the sequence but also by its SAX parameters. Table 11 shows some of the features selected by the mtSAX-SEQL+LR algorithm on the GunPoint UCR time series dataset.

6.2 Visualizing SAX features

Two examples of time series from the GunPoint dataset are shown in Fig. 14. The time series records the motion of the hand when pointing (Point class) or drawing a gun (Gun class). The distinction between two time series can be observed at both the beginning, where a small bump describes gun-drawing action, and the end, where a little dip suggests the hand moved past the holster because there is no gun. The highlighted regions were discovered by our mtSAX-SEQL+LR classifier by mapping the matched features back to the raw time series (Algorithm 7).

Algorithm 7 Mapping symbolic features back to the original time series.

```
function FINDSEGMENTS(feature, timeseries)
    Initialize the meta time series $mtts = zeros(lengthof(timeseries))$
    $sax = SAXtransform(timeseries, feature.l, feature.w, feature.\alpha)$
    Find all the locations of $feature.sequence$ in $sax$.
    for all $loc$ in $locations$ do
        $mtts[loc] += feature.coef/locations.size()$
    end for
    return $mtts$
end function
```

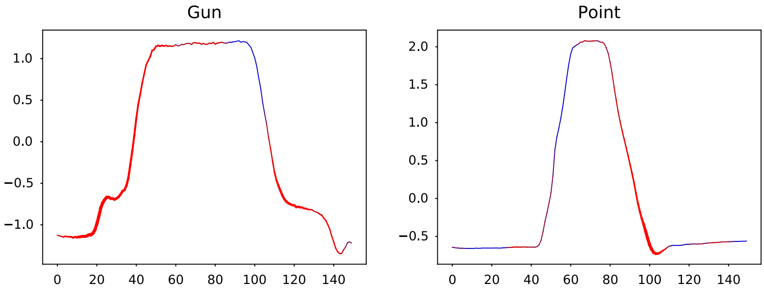

Fig. 14 Example from the GunPoint dataset. The discriminative regions for the respective class are thickened and highlighted with red color

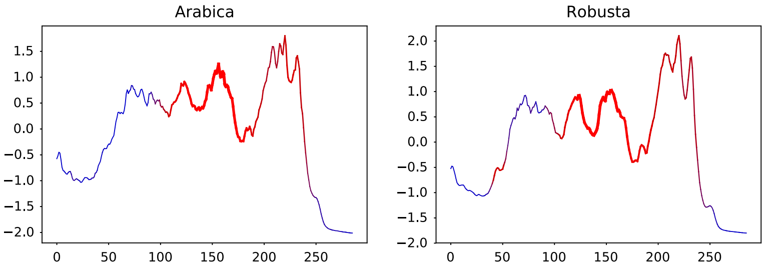

Fig. 15 Example from the Coffee dataset. The discriminative regions are thickened and highlighted with red color

Figure 15 presents another two examples from the Coffee dataset: one from the Arabica class and one from the Robusta class. The highlighted regions correspond to the caffeine and chlorogenic acid components of the coffee blends (Briandet et al. 1996).

Our discoveries on the Coffee and GunPoint datasets have been well-documented by similar studies (Ye and Keogh 2009; Senin and Malinchik 2013; Grabocka et al. 2014). This suggests that our method is capable to identify the key discriminative regions of the time series. In the next section, we present a detailed case study in which we study the interpretability of our method on a real-world problem.

7 Case study: human motion time series classification

To comprehensively investigate the interpretability of the proposed algorithms, we conducted a case-study on a dataset collected and well understood by the authors. The dataset describes the assessment of motion patterns during jumping performance testing in athletes. Below we describe the problem, the data collection process and the analysis of TSC algorithms.

7.1 Problem description

The assessment of power output is an important component of athlete performance testing. Lower limb power is commonly assessed using the Countermovement Jumps (CMJ) as it is a simple test and has been shown to be related to maximal and explosive strength performance (Nuzzo et al. 2008). In order for a CMJ test to be considered acceptable, athletes must jump with adherence to acceptable technique. The athlete stands upright with weight evenly distributed over both feet. Hands are placed on the hips and stay there throughout the test. When ready, the athlete squats down until the knees are bent at 90 degrees, then immediately jumps vertically as high as possible, landing on both feet at the same time. Commonly observed deviations from acceptable technique, which indicate a failed test, include the athlete bending their legs in the air and stumbling upon landing (Markovic et al. 2004).

Traditionally, CMJ tests are measured with large, cumbersome equipment such as force plates and infrared-based systems (Glatthorn et al. 2011). More recently, it has been shown that accelerometer-based assessment of CMJ tests can improve equipment affordability and portability (O'Reilly et al. 2018; Picerno et al. 2011). With the arrival of these new systems for measuring CMJ performance, it is important to ensure that they can automatically detect aberrant CMJ technique, which indicates a failed test, so that incorrectly completed tests are not saved as part of an athlete's performance profile. In this case-study we collected a dataset which captured acceptable CMJ tests and failed CMJ tests whereby the athlete bended their legs during flight or stumbled upon landing (3 classes of jumps: Normal, Bending, Stumble). The aims of the study were to compare the classification techniques presented in this paper to other methods commonly used in the literature, with regards to accuracy, time/space efficiency and interpretability.

7.2 Data collection

Ten participants (3 females, 7 males, age: 26.6 ± 2.2, weight: 80.1 ± 7.4 kg , height: 1.8±0.1 m) were recruited for this case-study. The Human Research Ethics Committee at University College Dublin approved the study protocol and written informed consent was obtained from all participants before testing. Participants did not have a current or recent musculoskeletal injury that would impair performance of CMJs. Participants were equipped with a Shimmer 3 (Shimmer, Dublin, Ireland) inertial measurement unit (IMU) on their dominant foot. The IMU was configured to stream wide range, tri-axial accelerometer data (±16 g) at 1024 Hz. Each participant completed 20 CMJs with acceptable form, 20 jumps with their legs bending during flight and 20 jumps with a stumble upon landing. The resulting 3-class dataset consists of 200 files of IMU data in the acceptable form class and, due to Bluetooth dropping twice during data collection, 199 files of IMU data in each of the 'legs bending' and 'stumble on landing' classes. The length of the IMU time series signals in each file ranged from 1231 to 6710 samples.

7.3 Data analysis

Acceleration magnitude was first computed from the accelerometer x, y and z signals whereby:

$$A_m = \sqrt{A_x^2 + A_y^2 + A_z^2}.$$

The signal used, as included in the data donated with this paper, was the acceleration magnitude inclusive of the inertial and gravitational accelerations acting on the sensor device. This signal is appropriate for computing jump performance metrics, e.g., time in the air, jump height and classifying jump technique quality as described in this case-study. For the purposes of jump assessment, the removal of gravitational acceleration from the sensor's signal(s) is not important and has not been included in the methodology. Classification was then completed using the methods introduced in this paper, state-of-the-art methods from the TSC literature and feature-based methods (Table 12). The feature-based methods analysed were Support Vector Machines (SVM) and Random-Forests. The features used were mean, root-mean-square, standard deviation, kurtosis, median, skewness, range, variance, max value, minimum value, energy, 25th percentile, 75th percentile, level crossing rate, fractal dimension, index of minimum value, index of maximum value, number of signal peaks and length of signal epoch (O'Reilly et al. 2017). The Random-Forest method used 1000 trees. The SVM used a quadratic kernel function and a box constraint of 1. All classifiers were trained and evaluated using three settings. In the first setting, each full signal epoch was used (full signals data). In the second setting, the epoch was cropped whereby the active region of the jump was extracted (cropped signals data). This was completed by first computing the mean of the first 100 samples of the signal, the start point of the epoch was then set as the first value in the signal which was three times greater than the mean value. The end sample of the epoch was found by iterating backwards across the signal epoch and identifying the same threshold. In the third setting, all the cropped signal epochs were re-sampled to a fixed length of 500 samples (cropped and resampled to fixed length signals data). For this three-class (Normal, Bending, Stumble) classification problem the training set was the data from 70% of the participants and the test set was data from 30% of participants. No training and test data were from the same participants. The interpretability of all classification methods was considered by a domain expert in jump technique biomechanics and exercise classification with wearable sensors.

7.4 Results

The classification error rate, runtime, and memory usage for each technique are shown in Tables 12, 13, and 14 respectively. For memory usage, we monitored the memory activity of the process with *psrecord*[6] and report only the highest peak. Note that we also tested HIVE-COTE for this case study, but the algorithm failed to complete the

[6] https://github.com/astrofrog/psrecord.

Table 12 Classification error rate of each method compared on full signals (F), cropped signals (C) and cropped and re-sampled to fixed length signals (CR)

Methods	F	C	CR
mtSAX-SEQL+LR	**0.123**	0.056	0.028
WEASEL	(-)	**0.022**	**0.017**
BOSS	(-)	0.251	0.117
BOSS VS	0.458	0.162	0.045
FCN	(–)	(–)	0.045
ResNet	(–)	(–)	0.067
SVM	0.195	0.078	0.084
RandomForest	0.251	0.061	0.039
1NN-DTW	(–)	(–)	0.061

(-) The algorithm throws out-of-memory error in the experiment. (–) The implementation of the algorithm only supports fixed length time series. Best method in bold

Table 13 Runtime in seconds of each method compared on full signals (F), cropped signals (C) and cropped and re-sampled to fixed length signals (CR)

Methods	F	C	CR
mtSAX-SEQL+LR	1717.288	139.045	102.473
WEASEL	(-)	688.523	307.247
BOSS	(-)	724.522	406.285
BOSS VS	127.658	17.161	13.133
FCN	(–)	(–)	8900.759
ResNet	(–)	(–)	14259.69
SVM	**13.1**	**11.05**	**10.51**
RandomForest	18.89	16.33	15.76
1NN-DTW	(–)	(–)	1167.986

(-) The algorithm throws out-of-memory error in the experiment. (–) The implementation of the algorithm only supports fixed length time series. Best method in bold

experiment in a reasonable amount of time on our system (we stopped it after 5h of training), hence we do not report results for this method.

When classifiers were trained and evaluated with the full signal epochs (variable-length time series), the mtSAX-SEQL+LR algorithm produced the lowest error rate.[7] A number of methods could not be evaluated on the full signals data (F) as the varying length of the time series was not supported by the method or attempting to train the method caused out-of-memory errors. The error rate dropped for all methods when classifiers were trained and evaluated with the domain specific cropped signal epochs (C). In this scenario WEASEL has the lead, with the mtSAX-SEQL+LR, SVM and Random-Forest methods also performing well. In the final scenario, whereby the

[7] The error rates of mtSAX-SEQL+LR, mtSFA-SEQL+LR and mtSS-SEQL+LR were comparable, but we only discuss mtSAX-SEQL+LR here since it is also interpretable. Note that we used a default number of symbolic representations/resolutions for mtSAX-SEQL+LR (increasing the window length with a step $\sqrt{L}$). By increasing the number of SAX representations, we can further improve the accuracy of mtSAX-SEQL+LR, as also discussed in Sect. 5.3.

Table 14 Memory usage in MBs of each method compared on full signals (F), cropped signals (C) and cropped and re-sampled to fixed length signals (CR)

Methods	F	C	CR
mtSAX-SEQL+LR	537.762	**127.02**	**117.559**
WEASEL	(-)	2860.457	2398.332
BOSS	(-)	2688.645	1973.254
BOSS VS	3498.031	1552.613	1494.125
FCN	(–)	(–)	801.609
ResNet	(–)	(–)	892.578
SVM	**497.609**	495.707	502.438
RandomForest	528.016	519.609	520.039
1NN-DTW	(–)	(–)	132.562

(-) The algorithm throws out-of-memory error in the experiment. (–) The implementation of the algorithm only supports fixed length time series. Best method in bold

cropped signals were all re-sampled to a fixed length of 500 samples (CR), almost all methods achieved a very low error rate.

In terms of running time (Table 13), feature-based classifiers (SVM and Random Forest) have a clear advantage, with only BOSS VS as the only comparable method when the signals are cropped (in both C and CR columns). We argue that this advantage is mainly gained from the domain knowledge (to handcraft features), to which the other methods are oblivious. Deep learning methods (FCN and ResNet) took the longest time to finish the experiments. On the other hand, mtSAX-SEQL+LR shows its efficiency in memory usage (Table 14). In this category, the SFA-based family of classifiers (BOSS, BOSS VS, and WEASEL) reported very high memory activity. BOSS and WEASEL even failed to complete the experiments in the case of full signals (F). Nevertheless, it is important to note that running time and memory activity strongly tie to the implementations of the algorithms, which are usually written in different programming languages (C++, Java, Python, and Matlab in this case), hence might not fully reflect the efficiency of said algorithms.

For each classifier evaluated, the interpretability of the method was considered by the domain expert. For the feature-based methods (SVM and RandomForest) and the other methods performing well (BOSS and WEASEL) the reasons for successful classification and for misclassification were largely unknown. The domain expert cited a key advantage of the mtSAX-SEQL+LR method to be the ability to visualize the sub-sections of signals which are pertinent to differentiating the classes. For instance, this approach allows to clearly highlight the segment of the signal which represents a stumble upon landing (Fig. 16). It also identified and highlighted the exact portion in the signal whereby participants were bending their legs in the air (Fig. 16). This was considered particularly useful when the visualizations were placed side by side to a normal jump. This allowed the domain expert to quickly understand the differences in acceleration profiles between jumps completed with acceptable technique and with known deviations from correct technique. Additionally, the mtSAX-SEQL+LR method was deemed useful for identifying less intuitive differences between classes. For instance the highlighted portion on the first landing spike in the stumble class

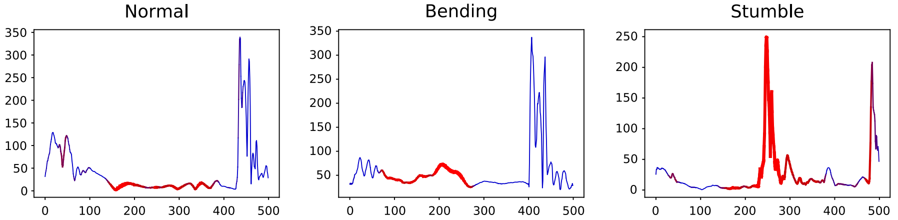

Fig. 16 Three different motions made by a person in a CMJ test. Signals were cropped and re-sampled to fixed length. The red-highlighted segments are features selected by the linear classifier mtSS-SEQL+LR and suggest the discriminative segments of the motion for the respective class

(Fig. 16) suggests that when an athlete lands correctly versus when they stumble, there is a different acceleration profile following initial contact. This was deemed new insight for the domain expert.

7.5 Discussion

Whilst many of the methods evaluated in Table 12 produced favourable accuracy results following the pre-processing steps of cropping the signal to the region representing the jumping motion and then re-sampling the time series to a fixed length, it is interesting to see that the mtSAX-SEQL+LR method achieved acceptable accuracy when the classifier was trained and tested on the raw data. This suggests that this method has a better inherent ability to factor out irrelevant information in raw signals, than the feature-based methods, the BOSS/WEASEL or the deep learning FCN/ResNet methods. It is also of interest to note that the mtSAX-SEQL+LR method was able to perform training/testing on the varying lengths of signal epochs without the necessity of altering the data by padding or re-sampling the time series to a pre-defined fixed length.

With regards to the interpretability of the presented methods, the domain expert stated that the visualizations from the mtSAX-SEQL+LR method contributed to an understanding of both how the classifiers actually worked and in understanding the differences in acceleration profiles between the various classes. They saw particular potential for the presented methods to educate athletes, using wearable sensor data, on the portions of their movements which need correcting in order to perfect a skillful, complex movement, e.g., squatting exercise or tennis serve. All the data from this case-study was donated to the UCR and UEA Time Series Classification Repository and released together with the source code for this paper to allow reproducibility and to enable the development of accurate and interpretable methods on raw, real-world, time series problems and datasets.

8 Conclusion

The goal of this study is to explore the impact of combining multiple resolutions and multiple domains of time series symbolic representations with efficient sequence classifiers, while posing both accuracy and classifier interpretability as desirable properties. In the TSC area it has been commonly perceived that DTW-based methods are hard to beat and that ensemble methods (e.g., COTE), and lately deep learning methods (e.g., FCN), offer the best accuracy. However, the series of SFA-papers (Schäfer and Högqvist 2012; Schäfer 2015, 2016; Schäfer and Leser 2017a) has shown otherwise. In particular the WEASEL method is an accurate algorithm that uses a linear classifier in a large-dimensional SFA-words space. As discussed in our experiments, WEASEL faces memory challenges due to the large feature space it uses, and is not interpretable. Our concerns about interpretability and efficiency mean that even the most accurate classifiers can still be less desirable. With this paper, we have shown that linear classifiers are strong competitors with regard to accuracy, and that symbolic representations are a powerful tool for time series analysis.

In summary, we have studied two notable symbolic representations of time series (SAX and SFA) and proposed a time series classification framework that can utilize both representations at different resolutions. Due to its pruning ability, our core classifier (SEQL) can navigate the vast symbolic-words space efficiently. We showed that symbolic approximation at multiple resolutions is an effective approach, instead of trying to find an optimal symbolic representation. Furthermore, our classifiers work with different symbolic representations, thus effectively learn from a multiple domain feature space without the need of incorporating various learning algorithms. We think that this characteristic has great potential as our classifier can theoretically accommodate other symbolic representations in the future. In practice, this flexibility means representations and resolutions can be chosen according to the problem and application domain.

We proposed 8 different SEQL-based algorithms using the SAX and SFA symbolic representations. To showcase our contributions, we tested our proposals with the full UCR Time Series Archive and demonstrated that they are strongly competitive against the state-of-the-art, including against complex algorithms such as large ensembles (COTE) or deep learning methods (FCN). While ensemble and deep learners are well-known for their accuracy, they are also notorious for their high demand of computing resources. On the other hand, our SEQL-based methods are more efficient due to the effective combination of symbolic representations and sequence learning algorithm. The time and space complexity of our algorithms also enables them to scale well for large datasets. The outcome of our best SEQL-based TSC algorithm is a linear model, which enables us to interpret the classification decision, a property which is desirable for time series analysis.

Various ways of interpreting the resulting models were also discussed in this paper. In particular, we discussed interpreting the classification decision on well known UCR problems, as well as presented a case-study on athlete performance testing, where we can relate the algorithmic decisions to real-world domain knowledge. We have also reported and discussed the accuracy, time and space efficiency for all the methods evaluated during the case-study. We pointed out that existing TSC approaches require pre-processing of the raw signal and need extensive computation resources to achieve high accuracy. This opens interesting research directions in resource-constraint TSC where the classifiers have access to limited computation resources (e.g., limited memory) for training and prediction. Such applications are already common in the deployment of TSC on mobile phones, and we intend to extend our methods to further improve their efficiency for this constrained setting. For the future, we also see a lot of potential on extending these methods for multivariate time series classification problems, as well as continuing to work on human-friendly ways of explaining the classification decisions.

Acknowledgements We would like to thank the anonymous reviewers for their detailed and constructive feedback. We would also like to gratefully acknowledge the work by researchers at University of California Riverside, USA (especially Eamonn Keogh and his team) and researchers at University of East Anglia, UK (especially Tony Bagnall and his team) and their effort in collecting, updating and making available the UCR and UEA time series classification benchmarks. We want to thank all researchers in time series classification who have made their data, code and results open source and have helped the reproducibility of research methods in this area. We acknowledge financial support for this work by Science Foundation Ireland (SFI) under grant number 12/RC/2289 (Insight Centre for Data Analytics).

References

Bagnall A, Lines J, Hills J, Bostrom A (2015) Time-series classification with cote: the collective of transformation-based ensembles. IEEE Trans Knowl Data Eng 27(9):2522–2535

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Discov 31(3):606–660. https://doi.org/10.1007/s10618-016-0483-9

Baydogan MG, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Trans Pattern Anal Mach Intell 35(11):2796–2802. https://doi.org/10.1109/TPAMI.2013.72

Benavoli A, Corani G, Mangili F (2016) Should we really use post-hoc tests based on mean-ranks? J Mach Learn Res 17(5):1–10

Bostrom A, Bagnall A (2015) Binary shapelet transform for multiclass time series classification. In: Madria S, Hara T (eds) Big data analytics and knowledge discovery. Springer International Publishing, Cham, pp 257–269

Briandet R, Kemsley EK, Wilson RH (1996) Discrimination of arabica and robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics. J Agric Food Chem 44(1):170–174. https://doi.org/10.1021/jf950305a

Calvo B, Santaf G (2016) scmamp: statistical comparison of multiple algorithms in multiple problems. R J 8(1):248–256. https://doi.org/10.32614/RJ-2016-017

Castro N, Azevedo P (2010) Multiresolution Motif Discovery in Time Series, pp 665–676. https://doi.org/10.1137/1.9781611972801.73

Chen JS, Moon YS, Yeung HW (2005) Palmprint authentication using time series. In: Kanade T, Jain A, Ratha NK (eds) Audio- and video-based biometric person authentication. Springer, Berlin, pp 376–385

Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The ucr time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/

Costa da Silva J, Klusch M (2007) Privacy-preserving discovery of frequent patterns in time series. In: Perner P (ed) Advances in data mining. Theoretical aspects and applications. Springer, Berlin, pp 318–328

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Garcia S, Herrera F (2008) An extension on ”statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. J Mach Learn Res 9:2677–2694

Glatthorn JF, Gouge S, Nussbaumer S, Stauffacher S, Impellizzeri FM, Maffiuletti NA (2011) Validity and reliability of optojump photoelectric cells for estimating vertical jump height. J Strength Cond Res 25(2):556–560

Gordon D, Hendler D, Rokach L (2012) Fast randomized model generation for shapelet-based time series classification. CoRR arXiv:1209.5038

Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2014) Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, KDD ’14, pp 392–401, https://doi.org/10.1145/2623330.2623613

Ifrim G, Wiuf C (2011) Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, KDD ’11, pp 708–716, https://doi.org/10.1145/2020408.2020519

Ismail Fawaz H, Forestier G, Weber J, Idoumghar L, Muller PA (2019) Deep learning for time series classification: a review. Data Mining Knowl Discov. https://doi.org/10.1007/s10618-019-00619-1

Kasten EP, McKinley PK, Gage SH (2007) Automated ensemble extraction and analysis of acoustic data streams. In: 27th International conference on distributed computing systems workshops (ICDCSW’07), pp 66–66, https://doi.org/10.1109/ICDCSW.2007.25

Kate RJ (2016) Using dynamic time warping distances as features for improved time series classification. Data Mining Knowl Discov 30(2):283–312. https://doi.org/10.1007/s10618-015-0418-x

Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2001) Dimensionality reduction for fast similarity search in large time series databases. Knowl Inf Syst 3(3):263–286. https://doi.org/10.1007/PL00011669

Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery, ACM, New York, NY, USA, DMKD ’03, pp 2–11, https://doi.org/10.1145/882082.882086

Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Mining Knowl Discov 15(2):107–144. https://doi.org/10.1007/s10618-007-0064-z

Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. J Intell Inf Syst 39(2):287–315. https://doi.org/10.1007/s10844-012-0196-5

Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Discov 29(3):565–592. https://doi.org/10.1007/s10618-014-0361-2

Lines J, Davis LM, Hills J, Bagnall A (2012) A shapelet transform for time series classification. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, KDD ’12, pp 289–297, https://doi.org/10.1145/2339530.2339579

Lines J, Taylor S, Bagnall A (2016) Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In: 2016 IEEE 16th international conference on data mining (ICDM), pp 1041–1046, https://doi.org/10.1109/ICDM.2016.0133

Markovic G, Dizdar D, Jukic I, Cardinale M (2004) Reliability and factorial validity of squat and countermovement jump tests. J Strength Cond Res 18(3):551–555

Nguyen TL, Gsponer S, Ifrim G (2017) Time series classification by sequence learning in all-subsequence space. In: 2017 IEEE 33rd international conference on data engineering (ICDE), pp 947–958, https://doi.org/10.1109/ICDE.2017.142

Nuzzo JL, McBride JM, Cormie P, McCaulley GO (2008) Relationship between countermovement jump performance and multijoint isometric and dynamic tests of strength. J Strength Cond Res 22(3):699–707. https://doi.org/10.1519/jsc.0b013e31816d5eda

O’Reilly M, Caulfield B, Ward T, Johnston W, Doherty C (2018) Wearable inertial sensor systems for lower limb exercise detection and evaluation: a systematic review. Sports Medicine pp 1–26

O’Reilly MA, Whelan DF, Ward TE, Delahunt E, Caulfield BM (2017) Classification of deadlift biomechanics with wearable inertial measurement units. J Biomech 58:155–161. https://doi.org/10.1016/j.jbiomech.2017.04.028

Picerno P, Camomilla V, Capranica L (2011) Countermovement jump performance assessment using a wearable 3d inertial measurement unit. J Sports Sci 29(2):139–146, https://doi.org/10.1080/02640414.2010.523089, pMID: 21120742

Rakthanmanon T, Keogh E (2013) Fast shapelets: A scalable algorithm for discovering time series shapelets. In: Proceedings of the thirteenth SIAM conference on data mining (SDM), SIAM, pp 668–676

Schäfer P (2015) The boss is concerned with time series classification in the presence of noise. Data Min Knowl Discov 29(6):1505–1530

Schäfer P (2016) Scalable time series classification. Data Min Knowl Discov 30(5):1273–1298. https://doi.org/10.1007/s10618-015-0441-y

Schäfer P, Högqvist M (2012) 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, ACM, New York, NY, USA, EDBT ’12, pp 516–527, https://doi.org/10.1145/2247596.2247656

Schäfer P, Leser U (2017) Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on conference on information and knowledge management, ACM, New York, NY, USA, CIKM ’17, pp 637–646, https://doi.org/10.1145/3132847.3132980

Schäfer P, Leser U (2017) Multivariate time series classification with WEASEL+MUSE. CoRR arXiv:1711.11343

Senin P, Malinchik S (2013) Sax-vsm: Interpretable time series classification using sax and vector space model. In: 2013 IEEE 13th international conference on data mining (ICDM), pp 1175–1180, https://doi.org/10.1109/ICDM.2013.52

Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Discov 26(2):275–309

Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: a strong baseline. In: 2017 international joint conference on neural networks (IJCNN), pp 1578–1585, https://doi.org/10.1109/IJCNN.2017.7966039

Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 947–956

Ye L, Keogh E (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min Knowl Discov 22(1):149–182. https://doi.org/10.1007/s10618-010-0179-5

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Thach Le Nguyen[1] · Severin Gsponer[1] · Iulia Ilie[1] · Martin O’Reilly[1] · Georgiana Ifrim[1]

✉ Thach Le Nguyen
thach.lenguyen@insight-centre.org

✉ Georgiana Ifrim
georgiana.ifrim@insight-centre.org

Severin Gsponer
severin.gsponer@insight-centre.org

Iulia Ilie
iulia.ilie@insight-centre.org

Martin O’Reilly
martin.oreilly@insight-centre.org

[1] Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland