Data Min Knowl Disc (2015) 29:1505–1530
DOI 10.1007/s10618-014-0377-7

The BOSS is concerned with time series classification in the presence of noise

Patrick Schäfer

Received: 28 March 2014 / Accepted: 9 August 2014 / Published online: 16 September 2014

Abstract Similarity search is one of the most important and probably best studied methods for data mining. In the context of time series analysis it reaches its limits when it comes to mining raw datasets. The raw time series data may be recorded at variable lengths, be noisy, or are composed of repetitive substructures. These build a foundation for state of the art search algorithms. However, noise has been paid surprisingly little attention to and is assumed to be filtered as part of a preprocessing step carried out by a human. Our Bag-of-SFA-Symbols (BOSS) model combines the extraction of substructures with the tolerance to extraneous and erroneous data using a noise reducing representation of the time series. We show that our BOSS ensemble classifier improves the best published classification accuracies in diverse application areas and on the official UCR classification benchmark datasets by a large margin.

Keywords Time series · Classification · Similarity · Noise · Fourier transform

1 Introduction

Time series are recorded from sensors and other input sources over time. Application domains include personalised medicine,[1] human walking motions (Ye and Keogh 2011), anthropology (Ye and Keogh 2009), security (Mueen et al. 2011), historical documents (Ye and Keogh 2009), astronomy (Rakthanmanon and Keogh 2013),

[1] The BIDMC congestive heart failure database: http://www.physionet.org/physiobank/database/chfdb/. Accessed 2014.

Responsible editors: Toon Calders, Floriana Esposito, Eyke Hüllermeier, Rosa Meo.

P. Schäfer (✉)
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: patrick.schaefer@zib.de

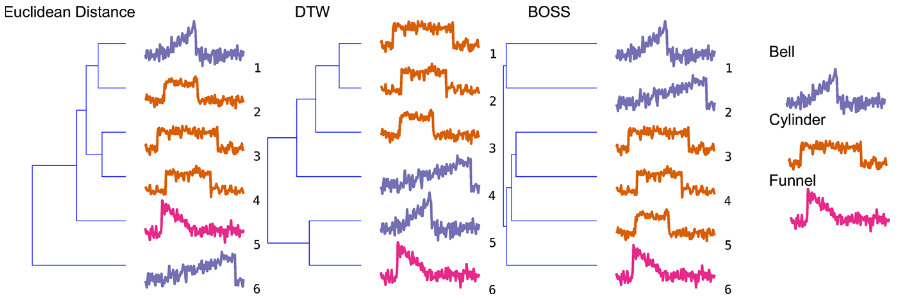


Fig. 1 Hierarchical clustering of the Cylinder–Bell–Funnel dataset based on three similarity metrics. There are three types of curves: cylinder, bell, funnel

spectrographs (Ye and Keogh 2009), for example. While a human has an intuitive understanding of similarity, this task becomes very complex for a computer. It is non trivial to extract a statistical model from time series as these may be non-stationary, and show varying statistical properties with time. Data mining algorithms on the other hand, degenerate due to the high dimensionality of the time series and noise (Keogh and Kasetty 2002). Existing techniques can be categorised as *shape-based* and *structure-based* (Ding 2008). Shape-based techniques use a *similarity measure* in combination with 1-nearest-neighbor search. These are competitive on pre-processed datasets (Ding 2008) but fail on long or noisy data. *Structure-based* techniques transform a time series into a different representation or extract feature vectors from the time series like characteristic patterns (Mueen et al. 2011; Hu et al. 2013; Zakaria et al. 2012; Lin et al. 2012). This comes at a high computational cost. Typical similarity metrics are the Euclidean distance (ED) or dynamic time warping (DTW) (Sakoe and Chiba 1978; Rakthanmanon et al. 2012a, b). While DTW is four decades old, it is highly competitive and used as the reference Ding (2008). DTW provides warping invariance which is a peak-to-peak and valley-to-valley alignment of two time series. This fails if there is a variable number of peaks and valleys.

Figure 1 shows a hierarchical clustering of the first six samples from the synthetic Cylinder–Bell–Funnel (CBF) dataset. This synthetic time series benchmark dataset is widely used and contains three basic shapes: cylinders, bells and funnels. For the human eye the distinguishing power of the first two distance measures is very disappointing. The ED fails to cluster the funnel curves 1 and 6 as it does not provide horizontal alignment (phase invariance). DTW provides warping invariance, but still does not give a satisfying clustering as the funnel curves 4 and 5 are separated. Our BOSS model clusters the funnel curves 1–2 and cylinder curves 3–5 correctly. This toy example illustrates the difficulties for time series similarity. In general, several sources of invariance like *amplitude/offset, warping, phase, uniform scaling, occlusion, and complexity* have been presented in Batista et al. (2011). The CBF dataset requires invariance to phase (horizontal alignment), warping (local scaling), occlusion (noise) and amplitude/offset.

We believe that *invariance to noise* was paid too little attention to, as most algorithms operate directly on the raw data. To illustrate the relevance of noise to the classification task, we performed another experiment on the CBF data. All time series were first z-normalised to have a standard deviation (SD) of 1. We then added Gaussian noise

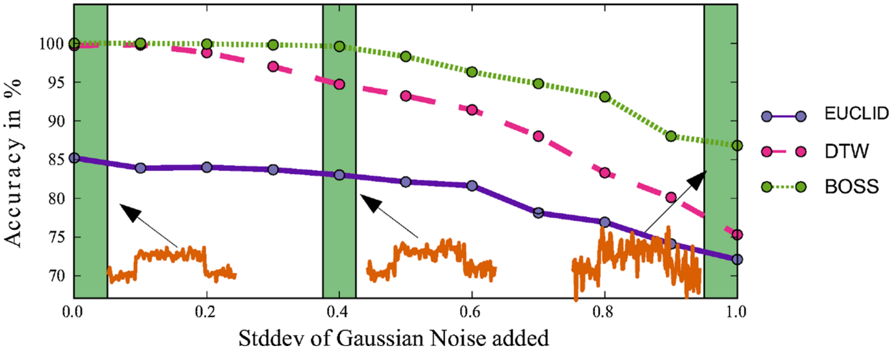


Fig. 2 Effects of Gaussian noise on Cylinder–Bell–Funnel dataset

with an increasing SD of 0 to 1.0 to each time series, equal to a noise level of 0–40 %. Figure 2 shows that DTW and BOSS provide the best classification accuracies. With an increase of noise the classification accuracies decrease. The BOSS classifier is very robust to noise and remains stable up to a noise level of 40 %, whereas DTW degenerates starting from a noise level of 10 %.

Our **B**ag-**of**-**S**FA-**S**ymbols (BOSS) model is a structure-based similarity measure that applies noise reduction to the raw time series. It first extracts substructures (patterns) from a time series. Next, it applies low pass filtering and quantisation to the substructures, which reduces noise and allows for string matching algorithms to be applied. Two time series are then compared based on the differences in the set of noise reduced patterns. As opposed to rivalling methods the BOSS offers multiple advantages: (a) it is fast, (b) it applies noise reduction, (c) invariance to offsets is treated as a parameter, and (d) it is a structure based similarity measure. As a result the BOSS is as fast as DTW but much more accurate than DTW and state of the art classifiers. Our contributions are as follows:

- We present our BOSS model that combines the noise tolerance of the Symbolic Fourier Approximation (SFA) (Schäfer and Högqvist 2012) with the structure-based representation of the bag-of-words model (Lin et al. 2012) (Sect. 3).
- We present several optimisation strategies like reducing the computational complexity of SFA from $O(w \log w)$ to $O(1)$, for windows of size w (Sect. 4).
- We present the *BOSS ensemble classifier* based on multiple BOSS models at different window lengths (Sect. 5).
- We show (Sect. 7) that the BOSS ensemble classifier (a) achieves a up to 10 percentage points higher accuracy than any rivalling classification model on real datasets in diverse application areas, (b) is as fast as DTW and up to 13-times as fast as rivalling structure based classifiers and (c) shows the best test accuracies on the UCR time series benchmark datasets.

2 Background

Before going into the details of our BOSS model, we present the building blocks in Fig. 3 based on a sample time series. First, sliding windows of fixed length are

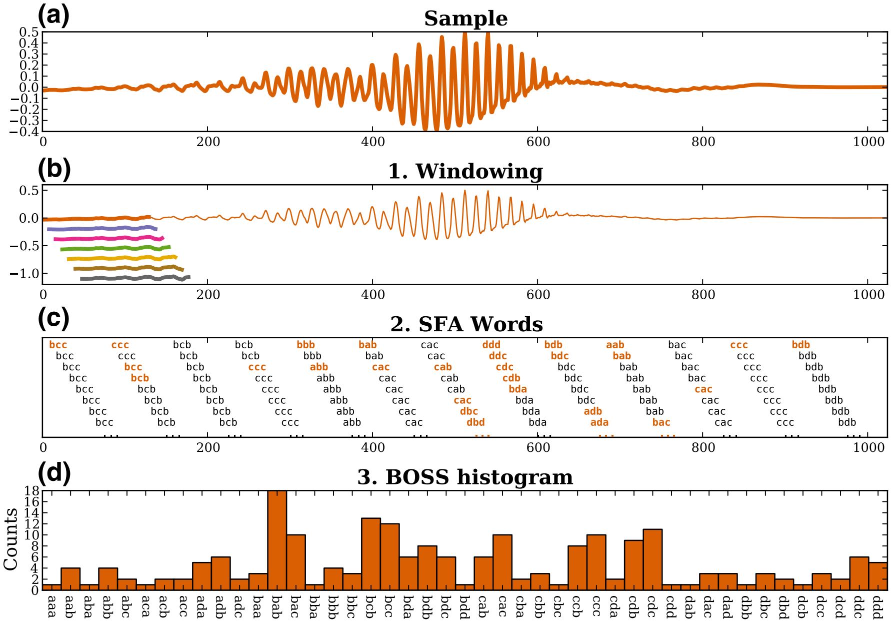


Fig. 3 The BOSS model is extracted from a sample time series using word length 3 and 4 symbols (**a**–**d**). The *black* SFA words are skipped due to numerosity reduction

extracted from a time series. Next, a symbolic representation called SFA (Schäfer and Högqvist 2012) is applied to each sliding window. SFA provides low pass filtering and quantisation to reduce noise. This results in a sequence of symbols (*SFA word*) for each sliding window. The histogram of SFA words (Fig. 3 bottom right) is then used as the indicator for structural similarity.

2.1 Definitions

A *time series* is a sequence of $n \epsilon \mathbb{N}$ real values, which are recorded over time:

$$T = (t_1, \ldots, t_n) \tag{1}$$

This time series is split into a set of subsequences, named *windows* hereafter, using a *windowing* function.

Definition 1 Windowing: A time series $T = (t_1, \ldots, t_n)$ of length n is split into fixed-size windows $S_{i;w} = (t_i, \ldots, t_{i+w-1})$ of length w using a windowing function. Two consecutive windows at offset i and $i+1$ *overlap* in $w-1$ positions:

$$windows(T, w) = \left\{ \underbrace{S_{1;w}}_{(t_1,\ldots,t_w)}, \underbrace{S_{2;w},}_{(t_2,\ldots,t_{w+1})} \ldots, S_{n-w+1;w} \right\} \tag{2}$$

To obtain a consistent scale and vertical alignment (offset and amplitude invariance), each window is typically z-normalised by subtracting the mean and dividing it by the standard deviation.

2.2 From real values to words

The SFA (Schäfer and Högqvist 2012) is a symbolic representation of time series. That is, a real valued time series is represented by a sequence of symbols, named *SFA word*, using a finite alphabet of symbols. This transformation allows for string matching algorithms like hashing and the bag of words representation to be applied. Figure 3 (bottom left) illustrates SFA words over the sliding windows of a time series using 3 characters and 4 symbols (a–d). The SFA transformation aims at:

- **Low pass filtering:** Rapidly changing sections of a signal are often associated with noise. These can be removed by a low pass filter. The SFA word length determines the number of Fourier coefficients to be used and thereby the bandwidth of the low pass filter.
- **String representation:** SFA uses quantisation and uses character strings. Thereby it allows for string matching algorithms to be applied. The size of the alphabet determines the degree of quantisation, which has an additional noise reducing effect, but it might lead to a loss of information.

2.3 Symbolic Fourier Approximation (SFA)

SFA is composed of two operations (Fig. 4):

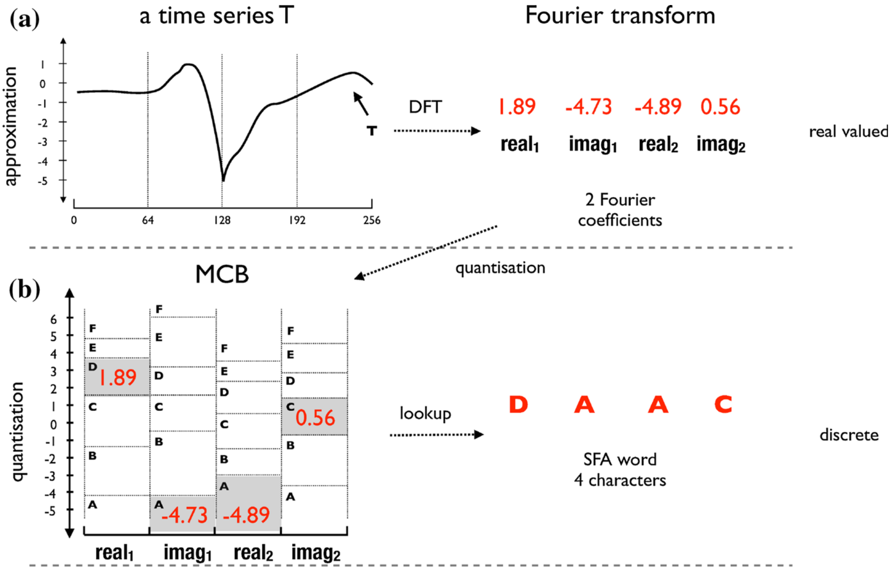


Fig. 4 SFA: A time series is **a** approximated (low pass filtered) using DFT and **b** quantised using MCB resulting in the SFA word *DAAC*

1. **Approximation** using the discrete Fourier transform (DFT) and
2. **Quantisation** using a technique called Multiple Coefficient Binning (MCB).

Approximation aims at representing a signal of length n by a transformed signal of reduced length l. Higher order Fourier coefficients represent rapid changes like dropouts or noise in a signal. The signal is low pass filtered by using the first $l \ll n$ Fourier coefficients.

Quantisation adds to noise reduction by dividing the frequency domain into frequency ranges and mapping each real valued Fourier coefficient to its range. MCB quantisation is data adaptive and thereby minimises the loss of information introduced by quantisation.

2.3.1 Approximation (DFT)

The DFT decomposes a signal T of length n into a sum of orthogonal basis functions using sinusoid waves. Each wave is represented by a complex number $X_u = (real_u, imag_u)$, for $u = 0, 1 \ldots, n-1$, called a Fourier coefficient:

$$DFT(T) = X_0 \ldots X_{n-1} = (real_0, imag_0, \ldots real_{n-1}, imag_{n-1}) \tag{3}$$

The n-point DFT of a discrete signal of one variable $T(x),\ x = 0, 1, \ldots, n-1$, is given by the equation:

$$X_u = \frac{1}{n} \sum_{x=0}^{n-1} T(x) \cdot e^{-j2\pi ux/n}, \text{ for } u \epsilon [0, n)\,,\ j = \sqrt{-1} \tag{4}$$

The first Fourier coefficients correlate to lower frequency ranges or the slowly changing sections of a signal. The higher order coefficients correlate to higher frequency ranges or rapidly changing sections of a signal. The first Fourier coefficients are commonly used to describe a signal, thereby low pass filtering and smoothing the signal. The first Fourier coefficient is equal to the mean value of the signal and can be discarded to obtain offset invariance (vertical shifts).

2.3.2 Quantisation (MCB)

The MCB quantisation intervals are computed from the samples. A matrix $A = \left(a_{ij}\right)_{i=1,\ldots,N;j=1,\ldots,l}$ is built from the Fourier transformations of the N training samples using only the first $\frac{l}{2}$ Fourier-coefficients—equal to an SFA word of length l with $\frac{l}{2}$ real and $\frac{l}{2}$ imaginary values. The i-th row of matrix A corresponds to the

Fourier transform of the i-th sample T_i:

$$A = \begin{pmatrix} DFT(T_1) \\ . \\ DFT(T_i) \\ . \\ DFT(T_N) \end{pmatrix} = \begin{pmatrix} real_{11} & imag_{11} & \ldots & real_{1\frac{l}{2}} & imag_{1\frac{l}{2}} \\ . & . & \ldots & . & . \\ real_{i1} & imag_{i1} & \ldots & real_{i\frac{l}{2}} & imag_{i\frac{l}{2}} \\ . & . & \ldots & . & . \\ real_{N1} & imag_{N1} & \ldots & real_{N\frac{l}{2}} & imag_{N\frac{l}{2}} \end{pmatrix} = \left(C_1 \ldots C_j \ldots C_l \right)$$

The j-th column C_j corresponds to either the real or imaginary values of all N training signals. Each column is sorted by value and then partitioned into c equi-depth bins.

Given the sorted columns C_j, with $j = 1, \ldots, l$, and a finite alphabet Σ of size c: MCB determines $c + 1$ breakpoints $\beta_j(0) < \ldots < \beta_j(c)$ for each column C_j, by applying equi-depth binning. Using an alphabet of size c and $\frac{l}{2}$ Fourier coefficients, MCB results in a total of l sets of $c + 1$ intervals. Figure 4 (bottom left) illustrates the intervals for $c = 6$ and $l = 4$.

Finally, we label each bin by assigning the a-th symbol of the alphabet Σ to it. For all pairs (j, a) with $j = 1, \ldots, l$ and $a = 1, \ldots, c$:

$$\left[\beta_j(a-1), \beta_j(a)\right) \triangleq symbol_a \tag{5}$$

When it comes to time series classification the precomputed and labelled MCB intervals are obtained from a train dataset. Based on the MCB intervals we compute the SFA words for both the train and test data.

2.3.3 *SFA transformation*

The SFA word is obtained from a Fourier transformed time series by a simple lookup using the precomputed MCB intervals (Fig. 4 bottom).

Definition 2 SFA Word: the symbolic representation $SFA(T) = s_1, \ldots, s_l$ of a time series T with approximation $DFT(T) = t_1^{'}, \ldots, t_l^{'}$ is a mapping $SFA : \mathbb{R}^l \rightarrow \Sigma^l$ of the real and imaginary value to a symbol over the alphabet Σ of size c. Specifically, the j-th value $t_j^{'}$ is mapped to the a-th symbol, if it falls into its interval:

$$\left(\beta_j(a-1) \leq t_j^{'} < \beta_j(a)\right) \Rightarrow s_j \equiv symbol_a \epsilon \Sigma \tag{6}$$

Figure 4 bottom right illustrates this mapping. The resulting SFA word is $DAAC$ for $DFT(T) = (1.89, -4.73, -4.89, 0.56)$.

2.4 Related work

Existing classification algorithms either (a) try to find a similarity metric that resembles our intuition of similarity (shape-based) or (b) extract feature vectors or model parameters from the data to make existing data mining algorithms applicable (structure-based)

(Ding 2008; Liao Warren 2005). The UCR time series classification datasets[2] have been established as the benchmark (Mueen et al. 2011; Lin et al. 2012; Ding 2008; Bagnall et al. 2012 and also see Footnote 2). We focus on these approaches in our analysis.

Shape-based techniques are based on a similarity metric in combination with 1-nearest-neighbour (1-NN) classification. Examples include 1-NN ED, 1-NN Longest Common Subsequence (Vlachos et al. 2002), or 1-NN DTW (Sakoe and Chiba 1978; Rakthanmanon et al. 2012a, b). DTW has shown to be a highly competitive classifier on time series datasets and is used as a reference (Ding 2008). The problem with shape-based techniques is that they fail to classify noisy or long data containing characteristic substructures.

Structure-based techniques extract higher-level feature vectors or build a model from the time series prior to the classification task using classical data mining algorithms like support vector machines (SVMs), decision trees, or random forests. Techniques for extracting feature vectors include the DFT (Agrawal et al. 1993), indexable piecewise linear approximation (Chen et al. 2007), SFA (Schäfer and Högqvist 2012), or Symbolic Aggregate approXimation (SAX) (Lin et al. 2007), to name but a few examples. These transformations use the whole time series. In contrast Shapelets classifiers (Mueen et al. 2011; Ye and Keogh 2011; Rakthanmanon and Keogh 2013) extract representative variable-length subsequences (called *shapelets*) from a time series for classification. A decision tree is build using these shapelets within the nodes of the tree. A distance threshold is used for branching.

The *bag-of-patterns* (BOP) model (Lin et al. 2012) is the closest to our work. BOP extracts substructures as higher-level features of a time series. BOP transforms these substructures using a quantisation method called SAX and uses the ED as a similarity metric. SAX-VSM (Senin and Malinchik 2013) builds on BOP by the use of ts-idf weighting of the bags and Cosine similarity as similarity metric. It uses one bag of words for each class, instead of one bag for each sample. In contrast BOSS uses SFA (Schäfer and Högqvist 2012), the offset invariance as a model parameter, a different similarity metric, an ensemble of BOSS models, and we present multiple optimisation techniques. Time-series bitmaps (Kumar et al. 2005) are a visualisation tool for time series datasets based on a histogram of SAX words. The approach is similar to the BOP model.

SFA has been introduced in Schäfer and Högqvist (2012) in the context of similarity search on massive time series datasets using the SFA trie. This work focuses on time series classification and clustering (rather than indexing) and extends our previous work on SFA by introducing the momentary Fourier transform (MFT) (Albrecht et al. 1997) to SFA and the BOSS model based on SFA words over sliding windows of a time series.

SFA uses the DFT and SAX uses mean values (PAA) to approximate a time series. Both, have a noise cancelling effect by smoothing a time series. One disadvantage of using mean values is that these have to be recalculated when changing the resolution—i.e. from weekly to monthly mean values. The resolution of DFT can be incrementally

[2] UCR Time Series Classification/Clustering Homepage: http://www.cs.ucr.edu/~eamonn/time_series_data. Accessed 2014.

adapted by choosing an arbitrary subset of Fourier coefficients without recalculating the DFT of a time series. Maximising the train accuracy while increasing the number of Fourier coefficients is the core idea of our algorithm in Algorithm 3. Dropping the rear mean values of a SAX word is equal to dropping the rear part of a time series. To avoid this, we would have to recalculate all SAX transformations each time we chose to represent a time series by a different SAX word length.

3 The Bag-of-SFA-Symbols (BOSS)

The BOSS model (Fig. 5) describes each time series as an unordered set of substructures using SFA words. The approach has multiple advantages:

- it is fast, as hashing is used to determine the similarity of substructures (SFA words),
- it applies noise reduction,
- it provides invariance to phase shifts, offsets, amplitudes and occlusions.

3.1 The BOSS model

Our model has four parameters:

- **the window length** $w \epsilon \mathbb{N}$: represents the size of the substructures.
- **mean normalisation** $mean \epsilon [true, false]$: set to true for offset invariance.
- the **SFA word length** $l \epsilon \mathbb{N}$ **and alphabet size** $c \epsilon \mathbb{N}$: used for low pass filtering and the string representation.

First, sliding windows of length w are extracted from a time series. Intuitively w should roughly represent the size of the substructures within the time series. Next, each sliding window is normalised to have a standard deviation of 1 to obtain amplitude invariance. The parameter $mean$ determines if the mean value is to be subtracted from each sliding window to obtain offset invariance. The mean normalisation is treated as a parameter of our model and can be enabled or disabled. For example, heart beats should be compared using a common baseline but the pitch of a bird sound can be significant for the species. Finally, the SFA transformation is applied to each real valued sliding window. The BOSS model transforms a time series into an unordered set of SFA words. Using an unordered set provides invariance to the horizontal alignment of each substructure within the time series (phase shift invariance). In stable sections of

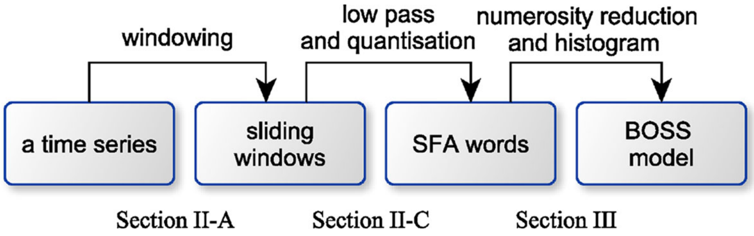


Fig. 5 The BOSS workflow

a signal the SFA words of two neighbouring sliding windows are very likely to be identical. To avoid outweighing stable sections of a signal, *numerosity reduction* (Lin et al. 2012, 2007) is applied. That is, the first occurrence of an SFA word is registered and all duplicates are ignored until a new SFA word is discovered. In Fig. 3 the first SFA words are identical:

$$S = \mathbf{bcc}\ bcc\ bcc\ bcc\ bcc\ bcc\ bcc\ bcc\ \mathbf{ccc}\ ccc\ \mathbf{bcc}\ \mathbf{bcb}\ bcb\ bcb\ bcb\ \ \ldots$$

Applying numerosity reduction to S this leads to:

$$S' = bcc\ ccc\ bcc\ bcb\ \ \ldots$$

From these SFA words a histogram is constructed, which counts the occurrences of the SFA words. In the above example the BOSS histogram of S' is:

$$B : bcc = 2,\ ccc = 1,\ bcb = 1, \ldots$$

This BOSS histogram ignores the ordering of the occurrences of the SFA words within a time series. This provides phase invariance of the substructures and thereby eliminates the need for preprocessing the samples by a domain expert for approximate alignment of the substructures.

Definition 3 BOSS: Given are a time series T, its sliding windows $S_{i;w}$ and SFA transformations $SFA(S_{i;w}) \epsilon \Sigma^l$, for $i = 1, 2, \ldots, (n - w + 1)$. The BOSS histogram $B : \Sigma^l \to \mathbb{N}$ is a function of the SFA word space Σ^l to the natural numbers. The number represents the occurrences of an SFA word within T counted after numerosity reduction.

Algorithm 1 The BOSS transformation.

```
map<string,int> BOSSTransform(sample,w,l,c,mean)
(1) map<string,int> boss
(2) for s in sliding_windows(sample,w)
(3)   word = SFA(s,l,c,mean)
(4)   if word != lastWord     // numerosity reduction
(5)     boss[word]++          // increase histogram counts
(6)   lastWord = word
(7) return boss
```

BOSS transformation (Algorithm 1): The algorithm extracts sliding windows of length w from the sample (line 2) and determines SFA words (line 3) with length l and alphabet size c. Mean normalisation is obtained by dropping the first Fourier coefficient in each SFA word. Finally, a new word is added to the histogram (line 5), if two subsequent words are different (numerosity reduction).

3.2 BOSS distance

Two time series are considered similar, if they share the same set of SFA words. Figure 6 illustrates the BOSS histograms for abnormal and normal walking motions.

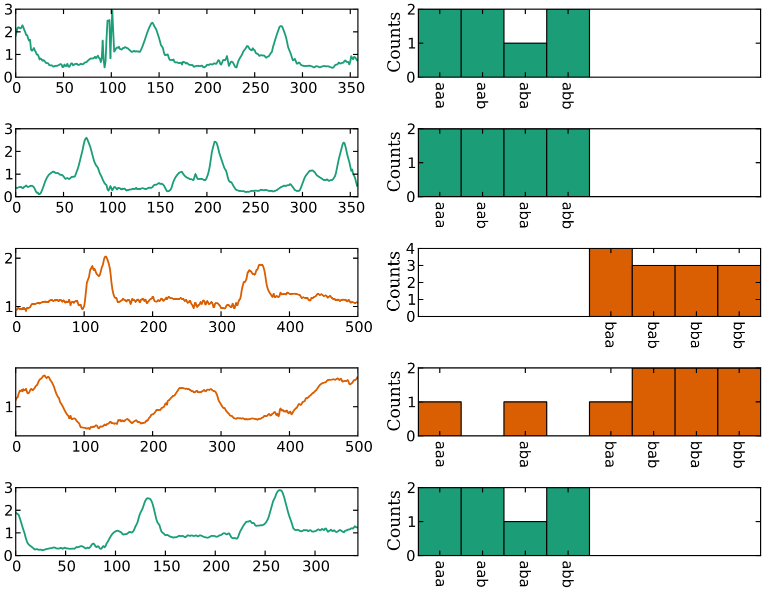


Fig. 6 The BOSS histograms of normal (*green*) and abnormal (*orange*) walking motions

There is noise, erroneous data (a peek in the first motion) and the motions are not aligned. Still the BOSS histograms for the normal walking motions 1, 2, 5 are very similar, while the histograms of the abnormal motions 3, 4 clearly differ.

When comparing two time series, the absence of SFA words has two reasons: noise distorts the substructures or a substructure is not contained in another signal. Consider two identical signals, whereas the second signal contains extraneous data at the beginning, i.e. as the sensor was not connected. These signals will have identical BOSS histograms except for the SFA words at the beginning of the second recording. We have to ignore these SFA words for the signals to become identical. Thus, we chose to ignoring missing substructures in our distance measure. The BOSS distance is a modification of the ED: we omit all SFA word counts of 0 in the query when computing the pairwise SFA word count differences. For example, the BOSS model of the first and fourth motion in Fig. 6 is:

$$\begin{array}{rcccccccc} & aaa & aab & aba & abb & baa & bab & bba & bbb \\ B_1 = & 2 & 2 & 1 & 2 & 0 & 0 & 0 & 0 \\ B_4 = & 1 & 0 & 1 & 0 & 1 & 2 & 2 & 2 \\ D(B_1, B_4) = & (2-1)^2 & +(2)^2 & +(1-1)^2 & +(2)^2 & +\mathbf{0} & +\mathbf{0} & +\mathbf{0} & +\mathbf{0} \\ D(B_4, B_1) = & (2-1)^2 & +\mathbf{0} & +(1-1)^2 & +\mathbf{0} & +(1)^2 & +(2)^2 & +(2)^2 & +(2)^2 \end{array}$$

The resulting pairwise BOSS distances are: $D(B_1, B_4) = 9$ and $D(B_4, B_1) = 14$.

Definition 4 BOSS distance: Given two BOSS histograms $B_1 : \Sigma^l \rightarrow \mathbb{N}$ and $B_2 : \Sigma^l \rightarrow \mathbb{N}$ of two time series T_1 and T_2, the BOSS distance is defined as:

$$D(T_1, T_2) = dist(B_1, B_2) \tag{7}$$

with

$$dist(B_1, B_2) = \sum_{a \epsilon B_1;\, B_1(a)>0} [B_1(a) - B_2(a)]^2 \tag{8}$$

The BOSS distance is not a distance metric as it neither satisfies the symmetry condition nor the triangle inequality. As a consequence the BOSS distance does not allow for indexing (triangle inequality) and the nearest neighbour of X may not be the nearest neighbour of Y (symmetry). In the context of time series classification the BOSS distance gave the best classification accuracy. However, other distance metrics such as the ED or Cosine similarity may be applied, if the two conditions have to be met.

4 Optimisation of the BOSS model

4.1 Incremental Fourier transform

The SFA transformation is dominated by the runtime of a single DFT. As part of Algorithm 1, $n - w + 1$ sliding windows of length w are extracted from a sample of length n. A single DFT of a window of length w has a complexity of $O(w \log w)$, which is time consuming considering we need only $l \ll w$ Fourier coefficients. Let us assume that we are interested in the first $l \ll w$ Fourier coefficients of the sliding windows $\left\{S_{1;w}, \ldots, S_{n-w+1;w}\right\}$. A sliding window at time interval i is inferred from its predecessor by one summation and one subtraction:

$$S_{i;w} = S_{i-1;w} + x_i - x_{i-w} \ , \ \text{for } i > 1 \tag{9}$$

The MFT (Albrecht et al. 1997) makes use of this recursive property as the first l Fourier coefficients at the time interval $i : X_{i;0} \ldots X_{i;l-1}$ can be computed from the previous time interval $i - 1 : X_{i-1;0} \ldots X_{i-1;l-1}$ using:

$$\begin{pmatrix} X_{i;0} \\ X_{i;1} \\ . \\ . \\ X_{i;l-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & . & 0 \\ 0 & \upsilon^{-1} & . & . \\ . & 0 & . & . \\ . & . & . & 0 \\ 0 & . & . & \upsilon^{-(l-1)} \end{pmatrix} \cdot \begin{pmatrix} X_{i-1;0} + x_i - x_{i-w} \\ X_{i-1;1} + x_i - x_{i-w} \\ . \\ . \\ X_{i-1;l-1} + x_i - x_{i-w} \end{pmatrix} \tag{10}$$

with the definition of $\upsilon^k = e^{-j2\pi k/n}$ and imaginary number $j = \sqrt{-1}$. In this representation each Fourier coefficient at time interval i can be independently computed from time $i - 1$ using only $O(1)$ complex multiplications and summations: $X_{i;f} = \upsilon^{-f}(X_{i-1;f} + (x_i - x_{i-w}))$.

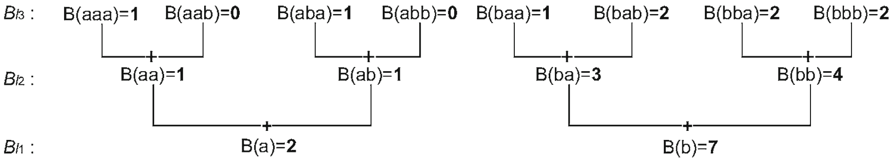
By use of the MFT the computational complexity to compute l SFA features is reduced to $O(l)$ for all but the first window, which has a complexity of $O(w \log w)$. Thus, the computational complexity for all windows is reduced from $O(n \cdot w \log w)$ to:

$$O(nl + w \log w) = O(nl) = O(n), \ \text{ with } l \ll w \ll n \tag{11}$$

4.2 Incremental refinement of SFA word lengths

Using smaller SFA word lengths has the effect of reducing noise but you also loose signal details. A core idea of our BOSS model is to test different SFA word lengths to find the optimal trade-off between word length and noise reduction by maximising the classification accuracy on a train dataset. To avoid redundant computations we can vary the length of an SFA word by first calculating long SFA words and then dropping the rearward characters which are equal to higher-order Fourier coefficients. The ability to vary the SFA word length on the fly is one of the main differences to the symbolic representation SAX used in the bag-of-pattern representation, as stated in the background.

Our rationale is to (a) calculate the SFA transformation for the largest required SFA word length l and (b) incrementally remove the last character(s) from each word. However, we still have to adapt the BOSS histograms when changing the SFA word length. Fortunately, this can be achieved by a simple addition of counts from the histograms of the larger SFA word lengths. We use the 4th sample from Fig. 6 to illustrate this operation using an alphabet Σ of size 2:



The histograms are incrementally updated by dropping the last character(s) and recounting the occurrences: i.e., *bba* and *bbb* are merged to *bb* and the counts of both add up to 4. In general the following holds for histograms $B_1 : \Sigma^{l_1} \rightarrow \mathbb{N}$ at word length l_1 and and $B_2 : \Sigma^{l_2} \rightarrow \mathbb{N}$ at word length $l_2 = l_1 + 1$ with SFA alphabet Σ:

$$B_1(\alpha) = \sum_{\beta \epsilon \Sigma} B_2(\alpha\beta), \quad \text{with} \quad \alpha \epsilon \Sigma^{l_1} \quad \text{and} \quad \alpha\beta \epsilon \Sigma^{l_2} \tag{12}$$

4.3 Lower bounding of SFA world lengths

An important observation is that the smaller SFA word length l_1 can be used to lower bound the distance computations on the larger word lengths $l_2 > l_1$ to avoid unnecessary computations. That means we can use the distance on length l_1 to decide, if we have to test l_2.

Proof Given alphabet size c, two BOSS histograms $B_{1;l_1}$ and $B_{2;l_1}$ at word length l_1 and $B_{1;l_2}$ and $B_{2;l_2}$ at word length $l_2 = l_1 + 1$, the following applies:

$$(l_2 = l_1 + 1) \Rightarrow \left(\frac{1}{c} \cdot dist(B_{1;l_1}, B_{2;l_1}) \leq dist(B_{1;l_2}, B_{2;l_2}) \right) \tag{13}$$

Proof idea: Given any SFA word $a \epsilon \Sigma^{l_1}$, and the SFA words $(ab) \epsilon \Sigma^{l_2}$ derived from concatenating a of length l_1 with a symbol $b \epsilon \Sigma$. The following applies:

$$\frac{1}{c}\left[B_{1;l_1}(a) - B_{1;l_1}(a)\right]^2 = \underbrace{\left[\frac{B_{1;l_1}(a)}{c} - \frac{B_{2;l_1}(a)}{c}\right]^2}_{(\overline{x}-\overline{y})^2} \leq \sum_{b \epsilon \Sigma} \underbrace{\left[B_{1;l_2}(ab) - B_{2;l_2}(ab)\right]^2}_{(x_i - y_i)^2} \tag{14}$$

as $B_{1;l_1}(a) = \sum_b B_{1;l_2}(ab)$ and $B_{2;l_1}(a) = \sum_b B_{2;l_2}(ab)$. Equation 14 mimics the formula that was proven in Keogh et al. (2001): $c(\overline{x} - \overline{y})^2 \leq \sum (x_i - y_i)^2$. Our proof ends by extending Eq. 14 to all SFA words in $B_{1;l_1}$:

$$\begin{aligned} \frac{1}{c} \cdot dist(B_{1;l_1}, B_{2;l_1}) &= c \sum_{a \epsilon B_{1;l_1}} \left[B_{1;l_1}(a) - B_{2;l_1}(a)\right]^2 \\ &\leq \sum_{a \epsilon \Sigma^{l_1}} \sum_{b \epsilon \Sigma} \left[B_{1;l_2}(ab) - B_{2;l_2}(ab)\right]^2 \\ &= \sum_{a \epsilon \Sigma^{l_2}} \left[B_{1;l_2}(a) - B_{2;l_2}(a)\right]^2 = dist(B_{1;l_2}, B_{2;l_2}) \end{aligned} \tag{15}$$

We have to skip normalising the histograms to allow for this lower bounding between different word lengths.

5 The BOSS classifier

The classification of time series has gained increasing interest over the past decade (Bagnall et al. 2012; Ding 2008 and also see Footnote 2). The time series classification task aims at assigning a class label to a time series. For this the features (the model) to distinguish between the class labels are trained based on a labelled train dataset. When an unlabelled query time series is recorded, the model is applied to determine the class label.

Algorithm 2 Predict: 1-nearest-neighbor classification using the BOSS model.

```
TimeSeries predict(qId,samples,histograms)
(bestDist, bestTs) = (MAX_VALUE, NULL)
for i in 1..len(samples)            // search for the 1-NN
  dist = 0
  // iterate only those words with a count > 0!
  for (word,count) in histograms[qId]
    dist += (count-histograms[i].get(word))^2
  if dist < bestDist                // store current 1-NN
    (bestDist, bestTs) = (dist, samples[i])
return bestTs
```

Algorithm 3 Fit: Train the parameters using leave-one-out cross validation.

```
[(int,int,int,histogram[])] fit(samples,labels,mean)
( 1)  scores = [], maxF=16, c=4, minSize = 10
( 2)  for w = maxSize down to minSize           // search all window lengths
( 3)    for i in 1..len(samples)                // obtain histograms
( 4)      hist[i]=BOSSTransform(samples[i],w,maxF,c,mean)
( 5)    bestCor=0, bestF=0
( 6)    for f in {8,10..maxF}                   // search all word lengths
( 7)      histShort = shortenHistogram(hist, f)  // incremental refinement
( 8)      correct=0
( 9)      for qId in 1..len(samples)            // leave-one-out
(10)        best = predict(qId,samples\{sample[qId]},histShort)
(11)        if labels(best)==labels(sample) correct++
(12)      if correct > bestCor                  // store best
(13)        (bestCor, bestF) = (correct, f)
        // store scores for each window length
(14)    scores.push((correct, w, bestF, hist))
(15)  return scores
```

Prediction (Algorithm 2): The BOSS classifier is based on 1-NN classification and the BOSS model. We chose to use 1-NN classification as it is very robust and doesn't introduce any parameters for model training. Given a query, the *predict*-method in Algorithm 2 searches for the 1-NN within a set of samples by minimising the distance between the query and all samples (*predict* lines 2ff). The lookup operation *histograms[i].get(word)* is a bottleneck as it is iterated for each sample (*predict* line 5). Thus, we implemented each BOSS histogram as a map to allow for constant time lookups.

Training (Algorithm 3): We use grid-search over the parameter space *window length* $w \epsilon [10, n]$, *SFA word length* $f \epsilon \{8, 10, 12, 14, 16\}$ and *alphabet size* $c = 4$ using leave-one-out cross-validation to train the BOSS classifier from a set of train samples. All window lengths (*fit* line 2ff) are iterated to obtain for each window length the optimal SFA word length. Based on the incremental refinement in Sect. 4.2 the first BOSS histograms are constructed using the longest word length $max\,F$ (*fit* lines 3–4). Shorter word lengths are then tested by dropping the last characters of each word and rebuilding the histogram (*fit* line 7). In case of an accuracy tie between two word lengths, the smaller word length is kept (*fit* lines 12–13). This follows the assumption that the patterns should be kept as simple as possible and therefor a stronger noise reduction is generally preferable. Finally, the accuracy counts for each pair of window length and SFA word length are returned (*fit* line 15). The *mean* normalisation parameter is a Boolean parameter, which is constant for a whole dataset and not set per sample. If set to *true*, the first Fourier coefficient (DC coefficient) is dropped to obtain offset invariance. We empirically observed that a constant alphabet size of 4 was optimal for most datasets. This observation is in accordance with SAX (Lin et al. 2007, 2012) which reported 4 symbols to work best for most time series datasets. Thus, we keep the alphabet size c fixed to four symbols.

5.1 BOSS Ensemble Classifier

By intuition every dataset is composed of substructures at multiple window lengths caused by different walking motions, heart beats, duration of vocals, or length of shapes. For example, each human may have a different length of a gait cycle. Thus, we represent each sample by a set of window lengths using an ensemble technique.

Algorithm 4 Predict: The BOSS Ensemble Classifier.

```
String predictEnsemble(qId,samples,windowScores)
    // stores for each window length a label
(1) windowLabels = []
    // determine best accuracy
(2) maxCorrect = max([correct | (correct,_,_,_) in windowScores])
    // determine the label for each window length
(3) for (correct, _, _, histograms) in windowScores
(4)   if (correct > maxCorrect * FACTOR)
(5)     windowLabels[len] = labels(predict(qId,samples,histograms))
(6) return most frequent label from windowLabels
```

The BOSS classifier in Algorithm 2 predicts the classification accuracy using one fixed window length. In the following we represent each time series by multiple window lengths to allow for different substructural sizes. The *fit*-method in Algorithm 3 returns a set of scores resulting from each window length on the train samples. The BOSS ensemble classifier (Algorithm 4) classifies a query using the best window sizes. It first obtains the best accuracy *maxCorrect* from the set of window scores (line 2) which result from the train data. All window lengths that have an accuracy that lies within a user defined constant threshold *factor* ϵ (0, 1] multiplied by this best score are used for prediction (lines 3–5):

$$score.correct > maxCorrect \cdot factor$$

A label is determined for each window length based on the 1-NN to the query (line 5). Finally, the most frequent class label is chosen from the set of labels (line 6). In our experiments a constant factor set to 0.92 or 0.95 was best throughout most datasets. Our web page contains the C++-code of the BOSS ensemble classifier.

6 Computational complexities

The BOSS model (Algorithm 1): The BOSS model has a runtime that is linear in n: there are $n - w + 1$ sliding windows in each time series of length n. Using the MFT, the SFA transformation for word length l have a complexity of (Eq. 11):

$$\begin{aligned} T(BOSS) &= O(w \log w + l \cdot (n - w)) \\ &= O(n) \qquad \text{with } l \ll w \ll n \end{aligned}$$

The BOSS distance (Algorithm 2 lines 4–5): The computational complexity of the BOSS distance is linear in the length of the time series n. Each BOSS histogram contains at most $n - w + 1$ SFA words. A histogram lookup for an SFA word has a constant time complexity by the use of hashing. This results in a total complexity that is linear in n:

$$T(BOSSDistance) = O(n - w + 1) = O(n)$$

While the computational time is bound by the time series length n, the actual number of *unique* SFA words is much smaller due to the numerosity reduction.

The BOSS Classifier Predict (Algorithm 2): The computational complexity of the *predict*-method performs a 1-NN search over the N samples using the BOSS distance calculations (line 2ff):

$$T(Predict) = O(N \cdot T(BOSSDistance)) = O(N \cdot n)$$

The BOSS ensemble classifier increases this runtime by a constant factor by testing a constant number of window lengths.

The BOSS classifier fit (Algorithm 3): The computational complexity of the *fit*-method results from leave-one-out cross-validation in combination with the 1-NN search. To obtain the best window lengths, at most n window lengths have to be tested to predict N labels each. This results in a computational complexity quadratic in the number of samples N and in the time series length n:

$$\begin{aligned} T(Fit) &= O(Nn[T(BOSS) + T(Predict)]) \\ &= O(Nn^2 + N^2n^2) = O(N^2n^2) \end{aligned}$$

If the length of patterns within a dataset is known ahead of time, the computational complexity can be trivially reduced to $O(Nn + N^2n)$ by testing only window lengths that are roughly equal to the pattern length.

7 Experimental evaluation

We evaluated the BOSS ensemble classifier using case studies and established benchmark datasets. Our web page reports all raw numbers and contains source codes (Webpage, The BOSS 2014). The BOSS ensemble classifier was implemented in JAVA and C++. All experiments were performed on a shared memory machine running LINUX with 8 Quad Core AMD Opteron 8358 SE processors, using the JAVA implementation and JAVA JDK x64 1.7. In all experiments we optimised the parameters of the classifiers based on the train dataset. The optimal set of parameters is then used on the test dataset. For example, the BOSS ensemble classifier requires two parameters: *factor* and *mean* (compare Sect. 5.1). We use the term *BOSS or BOSS classifier* as an equivalent to the BOSS ensemble classifier.

7.1 Case studies

7.1.1 Astronomy

It is easy to get large amounts of data, but it can be very time consuming to obtain labels for each data item. Thus, it is difficult to obtain large amounts of labelled data.

The *StarlightCurves* dataset is one of the largest freely available datasets (see Footnote 2) that consists of $N = 9236$ starlight curves, each of length $n = 1024$. There are three types of star objects: *Eclipsed Binaries* (purple), *Cepheids* (blue) and *RR Lyrae Variables* (green). This dataset is of particular interest as there are dozens of papers referencing this dataset.

Hierarchical clustering: Figure 7 illustrates a hierarchical clustering of the data. The *Cepheids* and *RR Lyrae Variables* have a similar shape and are difficult to separate. Both, the ED and DTW result in a visually unpleasing clustering, as they fail to separate Cepheids from RR Lyrae Variables. BOSS performs best in separating these two classes, which is a result of the noise reduction of SFA and the phase invariance of the BOSS model.

Classification: The BOSS classifier outperforms previous approaches in terms of classification accuracy. The 1-NN DTW classifier achieves a test accuracy of 90.7 % and the highest reported test accuracy is 93.68 % (Rakthanmanon and Keogh 2013). Our BOSS classifier has a test accuracy of 97.6 % (Table 1), which is the best published accuracy.

Scalability: We test the scalability based on subsets of 100 to 9,236 samples. Figure 8b shows four curves: (a) BOSS *train* including grid search on the parameter space (b) BOSS *predict*, (c) 1-NN DTW *predict* and (d) 1-NN Euclidean *predict*. The DTW is implemented using the lower bounding techniques presented in Rakthanmanon et al. (2012a), which result in a close to linear runtime. BOSS predict has the same asymptotic runtime as the DTW, yet yields in a much higher accuracy. BOSS predict takes at most 4 min for 1,000 samples and 56 minutes for 9,236 samples. The best rivalling method in Rakthanmanon and Keogh (2013) reports a more than 13 times higher runtime (52 min), to obtain a lower accuracy. The BOSS *prediction* takes at most 0.36 s on average for a single query. We conclude that the BOSS classifier is as fast as 1-NN DTW but significantly more accurate.

Parameter space: Figure 8b shows that the BOSS classifier is very robust to the choice of parameters *window length* and *number of features* on this dataset. A small number

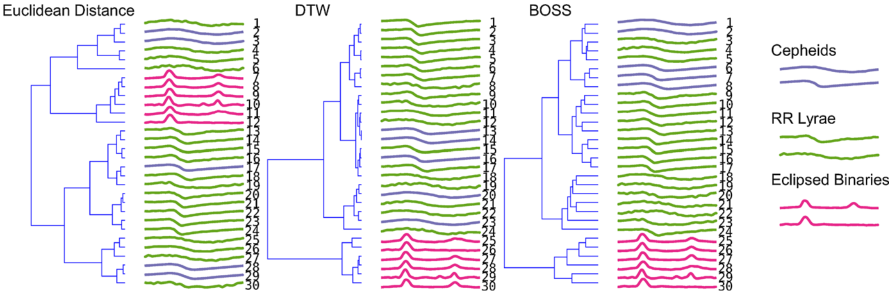


Fig. 7 Hierarchical clustering of StarlightCurves. There are three types of *star* objects: *Eclipsed Binaries*, *Cepheids* and *RR Lyrae Variables*

Table 1 Test accuracies along with test accuracies of the best rivalling methods and DTW (without a warping window)

Dataset	Best rival (%)	DTW (%)	BOSS (%)	BOSS parameters
Anthropology (Arrowhead)	80 (Ye and Keogh 2011)	66.3	88.6	$factor: 0.95, mean: T$
Medicine (BIDMC)	92.4 (Hu et al. 2013)	62.8	100	$factor: 0.95, mean: F$
Security (Passgraph)	70.2 (Mueen et al. 2011)	71.8	74	$factor: 0.95, mean: F$
Historical document (Shield)	89.9 (Ye and Keogh 2011)	86	90.7	$factor: 0.95, mean: T$
Astronomy (StarlightCurves)	93.7 (Rakthanmanon and Keogh 2013)	90.7	97.6	$factor: 0.95, mean: F$
Motions (Toe segmentation)	91 (Ye and Keogh 2011)	66.2	98.2	$factor: 0.95, mean: T$
Spectrographs (Wheat)	72.6 (Ye and Keogh 2011)	71.3	82.6	$factor: 0.95, mean: T$

of features is favourable, which is equivalent to a strong reduction in noise (low pass filter). We observed similar patterns on the other case studies. We omit these plots for the sake of brevity.

7.1.2 Human walking motions

The CMU[3] contains walking motions of four subjects. The authors (Ye and Keogh 2011) provide multiple segmentation approaches and we used their first segmentation approach. Each motion was categorised by the labels *normal walk* (green) and *abnormal walk* (orange). The data were captured by recording the z-axis accelerometer values of either the right or the left toe. The difficulties in this dataset result from variable length gait cycles, gait styles and paces due to different subjects throughout different activities including stops and turns. A normal walking motion consists of up to three repeated similar patterns.

Hierarchical clustering: Figure 9 shows a hierarchical clustering of the walking motions. The ED fails to identify the abnormal walking styles, thus these are not clearly separated from the normal walking motions. DTW provides invariance to phase shifts by a peak-to-peak and valley-to-valley alignment of the time series. This still does not result in a satisfying clustering as the abnormal and normal walking patterns are intermingled. As part of our BOSS model the patterns from the walking motions are extracted and noise reduction is applied. As a result the separation of the normal walking motions from the abnormal walking motions is much clearer with just the 19th walking motion being out of place.

Classification: The 1-NN DTW classifier gives a test accuracy of 66 %. The best reported accuracy in literature Ye and Keogh (2011) is 91 %. Training the BOSS classifier using grid search took about a second. This results in a test classification

[3] CMU Graphics Lab Motion Capture Database: http://mocap.cs.cmu.edu/. Accessed 2014.

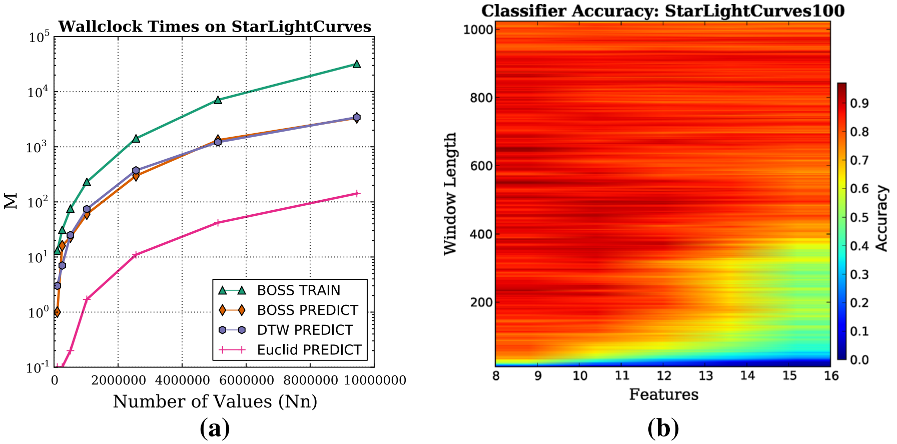


Fig. 8 **a** Wallclock times on the StarlightCurves dataset. **b** BOSS parameter space on the StarlightCurves dataset

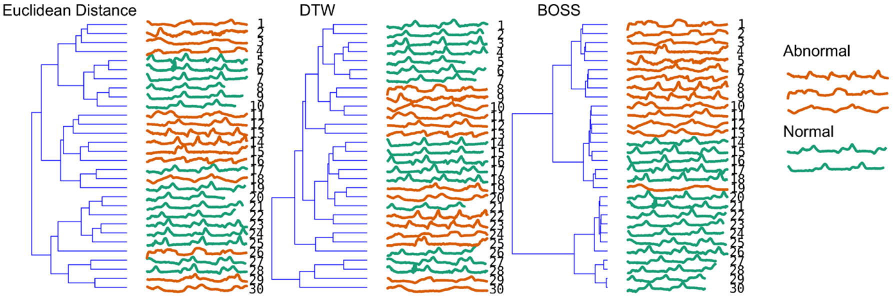


Fig. 9 Hierarchical clustering of human walking motions. There are two motions: *normal walk* and *abnormal walk*

accuracy for the BOSS classifier of 98.2 % (Table 1), which is by far the best reported accuracy.

7.1.3 Anthropology, historical documents, personalised medicine, spectrography and security.

We complement the case studies using datasets covering personalised medicine, anthropology, historical documents, mass spectrography and security (Fig. 10). *Passgraph* (Mueen et al. 2011) represents grids of dots, which a user has to connect to gain access to a resource like his smartphone. *Arrowheads* (Ye and Keogh 2009) is a dataset representing the shape of projectile points of variable lengths. *Shield* (Ye and Keogh 2009) contains heraldic shields of variable lengths. *Wheat* (Ye and Keogh 2009) is dataset of spectrographs of wheat samples grown in Canada clustered by year. The BIDMC Congestive Heart Failure Database (see Footnote 1) is a dataset that contains ECG recordings (heartbeats) of different subjects. These suffer from severe congestive heart failures. The results in Table 1 show that the BOSS classifier is applicable to

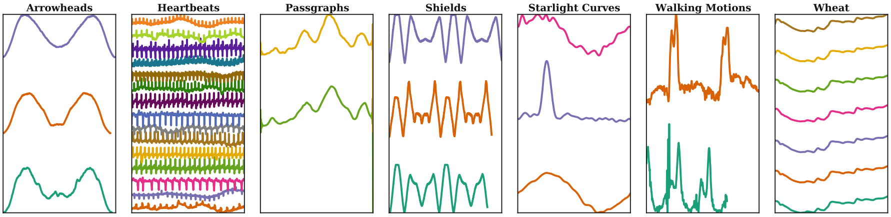


Fig. 10 One sample for each class of the case studies

a large scope of application areas including raw, extraneous, erroneous, and variable length data. It performs significantly better than the best, specialised rivalling methods by up to 10 percentage points. The accuracy gap to DTW is up to 37 percentage points.

7.2 UCR benchmark

The BOSS classifier is compared to state of the art classifiers like *structure-based* shapelets (Mueen et al. 2011) and bag-of-patterns (Lin et al. 2012) or *shape-based* 1-NN classifiers using ED or DTW with the optimal warping window. Additionally, more complex classifiers such as SVM with a quadratic and cubic kernel and a tree based ensemble method (random forest) were benchmarked. The classifiers were evaluated using time series datasets from the UCR time series classification archive (see Footnote 2). Each dataset provides a *train/test* split. By the use of these train/test splits, the results are comparable to those previously published in Bagnall et al. (2012), Batista et al. (2011), Ding (2008), Fast Shapelet Results (2012), Lin et al. (2012), Mueen et al. (2011), Rakthanmanon and Keogh (2013). All our results show the test accuracy of the classifiers. The BOSS ensemble classifier is trained using a constant *factor* : 0.92 and $mean : \{true, false\}$. The latter is selected based on the train datasets.

The scatter plots (Fig. 11) show a pair-wise comparison of two classifiers. In these plots each dot represents the test accuracies of the two classifiers on one dataset. The farther a dot is located from the diagonal line, the greater the difference in accuracy. A dot below the line indicates that the BOSS classifier is more precise than the rivalling method.

The scatter plots show that the BOSS is significantly better than each of the rivalling shape-based methods, structure-based methods and complex classifiers on a majority of the 32 datasets. These datasets have been preprocessed by a human for approximate horizontal alignment, still the BOSS classifier performs significantly better than the rivalling approaches.

The BOSS achieves a perfect test accuracy of 100 % on 6 datasets and a close to optimal accuracy on several other datasets. For most datasets there is a huge gap between the accuracy of the BOSS classifier and the rivalling methods. The 1-NN DTW classifier (with an optimal warping window) is used as a reference, as it has shown to be highly competitive (Ding 2008). However, DTW performs much worse than BOSS on a majority of datasets. This is a remarkable result, as it implies that either

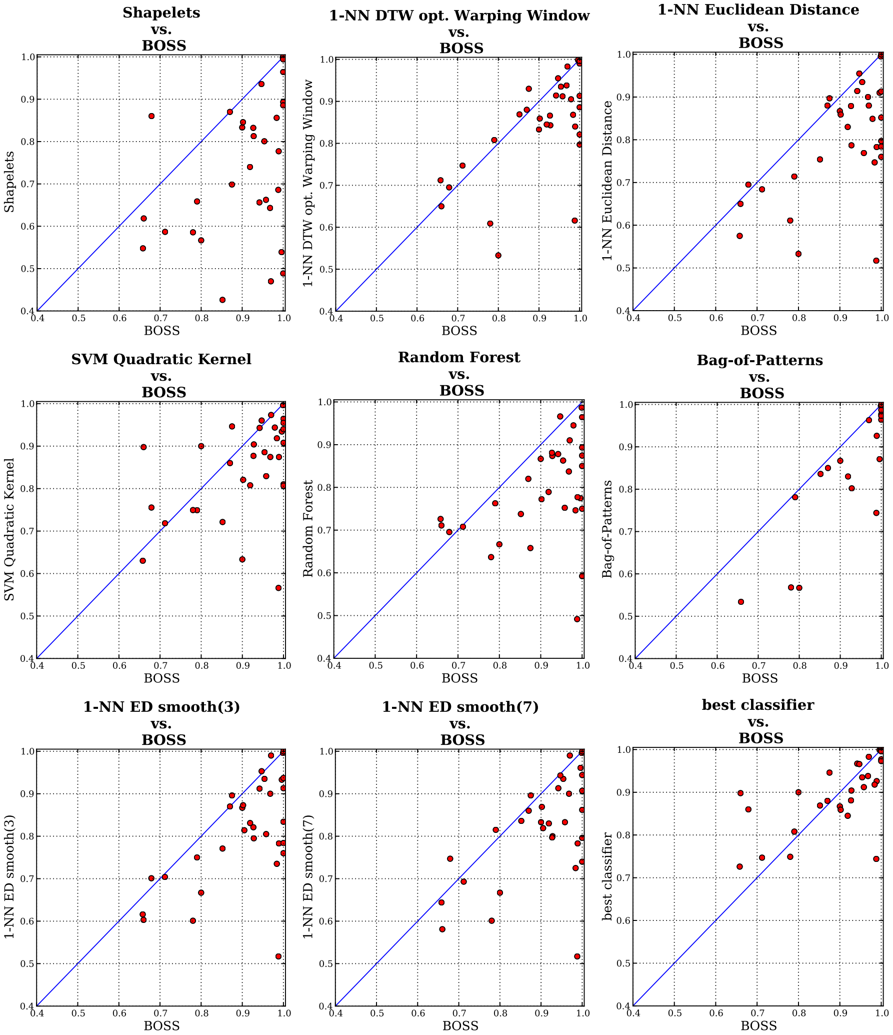


Fig. 11 Classifier accuracies on test subsets for the BOSS ensemble classifier versus rivalling methods. Each *dot* represents a dataset. A *dot* below the *line* indicates that the BOSS ensemble classifier is more accurate than its rival

(a) most time series datasets do not require time warping, or (b) the BOSS implicitly provides some kind of time warping. This remains part of our ongoing research.

7.2.1 Invariance to noise

To underline the influence of noise, we applied different levels of smoothing to the data prior to the classification task using matlab's *smooth*-function prior to the 1-NN ED classification. The results are presented in the two scatter plots in Fig. 11 (bottom): 1-NN ED smooth(3) and smooth(7). When smoothing is applied the 1-NN ED classifier accuracy improves by more than 10 percentage points on three datasets (FaceAll, synthetic_control, Beef) with hard coded parameters of 3 or 7. This underlines the importance of smoothing a signal to counter noise. The BOSS classifier optimises the amount of noise reduction as part of the training and outperforms both smoothed 1-NN ED classifiers even without the additional use of the *smooth*-function. This is a result of noise cancelling and also to the invariances provided by our BOSS model.

7.2.2 Building a golden classifier

We showed that the BOSS classifier is better than every single classifier presented in this paper for a majority of datasets. To give a complete view, we assume that we could predict ahead of time which of the 7 classifiers (Shapelets, Fast Shapelets, 1-NN ED, 1-NN DTW, SVM, Random Forest, Bag-of-Patterns) will give the best accuracy for a dataset and use the classifier on this particular dataset. The scatter plot in Fig. 11 (bottom right) shows the results. When compared to the best of 7 classifiers our BOSS classifier performs better on 17 datasets, scores a tie on 2 datasets and is worse on 13 datasets. We can not claim that the BOSS classifier is the best classifier to use on all datasets. However, in total it is competitive to a combination of 7 state of the art classifiers.

7.3 Texas sharpshooter plot

The Texas sharpshooter plot (Batista et al. 2011) illustrates a method to predict ahead of time if one algorithm is better than another algorithm in terms of classification accuracy. The aim is to predict the test accuracy for the 1-NN ED and the BOSS classifier based on the accuracy on the train data. The gain in accuracy when using the

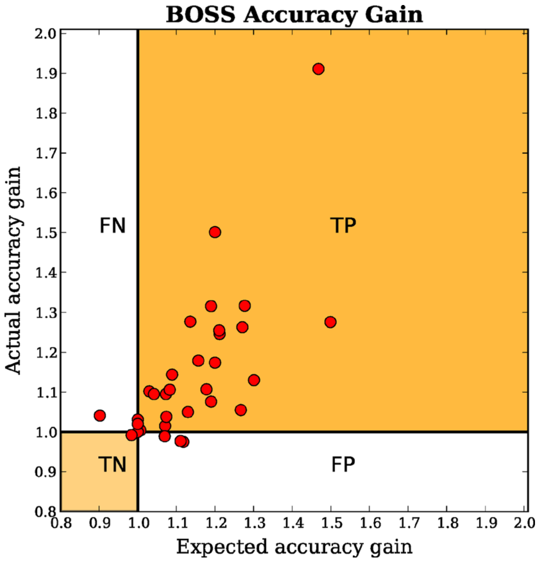


Fig. 12 Expected accuracy gain from train data compared to actual accuracy gain on test data

BOSS classifier as a replacement of 1-NN ED can be measured by:

$$\text{gain} = \frac{\text{accuracy BOSS classifier}}{\text{accuracy 1-NN ED}}$$

Gain values greater than 1 indicate that the BOSS classifier is better than 1-NN ED for one particular dataset. The gain is measured on both the train and test dataset splits. The plot in Fig. 12 shows the actual gain on the test dataset versus the expected gain on the train dataset. There are four regions of interest:

- *True Positive (TP)*: We expected the accuracy to improve and were correct. 27 out of 32 datasets fall into this region.
- *False Negative (FN)*: We expected the accuracy to drop but it increased. This is a lost chance to improve (MedicalImages).
- *True Negative (TN)*: We correctly predicted the accuracy to decrease. One dataset falls into this region (ItalyPowerDemand).
- *False Positive (FP)*: We expected the accuracy to improve but it decreased. This is the bad region as we lost accuracy by deciding to use the BOSS classifier. Three datasets (CinC_ECG_torso, ECG200, SonyAIBORobotSurface) fall into this region. However, for all of these datasets the loss in accuracy is less than 2 percentage points.

7.4 Impact of design decisions

The BOSS model is based on three design decisions:

1. The *BOSS distance* as opposed to the commonly used Euclidean distance or Cosine similarity.
2. *MCB using equi-depth binning* as opposed to MCB using equi-width binning.
3. *Mean normalisation* as a parameter as opposed to always normalising the mean of all windows.

We chose to use 1-NN classification as it doesn't introduce any new parameters for model training which allows us to focus on the BOSS model. Thus we omit to study the effects of different classification algorithms. The scatter plots in Fig. 13 justify

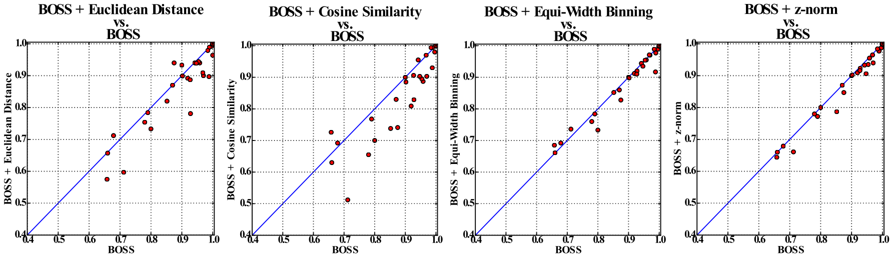


Fig. 13 Classifier accuracies on test subsets for the BOSS ensemble classifier using two different distance metrics, a different binning technique or always apply z-normalisation

the use of each of the design decisions. Overall the BOSS distance showed a better or equal accuracy on 21 datasets when compared to ED or Cosine similarity. The ED and Cosine similarity performed equally worse with 8 and 10 ties/wins respectively. However, these can be applied if a distance metric to satisfy the symmetry condition or the triangle inequality is required as for indexing. The difference between equi-depth and equi-width binning is marginal, whereas equi-depth performed slightly better or equal to equi-width on 21 out of 32 datasets. As for mean normalisation the accuracies increased by up to 6.5 percentage points (Lighting2) when treated as a parameter.

8 Conclusion

The time series classification task is complicated by extraneous, erroneous, and unaligned data of variable length. Human assistance is commonly used to prepare the data so that similarity search algorithms can be applied. We introduce the BOSS model based on the structural representation of the bag-of-words model and the tolerance to extraneous and erroneous data of the SFA representation. It allows for fast data analytics on raw time series datasets as it is very robust to noise and compares two time series based on their higher-level substructures. The BOSS ensemble classifier is based on 1-NN classification and represents each time series by multiple BOSS models at different substructural sizes. Optimisation techniques are presented to reduce the computational complexity of the BOSS classifier prediction up to the level of DTW while being much more accurate. As part of our experimental evaluation we show that the BOSS ensemble classifier improves the best published test accuracies in diverse application areas. Finally, the BOSS ensemble classifier performs significantly better than the state of the art classifiers on the UCR benchmark datasets.

Acknowledgments The author would like to thank the anonymous reviewers, Claudia Eichert-Schäfer, Florian Schintke, Florian Wende, and Ulf Leser for their valuable comments on the paper and the owners of the datasets.

References

Agrawal R, Faloutsos C, Swami A (1993) Efficient similarity search in sequence databases. Foundations of Data Organization and Algorithms.

Albrecht S, Cumming I, Dudas J (1997) The momentary fourier transformation derived from recursive matrix transformations. In: IEEE Digital Signal Processing Proceedings.

Bagnall A, Davis LM, Hills J, Lines J (2012) Transformation based ensembles for time series classification. In: SDM. SIAM/Omnipress.

Batista G, Wang X, Keogh EJ (2011) A complexity-invariant distance measure for time series. In: SDM. SIAM/Omnipress.

Chen Q, Chen L, Lian X, Liu Y, Yu JX (2007) Indexable PLA for efficient similarity search. In: VLDB. ACM.

Ding H (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. VLDB Endowment.

Fast Shapelet Results: (2012) http://alumni.cs.ucr.edu/rakthant/FastShapelet/

Hu B, Chen Y, Keogh E (2013) Time series classification under more realistic assumptions. In: SDM.

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.

Keogh E, Kasetty S (2002) On the need for time series data mining benchmarks: a survey and empirical demonstration. In: Proceedings of the 8th KDD, ACM, pp. 102–111.

Kumar N, Lolla VN, Keogh EJ, Lonardi S (2005) Ratanamahatana, C.A.: Time-series bitmaps: a practical visualization tool for working with large time series databases. In: SDM.

Lin J, Keogh EJ, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144.

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.

Mueen A, Keogh EJ, Young N (2011) Logical-shapelets: an expressive primitive for time series classification. In: KDD. ACM.

Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: ACM SIGKDD. ACM.

Rakthanmanon T, Campana BJL, Mueen A, Batista GEAPA, Westover M, Zakaria J, Keogh EJ (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: KDD. ACM.

Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: SDM.

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 1:43–49

Schäfer P, Högqvist M (2012) SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: EDBT. ACM.

Senin P, Malinchik S (2013) SAX-VSM: Interpretable time series classification using SAX and vector space model. In: IEEE 13th International Conference on Data Mining (ICDM) 2013.

Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: ICDE, San Jose.

Liao Warren T (2005) Clustering of time series data-a survey. Pattern Recognit 38(11):1857–1874

Webpage, The BOSS (2014) http://www.zib.de/patrick.schaefer/boss/

Ye L, Keogh EJ (2009) Time series shapelets: a new primitive for data mining. In: KDD. ACM.

Ye L, Keogh EJ (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min Knowl Discov 22(1–2):149–182.

Zakaria J, Mueen A, Keogh EJ (2012) Clustering time series using unsupervised-shapelets. In: ICDM. IEEE Computer Society.