Noname manuscript No.
(will be inserted by the editor)

Scalable Time Series Classification

Patrick Schäfer

the date of receipt and acceptance should be inserted later

Abstract Time series classification tries to mimic the human understanding of similarity. When it comes to long or larger time series datasets, state-of-the-art classifiers reach their limits because of unreasonably high training or testing times. One representative example is the 1-nearest-neighbor DTW classifier (1-NN DTW) that is commonly used as the benchmark to compare to. It has several shortcomings: it has a quadratic time complexity in the time series length and its accuracy degenerates in the presence of noise. To reduce the computational complexity, early abandoning techniques, cascading lower bounds, or recently, a nearest centroid classifier have been introduced. Still, classification times on datasets of a few thousand time series are in the order of hours. We present our Bag-Of-SFA-Symbols in Vector Space (BOSS VS) classifier that is accurate, fast and robust to noise. We show that it is significantly more accurate than 1-NN DTW while being multiple orders of magnitude faster. Its low computational complexity combined with its good classification accuracy makes it relevant for use cases like long or large amounts of time series or real-time analytics.

Keywords Time Series · Classification · Data Mining · Symbolic Representation

1 Introduction

Time series are a collection of values sequentially recorded from sensors or live observations over time. In the last decades there has been an enormous increase in data volumes and it is expected to reach 100 zettabytes (10^{21}) by 2020 [1]. At the same time, sensors for recording time series have become cheap and omnipresent as in RFID chips, wearable sensors (wrist bands, smartphones), smart homes [2], or event-based systems [3]. A smart-meter with a sampling rate of one value per

Patrick Schäfer
Zuse Institute Berlin
Takustr. 7
14195 Berlin
Tel.: +49-30-84185-168
E-mail: patrick.schaefer@zib.de

	Train Complexity	Test Complexity		Accuracy
		lower bound	upper bound	
1-NN ED [4]	-	$\Omega(n+N)$	$O(Nn)$	
Shapelets [5,6,7]	$O(N^2n^3)$ to $O(Nn^2)$		$O(n)$	+/++
1-NN DTW [8,4]	-	$\Omega(n^2+N)$	$O(Nn^2)$	+
1-NN DTW CV [8,4]	$O(N^2n^2)$	$\Omega(nr+N)$	$O(Nnr)$	+
SVM	$O(N^2n)$ to $O(N^3n)$		$O(n)$	+
Shotgun Classifier [9]	$O(N^2n^3)$	$\Omega(n^2+N)$	$O(Nn^2)$	+
Ensemble PROP [10]	$O(N^2n^2)$		$O(Nn^2)$	++
Ensemble COTE [11]	$O(N^2n^4)$		$O(Nn^2)$	++
1-NN BOSS [12]	$O(N^2n^2)$	$\Omega(n+N)$	$O(Nn)$	++
1-NN BOSS VS	$O(Nn^{\frac{3}{2}})$		$O(n)$	+

Table 1 Computational complexity (upper $O(...)$ and lower bounds $\Omega(...)$) of state-of-the-art classifiers for n=length, N=number of time series, r=warping window constraint.

minute records more than 0.5 million values a year. Given a company with millions of customers, this accounts for trillions (10^{12}) of measurements. A goal is to extract hidden knowledge from that raw data.

The availability of the UCR time series benchmark datasets [13] has led to a wealth of time series classification algorithms. The *classification accuracy* has been the key metric to evaluate new time series classification methods [10,14]. The use of the UCR datasets has led to unreasonable high computation complexities (Table 1), as the largest datasets contain only a few thousand time series of a few thousand values length. A key challenge is to provide *scalability in classification times* with high accuracy. Common classifiers are 1-Nearest Neighbor Dynamic Time Warping (1-NN DTW), the 1-NN Euclidean Distance (1-NN ED) or most recently our noise robust 1-NN Bag-of-SFA-Symbols (1-NN BOSS) classifier [12]. The 1-NN DTW classifier is commonly used as the benchmark to compare to [14, 10].

Figure 12 shows for each state-of-the-art classifier the total wall-time on 91 public time series datasets using a single CPU core. The datasets are ordered by increasing time series length. In total the datasets account for roughly $N = 50000$ train and $N = 100000$ test time series. The 1-NN DTW classifier with a warping window constraint (1-NN DTW CV) takes more than **2000** CPU hours until completion, when using the state-of-the-art implementation [4]. 1-NN DTW finishes within **80** CPU hours and it does not require training. The 1-NN BOSS classifier [12] finishes after roughly **97** CPU hours. Our *1-NN Bag-Of-SFA-Symbols in Vector Space* (1-NN BOSS VS) classifier takes roughly **7** CPU hours, which is one to three orders of magnitude faster. These empirical results confirm the computational complexities in Table 1. To put a factor of 10^3 into relation: we can run 1-NN DTW CV on a cluster of 4000 cores for one day [8], or spent one to two days with 1-NN BOSS VS on commodity hardware and a 4 core CPU, resulting in a similar or better classification accuracy.

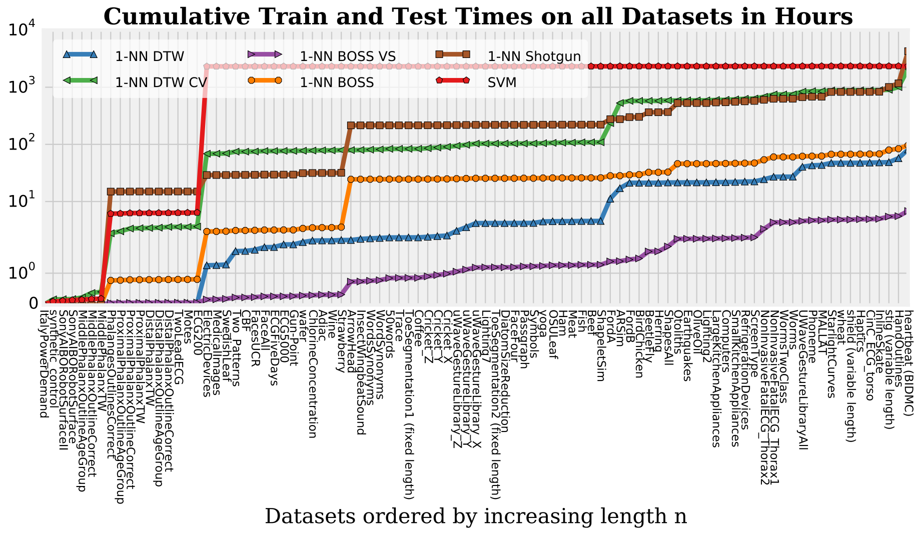


Fig. 1 Cumulative runtime on 91 time series datasets (Table 2) using a single CPU core. The datasets are ordered by increasing time series length.

In our previous work we introduced the *Bag-Of-SFA-Symbols (BOSS)* model [12]. It combines the string representation and noise tolerance of the Symbolic Fourier Approximation (SFA) [15] representation with the bag-of-words model. First, subsequences are extracted from a time series. Next, the SFA transformation is applied to each subsequence resulting in a string (SFA word) for each subsequence. The BOSS model is a histogram over these SFA words and thus describes the structure of a time series. It offers a very high classification accuracy. Its computational complexity limits its utility for large datasets.

In this work we address the scalability in classification times while maintaining high accuracy with robustness to noise. The *Bag-Of-SFA-Symbols in Vector Space (BOSS VS)* model extends the BOSS model by a compact representation of classes instead of time series. It uses the *term frequency - inverse document frequency* (tf-idf) model, which is built for each class. This significantly reduces the computational complexity, highlights characteristic SFA words, and has an additional noise reducing effect. We significantly reduce the parameter space to improve train times. Our 1-NN BOSS VS is (a) significantly more accurate than the benchmark 1-NN DTW classifier with or without a warping window constraint, and (b) multiple orders of magnitude faster than the most accurate state-of-the-art classifiers. Our contributions are as follows:

- This is the first work that compares state-of-the-art classifiers regarding classification times.
- We present the background of the SFA representation and the BOSS model in Section 2.
- We present the 1-NN BOSS VS model that combines the noise tolerance of the BOSS model with fast train and test times due to the use of the vector space model in Section 3.

- We present case studies which underline that our BOSS VS classifier is orders of magnitude faster than state-of-the-art classifiers in Section 5.1.
- We present an exhaustive study using 91 time series datasets, which shows that our 1-NN BOSS VS classifier is significantly more accurate than 1-NN DTW, while being up to 4 orders of magnitude faster than state-of-the-art (Section 5.2). The 1-NN BOSS VS classifier is among the most accurate classifiers in our experiments.
- Finally, we show the impact of our design decisions to the BOSS VS model (Section 5.3).

2 Background and Related Work

2.1 Definitions

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

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

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-normalized by subtracting its mean and dividing it by its standard deviation.

2.2 From Real Values to Words

The Symbolic Fourier Approximation (SFA) [15] 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. The SFA transformation aims at:

- **Noise removal:** Rapidly changing sections of a signal are often associated with noise. The SFA word length determines the number of Fourier coefficients and thereby the bandwidth of the low-pass filter.
- **String representation:** It allows for string algorithms like the bag of words to be used. The size of the quantization alphabet has an additional noise reducing effect.
- **Frequency domain:** We can choose an arbitrary subset of Fourier coefficients without recalculating the Fourier transform. Adding Fourier coefficients to an SFA word adds details and reduces the reconstruction error between the transformed and the original time series.

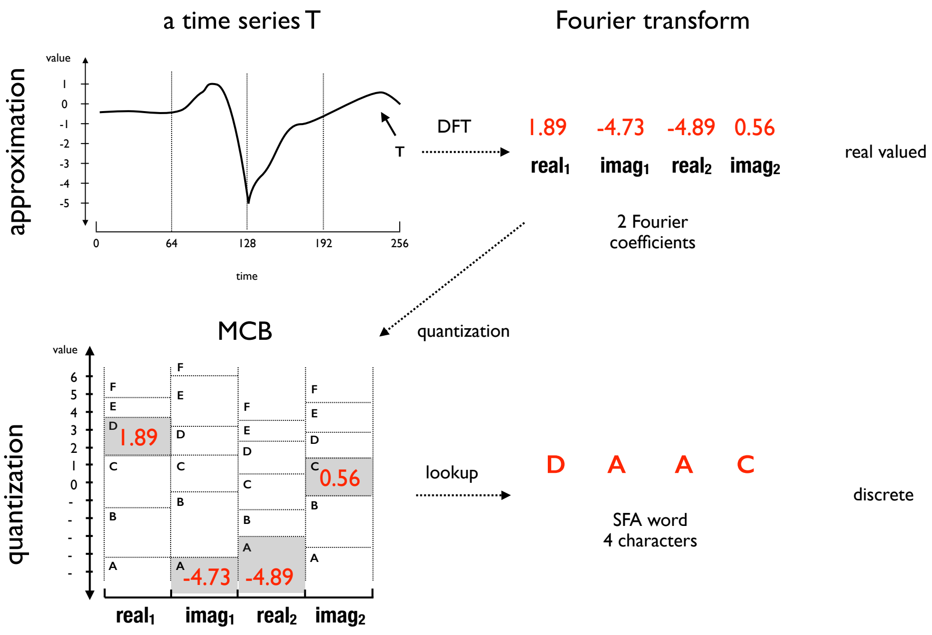


Fig. 2 SFA: A time series is (a) approximated (low-pass filtered) using DFT and (b) quantized using MCB resulting in the SFA word *DAAC*. [12,15]

2.3 Symbolic Fourier Approximation (SFA)

SFA has two parameters (Figure 2):

– The SFA **word length** $l \in \mathbb{N}$ represents the number of Fourier coefficients for approximation. Commonly, the first Fourier coefficients are used. A smaller SFA word length correlates to stronger noise reduction by using less Fourier coefficients.
– The SFA **alphabet size** $c \in \mathbb{N}$ is used for quantization. A smaller alphabet size results in stronger noise reduction.

The *approximation step* aims at representing a time series 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 $\frac{l}{2} \ll n$ Fourier coefficients. *Quantization* adds to noise reduction by dividing the frequency domain into frequency bins and mapping a Fourier coefficient to its bin. In essence MCB quantization determines equi-depth bins to map the real and imaginary part of the Fourier coefficients separately to symbols. As part of MCB a separate histogram for each real and imaginary part is built using all train samples. The histograms are then partitioned using equi-depth binning. Figure 2 bottom right illustrates the SFA transformation. A time series is transformed using DFT resulting in a vector of real values $(1.89, -4.73, -4.89, 0.56)$. The vector is quantized to the SFA word *DAAC* using the precomputed MCB bins.

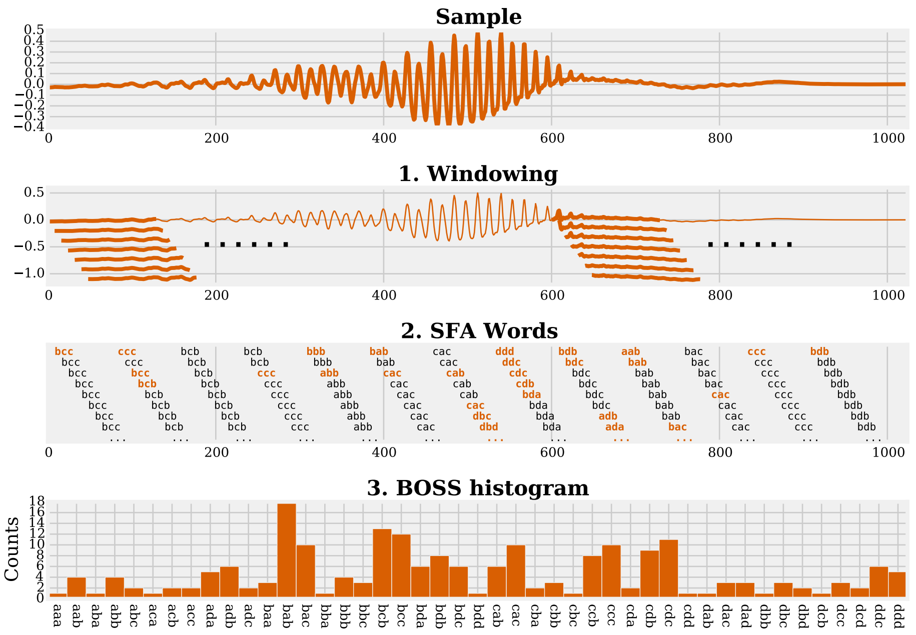


Fig. 3 The BOSS model [12] is extracted from a sample time series using word length 3 and 4 symbols (a-d). The black SFA words are skipped due to numerosity reduction.

Algorithm 1 The BOSS transformation.

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

2.4 The Bag-of-SFA-Symbols (BOSS) Model

The BOSS model describes each time series as an unordered set of windows (subsequences) using SFA words. It has four parameters (Figure 3):

- **The window length** $w \epsilon \mathbb{N}$: Represents the length of the windows.
- **Mean normalization** $mean \epsilon [true, false]$: Set to true for offset invariance.
- The two SFA parameters **word length** $l \epsilon \mathbb{N}$ **and alphabet size** $c \epsilon \mathbb{N}$: Used for low-pass filtering and the string representation.

The BOSS model (Algorithm 1) transforms a time series into an unordered set of SFA words. Using an unordered set provides invariance to the horizontal alignment of the substructure contained in the time series (phase shift + local scaling invariances). First, sliding windows of length w are extracted (line 3). Intuitively w roughly represents the size of the characteristic patterns within a time series.

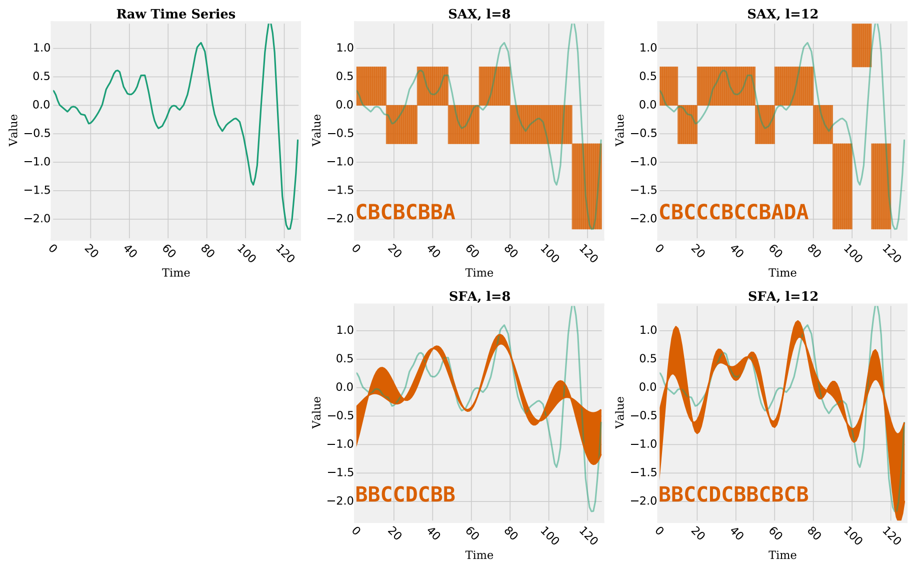


Fig. 4 For SAX (top), we (a) have to recalculate all symbols (mean values), or (b) drop the rear part of the time series when changing the word length l. For SFA (bottom), the symbols (Fourier coefficients) of a smaller word length are always a prefix of the larger word lengths.

Each sliding window is normalized 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. For example, heartbeats should be compared using a common baseline (mean=true) but the pitch of a bird sound can be significant for the species (mean=false). Finally, the SFA transformation is applied to each real-valued sliding window with a length l and an alphabet size c (line 4). In stable sections of a signal, the SFA words of two neighboring sliding windows are very likely to be identical. Thus, the first occurrence of an SFA word is counted and all duplicates are ignored until a new SFA word is detected (line 6). This is called *numerosity reduction* [16,17].

Definition 2 Bag-Of-SFA-Symbols (BOSS): Given are a time series T, and the SFA transformations $SFAs(T) = \{SFA(S)\,|S \,\epsilon\, windows(T,w,1)\,\}$ of the sliding windows. The BOSS histogram (BOSS model) $B: \ \Sigma^l \to \mathbb{N}$ is a function of the SFA word space Σ^l to the natural numbers. The number represents the occurrences of an SFA word within $SFAs(T)$ counted after numerosity reduction.

2.5 Related Work

Classical data mining algorithms like SVMs, decision trees, rotation, random forests, or Naive Bayes have been used in the context of time series [18]. However, these did not perform better than the 1-NN DTW classifier, which is commonly used as the benchmark to compare to [10,14]. Its computational complexity is $O(Nn^2)$ for dataset size N and time series length n. The best case computational complexity has been reduced by the use of early abandoning techniques and cascading lower bounds to prune off unpromising candidates in the UCR suite [4]. These lower bounds have constant to linear time to compute with an increasing tightness of lower bounds. Most recently a 1-NN DTW nearest centroid classifier has been presented [19] to reduce the test complexity of 1-NN DTW to $O(n)$ at the cost of an excessive training complexity. The 1-NN DTW CV classifier sets a warping window constraint through cross-validation. This reduces the test complexity to $O(Nnr)$ for warping window constraint $\frac{1}{100} \leq r \leq 1$, at the cost of an excessive train complexity of $O(N^2n^2)$.

Shapelet classifiers [5,6] extract representative variable-length subsequences from time series and use a decision tree for classification. These classifiers have a high computational complexity for training of $O(N^2n^3)$ [5] to $O(Nn^2)$ [6]. The 1-NN Shotgun classifier [9] divides the query into disjoint subsequences and slides each query window over the sample to find the position that minimizes the Euclidean distance. The Shotgun classifier has a high $O(N^2n^3)$ complexity for training. Both, Shapelet and the Shotgun Classifier are based on the Euclidean distance and are therefore sensitive to noisy data. Learning Shapelets (LS) [7] consider the Shapelets to be parameters to be optimized, rather than extracting the candidate Shapelets from the training dataset. It can generate arbitrary Shapelets, that build a hyperplane for classification.

Our previously published BOSS model [12] is based on the Symbolic Fourier Approximation (SFA) [15] and the bag-of-words model. SFA is a symbolic representation of time series like Symbolic Aggregate approXimation (SAX) [17]. Unlike SAX, which uses mean values (PAA) to approximate a time series, SFA uses DFT coefficients. Both, have a noise canceling effect. The resolution of SFA can be dynamically adapted by choosing any arbitrary subset of Fourier coefficients without recalculating the DFT of a time series (Figure 4). In contrast, mean values have to be recalculated when changing the resolution - i.e. from weekly to monthly mean values. The computational complexity limits its utility for large or long datasets.

The bag-of-patterns (BOP) model [16] is the closest to the BOSS model. BOP extracts substructures as higher-level features of a time series. BOP transforms these substructures using SAX for quantization and the Euclidean distance as similarity metric. SAX-VSM [20] is the successor of the BOP model. It extends the BOP model by the use of the *tf-idf* weighing 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. SAX-VSM has a huge parameter space of $O(n^2)$ parameters and has to recalculate all SAX coefficients for each new set of parameter. This results in an unreasonably high train time of $O(Nn^3)$. The bag-of-features framework (TSBF) [21] extracts random subsequences at random lengths from a time series and builds a supervised codebook generated from a random forest classifier. Its computational complexity relates to the complexity of the random forest classifier and the number of time series subsequences extracted.

Ensemble classifiers utilize other classifiers. These have the computational complexity of the slowest used classifier, as these have to train and run all classifiers to predict a label. The Elastic Ensemble (PROP) classifier [10] builds an ensemble of 11 classifiers including 1-NN DTW CV, 1-NN DTW, 1-NN LCSS, 1-NN ED, Shapelets, etc. The COTE ensemble [11] classifier is based on an ensemble of 35 classifiers and feature extraction techniques. It maximizes the classification accuracy at the cost of an enormous computational complexity.

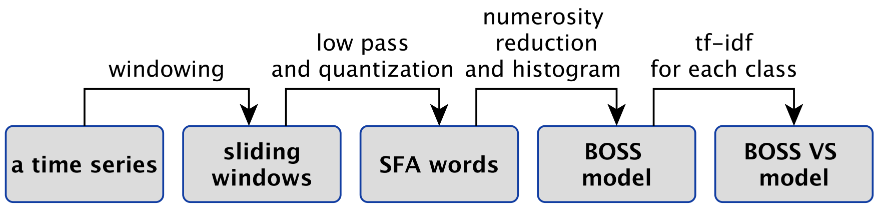


Fig. 5 From a time series to the BOSS VS model.

Time series bitmaps [22] are a time series representation. First, subsequences are extracted and transformed using SAX. This first step is equivalent to the BOP model. Next, even smaller sub-words are extracted from each SAX word. The counts of sub-words are arranged in a grid. Like BOP, SAX-VSM and the BOSS model it applies noise reduction and is suited for long time series. The approach differs from the others as it counts sub-words rather than directly using the count of words. Bitmaps have been used as a visualization tool, but also for anomaly detection, clustering and classification on two long time series datasets. We focus on classification. For the two datasets used in [22] our 1-NN BOSS VS achieves a perfect 100% classification accuracy. The count of sub-words is not applicable to the BOSS VS model, as each Fourier coefficient corresponds to a specific frequency and cannot be reordered arbitrarily.

In contrast our 1-NN BOSS VS has a small parameter space with just $O(\sqrt{n})$ parameters, leading to very low train times, and uses a compact representation of classes, leading to a low test complexity of just $O(n)$. Its accuracy is very competitive when compared to state of the art.

3 The BOSS in Vector Space: A Noise-Robust Similarity Model for Long and Large Time Series Datasets

3.1 Motivation

The BOSS VS model combines the BOSS model with the vector space model. The vector space model has first been introduced for representing text documents as vectors of keywords. The Cosine similarity is used to compare the similarity of documents. Since then it has been applied to other domains like audio [23] or time series retrieval [16, 20]. In the vector space model the *term frequency inverse document frequency* (*tf-idf*) model is often used [24]. The term frequency refers to the occurrence of words in a document. The *tf-idf* measure is used to weigh the term frequencies in the vector to give a higher weight to representative terms (words) of a class. In our model the *term* is an SFA word and a *document* corresponds to a time series.

Figure 5 illustrates the BOSS VS model: First, a time series is transformed to its BOSS model (Section 2.4). Next, a tf-idf vector is computed for each *class label*, as opposed to each time series. The tf-idf vector serves as a model for each class. It has several advantages:

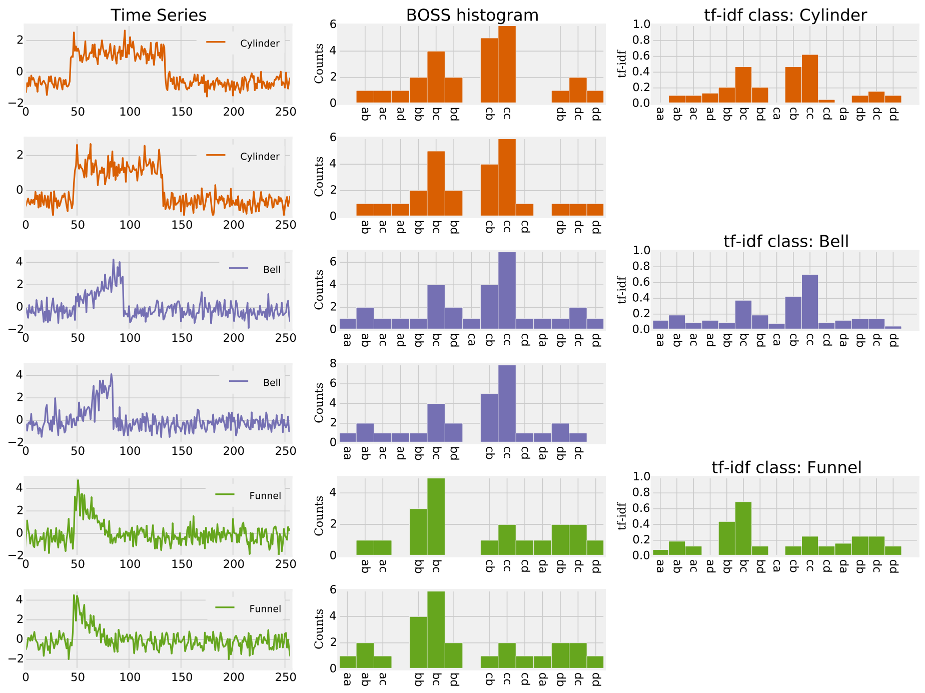


Fig. 6 Columns from left to right: The SFA word frequencies are stored in the BOSS histograms for each time series (center). Next, the *tf-idf* vectors are constructed for each class (right).

1. It is suited for *long time series* as it extracts subsequences and compares two time series based on their structural similarity;
2. It applies *noise reduction* by the use of SFA;
3. It provides invariances to phase shifts, local scaling, offsets, amplitudes, and occlusions;
4. It is *fast* as it uses a compact representation of classes instead of time series. It thereby minimizes the influence of erroneous and extraneous data within a single time series, and significantly reduces the computational complexity;
5. SFA words that occur frequently across all classes are given a lower *idf* weight. This adds to the occlusion invariance and has a noise reducing effect.

Properties (1) to (3) were introduced with the BOSS model [12]. The BOSS VS model adds properties (4) and (5) to the BOSS model.

3.2 The BOSS VS Model

Our BOSS VS model (Figure 6) has the same four parameters as the BOSS model:

– The **window length** $w \in \mathbb{N}$: Represents the size of the substructures.
– **Mean normalization** $mean \in [true, false]$: Set to true for offset invariance.
– The **SFA word length** $l \in \mathbb{N}$ and **alphabet size** $c \in \mathbb{N}$: Used for low-pass filtering and the string representation.

First, each time series is transformed to its BOSS histogram. Next, the *tf-idf* matrix is constructed using all BOSS histograms. It contains one *tf-idf* vector for each class (Cylinder, Bell, Funnel in Figure 6).

We use an approach presented in [20] and calculate the *inverse document frequency* (*idf*) for *each class* as opposed to *each time series*. The term frequency (*tf*) for an SFA word t of a time series T is given by:

$$tf(t,T) = \begin{cases} 1+\log(B_T(t)) & , \textit{if } B_T(t) > 0 \\ 0 & , \textit{otherwise} \end{cases} \tag{3}$$

with $B_T(t)$ being the BOSS histogram, which represents the frequency of an SFA word t in the specific time series T. In the same manner the *term frequency (tf)* for an SFA word t within a class C is given by:

$$\textit{tf}(t,C) = \begin{cases} 1+\log(\sum_{T \epsilon C} B_T(t)) & , \textit{if } \sum_{T \epsilon C} B_T(t) > 0 \\ 0 & , \textit{otherwise} \end{cases} \tag{4}$$

The *inverse document frequency (idf)* captures how relevant an SFA word is across all time series T within a class C:

$$\textit{idf}(t,C) = \log(1 + \frac{|\textit{CLASSES}|}{\underbrace{|\{C \,|T \epsilon C \wedge B_T(t) > 0\}|}_{\textit{number of classes that contain } t}}) \tag{5}$$

This *idf* for an SFA word represents the total number of classes divided by the number of classes this SFA word occurs in. A high *idf* value is obtained by SFA words that occur only in a specific class.

The *tf-idf* of an SFA word t within a class C is thus defined as:

$$\begin{aligned} \textit{tfidf}(t,C) &= \textit{tf}(t,C) \cdot \textit{idf}(t,C) & (6) \\ &= (1+\log(\sum_{T \epsilon C} B_T(t))) \cdot \log(1 + \frac{|\textit{CLASSES}|}{|\{C \,|T \epsilon C \wedge B_T(t) > 0\}|}) & (7) \end{aligned}$$

High *tf-idf* weights are obtained by SFA words with a high frequency that occur only in a specific class. Thus, SFA words that are common within all classes receive a low weight and are thereby filtered out.

3.3 The BOSS VS Distance

The similarity of a *tf* vector of query Q to an *tf-idf* class vector of sample C can then be computed using the Cosine similarity metric:

$$D_{BOSSVS}(Q,C) = \frac{\vec{Q} \cdot \vec{C}}{\left\|\vec{Q}\right\| \cdot \left\|\vec{C}\right\|} = \frac{\sum_{t \epsilon Q} \textit{tf}(t,Q) \cdot (\textit{tfidf}(t,C)+1)}{\sqrt{\sum_{t \epsilon Q} (\textit{tf}(t,Q))^2} \sqrt{\sum_{t \epsilon C} (\textit{tfidf}(t,C))^2}} \tag{8}$$

3.4 Computational Complexity:

The BOSS model: The BOSS model has a computational complexity that is linear in the time series length n [12] :

$$T(BOSS) = O(n)$$

Algorithm 2 Predict: 1-NN classification using the BOSS VS model.

```
String predict(map<String,int> tf, map<String,int>[] tfIdfs)
  (double maxSim, String bestClass) = (0, NULL)
  for int classId in [1..len(tfIdfs)] // search classes
    double cosSim = dotProduct(tf, tfIdfs[classId])
    if cosSim > maxSim  // store the best class
      (maxSim, bestClass) = (cosSim, classId)
  return label(bestClass)
```

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

```
(int,int,int,map<String,int>) fit(TimeSeries[] samples, bool mean)
  int maxL=16, int c=4
  int bestCor=0, int bestL=0, int bestW=0
  map<String,int>[] bestTfIdfs = []
  for (int w = 10; w <= n; w += sqrt(n-10)) // iterate sqrt(n) window
       lengths
    map<String, int>[] sHist = []
    for int i in [1..len(samples)] // obtain the bags
      sHist[i] = BOSSTransform(samples[i], w, maxL, c, mean)
    for int l in [maxL,...,8,6,4]  // test all word lengths
      map<String, int>[] bags = createHistogram(sHist, f)
      map<String,int>[] tfIdfs = calcTfIdf(bags, l)  // tf-idf matrix
      int correct=0
      for int qId in [1..len(samples)]  // leave-one-out
        String bestClass = predict(bags[qId], tfIdfs)
        if bestClass has correct label then correct++
      if correct > bestCor  // keep best
        (bestCor, bestL, bestW, bestTfIdfs) = (correct, l, w, tfIdfs)
  return (bestCor, bestL, bestW, bestTfIdfs) // best parameters
```

Cosine Similarity: 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:

$$T(BOSSVSDistance) = O(n-w+1) = O(n) \tag{9}$$

Due to duplicates and numerosity reduction the actual number of *unique* SFA words is typically much smaller than n .

4 Time Series Classification

Classification describes the task of assigning a label to an unlabeled time series Q using a trained model from labeled samples. It requires the *tf-idf* weight matrix to be computed for each class (i.e., Cylinder, Bell and Funnel) based on a train dataset (Algorithm 2). An unlabeled time series Q is assigned to the class C that maximizes the Cosine similarity (Algorithm 3):

$$label(Q) = \underset{C \epsilon CLASSES}{argmax} \left(D_{BOSSVS}(Q,C)\right)$$

4.1 The BOSS VS Classifier Algorithm

Prediction (Algorithm 2): The algorithm is based on 1-Nearest-Neighbor (1-NN) classification using the *tf-idf* weight matrix. The use of classes instead of the time series significantly reduces the computational complexity for classification. First, all classes (line 3) are iterated and the dot product between the *tf* vector of the query and the *tf-idf* weight matrix for each class is calculated (line 4). The class that maximizes the Cosine similarity is chosen as the query's class label (lines 5–6).

Fit (Algorithm 3): We use grid search in combination with cross-validation to estimate the parameters for *mean*, w, and l. The SFA alphabet size is fixed to $c = 4$, which is sufficient for a high classification accuracy on most datasets [12]. The *mean* normalization is a boolean parameter. If set to *true*, the first Fourier coefficient (DC coefficient) is dropped to obtain offset invariance. At the end of the training phase, we obtain *tf-idf* vectors for each class of the train dataset. This *tf-idf* matrix is the compressed representation of the train dataset and used as the model for classification.

Algorithm 3 iterates all $\sqrt{n}$ window lengths (line 5) and obtains the BOSS model for each window length (lines 7–8). Next, the *tf-idf* weight matrix is computed for each class based on the BOSS models of each time series and a concrete SFA word length (lines 10–11). By removing symbols from an SFA word, we can avoid recomputing the Fourier transform for smaller word lengths. Leave-one-out cross-validation is performed for each sample to predict the best class (lines 13–15). Finally, the best configuration is returned (line 18).

We choose the SFA word lengths from $\{4, 6, 8, 10, 12, 14, 16\}$, in total seven values. Mean normalization can be set to true or false. Searching for the optimal window length can be computationally expensive, as there are at most n windows for time series length n. To significantly reduce training times, we decided to train using only the $\sqrt{n}$ windows at equivalent distance in the interval $[10, n]$. As the BOSS model is robust regarding the choice of window lengths [12], this has no significant effect on the accuracy. In total the BOSS VS parameter space has the size: $2 \cdot 7 \cdot \sqrt{n} = O(\sqrt{n})$. The BOSS model has a parameter space of size $O(n)$, leading to higher train times. We will analyze the effects of these design decisions in Section 5.3.

4.2 Computational Complexity

Predict (Algorithm 2): The computational complexity for the classification is given by a 1-NN search over the $|CLASSES|$ classes using the Cosine similarity:

$$\begin{aligned} T(\mathit{BOSS\ VS\ Predict}) &= O(|CLASSES| \cdot T(BOSSVSDistance)) \\ &= O(|CLASSES| \cdot n) = O(n) \end{aligned} \tag{10}$$

The computational complexity is constant in the number of samples N.

Fit (Algorithm 3) The computational complexity of the train phase results from (a) building N histograms, one for each of N time series, and (b) building $|CLASSES|$ *tf-idf* vectors, one for each class C. This is done for $\sqrt{n}$ window lengths:

$$\begin{aligned} T(\mathit{BOSS\ VS\ Fit}) &= O(\sqrt{n} \cdot N \cdot [T(BOSS) + T(\mathit{BOSS\ VS\ Predict})]) \\ &= O(Nn^{\frac{3}{2}}\,|CLASSES|) = O(Nn^{\frac{3}{2}}) \end{aligned}$$

Note that this computational complexity is even lower than the test complexity of the 1-NN DTW classifier with $O(Nn^2)$ (Table 1).

45 UCR datasets [13]: *50words, Adiac, Beef, CBF, ChlorineConcentration, CinC_ECG_torso, Coffee, Cricket_X, Cricket_Y, Cricket_Z, DiatomSizeReduction, ECG200, ECGFiveDays, FaceAll, FaceFour, FacesUCR, Fish, Gun-Point, Haptics, InlineSkate, ItalyPowerDemand, Lighting2, Lighting7, MALLAT, MedicalImages, Motes, NonInvasiveFatalECG_Thorax1, NonInvasiveFatalECG_Thorax2, OliveOil, OSULeaf SonyAIBORobotSurface, SonyAIBORobotSurfaceII, StarlightCurves, SwedishLeaf, Symbols, synthetic_control, Trace, Two_Patterns, TwoLeadECG, uWaveGestureLibrary_X, uWaveGestureLibrary_Y, uWaveGestureLibrary_Z, wafer, WordsSynonyms, yoga*
46 Datasets from [25,26,14,16,5,6,20,13]: *ArrowHead, ARSim, BeetleFly, BirdChicken, Computers, DistalPhalanxOutlineAgeGroup, DistalPhalanxOutlineCorrect, DistalPhalanxTW, Earthquakes, ECG5000, ElectricDevices, FordA, FordB, Ham, HandOutlines, heartbeat (BIDMC), Herring, InsectWingbeatSound, LargeKitchenAppliances, Meat, MiddlePhalanxOutlineAgeGroup, MiddlePhalanxOutlineCorrect, MiddlePhalanxTW, Otoliths, Passgraph, PhalangesOutlinesCorrect, Phoneme, ProximalPhalanxOutlineAgeGroup, ProximalPhalanxOutlineCorrect, ProximalPhalanxTW, RefrigerationDevices, ScreenType, ShapeletSim, ShapesAll, shield, SmallKitchenAppliances, stig, Strawberry, ToeSegmentation1, ToeSegmentation2, UWaveGestureLibraryAll, wheat, Wine, WordSynonyms, Worms, WormsTwoClass*

Table 2 The 91 public time series datasets used in the experiments.

5 Experiments

We evaluated the BOSS VS model using two case studies for long time series and 91 public time series benchmark datasets (Table 2). Our web page contains all raw numbers and the C++ source codes [27]. The BOSS VS classifier was implemented in C++ using OpenMP and JAVA. All experiments were performed using the JAVA implementation on a shared memory machine running LINUX with 8 Quad Core AMD Opteron 8358 SE processors, and JAVA JDK x64 1.8. For all experiments the time series datasets were z-normalized prior to the experiments. All experiments consist of two phases: model building using the train dataset and testing the classification accuracy using the test dataset.

The 1-NN BOSS VS classifier is compared to state-of-the-art classifiers including nearest-neighbor based classifiers like 1-NN Fast Shapelets [6], Learning Shapelets (LS) [7], time series bag-of-features (TSBF) [21], 1-NN bag-of-patterns [16], 1-NN Shotgun [9], 1-NN ED or 1-NN DTW with the optimal warping window constraint [4], SAX-VSM [20], 1-NN BOSS classifier [12], ensemble techniques Elastic Ensemble (PROP) [10] and COTE [11], support vector machines (SVM) with a quadratic and cubic kernel, Naive Bayes classifier, and a tree based ensemble method (random forest). Where possible we used the implementations given by the authors, or python using the *sklearn*-framework for the SVM, Naive Bayes, and random forest benchmarks.

5.1 Case Studies for Long Time Series

We present two case studies for long time series that underline the high accuracy combined with the fast classification times offered by our BOSS VS model. These two datasets are two of the largest public datasets available.

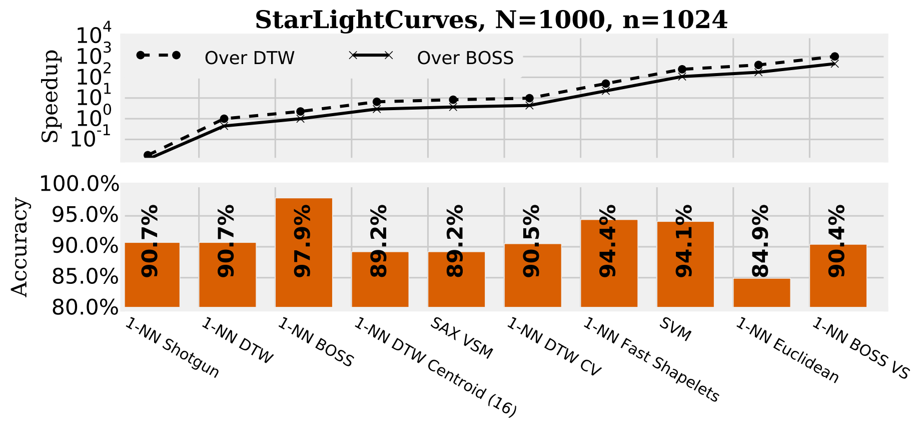


Fig. 7 Speedup and classification accuracy for starlight curves using 32 cores.

5.1.1 Starlight Curves

The StartLightCurves dataset [13] contains $N = 1000$ train and $N = 8236$ test samples each of length 1024. Figure 7 illustrates (from top to bottom):

1. The speedup of each classifier over 1-NN DTW (dotted line) and 1-NN BOSS (straight line),
2. The classification accuracy.

Training the classifiers takes from seconds (BOSS VS) up to several days (SAX-VSM, DTW centroid) using 32 cores. We have added a video to our website to illustrate differences in test times [27].

The 1-NN DTW, using the state-of-the-art implementation [4], takes 7 minutes on our 32 core machine for prediction. 1-NN DTW CV uses a warping window constraint set through cross validation and is two orders of magnitude slower than the 1-NN BOSS VS in prediction time with 0.4 seconds, and it takes roughly one hour to train using all cores. The 1-NN BOSS classifier takes three minutes for prediction, six minutes for training, and has the best accuracy of 97.9%, due to its invariances to noise, phase shifts, offsets, amplitudes and occlusions. The Euclidean distance based classifier is fast but the least accurate. Our 1-NN BOSS VS classifier has a competitive classification accuracy and takes only 0.4 seconds for prediction which is three orders of magnitude faster than 1-NN DTW, and 2.5 orders of magnitude faster than the 1-NN BOSS classifier.

This dataset underlines the utility of the 1-NN BOSS VS for long time series.

5.1.2 Personalized Medicine: Heartbeat BIDMC

The BIDMC Congestive Heart Failure Database[1] contains ECG recordings of 15 subjects, who suffer from severe congestive heart failures. The recordings contain noisy or extraneous data, when the recordings started before the machine was connected to the patient. ECG signals show a high level of redundancy due to

[1] http://www.physionet.org/physiobank/database/chfdb/ (2015)

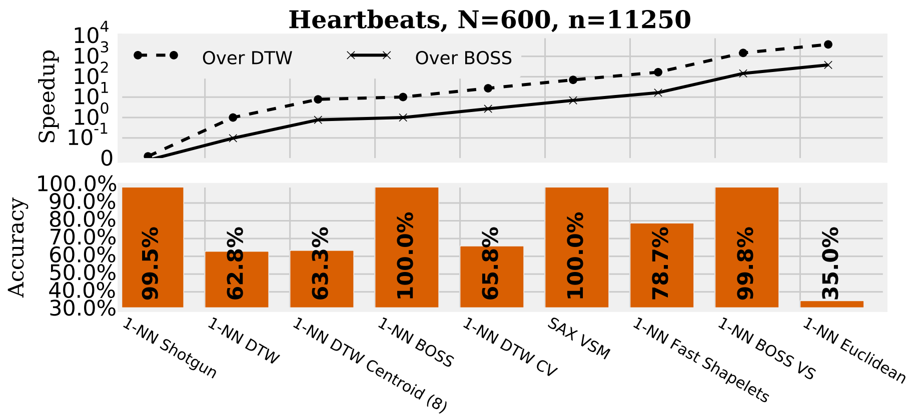


Fig. 8 Speedup and classification accuracy for ECG recordings using 32 cores. SVM was not benchmarked due to the different length of the train and test set. Fast Shapelets timed out after 6 days of training.

repetitive heartbeats but even a single patient can have multiple different heartbeats. The train/test split [28] contains $N = 600$ train and $N = 600$ test samples. The train and test splits have different lengths of $n = 3750$ and $n = 11250$.

Figure 8 shows that the 1-NN BOSS, the 1-NN BOSS VS and the 1-NN Shotgun classifiers offer the best classification accuracy. The other classifiers have a much lower classification accuracy. This is not surprising as the data is noisy and requires invariance to phase shifts in order to cope with the periodic ECG patterns. Train times vary from minutes (BOSS, BOSS VS) to days (SAX-VSM, Shotgun classifier, and DTW centroid) using 32 cores. 1-NN DTW has the highest test time with 30 minutes. Our 1-NN BOSS VS classifier offers a close to perfect accuracy with 99.8%, has a test time of 2s and a train time of 150s. The test time is three orders of magnitude lower than that of 1-NN DTW and two orders of magnitude lower than the 1-NN BOSS classifier. Even when combining the test and train times, the 1-NN BOSS VS is one order of magnitude faster than the 1700s 1-NN DTW classifier's test time.

This dataset underlines that common distance measures degenerate for long time series of with thousands of values. Our BOSS VS scores a close to perfect accuracy.

5.2 Classification Accuracy

All classifiers were evaluated using the same 91 public time series datasets (see Table 2):

- 84 datasets taken from the UCR time series classification archive [13].
- 7 datasets taken from other publications [25,26,14,16,5,6,20].

Each dataset provides a *train/test* split. There might be a confusion with the terms "*training*", "*validation*", and "*test*" sets used in supervised learning. The train split is used as the training and validation set. The test split is used as the

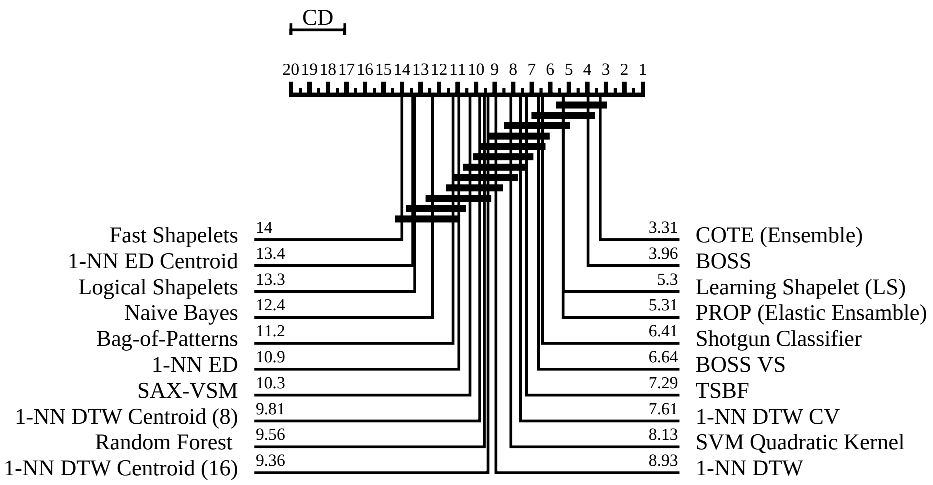


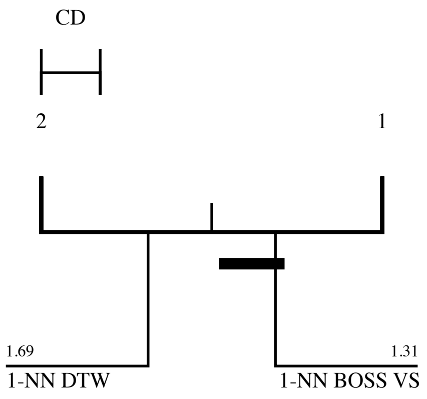
Fig. 9 Critical difference diagram for state-of-the-art classifiers on the 91 datasets. The best classifiers are to the right. Critical difference is 2.9. Ensemble classifiers like Elastic Ensemble (PROP) [10] or COTE [11] make use of dozens of classifiers to produce a label.

test set. By the use of these train/test splits, the results are comparable to previous publications. All our results are based on the *test accuracy* of the classifiers using the test split. We used the fixed length versions of each dataset, except for *stig* and *shield*.

Classification Accuracy

Figure 9 shows a critical difference diagram over the average ranks of the classifiers as introduced in [29]. The classifiers with the lowest (best) ranks are to the right. The group of classifiers that are not significantly different in their rankings are connected by a bar. The critical difference (CD) length is shown above the graph.

– The 1-NN BOSS classifier is in the group of the most accurate time series classifiers. This confirms our claims that structural similarity (bag of words) in combination with noise reduction (SFA) is important for time series similarity.
– The strength of the 1-NN BOSS VS classifier lies in its computational complexity rather than its high classification accuracy. It is in the group of the best classifiers: COTE (Ensemble), 1-NN BOSS, Learning Shapelets (LS), or Elastic Ensemble (PROP).
– The 1-NN DTW and 1-NN DTW CV classifiers are commonly used as benchmarks to compare to [14, 8, 10]. Both perform worse than our 1-NN BOSS VS.
– Most other classifiers perform significantly worse than 1-NN BOSS VS, including Euclidean distance based classifiers, Naive Bayes, the SAX-VSM classifier that builds on SAX and the vector space model, or Shapelet classifiers.
– The Elastic Ensemble (PROP) [10] is an ensemble of 11 distinct time series classifiers. The COTE [10] ensemble is based on 35 distinct time series classifiers. Their excellent accuracy comes at the cost of an enormous classification time, as they have to predict a label for each distinct classifier first.



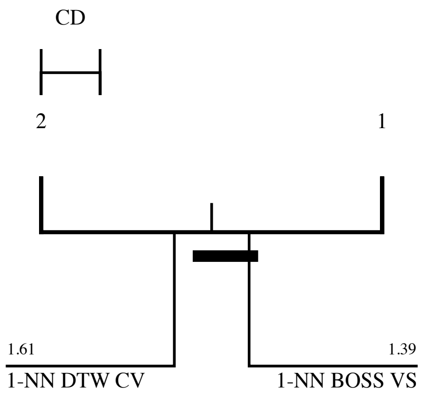


Fig. 10 Critical difference diagram for BOSS VS vs 1-NN DTW (left) and 1-NN DTW CV (right) on the 91 datasets. Critical distance is 0.17 for both.

Implications: The 1-NN BOSS is among the most accurate classifiers. The 1-NN BOSS VS classifier is extremely fast and provides better accuracy than the benchmark 1-NN DTW classifier. Ensemble classifiers like Elastic Ensemble (PROP) [10] or COTE [11] use large numbers of classifiers, and give high accuracy at the cost of an enormous computational complexity.

Is 1-NN BOSS VS significantly more accurate than 1-NN DTW with a full warping window?

The 1-NN DTW classifier is commonly used as the benchmark to compare to [8, 25, 14]. Thus, we test if the 1-NN BOSS VS classifier performs significantly better than the 1-NN DTW or 1-NN DTW CV classifiers on the 91 time series datasets (Figure 10).

Critical difference diagram: Figure 10 shows the critical difference diagrams. The critical difference is 0.17 in both cases. The 1-NN BOSS VS is significantly more accurate than 1-NN DTW with a difference in ranks of $1.69 - 1.31 > 0.17$ and $1.61 - 1.39 > 0.17$ and:

- 1-NN BOSS VS has 62 wins, 3 draw, and 26 losses over 1-NN DTW with a full warping window.
- 1-NN BOSS VS has 54 wins, 4 draw, 33 losses over 1-NN DTW CV (see the excel sheet with detailed results [27]).

Wilcoxon signed rank test: To validate the results, we performed a Wilcoxon signed rank test to check if 1-NN BOSS VS is significantly different from 1-NN DTW and 1-NN DTW CV. With a p-value of 0.025 for 1-NN DTW VS and 0.000086 for 1-NN DTW we can reject the null-hypothesis that any of the two 1-NN DTW classifiers is from the same distribution.

Implications: The 1-NN BOSS VS classifier is significantly more accurate than 1-NN DTW and 1-NN DTW CV, and orders of magnitude faster.

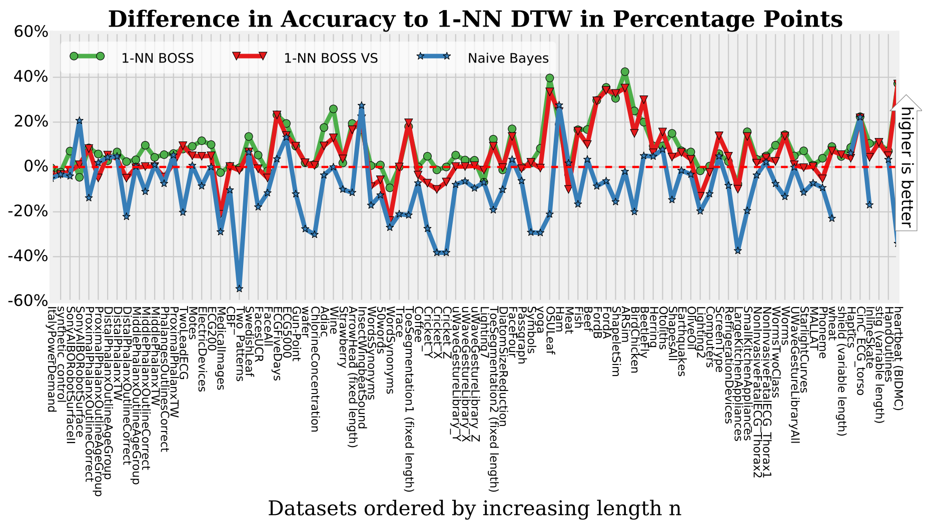


Fig. 11 Difference in accuracy on 91 time series datasets using 1-NN DTW as baseline. Datasets are ordered by increasing time series lengths as in Figure 7.

Classification Accuracy for Long Time Series

The 1-NN BOSS VS classifier is optimized for classification speed. Its utility would be limited, if it only scored well for small datasets but degenerates for long or large time series datasets. This is not the case. Figure 11 shows the differences in classification accuracy using the 1-NN DTW classifier as the baseline.

– For comparison, we plot the Naive Bayes classifier, which is a very fast classifier. Its accuracy is worse than that of the 1-NN DTW for most datasets and it is not applicable to variable length datasets such as *shield* and *stig*. The accuracy drops by up to 50% (Two_Patterns) compared to 1-NN DTW, or 70% compared to 1-NN BOSS (heartbeat BIDMC).
– The 1-NN BOSS classifier shows the highest accuracy and is better than 1-NN DTW for most datasets independent of the length of the time series.
– The 1-NN BOSS VS classifier performs better than 1-NN DTW for most of the long time series datasets to the right of the plot. The accuracy drops by just a few percentage points for most datasets compared to 1-NN BOSS.

Implications: The 1-NN BOSS VS classifier is optimized for classification times but also provides a high accuracy for long time series datasets. The accuracy drops by just a few percentage points for most datasets and it is orders of magnitude faster.

Pairwise comparison of wall-clock times

Figure 12 shows the wall-clock times of the four state-of-the-art classifiers in a pairwise comparison to our 1-NN BOSS VS classifier. We omit the PROP or COTE classifiers, as these cannot be faster than the 1-NN DTW or 1-NN DTW CV classifiers, that both use. Again our BOSS VS classifier significantly outperforms

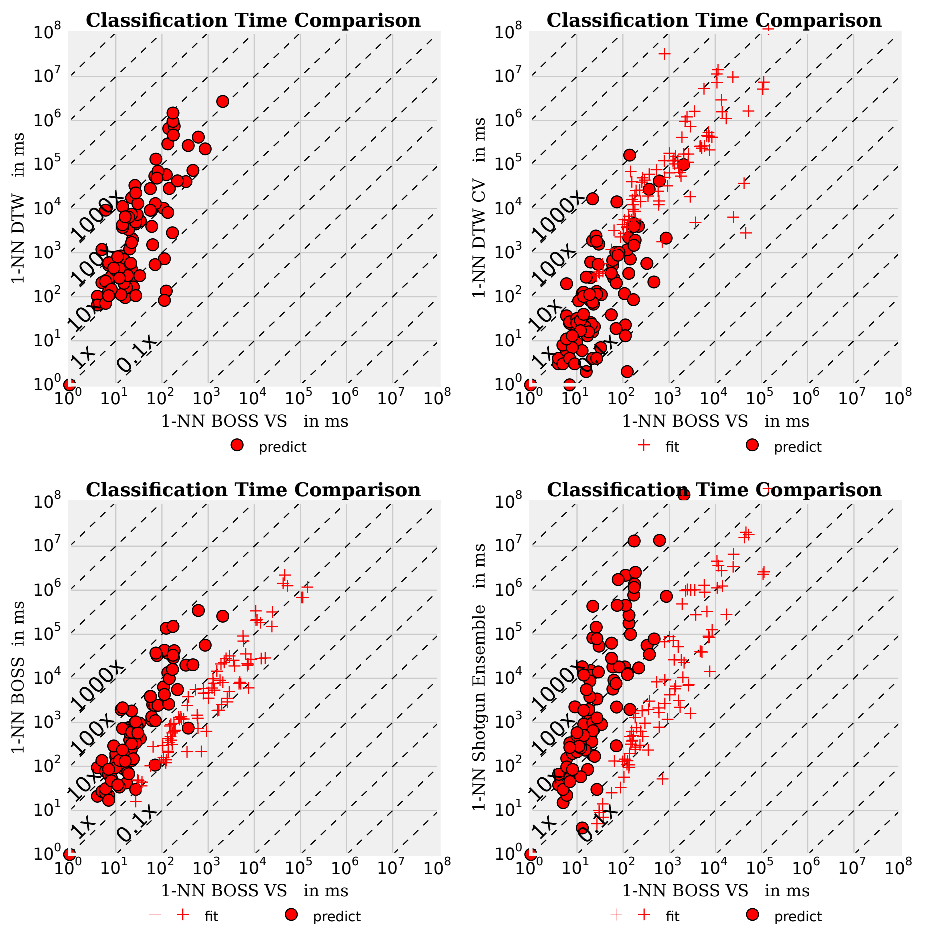


Fig. 12 Pairwise comparison of wall-clock times of the four most accurate classification algorithms compared to 1-NN BOSS VS.

the other classifiers by up to five orders of magnitude in terms of classification times.

- The 1-NN BOSS VS classifier is significantly faster than the 1-NN BOSS classifier in terms of train and test times.
- It seems to have similar train times to the 1-NN Shotgun classifier. When looking at the raw data, the 1-NN Shotgun classifier trains faster for very small datasets and is orders of magnitude slower for moderate to large sized datasets.
- 1-NN DTW CV requires a training phase to find the warping window constraint, resulting in reduced test times when compared to 1-NN DTW. When looking at the raw data, the 1-NN DTW CV classifier has similar test times for the small datasets and is orders of magnitude slower than 1-NN BOSS VS for moderate to large sized datasets (compare total time to completion in Figure 12).

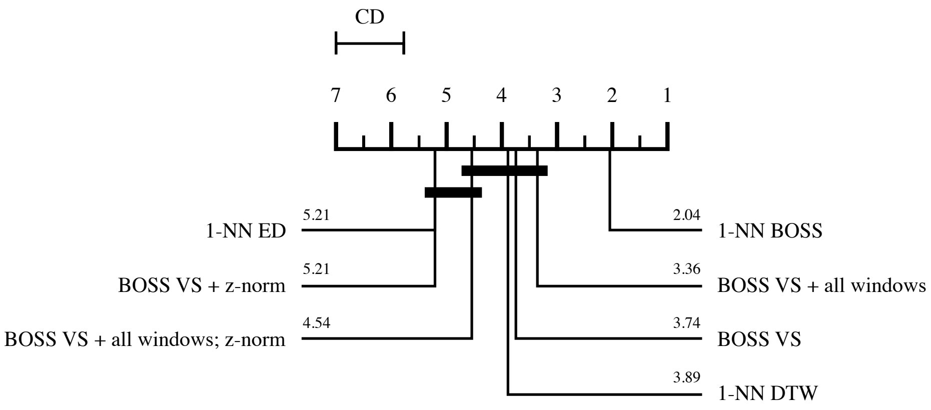


Fig. 13 Critical difference diagram for the design decisions made using the 45 UCR datasets. The best classifiers are to the right. Critical difference (CD) is 1.22.

– 1-NN DTW is one to four orders of magnitude slower in terms of test times than our 1-NN BOSS VS, and is never significantly faster.

Implications: The 1-NN BOSS VS is among the most accurate classifiers. It is (a) multiple orders of magnitude faster than the 1-NN BOSS classifier, 1-NN DTW, 1-NN Shotgun classifier, and state of the art, and (b) it is significantly more accurate than 1-NN DTW with or without a warping window constraint.

5.3 Impact of Design Decisions

We use all 45 UCR datasets [13] to test the impact of the design decisions of the BOSS VS model:

1. Testing a *subset* of $\sqrt{n}$ windows for training as opposed to using *all windows (BOSS VS + all windows)*.
2. *Mean normalization* as a parameter as opposed to always applying z-normalization *(BOSS VS + z-norm)* to all windows.

BOSS VS was designed to use a subset of windows and mean normalization as a parameter. Figure 13 shows that the mean normalization as a parameter performs significantly better than always norming the data (*"+z-norm"*). The use of $\sqrt{n}$ windows performs worse than the use of all windows (*"+all windows"*).

Implications: The BOSS VS model is (a) based on the use of a subset of windows for training and (b) mean normalization as a parameter. While the mean parameter improves the accuracy, the use of a subset of windows reduces accuracy. However, it is crucial for a low train time which is reduced to $O(Nn^{\frac{3}{2}})$.

5.4 When would we recommend using our BOSS VS model over our BOSS model?

BOSS VS equals and outperforms previous methods on 12 datasets: *BeetleFly, DistalPhalanxOutlineAgeGroup, ECGFiveDays, ShapeletSim, shield, Strawberry, Symbols,*

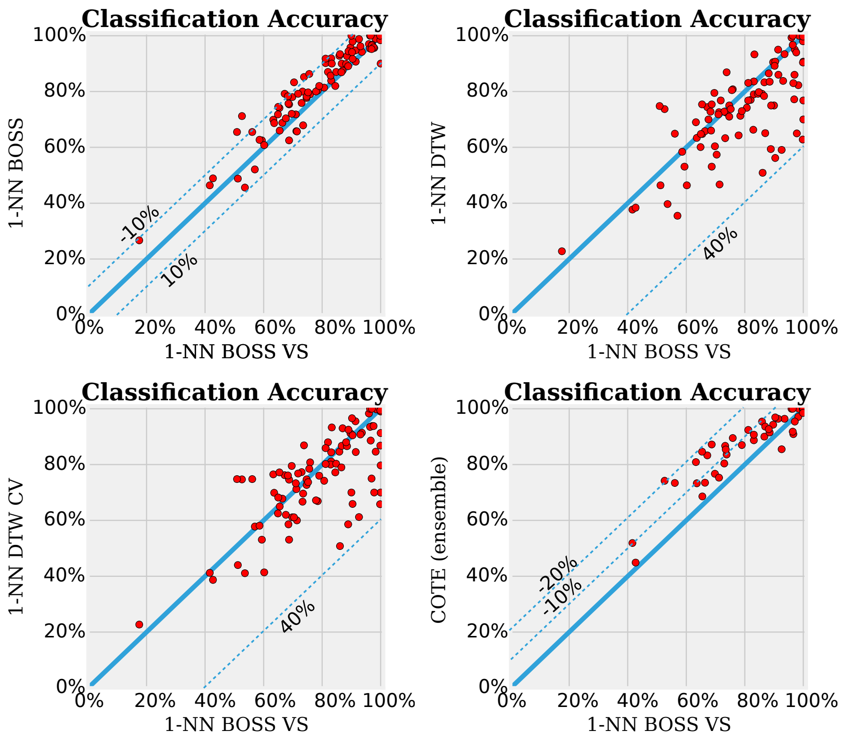


Fig. 14 Pairwise comparison of BOSS VS and state of the art on all 91 datasets.

ToeSegmentation1, Trace, TwoLeadECG, and wafer. It is state of the art for these datasets. For all datasets it is orders of magnitude faster than state of the art. It is well suited for repetitive (ECG/EEG signals, human motions), long time series or outline/shape-based datasets.

For the remaining datasets there is a trade-off between accuracy, energy consumption and processor time of a classifier. If processor time and energy consumption are not an issue, the best classifiers to date are the 1-NN BOSS classifier [12], Learning Shapelet (LS) [7], or the two ensemble techniques Elastic Ensemble (PROP) [10] and COTE [11]. The 1-NN BOSS classifier offers a very high accuracy for long or noisy time series datasets, as it compares two time series based on a structural level. It has a high computational complexity for large time series datasets. The 1-NN BOSS VS classifier has a low computational complexity with a sightly decreased accuracy. Figure 14 shows a pairwise comparison of classification accuracies. Each point represents the accuracy on one dataset. A point below the straight blue line indicates that BOSS VS is more accurate. The difference to the BOSS classifier (left) in accuracy is negligibly small with -10 to 10 percentage points for most datasets. Its accuracy drops by up to 20 percentage points when compared to the COTE ensemble, which ensembles 35 classifiers and as such is very slow. Meanwhile the BOSS VS is up to 40 percentage points better than 1-NN DTW (center) or 1-NN DTW CV classifiers (right).

Implications: The 1-NN BOSS VS advances state of the art on 12 datasets. It is an ideal choice for mining repetitive, long or large amounts of time series, where it is orders of magnitude faster than state of the art with a sightly decreased accuracy. In some real-time data analytics or embedded computing use cases it is the best viable solution to date.

6 Conclusion

In the context of mining large datasets and real-time analytics there is a need for time series classification algorithms with (a) a low test time to allow for mining large datasets, (b) tolerance to noise to provide high classification accuracy and (c) a moderate train time to allow for frequent model updates. This work introduces BOSS in Vector Space (BOSS VS) that is multiple orders of magnitude faster than state-of-the-art classifiers and has a high classification accuracy due to invariances to noise, phase shifts, offsets, amplitudes and occlusions. An exhaustive evaluation using 91 public time series benchmark datasets shows that it is (a) significantly more accurate than 1-NN DTW with or without a warping window, and (b) competitive to state-of-the-art classifiers with regards to classification accuracy while being orders of magnitude faster in terms of classification times.

Acknowledgment

The author would like to thank Claudia Eichert-Schäfer, Florian Schintke, the anonymous reviewers and the owners of the datasets.

Compliance with Ethical Standards

Funding: This project was motived and partially funded by the German Federal Ministry of Education and Research through the project "Berlin Big Data Center (BBDC)", Funding mark: 01IS14013A.

Conflict of Interest: The author P. Schäfer received research grants from this project.

References

1. Urbanski, J., Weber, M.: Big Data im Praxiseinsatz–Szenarien, Beispiele, Effekte. `http://www.bitkom.org/files/documents/BITKOM_LF_big_data_2012_online(1).pdf` (2012)
2. Jerzak, Z., Ziekow, H.: The DEBS 2014 Grand Challenge. In: Proceedings of the 2014 ACM International Conference on Distributed Event-based Systems, ACM (2014) 266–269
3. Mutschler, C., Ziekow, H., Jerzak, Z.: The DEBS 2013 grand challenge. In: Proceedings of the 2013 ACM International Conference on Distributed Event-based Systems, ACM (2013) 289–294
4. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 2012 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2012) 262–270

5. Mueen, A., Keogh, E.J., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 2011 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2011) 1154–1162
6. Rakthanmanon, T., Keogh, E.: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM (2013)
7. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proceedings of the 2014 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2014) 392–401
8. Bagnall, A., Lines, J.: An Experimental Evaluation of Nearest Neighbour Time Series Classification. arXiv preprint arXiv:1406.4757 (2014)
9. Schäfer, P.: Towards Time Series Classification without Human Preprocessing. In: Machine Learning and Data Mining in Pattern Recognition. Springer (2014) 228–242
10. Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery **29**(3) (2014) 565–592
11. Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. IEEE Transactions on Knowledge and Data Engineering **27**(9) (2015) 2522–2535
12. Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29**(6) (2015) 1505–1530
13. UCR Time Series Classification/Clustering Homepage. `http://www.cs.ucr.edu/~eamonn/time_series_data` (2014)
14. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representationsx and distance measures. In: Proceedings of the VLDB Endowment. Number 2, VLDB Endowment (2008) 1542–1552
15. Schäfer, P., Högqvist, M.: SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: Proceedings of the 2012 International Conference on Extending Database Technology, ACM (2012) 516–527
16. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39**(2) (2012) 287–315
17. Lin, J., Keogh, E.J., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining and knowledge discovery **15**(2) (2007) 107–144
18. Esling, P., Agon, C.: Time-series data mining. ACM Computing Surveys **45**(1) (2012) 12:1–12:34
19. Petitjean, F., Forestier, G., Webb, G.I., Nicholson, A.E., Chen, Y., Keogh, E.: Dynamic Time Warping averaging of time series allows faster and more accurate classification. In: Proceedings of the 2014 IEEE International Conference on Data Mining, IEEE (2014) 470–479
20. Senin, P., Malinchik, S.: SAX-VSM: Interpretable time series classification using SAX and vector space model. In: Proceedings of the 2013 IEEE International Conference on Data Mining, IEEE (2013) 1175–1180
21. Baydogan, M.G., Runger, G., Tuv, E.: A bag-of-features framework to classify time series. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(11) (2013) 2796–2802
22. Kumar, N., Lolla, V.N., Keogh, E.J., Lonardi, S., Ratanamahatana, C.A.: Time-series bitmaps: a practical visualization tool for working with large time series databases. In: Proceedings of the 2005 SIAM International Conference on Data Mining, SIAM (2005) 531–535
23. Aucouturier, J.J., Defreville, B., Pachet, F.: The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. The Journal of the Acoustical Society of America **122**(2) (2007) 881–891
24. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM, **18**(11) (1975) 613–620
25. Bagnall, A., Davis, L.M., Hills, J., Lines, J.: Transformation Based Ensembles for Time Series Classification. In: Proceedings of the 2012 SIAM International Conference on Data Mining. Volume 12., SIAM (2012) 307–318
26. Batista, G., Wang, X., Keogh, E.J.: A Complexity-Invariant Distance Measure for Time Series. In: Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM (2011) 699–710
27. The BOSS in Vector Space Results. `http://www.zib.de/patrick.schaefer/bossVS/` (2015)

28. Hu, B., Chen, Y., Keogh, E.: Time Series Classification under More Realistic Assumptions. In: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM (2013) 578–586
29. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7** (2006) 1–30