Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

PATRICK SCHÄFER

Bag-Of-SFA-Symbols in Vector Space (BOSS VS)

Preprint

ZIB Report 15-30 (May 2015)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Bag-Of-SFA-Symbols in Vector Space (BOSS VS)

Patrick Schäfer

8th May 2015

Abstract

Time series classification mimics the human understanding of similarity. When it comes to larger datasets, state of the art classifiers reach their limits in terms of unreasonable 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 and has several shortcomings: it has a quadratic time and it degenerates in the presence of noise. To reduce the computational complexity lower bounding techniques or recently a nearest centroid classifier have been introduced. Still, execution times to classify moderately sized datasets on a single core are in the order of hours. We present our Bag-Of-SFA-Symbols in Vector Space (BOSS VS) classifier that is robust and accurate due to invariance to noise, phase shifts, offsets, amplitudes and occlusions. We show that it is as accurate while being multiple orders of magnitude faster than state of the art classifiers. Using the BOSS VS allows for mining massive time series datasets and real-time analytics.

1 Introduction

Time series are recorded from sensors and other input sources over time. Application domains include personalized medicine [1], human walking motions [2], anthropology [3], security [4], historical documents [3], astronomy [5], and spectrographs [3], for example. While a human has an intuitive understanding of the similarity of two time series, this task becomes very hard for a computer especially in the presence of noise. Typical similarity measures that resemble the human understanding of similarity are the Euclidean Distance (ED), Dynamic Time Warping (DTW) [6, 7] or most recently the noise robust Bag-of-SFA-Symbols (BOSS) [8].

Classification describes the task of finding a label to an unlabeled time series. Therefore a classifier has to produce a model from a set of labeled time series that takes this unlabeled time series as input and outputs its label. This process is referred to as testing or predicting. The task of building the model is referred to as training.

Two trends have evolved in data analytics: the availability of large datasets and machine learning on streaming data (aka real-time analytics). The first requires for an algorithm to handle large amounts of data in reasonable time. The latter is troublesome as the underlying source generating the data might change over time. This requires for a classifier to evolve its classification model by incremental updates or frequently rebuilding its model.

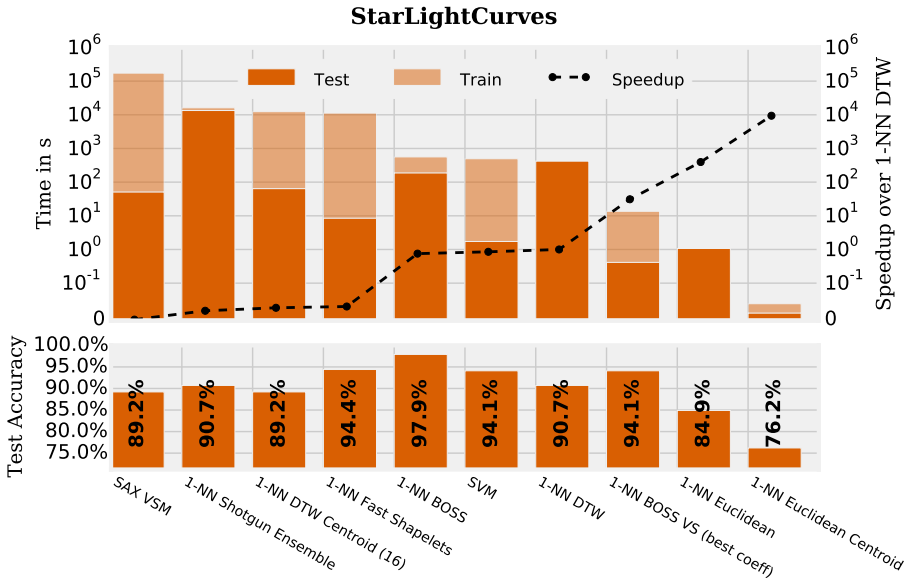

Figure 1: The accumulated train and test times and the accuracy on the StalightCurves dataset.

1-Nearest Neighbor (1-NN) classifiers are among the oldest classifiers, yet form the basis for most time series classification algorithms [9, 4, 6, 8, 10, 11]. Their idea is simple: to obtain the label to a query, one has to find the sample within a train dataset that minimizes a similarity measure, and assign its label to the query. These classifiers do not require any training, though the similarity measure might require training, like DTW with a warping window. The authors [11] argue that 1-NN DTW is among the best classifiers by performing an exhaustive study using techniques like C4.5, Naive Bayes, the Bayesian Network, SVM, or Rotation Forests. They conclude that "[...] a new algorithm is only of interest in terms of accuracy if it can significantly outperform 1-NN DTW [...]". The $O(Nn^2)$ complexity of 1-NN DTW has been reduced by the use of lower bounding techniques [6] and the introduction of a 1-NN DTW centroid classifier [12], for time series length n and size of the dataset N. 1-NN DTW is just one representative example. We believe that the computational complexity of most state of the art classifiers is excessive even when it comes to moderately sized datasets of $N \geq 10^3$ time series. Apparat from the obvious fact that a classifier has to provide fast testing times, the need for reasonable training times emerges with real-time analytics. We claim based on the statement in [11]:

> "A new algorithm is of interest if it can significantly outperform 1-NN DTW in terms of *accuracy* OR *classification times*".

The StartLightCurve dataset is a moderately sized dataset containing 1000 train samples and 8236 test samples each of length 1024. Figure 1 illustrates the test times, the train times, and the classification accuracy of state of the art time series classifiers. The dashed line represents the speedup over 1-NN DTW. Training the classifiers takes from seconds (BOSS VS) up to several days (SAX

VSM, DTW centroid). The 1-NN DTW takes more than 200 minutes on a single core machine and 7 minutes on a 32 core machine to classify the 8236 queries in parallel with an accuracy of 90.7%. The Euclidean Distance based classifiers are the fastest but show the most inaccurate results with 84.9% and 76.2%, as these provide no invariance to noise, horizontal shifts, warping, etc. The Bag-of-SFA-Symbols (BOSS) [8] classifier offers the best classification accuracy of 97.9% at the cost of high test and training times, due to its invariances to noise, phase shifts, offsets, amplitudes and occlusions. Support Vector Machines (SVMs) are very competitive with an accuracy of 94.1%, but have a high $O(N^2n)$ training complexity and cannot easily cope with variable length time series. We added a video to our website to illustrate differences in test times [13].

Our Bag-of-SFA-Symbols in Vector Space (BOSS VS) classifier offers the third best classification accuracy with 94.1%. It allows for *fast* testing combined with low training times. These are orders of magnitude lower than that of the other classifiers. The testing time is even lower than that of the 1-NN ED classifier. The BOSS VS has a test complexity of $O(n)$ that allows for the classification of massive time series datasets. Its moderate training complexity of $O(Nn^{\frac{3}{2}})$ allows for frequent model updates such as in real-time predictive analytics. Our contributions are as follows:

- We present the background of the Symbolic Fourier Approximation (SFA) and the BOSS model in Section 2.
- We present the BOSS VS classifier for fast and accurate time series classification in Section 3.3.
- We present two case studies for noisy time series that underline the low training, low testing times, and high classification accuracy of the BOSS VS in Section 4.1.
- We show the scalability to 2 billion values equal to 15 GB of the BOSS VS classifier in Section 4.2. The BOSS VS classifier is multiple orders of magnitude faster than other state of the art classifiers and has a moderate training complexity.
- The BOSS VS is very competitive with regards to classification accuracy on the established UCR benchmark datasets (Section 4.3).

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:

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

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:*

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

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) [14] 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 reduction:** 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 and thereby the bandwidth of the low pass filter.
- **String representation:** SFA uses quantization and character strings. This transformation allows for string matching algorithms like hashing or the bag of words to be applied. The size of the alphabet determines the degree of quantization and has an additional noise reducing effect, but it might lead to information loss.

2.3 Symbolic Fourier Approximation (SFA)

SFA is composed of two operations (Figure 2):

1. **Approximation** using the Discrete Fourier Transform (DFT) configured by the **SFA word length** $l \epsilon \mathbb{N}$ and
2. **Quantization** using a technique called Multiple Coefficient Binning (MCB) configured by the **alphabet size** $c \epsilon \mathbb{N}$ equal to the number of bins.

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

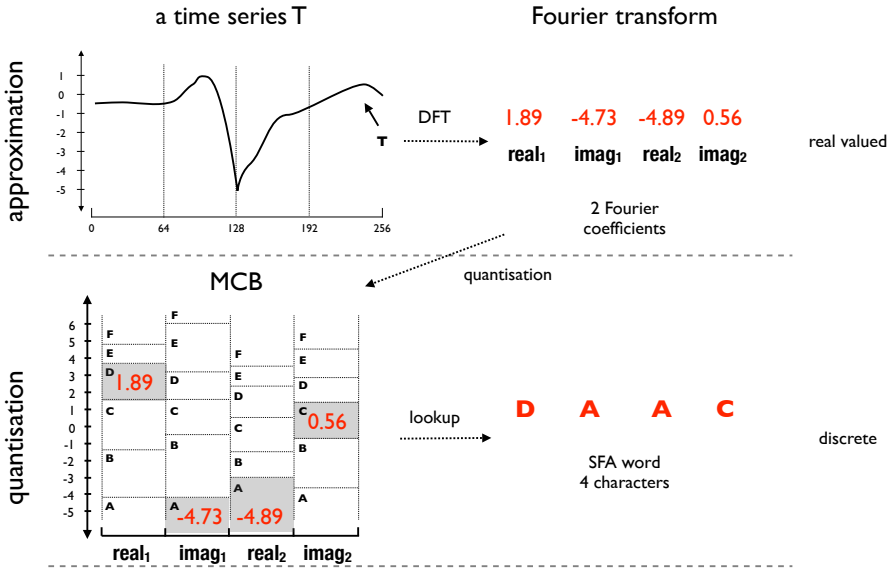

Figure 2: SFA: A time series is (a) approximated (low pass filtered) using DFT and (b) quantized using MCB resulting in the SFA word $DAAC$. [8, 14]

Modification to SFA In [14] we proposed to use the first $l \ll n$ Fourier coefficients. We noticed that for larger datasets, it is advantageous to use the l Fourier coefficients which have the largest area under the curve in the MCB histograms. This follows the assumption that if the value range is larger, the Fourier coefficient is more important. We study the effects in our experiments.

2.4 The Bag of SFA Symbols (BOSS)

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

- it is fast, as hashing can be used to measure the similarity of SFA words,
- it applies noise reduction,
- it provides invariance to phase shifts, offsets, amplitudes and occlusions.

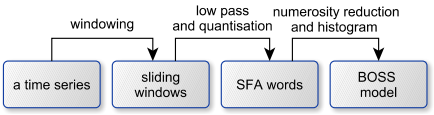

Figure 4: The BOSS workflow.[8]

2.5 The BOSS Model

The BOSS model (Figure 4) has four parameters:

- **the window length** $w \epsilon \mathbb{N}$: represents the length of the substructures.

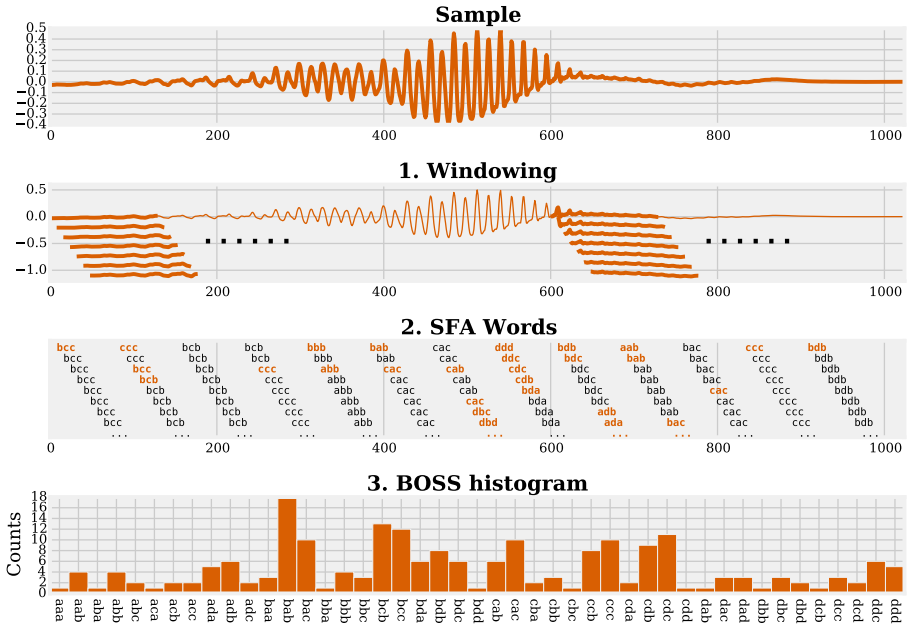

Figure 3: The BOSS model is extracted from a sample time series using word length 3 and 4 symbols (a-d). The black SFA words are skipped due to numerosity reduction.[8]

- **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.

First, sliding windows of length w are extracted from a time series. Intuitively w should roughly represent the length of the substructures within the time series. Next, each sliding window is normalized to have a standard deviation of 1 to obtain amplitude invariance. The parameter *mean* is treated as a parameter of our mode and determines if the mean value is to be subtracted from each sliding window to obtain offset invariance. 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 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 a signal the SFA words of two neighboring sliding windows are very likely to be identical. To avoid outweighing stable sections of a signal, *numerosity reduction* is commonly is applied [15, 16]. That is, the first occurrence of an SFA word is registered and all duplicates are ignored until a new SFA word is discovered. In Figure 3 the first SFA words are identical:

$$S \quad = \quad \textbf{bcc}\ bcc\ bcc\ bcc\ bcc\ bcc\ bcc\ bcc\ \textbf{ccc}\ ccc\ \textbf{bcc}\ \textbf{bcb}\ bcb\ bcb\ bcb\ \ldots$$

If numerosity reduction is applied to S this leads to:

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

```
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 counts
(6)      lastWord = word
(7)  return boss
```

Algorithm 1: The BOSS transformation.

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 2. *Bag-Of-SFA-Symbols (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 $B(t) : \; \Sigma^l \rightarrow \mathbb{N}$ 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.*

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 normalization is obtained by dropping the first Fourier coefficient in each SFA word. Finally, a new SFA word is added to the histogram (line 5), if two subsequent SFA words are different (line 4, numerosity reduction).

2.6 Related Work

Classical data mining algorithms like SVMs, decision trees, rotation or random forests have been used in the context of time series [17]. However, these did not perform better than the 1-NN DTW classifier [6], which is commonly used as the benchmark to compare to [11]. Its computational complexity is quadratic in the length n of the time series and linear in the size N of the training dataset: $O(Nn^2)$. Much effort has been spent to reduce the computational complexity by the use of lower bounding techniques [6] and most recently a nearest centroid classifier has been presented [12].

Shapelet classifiers [4, 18] extract representative variable-length subsequences (called *shapelets*) from a time series. A decision tree is built using these shapelets within the nodes of the tree. A distance threshold is used for branching. These classifiers have a high computational complexity for training of $O(N^2n^3)$ [4] to $O(Nn^2)$ [18]. The 1-NN Shotgun Classifier [10] divides the query into disjoint subsequences and slides each query window over the sample to find the position

that minimizes the Euclidean distance. Both, Shapelet and the Shotgun Classifiers are based on the Euclidean Distance and are therefore sensitive to noisy data.

In our previous work [8] we underlined the importance of the tolerance to noise for the time series classification task and showed that the BOSS model performs better than state of the art in terms of classification accuracy. In this work, we significantly reduce the computational complexity of the BOSS model to support the classification of large time series datasets. The BOSS model is based on the Symbolic Fourier Approximation (SFA) that has been introduced in [14] in the context of similarity search on massive time series datasets using the SFA trie. SFA is a symbolic representation of time series like Symbolic Aggregate approXimation (SAX) [16]. Unlike *SAX,* which uses mean values (PAA) to approximate a time series, SFA uses DFT coefficients. Both, have a noise canceling 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 adapted by choosing an arbitrary subset of Fourier coefficients without recalculating the DFT of a time series. Maximizing the train accuracy while increasing the number of Fourier coefficients is the core idea of the presented algorithms. Dropping the rear mean values of a SAX word is equal to dropping the rear part of a time series. To avoid this, one has to recalculate all SAX transformations each time we choose to represent a time series by a different SAX word length.

The *bag-of-patterns* (BOP) model [15] is the closest to our work. 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* [19] extends BOP by the use of the *tf-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. SAX-VSM has a huge parameter space and has to recalculate all SAX coefficients for each new set of parameters, as mentioned before. This results in an unreasonable high training time for even moderately sized datasets. In contrast our 1-NN BOSS VS uses SFA [14] that allows to adapt the length of the SFA words on the fly and has a much smaller parameter space (smaller training times). Furthermore, it uses the offset invariance as a model parameter.

3 The BOSS in Vector Space

The vector space model has been presented in information retrieval for representing text documents as vectors of keywords. Vector operations like cosine similarity are used to compare the similarity of documents. In the vector space model as proposed in [20] the *term frequency inverse document frequency* (*tf-idf*) model is used. The *tf-idf* measure is used to weigh each term frequency in the vector to give a higher weight to representative words of a class. In our model the *term* is an SFA word and a *document* is equal to a time series. We use an approach presented in [19] 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:

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

with $B_T(t)$ being the BOSS histogram which represents the raw frequency of an SFA word 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:

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

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

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

The *idf* for an SFA word represents the total number of classes divided by the number of classes this SFA word occurs in.

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

$$tfidf(t,C) = tf(t,C) \cdot idf(t,C) \quad (6)$$

$$= (1+\log(\sum_{T \epsilon C} B_T(t))) \cdot \log \frac{|C|}{|\{C\,|T \epsilon C \wedge B_T(t)>0\}|} \quad (7)$$

High 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.

The similarity of a *tf* vector Q to an *idf* class vector C is then computed using the Cosine similarity metric:

$$sim(Q,C) = \frac{\overrightarrow{Q} \cdot \overrightarrow{C}}{\left\|\overrightarrow{Q}\right\| \cdot \left\|\overrightarrow{C}\right\|} = \frac{\sum_{t \epsilon Q} tf(t,Q) \cdot tfidf(t,C)}{\sqrt{\sum_{t \epsilon Q}(tf(t,Q))^2}\sqrt{\sum_{t \epsilon C}(tfidf(t,C))^2}} \quad (8)$$

3.1 Intuition behind BOSS VS

The BOSS VS has several advantages compared to other approaches:

1. It applies noise reduction by using the SFA representation.
2. It provides invariance to phase shifts, offsets, amplitudes and occlusions by discarding the original ordering of the SFA words and normalization.
3. It highlights characteristic SFA words by the use of the *tf-idf* weight matrix. SFA words that occur frequently across all classes are given a lower *idf* weight. This has an additional noise reducing effect.
4. It uses a compact representation of classes instead of time series and thereby minimizes the influence of erroneous and extraneous data within a single time series, and it significantly reduces the computational complexity.

Properties (1) and (2) were presented with the BOSS model as presented in [8]. The BOSS VS adds properties (3) and (4).

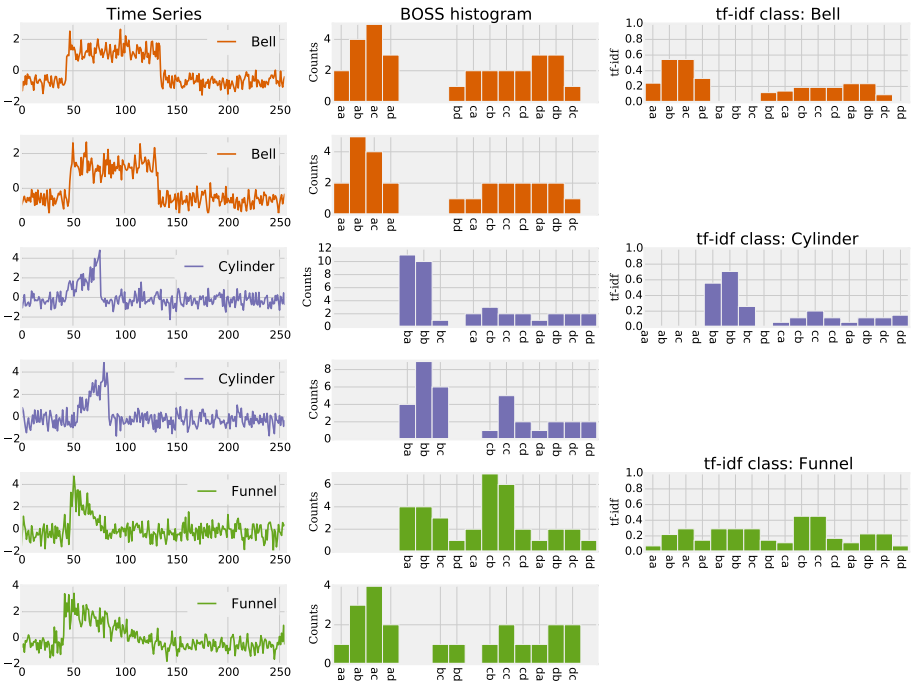

Figure 5: The BOSS VS: first, the SFA word frequencies are stored in the BOSS histograms from each time series. Finally, the *tf-idf* vectors are constructed from the BOSS histograms for each class.

3.2 Training

Figure 5 illustrates the training of the BOSS VS. Each time series is transformed to its BOSS histogram. Based on all BOSS histograms the *tf-idf* matrix is constructed. It contains one *tf-idf* vector for each class (i.e.: Cylinder, Bell, Funnel). Training the BOSS model aims at finding the parameters that maximize the accuracy on a train dataset:

- **the window length** $w \epsilon \mathbb{N}$.
- **mean normalization** $mean \epsilon [true, false]$.
- the **SFA word length** $l \epsilon \mathbb{N}$ **and alphabet size** $c \epsilon \mathbb{N}$.

We use grid search in combination with cross-validation to optimize the parameters for *mean*, w, l, and use a fixed alphabet size of $c = 4$. It is possible to use different alphabet sizes, but this would increase execution times. We empirically observed that a constant alphabet size of 4 was sufficient for a high classification accuracy.

The mean normalization is a boolean parameter, and increases the complexity by a constant factor of 2 when both *true* and *false* are tested. We choose the SFA word lengths from $\{4, 6, 8, 10, 12, 14, 16\}$. In total these are 7 values. 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 chose to train using only the $\sqrt{n}$ windows at equidistance in

```
(int,int,int,bool) fit(samples,labels,mean,windowLengths)
( 1)  maxL=16, c=4
( 2)  for w in windowLengths // search window lengths
( 3)    bags = BOSSTransform(samples,w,maxL,c,mean) // obtain the bags
( 4)    bestCor=0, bestL=0, bestW=0
( 5)    for l in {4,6..maxL}    // search word lengths
( 6)      tfIdfs = calcTfIdf(bags, l) // tf-idf matrix
( 7)      correct=0
( 8)      for qId in 1..len(samples) // leave-one-out
( 9)        best = predict(qId, bags[qId], tfIdfs)
(10)        if best has correct label then correct++
(11)      if correct > bestCor  // keep best
(12)        (bestCor, bestL, bestW) = (correct, l, w)
(13)  return (bestCor, bestL, bestW, mean) // return best parameters
```

Algorithm 2: Fit: Train the parameters using leave-one-out cross validation.

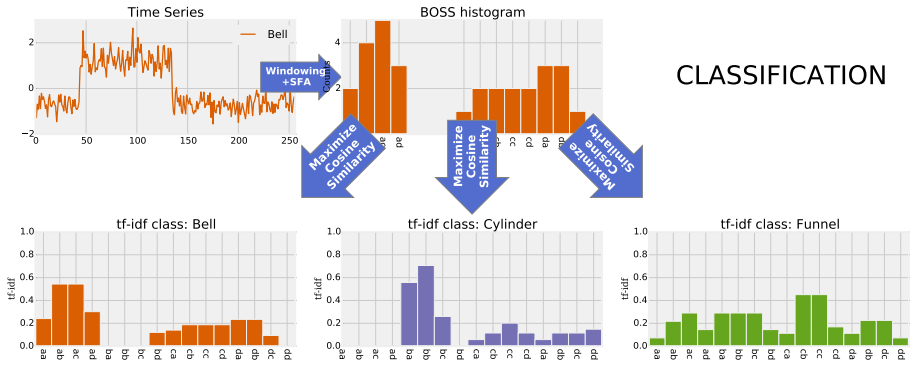

Figure 6: Classification of an unlabeled query using the BOSS VS: first the query is transformed to its BOSS histogram and *tf* vector and finally the Cosine similarity is maximized.

the interval $[10, n]$. In total this parameter space has the size: $2 \cdot 7 \cdot \sqrt{n}$. We will analyze the effects in our experiments.

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.

BOSS VS Fit (Algorithm 2) The algorithm iterates all $\sqrt{n}$ window lengths (line 2) and obtains the BOSS model for each window length (line 3). 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 5–6). Leave-one-out cross-validation is performed for each sample to predict the best class (lines 8–10). Finally, the best configuration is returned (line 13). The *mean* normalization parameter is a Boolean parameter to drop the DC Fourier coefficient. It is constant for a whole dataset and not set per sample.

3.3 Classification

Classification describes the task of assigning a label to an unlabeled time series Q. Figure 6 illustrates this classification task. It requires the *tf-idf* weight matrix to be computed for each class (i.e. Cylinder, Bell, and Funnel) based on

```
String predict(tf,samples,tfIdfs)
(1)  (maxSim, bestClass) = (0, NULL)
(2)  for classId in 1..len(classes) // search class
(3)    smilarity = dotProduct(tf, tfIdfs[classId])
(4)    if smilarity > maxSim  // store the best class
(5)      (maxSim, bestClass) = (similarity, classId)
(6)  return bestClass
```

Algorithm 3: Predict: Classification using the BOSS VS.

a train dataset. First, the unlabeled query Q has to be transformed into the *tf* vector using the BOSS model and the optimal features (window length, mean normalization, SFA word length) obtained from the training phase. Next, the Cosine similarity is calculated using the *tf-idf* weight matrix for each class C of the train dataset. Finally the unlabeled time series is assigned to the class C_i that maximizes the Cosine similarity:

$$label(Q) = \underset{C_i \epsilon C}{argmax}(sim(Q, C_i))$$

BOSS VS Classification (Algorithm 3) The algorithm is based on 1-nearest-neighbor (1-NN) classification using the *tf-idf* weight matrix. Using the classes instead of the time series significantly reduces the computational complexity for classification. First, all classes (line 2) are iterated and the cross product between the *tf* vector of the query and the *tf-idf* weight matrix for each class is calculated (line 3). The class that maximizes the cosine similarity is chosen (lines 4–5) as the query's class label.

3.4 Computational Complexity

The BOSS model 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. The transformation of one sliding window of length w is constant $O(l)$ in the window length (see [8]), whereas the first sliding window has to be calculated using the Fast Fourier Transform $O(w \log w)$:

$$\begin{aligned} T(BOSS) &= O(w \log w + l \cdot (n - w)) \\ &= O(n) \qquad \text{with } l \ll w \ll n \end{aligned} \tag{9}$$

Cosine Similarity The computational complexity of the cosine similarity 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:

$$T(BOSSDistance) = O(n - w + 1) = O(n) \tag{10}$$

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

Classification The computational complexity for classification is given by a 1-NN search over the $|C|$ classes using the cosine similarity calculations:

$$\begin{aligned} T(Predict) &= O(|C| \cdot T(BOSSDistance)) \\ &= O(|C| \cdot n) \end{aligned} \tag{11}$$

Rivalling methods have the complexity: 1-NN ED $O(Nn)$, 1-NN DTW $O(Nn^2)$, Bag-of-Pattern $O(Nn)$, or 1-NN BOSS $O(Nn)$ with number of samples N.

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

$$\begin{aligned} T(Fit) &= O(\sqrt{n} \cdot N \cdot [T(BOSS) + T(Predict)]) \\ &= O(Nn^{\frac{3}{2}}\,|C|) \\ &= O(Nn^{\frac{3}{2}}) \qquad \text{with } C \ll N \end{aligned} \tag{12}$$

4 Experiments

We claim that the BOSS VS offers a very good trade-off between classification accuracy and execution times. We evaluated this using two case studies and established benchmark datasets. Our web page reports all raw numbers and contains C++ source codes [13]. The BOSS VS classifier was implemented in C++ 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 1.7. All experiments consist of two phases: model building using the train dataset and testing the classification accuracy using the test dataset. It is impossible to benchmark all classification algorithms. We did our best to include as many state of the art algorithms as possible. The BOSS VS classifier is compared to state of the art classifiers like nearest neighbor based classifiers like 1-NN fast shapelets [18], 1-NN bag-of-patterns [15], 1-NN shotgun ensemble [10], 1-NN ED or 1-NN DTW with the optimal warping window [6], SAX VSM [19], 1-NN BOSS classifier [8], an ensemble technique *Proportional* (Prop) [21], or machine learning techniques such as support vector machines (SVM) with a quadratic and cubic kernel, and a tree based ensemble method (random forest). Where possible we used the implementations given by the authors and python using sklearn for the SVM benchmarks.

4.1 Case Studies

We present two case studies using (a) a moderately sized dataset resulting from personalized medicine and (b) a small dataset capturing human walking motions. Both datasets have in common that they contain noisy data. Figure 7 shows two examples of each class in the datasets.

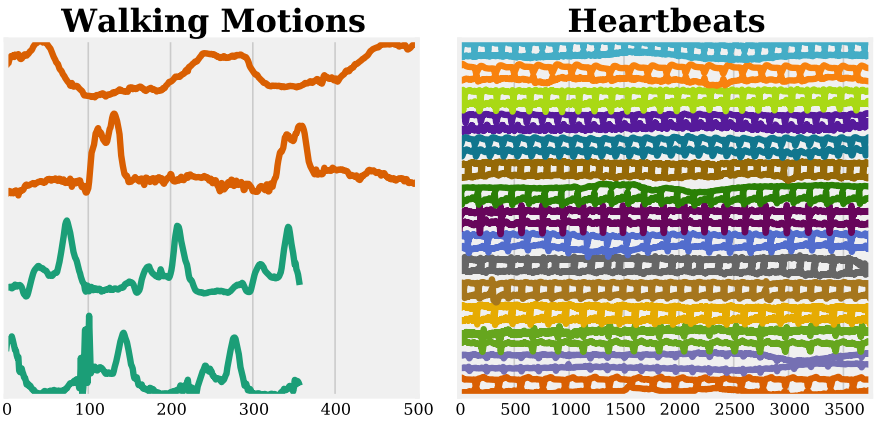

Figure 7: Two samples representing each class of the case studies: abnormal and normal walking motions, and ECG signals from 15 subjects.

4.1.1 Personalized Medicine: Heartbeat BIDMC

The BIDMC Congestive Heart Failure Database [1] consists of 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 repetitive heart beats but even a single patient can have multiple different heart beats. The total size of this dataset is equal to 9 million data points (10 hours sampled at at 250 Hz). We used the train/test split provided by [22] containing 600 samples each. The train and test splits have different lengths of 3750 and 11250.

Figure 8 shows that the 1-NN BOSS, the 1-NN BOSS VS and the 1-NN Shotgun Ensemble 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 horizontal shifts or time warping in order to cope with the periodic patterns. Train times vary from minutes (BOSS, BOSS VS) to days (SAX VSM, Shotgun classifier, and DTW centroid). 1-NN DTW has the highest test times with 30 minutes. The 1-NN Euclidean Centroid classifier has the lowest test time with 0.01s, but the worst accuracy. Our 1-NN BOSS VS offers the best trade-off between train/test times and accuracy. The test time is orders of magnitude lower than that of the more complex classifiers. Even when combining a test time of 2s and a train time of 150s, the 1-NN BOSS VS is one order of magnitude faster than the 1700s 1-NN DTW classifiers' test times.

4.1.2 Human Walking Motions: Toe Segmentation

This dataset [2] contains variable length walking motions of four subjects. We benchmarked the first segmentation approach provided by [3]. The data were captured by recording the z-axis accelerometer values of either the right or the left toe and categorized by the labels *normal walk* and *abnormal walk*. The

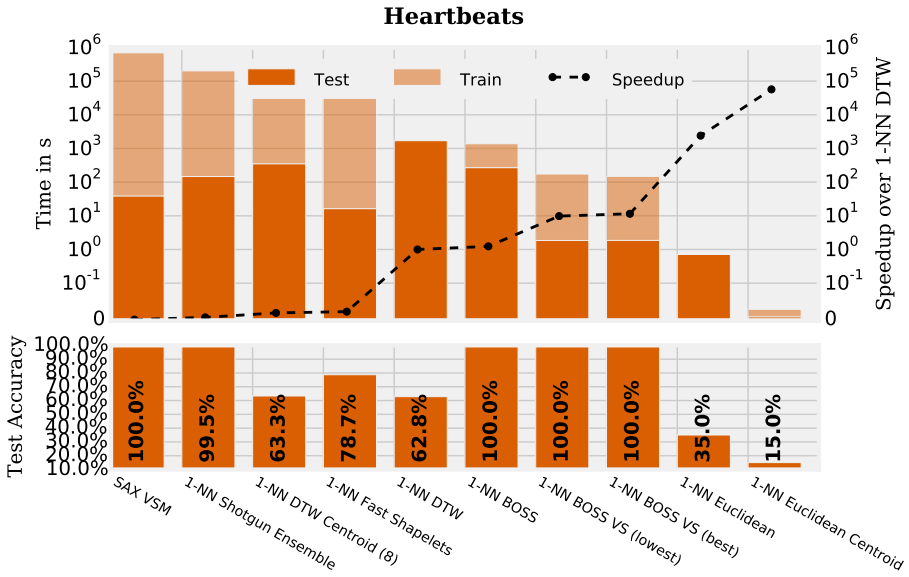

Figure 8: Accuracy and classification times for ECG recordings. We could not benchmark SVM due to variable time series lengths. Fast Shapelets timed out after 6 days of training.

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. This is a small dataset containing 40 train and 228 test samples of variable lengths.

Figure 9 shows that the 1-NN BOSS classifier offers the best classification accuracy with 97.4% followed by the 1-NN Shotgun Ensemble and 1-NN BOSS VS classifiers. The accuracy of the other classifiers drops significantly. The BOSS and BOSS VS classifiers perform well due to the periodicity in walking motions and the noise reducing effect of SFA. Train times vary from milliseconds to up to 2 hours (SAX VSM). 1-NN DTW has the highest test times with 2s. Again, the 1-NN Euclidean Centroid classifier has the lowest train and test times, but the worst accuracy.

Our 1-NN BOSS VS offers the best trade-off between train/test times and accuracy. It offers the second lowest test time and a very low train time. Note, that is was advantageous here to use the *best* Fourier coefficients instead of the lowest Fourier coefficients. This provides an increase of accuracy by 7 percentage points.

4.2 Scaling to a Billion Values

This synthetic dataset shows scalability in terms of wall-lock times. We test the scalability based on the Cylinder-Bell-Funnel (CBF) dataset using a variable number of time series N of 10^3 up to $8{\cdot}10^6$ and a fixed time series length $n = 256$ (Figure 10). The largest tested dataset has 2.048 billion values ($= 2 \cdot 10^9$) which is roughly equal to 15 GB of data. This dataset is widely used and contains

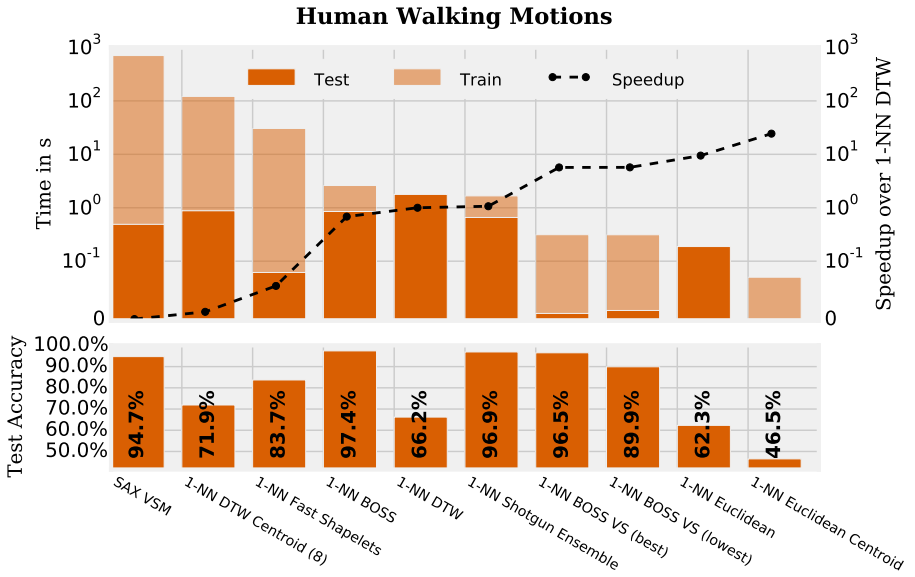

Figure 9: Accuracy and classification times for human walking motions. We could not benchmark SVM due to variable length time series.

three basic shapes: cylinders, bells and funnels. We performed 3000 predictions for each classifier. Not all classifiers could be tested for all dataset sizes due to timeouts or out of memory errors. We stopped a classifier when it took more than several days to train. This happened for 1-NN DTW Centroid, SVMs and the 1-NN Shotgun Ensemble.

We focus on execution time rather than accuracy here. For the sake of completeness, the BOSS VS scores an accuracy of 99.9%. Figure 10 shows two sets of curves for training and testing the classifiers. Training the BOSS VS classifier took a maximum of 4.4 hours for the largest dataset size of 2 billion values. These train times are quickly amortized given the presented 1-NN DTW test times. 1-NN DTW took 27 hours to classify the queries within 0.2 billion values, where 1-NN BOSS VS takes less than 1/50 of the time to train. We didn't continue to measure the 1-NN DTW times from there on.

1-NN BOSS VS testing took on average 10^{-1} seconds for all 3000 predictions regardless of the dataset size. The other classifiers scale with the size of the dataset, with the exception of 1-NN DTW centroid which has an enormous train time and timed out after 7 days for datasets larger than $2 \cdot 10^6$. The 1-NN BOSS runs out of memory at $2 \cdot 10^7$ values. In total the BOSS VS is multiple orders of magnitude faster than the other classifiers. It has moderate training times. These empirical results are not surprising given the test complexities of the other classifiers and that our 1-NN BOSS VS is just $O(n)$.

4.3 Classification Accuracy

All classifiers were evaluated using the same 32 time series datasets from the UCR time series classification archive [5] as previously published [23, 24, 9, 15, 4, 18, 19]. Each dataset provides a *train/test* split. By the use of these train/test

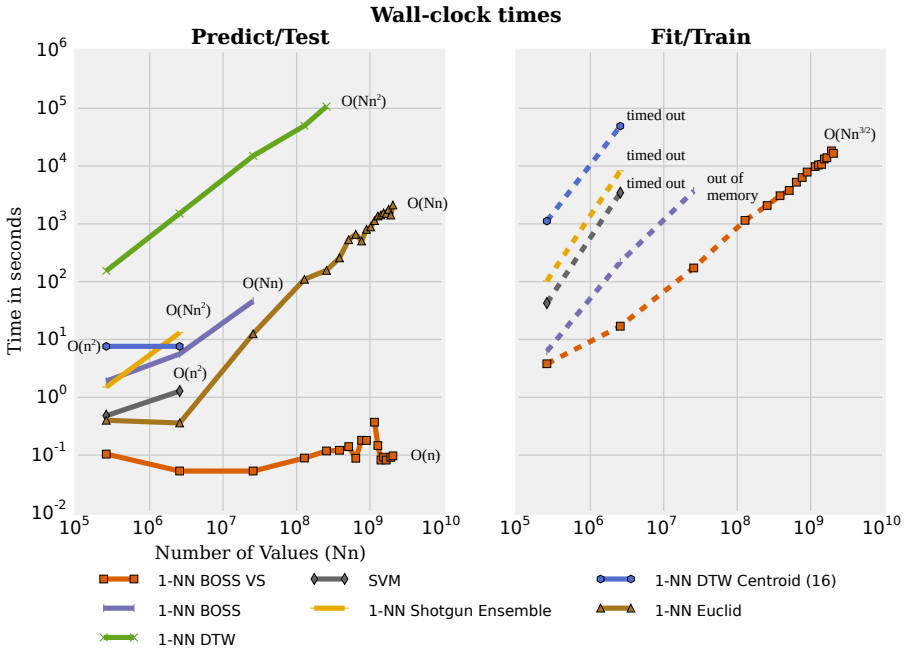

Figure 10: Scalability for an increasing number of time series. Dotted lines represent the train time, solid lines represent the test times.

splits, the results are comparable. All our results are based on the *test accuracy* of the classifiers.

Classification Accuracy Figure 11 shows a critical difference plot over the average ranks of the classifiers as introduced in [25]. The classifiers with the lowest (best) ranks are to the right. The group of classifiers that are not significantly different are connected by a bar. 1-NN BOSS performs best and has the lowest rank. The strength of the 1-NN BOSS VS classifier lies in its computational complexity rather than its high classification accuracy. Still it is not significantly different from the best classifiers 1-NN BOSS, PROP, SAX-VSM, 1-NN Shotgun ensemble, or 1-NN DTW. The 1-NN DTW classifier is commonly used as the benchmark to compare to [9, 11] and performs worse than 1-NN BOSS VS by almost 2 ranks.

Wall-clock time Figure 12 shows the wall-clock times of the four most accurate classifiers in a pairwise comparison to BOSS VS. PROP is an ensemble of classifiers including 1-NN DTW and as such can not be faster than 1-NN DTW. Again our BOSS VS classifier significantly outperforms the other classifiers by 2 to 4 orders of magnitude in terms of test times (predict). The BOSS VS is significantly faster than the BOSS in terms of train times (fit). It seems to have similar training times as the Shotgun ensemble classifier. When looking at the raw data, the Shotgun ensemble classifier trains faster for very small datasets and is orders of magnitude slower for moderate to large sized datasets. The authors of SAX VSM report training times for the UCR datasets. These are 4

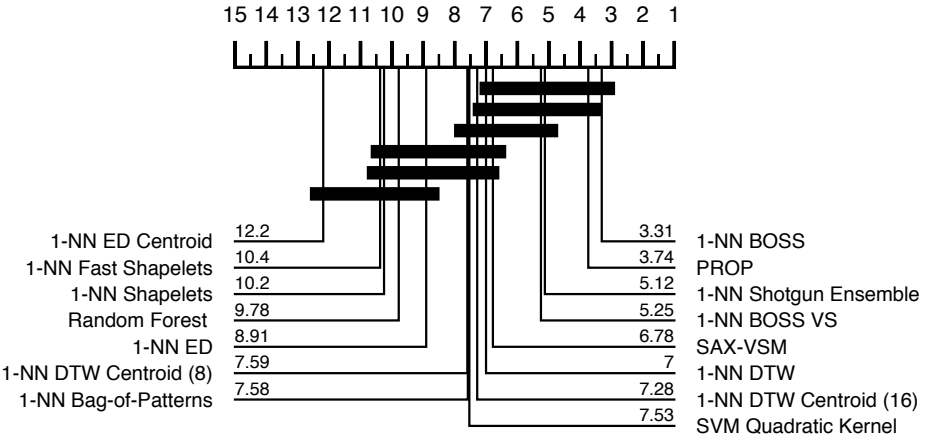

Figure 11: Critical difference diagram for state of the art classifiers on the UCR datasets. The best classifiers are to the right. Critical difference is 3.5.

to 5 orders of magnitude larger (in the order of weeks in total) than the BOSS VS training times. This might be a result of the large parameter space of the SAX VSM classifier. We could not verify these training times, and thus report the test times only. These are up to 2 orders of magnitude higher than those of the BOSS VS. We conclude that the BOSS VS is orders of magnitude faster than its competitors.

4.4 Impact of Design Decisions

The BOSS VS is based on four design decisions:

1. Testing a *subset* of $\sqrt{n}$ windows for training as opposed to using *all* windows.

2. Use mean normalization as a parameter or always *norm* the mean of all windows.

3. Using *numerosity reduction* to avoid outweighing stable sections of a signal as opposed to *no numerosity* reduction.

4. Using the *best* Fourier coefficients or using the *lowest* Fourier coefficients.

The BOSS VS was designed to use a subset of windows, mean normalization as a parameters, numerosity reduction, and the lowest Fourier coefficients (*Subset;Numerosity;Lowest).*

Figure 13 shows that the average score on the UCR datasets slightly improves when numerosity reduction is applied. Using the best Fourier coefficients seems to perform worse than using the lowest Fourier coefficients for the UCR datasets. However, it is advantageous for some datasets like the human walking motions (Figure 9) or the StarlightCurve dataset. Using the mean normalization as a parameter performs significantly better than always norming the data (*"Norm"*). The best 5 classifiers all use it as a parameter, whereas the worse

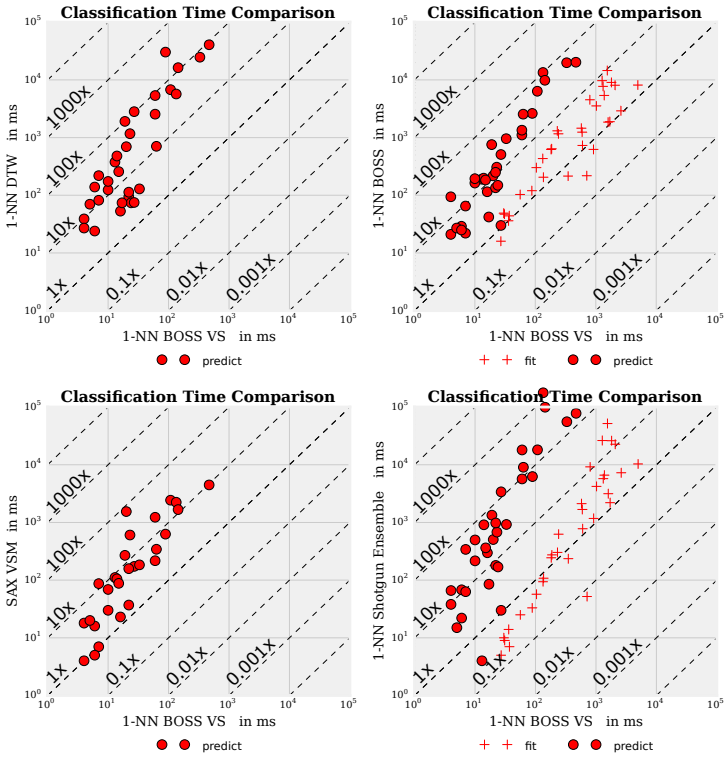

Figure 12: Predict (test) and fit (train) wall-clock times of the most accurate classification algorithms compared to 1-NN BOSS VS.

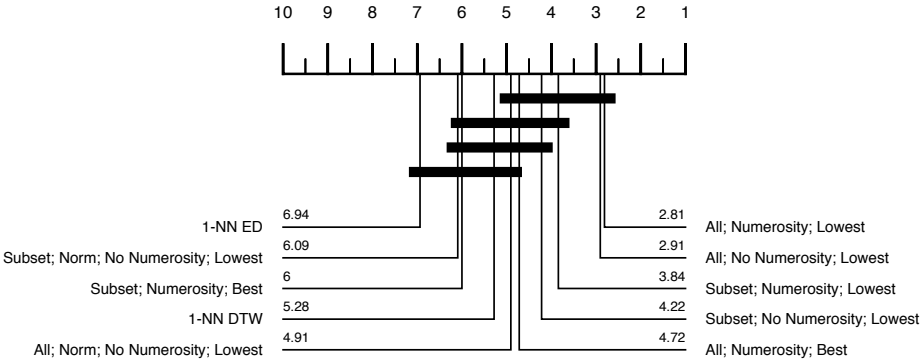

Figure 13: Critical difference diagram for different design decisions made. The best classifiers are to the right. Critical difference is 2.2. BOSS VS is *Subset;Numerosity;Lowest*.

classifiers always subtract the mean from the windows. Using $\sqrt{n}$ windows performs significantly worse than using all windows but is crucial for a low training time which is reduced by a factor of $\sqrt{n}$.

5 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 the BOSS in Vector Space that combines constant time classification and linear time training with high classification accuracy due to invariance to noise, phase shifts, offsets, amplitudes and occlusions. Our experimental evaluation shows the scalability to up to 2 billion values (15 GB) with a classification time that is multiple orders of magnitude faster than 1-NN DTW with the same level of classification accuracy. The BOSS VS beats competitors on two use cases for noisy data and it is in the group of best state of the art classifiers with regards to classification accuracy on the UCR benchmark datasets.

Acknowledgment

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

References

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

[2] CMU Graphics Lab Motion Capture Database. `http://mocap.cs.cmu.edu/` (2014)

[3] Ye, L., Keogh, E.J.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min. Knowl. Discov. (2011)

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

[5] UCR Time Series Classification/Clustering Homepage. `http://www.cs.ucr.edu/~eamonn/time_series_data` (2014)

[6] 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: ACM SIGKDD, ACM (2012)

[7] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing (1) (1978)

[8] Schäfer, P.: The boss is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery (2014)

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

[10] Schäfer, P.: Towards time series classification without human preprocessing. In: Machine Learning and Data Mining in Pattern Recognition. Springer (2014) 228–242

[11] Bagnall, A., Lines, J.: An experimental evaluation of nearest neighbour time series classification. arXiv preprint arXiv:1406.4757 (2014)

[12] 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: IEEE International Conference on Data Mining. (2014)

[13] Webpage, BOSS VS. `http://www.zib.de/patrick.schaefer/bossVS/` (2014)

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

[15] Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. J. Intell. Inf. Syst. (2012)

[16] Lin, J., Keogh, E.J., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining and Knowledge Discovery (2007)

[17] Esling, P., Agon, C.: Time-series data mining. ACM Computing Surveys (CSUR) **45**(1) (2012) 12

[18] Rakthanmanon, T., Keogh, E.: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: SDM. (2013)

[19] Senin, P., Malinchik, S.: Sax-vsm: Interpretable time series classification using sax and vector space model. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE (2013) 1175–1180

[20] Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18**(11) (1975) 613–620

[21] Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery (2014) 1–28

[22] Hu, B., Chen, Y., Keogh, E.: Time Series Classification under More Realistic Assumptions. In: SDM. 2013

[23] Bagnall, A., Davis, L.M., Hills, J., Lines, J.: Transformation Based Ensembles for Time Series Classification. In: SDM, SIAM / Omnipress (2012)

[24] Batista, G., Wang, X., Keogh, E.J.: A Complexity-Invariant Distance Measure for Time Series. In: SDM, SIAM / Omnipress (2011)

[25] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7** (2006) 1–30