REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	Technical Report	-

4. TITLE AND SUBTITLE
SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model

5a. CONTRACT NUMBER
W911NF-12-1-0037

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS
P Senin, S Malinchik

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Indiana University at Bloomington
Trustees of Indiana University
509 E 3RD ST
Bloomington, IN 47401 -3654

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
61766-NS-DRP.29

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
In this paper, we propose a novel method for characteristic patterns discovery in time series. This method, called SAX-VSM, is based on two existing techniques - Symbolic Aggregate approXimation and Vector Space Model. SAX-VSM is capable to automatically discover and rank time series patterns

15. SUBJECT TERMS
vector space models, time series classification, symbolic aggregate approximation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Alessandro Flammini
UU	UU	UU	UU		19b. TELEPHONE NUMBER 812-856-1830

Report Title

SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model

ABSTRACT

In this paper, we propose a novel method for
characteristic patterns discovery in time series. This method,
called SAX-VSM, is based on two existing techniques - Symbolic
Aggregate approXimation and Vector Space Model. SAX-VSM is
capable to automatically discover and rank time series patterns
by their importance to the class, which not only creates wellperforming
classifiers and facilitates clustering, but also provides
an interpretable class generalization. The accuracy of the method,
as shown through experimental evaluation, is at the level of the
current state of the art. While being relatively computationally
expensive within a learning phase, our method provides fast,
precise, and interpretable classification.

SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model

Pavel Senin
Information and Computer Sciences Department,
University of Hawaii at Manoa, Honolulu, HI, 96822
Email: senin@hawaii.edu

Sergey Malinchik
Lockheed Martin Advanced Technology Laboratories,
3 Executive Campus, Suite 600, Cherry Hill, NJ, 08002
Email: sergey.b.malinchik@lmco.com

***Abstract*—In this paper, we propose a novel method for characteristic patterns discovery in time series. This method, called SAX-VSM, is based on two existing techniques - Symbolic Aggregate approXimation and Vector Space Model. SAX-VSM is capable to automatically discover and rank time series patterns by their importance to the class, which not only creates well-performing classifiers and facilitates clustering, but also provides an interpretable class generalization. The accuracy of the method, as shown through experimental evaluation, is at the level of the current state of the art. While being relatively computationally expensive within a learning phase, our method provides fast, precise, and interpretable classification.**

I. Introduction

Time series classification is an increasingly popular area of research providing solutions to the wide range of fields including data mining, image and motion recognition, signal processing, environmental sciences, health care, and chemometrics. Within last decades, many time series representations, similarity measures, and classification algorithms were proposed following the rapid progress in data collection and storage technologies [1]. Nevertheless, to date, the best overall performing classifier in the field is a nearest-neighbor algorithm (1NN), that can be easily tuned for a particular problem by the choice of a distance measure, an approximation technique, or smoothing [4]. As pointed by dozens of papers, a simple "lazy" nearest neighbor classifier is accurate and robust, depends on a very few parameters and requires no training [1], [3], [4], [13]. However, while possessing these qualities, 1NN technique has a number of significant disadvantages, where the major shortcoming is that it does not offer any insight into the classification results. Another limitation is its need for a significantly large training set, that represents a class variance, in order to achieve a good accuracy. Finally, while having trivial initialization, 1NN classification is computationally expensive. Thus, the demand for a simple, efficient, and interpretable classification technique capable of processing of large data collections remains.

In this work, we address outlined above limitation by proposing an alternative to 1NN algorithm that provides a superior interpretability, learns efficiently from a small training set, and has a low computational complexity in classification.

The paper is structured as follows. Section II provides background into the existing algorithms and discusses relevant work. Section III, provides background for a proposed algorithm. In Section IV, we describe our algorithm, and in Section V, we evaluate its performance. Finally, we form our conclusions and discuss future work in Section VII.

II. Prior and related work

Almost all of the existing techniques for time series classification can be divided in two major categories [2]. The first category of classification techniques is based on shape-based similarity metrics - where distance is measured directly between time series points. Classical example of methods from this category is a nearest neighbor classifier built upon Euclidean distance [5] or SpADe [6]. The second category consists of classification techniques based on structural similarity metrics which employ some high-level representations of time series based on their global or local features. Examples from this category include classifier based on Discrete Fourier Transform [7] and a classifier based on Bag-Of-Patterns representation (BOP) [8]. The development of these distinct categories can be explained by differences in their performance: while shape-based similarity methods virtually unbeatable on short, often pre-processed time series data [3], they usually fail on long and noisy data sets [9], where structure-based solutions demonstrate a superior performance.

As possible alternatives to these two categories, two relevant to our work techniques, were recently proposed. The first technique is the time series shapelets algorithm, that was introduced in [10] and is featuring a superior interpretability and a compactness of delivered solution. A shapelet is a short time series "snippet", that is a representative of class membership and is used for a decision tree construction facilitating class identification and interpretability [11]. In order to find a branching shapelet, the algorithm exhaustively searches for a best discriminatory shapelet on data split via an information gain measure. The algorithm's classification is built upon the similarity measure between a branching shapelet and a full time series, defined as a distance between the shapelet and a closest subsequence in the series when measured by the nor-

malized Euclidean distance. This technique, potentially, combines the superior precision of shape-based exact similarity methods, and the high-throughput classification capacity and efficiency of feature-based approximate techniques. However, while demonstrating a superior interpretability, robustness, and similar to kNN algorithms performance, shapelets-based algorithms are computationally expensive ($O(n^2m^3)$, where n is a number of objects and m is the length of a longest time series), which makes difficult its adoption for many-class classification problems [12]. While a better solution was recently proposed ($O(nm^2)$), it is an approximate algorithm, that is based on iSAX approximation and indexing [18].

The second relevant to our work approach is the 1NN classifier built upon the Bag-Of-Patterns (BOP) representation of time-series [8]. BOP representation of a time series is equated to IR "bag of words" concept, and is obtained by extraction, symbolic approximation with SAX, and counting of occurrence frequencies of short overlapping subsequences (patterns) along the time series. By applying this procedure to a training set, algorithm converts the data into the vector space, where each of the original time series is represented by a pattern (a SAX word) occurrence frequency vector. These vectors are classified with 1NN classifier built upon Euclidean distance, or Cosine similarity on raw frequencies or with *tf∗idf* ranking. It was shown by the authors, that BOP has several advantages: it has a linear complexity ($O(nm)$), it is rotation-invariant and considers local and global structures simultaneously, and it provides an insight into patterns distribution through frequency histograms. Through an experimental evaluation the authors concluded, that the best classification accuracy of BOP-represented time series is achieved by using 1NN classifier based on Euclidean distance between frequency vectors.

Our proposed algorithm has similarities with aforementioned techniques. Similarly to shapelet-based approach, it finds time series subsequences which are characteristic representatives of a whole class, thus enabling superior interpretability. However, instead of recursive search for discriminating shapelets, our algorithm ranks by importance all potential candidate subsequences *at once* with a *linear computational complexity* of $O(nm)$. To achieve this, similarly to BOP, SAX-VSM converts all of the training time series into the vector space and computes tf∗idf ranking. But instead of building of n bags (for each of the training time series), our algorithm builds a *single bag of words for each of classes*, that effectively provides a compact solution of N weight vectors (N is the number of classes, $N << n$), and a fast classification time of $O(m)$.

As we shall show, these distinct features: the generalization of the class' patterns with a single bag and tf∗idf ranking, allow SAX-VSM to achieve high accuracy, and tolerate noise in data.

III. Background

SAX-VSM is based on two well-known techniques. The first technique is Symbolic Aggregate approXimation [14], which is a high-level symbolic representation of time series data. The second technique is a well known in Information Retrieval (IR) Vector Space Model [15]. By utilizing a sliding window subsequence extraction and SAX, our algorithm transforms labeled time series into collections of SAX words (terms). At the following step, it utilizes *tf∗idf* terms weighting for a classifier construction. The SAX-VSM classification relies on cosine similarity metric.

SAX algorithm, however, requires two parameters to be provided as an input, and as per today, there is no efficient solution for parameters selection known to the best of our knowledge. To solve this problem, we employ a global optimization scheme based on the divided rectangles (DIRECT) algorithm that does not require any parameters [16]. DIRECT is a derivative-free optimization process that possesses local and global optimization properties. It converges relatively quickly and yields a deterministic, optimized solution.

A. Symbolic Aggregate approXimation (SAX)

Symbolic representation of time series, once introduced [14], has attracted much attention by enabling an application of numerous string-processing algorithms, bioinformatics, and text mining tools to temporal data. The method provides a significant reduction of the time series dimensionality and a low-bounding to Euclidean distance metric, which guarantees no false dismissal [17]. These properties are often leveraged by other techniques, which embed SAX representation in their algorithms for indexing and approximation. For example, adoption of SAX indexing allowed significant shapelets discovery speed improvement in Fast-Shapelets [18] (but made the algorithm approximate).

Configured by two parameters - a desired word size w and an alphabet size A, SAX produces a symbolic approximation of a time-series T of a length n by compressing it into a string of the length w (usually $w << n$), whose letters are taken from the alphabet α ($|\alpha| = A$). At the first step of the algorithm, T is z-normalized (to unit of standard deviation) [19]. At the second step, a dimensionality of the normalized time series is reduced from n to w by obtaining its Piecewise Aggregate Approximation (PAA) [20]; for this, the normalized time series is divided into w equal-sized segments and mean values for points within each segment are computed. The aggregated sequence of these mean values forms PAA approximation of T. Finally, each of w PAA coefficients is converted into a letter of an alphabet α by the use of the lookup table. This table is pre-built by defining a set of breakpoints that divide the normalized time series distribution space into α equiprobable regions. The design of these tables rests on the assumption that normalized series tend to have Gaussian distribution [21].

B. Bag of words representation of time series

Following its introduction, SAX was shown to be an efficient tool for solving problems of finding motifs and discords in time series [17], [22]. The authors employed a sliding window-based subsequence extraction technique and augmented data structures (hash table in [22] and trie in

[17]) in order to build SAX words "vocabularies". Further, by analyzing words frequencies and locations, they were able to capture frequent and rare SAX words representing motifs and discords subsequences. Later, the same technique based on the combination of sliding window and SAX was used in the numerous works, most notably in time series classification using bag of patterns [8].

We also use this sliding window technique to convert a time series T of a length n into the set of m SAX words, where $m = (n - l_s) + 1$ and l_s is the sliding window length. By sliding a window of length l_s across time series T, extracting subsequences, converting them to SAX words, and placing these words into an unordered collection, we obtain the *bag of words* representation of the original time series T.

C. Vector Space Model (VSM) adaptation

We use Vector space model exactly as it is known in information retrieval (IR) [15]. Similarly to IR, we define and use terms *document*, *bag of words*, *corpus*, and *sparse matrix* in our workflow. Note however, that we use terms *bag of words* and *document* for abbreviation of an unordered collection of SAX words interchangeably, while in IR these usually bear different meaning, where a *document* usually presumes certain words ordering (semantics). Although, similar definitions, such as *bag of features* or *bag of patterns*, were previously proposed for techniques built upon SAX [8], we use *bag of words* since it reflects our workflow precisely. The term *corpus* is used for a structured collection of bags of words.

Given a training set, SAX-VSM builds bags of SAX-generated words representing each of the training classes and assembles them into a corpus. This corpus, by its construction, is a sparse *term frequency matrix*. Rows of this matrix correspond to the set of all SAX words found in *all classes*, while each column of the matrix denotes a class of the training set. Each element of this matrix is an observed frequency of a word in a class. Many elements of this matrix are zeros - because words extracted from one class are often not found in others (Figure 4). By its design, this sparse term frequency matrix is a dictionary of all SAX words extracted from all time series of a training set, which accounts for frequencies of each word in each of the training classes.

Following to the common in IR workflow, we employ the *tf∗idf* weighting scheme for each element of this matrix in order to transform a frequency value into the weight coefficient. The *tf∗idf* weight for a term is defined as a product of two factors: term frequency (*tf*) and inverse document frequency (*idf*). For the first factor, we use logarithmically scaled term frequency [23]:

$$\mathrm{tf}_{t,d} = \begin{cases} \log(1 + \mathrm{f}_{t,d}), & \text{if } \mathrm{f}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where t is the term, d is a bag of words (a **d**ocument), and $\mathrm{f}_{t,d}$ is a frequency of the term in a bag.

The inverse document frequency we compute as usual:

$$\mathrm{idf}_{t,D} = \log_{10} \frac{|D|}{|d \in D : t \in d|} = \log_{10} \frac{N}{\mathrm{df}_t} \quad (2)$$

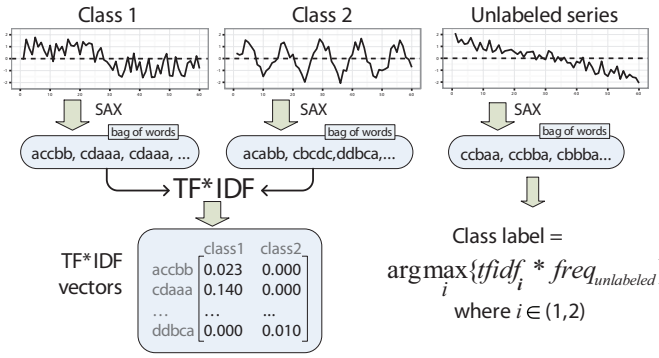

Fig. 1. An overview of SAX-VSM algorithm: at first, labeled time series are converted into bags of words using SAX; secondly, *tf∗idf* statistics is computed resulting in a single weight vector per training class. For classification, an unlabeled time series is converted into a term frequency vector and assigned a label of a weight vector which yields a maximal cosine similarity value. This is *ltc.nnn* weighting schema in SMART notation [23].

where N is the cardinality of corpus D (the total number of classes) and the denominator df_t is a number of documents where the term t appears.

Then, *tf∗idf* value for a term t in the document d of a corpus D is defined as

$$\mathrm{tf} * \mathrm{idf}(t, d, D) = \mathrm{tf}_{t,d} \times \mathrm{idf}_{t,D} = \log(1 + \mathrm{f}_{t,d}) \times \log_{10} \frac{N}{\mathrm{df}_t} \quad (3)$$

for the all cases where $\mathrm{f}_{t,d} > 0$ and $\mathrm{df}_t > 0$, or zero otherwise. Once all terms of a corpus are weighted, the columns of a sparse matrix are used as *class term weights vectors* that facilitate the classification using cosine similarity.

Cosine similarity measure between two vectors is based on their inner product. For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ that is:

$$\mathrm{similarity}(\boldsymbol{a}, \boldsymbol{b}) = cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||a|| \cdot ||b||} \quad (4)$$

IV. SAX-VSM CLASSIFICATION ALGORITHM

As many other classification techniques, SAX-VSM consists of two parts - the training phase, and the classification procedure.

A. Training phase

At first, algorithm transforms all labeled time series into symbolic representation. For this, it converts time series into SAX representation configured by four parameters: the sliding window length (*W*), the number of PAA frames per window (*P*), the SAX alphabet size (*A*), and by the numerosity reduction strategy (*S*) (the choice of these parameters we shall discuss later). Each of the subsequences, extracted with overlapping sliding window, is normalized to unit standard deviation before being processed with PAA [19]. If, however, the standard deviation value falls below a fixed threshold, the normalization procedure is not applied in order to avoid a possible over-amplification of a background noise.

By applying this conversion procedure to all time series from N training classes, algorithm builds a corpus of N bags, to which, in turn, it applies *tf∗idf* ranking. These steps result in N real-valued weight vectors of equal length representing N training classes.

As shown, because of the need to scan the whole training set, training of SAX-VSM classifier is computationally expensive

($O(nm)$). However, there is no need to maintain an index of training series, or to keep any of them in the memory at a runtime: the algorithm simply iterates over all training time series incrementally building a single bag of SAX words for each of training classes. Once built and processed with *tf∗idf*, corpus is also discarded - only a resulting set of N real-valued weight vectors is retained for classification.

B. Classification phase

In order to classify an unlabeled time-series, SAX-VSM transforms it into the terms frequency vector using exactly the same sliding window technique and SAX parameters that were used within the training phase. Then, it computes cosine similarity values between this terms frequency vector and N *tf∗idf* weight vectors representing the training classes. The unlabeled time series is assigned to the class whose vector yields the maximal cosine similarity value.

C. Sliding window size and SAX parameters selection

At this point of SAX-VSM classification algorithm development, it requires a sliding window size and SAX parameters to be specified upfront. Currently, in order to select optimal parameters values while knowing only a training data set, we use a common cross-validation scheme and DIRECT (DIviding RECTangles) algorithm, which was introduced in [26]. DIRECT optimization algorithm is designed to search for global minima of a real valued function over a bound constrained domain, thus, we use the rounding of a reported solution values to the nearest integer.

DIRECT algorithm iteratively performs two procedures - partitioning the search domain, and identifying potentially optimal hyper-rectangles (i.e., having potential to contain good solutions). It begins by scaling the search domain to a n-dimensional unit hypercube which is considered as potentially optimal. The error function is then evaluated at the center of this hypercube. Next, other points are created at one-third of the distance from the center in all coordinate directions. The hypercube is then divided into smaller rectangles that are identified by their center point and their error function value. This procedure continues interactively until error function converges. For brevity, we omit the detailed explanation of the algorithm, and refer the interested reader to [16] for additional details. Figure 2 illustrates the application of DIRECT to *SyntheticControl* data set problem.

D. Intuition behind SAX-VSM

First of all, by combining ***all*** SAX words extracted from ***all*** time series of single class into a ***single bag*** of words, SAX-VSM manages not only to capture observed intraclass variability, but to efficiently "generalize" it through smoothing with PAA and SAX.

Secondly, by partially discarding the original ordering of time series subsequences and through subsequence normalization, SAX-VSM is capable to capture, and to recognize characteristic subsequences in distorted by rotation or shift

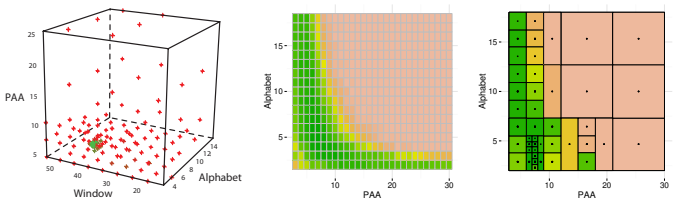

Fig. 2. Parameters optimization with DIRECT for *SyntheticControl* data set (6 classes). Left panel shows all points sampled by DIRECT in the space $PAA*Window*Alphabet$ where red points correspond to high error values in cross-validation experiments, while green points indicate low error values. Note the green points concentration at W=42. Middle panel shows an error-rate heat map when the sliding window size is fixed to 42; this figure was obtained by a complete scan of all 432 points of the slice. Right panel shows the optimized by DIRECT sampling. The optimal solution (W=42,P=8,A=4) was found by sampling of 43 points.

time series, as well, as to recover a signal from partially corrupted or altered by noise.

Thirdly, the *tf∗idf* statistics naturally "highlights" terms unique to a class by assigning them higher weights, while terms observed in multiple classes are assigned weights inversely proportional to their interclass presence frequency. This weighting scheme improves the selectivity of classification by lowering a contribution of "confusive" multi-class terms while increasing a contribution of class' "defining" terms to a final similarity value.

When combined, these features make SAX-VSM time series classification approach unique. Ultimately, algorithm compares a set of subsequences extracted from an unlabeled time series with a weighted set of all characteristic subsequences representing a whole of a training class. Thus, unknown time series is classified by its similarity not to a given number of "neighbors" (as in kNN or BOP classifiers), or to a pre-fixed number of characteristic features (as in shapelets-based classifiers), but by its combined similarity to all known discriminative subsequences found in a whole class during training.

This, as we shall show, contributes to the excellent classification performance on temporal data sets where time series have a very low intraclass similarity at the full length, but embed characteristic to the class subsequences.

V. Results

We have proposed a novel algorithm for time series classification based on SAX approximation of time series and Vector Space Model called SAX-VSM. Here, we present a range of experiments assessing its performance in classification and clustering and show its ability to provide insight into classification results.

A. Analysis of the classification accuracy

To evaluate our approach, we selected thirty three data sets. Majority of the data sets was taken from the UCR time series repository [27], the Ford data set was downloaded from IEEE World Congress on Computational Intelligence website [28], the ElectricDevices data set was downloaded from supporting website for [12]. Overall, SAX-VSM classification

Table I
Classifiers error rates comparison.

Data set	Nb. of classes	1NN-Euclidean	1NN-DTW	Fast Shapelet Tree	Bag Of Patterns	SAX-VSM
Adiac	37	0.389	0.396	0.515	0.432	**0.381**
Beef	5	0.467	0.467	0.447	0.400	**0.033**
CBF	3	0.148	0.003	0.053	0.013	**0.002**
Coffee	2	0.250	0.180	0.067	0.036	**0.0**
ECG200	2	**0.120**	0.230	0.227	0.140	0.140
FaceAll	14	0.286	**0.192**	0.402	0.219	0.207
FaceFour	4	0.216	0.170	0.089	0.011	**0.0**
Fish	7	0.217	0.167	0.197	0.074	**0.017**
Gun-Point	2	0.087	0.093	0.060	**0.002**	0.007
Lightning2	2	0.246	**0.131**	0.295	0.164	0.196
Lightning7	7	0.425	**0.274**	0.403	0.466	0.301
Olive Oil	4	0.133	0.133	0.213	0.133	0.100
OSU Leaf	6	0.483	0.409	0.359	0.236	**0.107**
Syn.Control	6	0.120	**0.007**	0.081	0.037	0.010
Swed.Leaf	15	0.213	0.210	0.270	**0.198**	0.251
Trace	4	0.240	**0.0**	0.002	**0.0**	**0.0**
Two patterns	4	0.090	**0.0**	0.113	0.129	0.004
Wafer	2	0.005	0.020	0.004	0.003	**0.0006**
Yoga	2	0.170	**0.164**	0.249	0.170	**0.164**

performance was found to be at the level of 1NN classifiers based on Euclidean distance, DTW, or BOP, and a shapelet-tree. This result is not surprising taking in account "No Free Lunch theorems" [29], which assert, that there will not be a single dominant classifier for all TSC problems.

Table I compares the performance of SAX-VSM and four competing classifiers: two state-of-the-art 1NN classifiers based on Euclidean distance and DTW, the classifier based on the recently proposed Fast-Shapelets technique [18], and the classifier based on BOP [8]. We selected these particular techniques in order to position SAX-VSM in terms of accuracy and interpretability. The presented comparison data sets selection is limited to the number of previously published or provided by the authors benchmark results for all of four competing classifiers. The performance of SAX-VSM for the rest of the data sets *will be made online along with our reference implementation if accepted*.

In our evaluation, we followed train/test split of the data (exactly as provided by UCR or other sources). We exclusively used train data in cross-validation experiments for selection of SAX parameters and numerosity reduction strategy using our DIRECT implementation. Once selected, the optimal set of parameters was used to assess SAX-VSM classification accuracy which is reported in the last column of the Table I.

B. Scalability analysis

For synthetic data sets, it is possible to create as many instances as one needs for experimentation. We used CBF [30] in order to investigate and compare the performance of SAX-VSM and 1NN Euclidean classifier on increasingly large data sets.

In one series of experiments, we varied a training size from ten to one thousand, while test data set size remained fixed to ten thousands instances. For small training data sets, SAX-VSM was found to be significantly more accurate than 1NN Euclidean classifier. However, by the time we had more than 500 time series in our training set, there was no statistically significant difference in accuracy (Fig. 3, left). As per the running time cost, due to the comprehensive training, SAX-VSM was found to be more expensive than 1NN Euclidean classifier on small training sets, but outperformed 1NN on large training sets. However, SAX-VSM allows to perform training offline and load *tf∗idf* weight vectors when needed. If this option can be utilized, our method performs classification significantly faster than 1NN Euclidean classifier (Fig. 3, right).

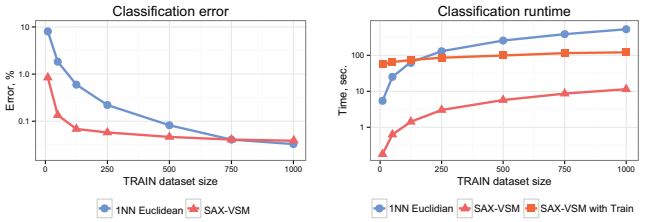

Fig. 3. Comparison of classification precision and run time of SAX-VSM and 1NN Euclidean classifier on CBF data. SAX-VSM performs significantly better with limited amount of training samples (left panel). While SAX-VSM is faster in time series classification, its performance is comparable to 1NN Euclidean classifier when training time is accounted for (right panel).

In another series of experiments we investigated the scalability of our algorithm with unrealistic training set sizes - up to one million of instances of each of CBF classes. As expected, with the grows of a training set size, the curve for a total number of distinct SAX words and curves for dictionary sizes of each of CBF classes reflected a significant saturation (Fig. 4, left). For the largest of training sets - one million instances of each class - the size of the dictionary peaked at 67'324 of distinct words (which is less than 10% of all possible words of length 7 from an alphabet of 7 letters), and the longest tf∗idf vector accounted for 23'569 values (Fig. 4, right). In our opinion, this result reflects two specificities: the first is that the diversity of words which are possible to encounter in CBF dataset is quite limited by its classes configuration and by our choice of SAX parameters (smoothing). The second specificity is that IDF (Inverse Document Frequency, Equation 2) efficiently limits the growth of dictionaries by eliminating those words, which are observed in all of them.

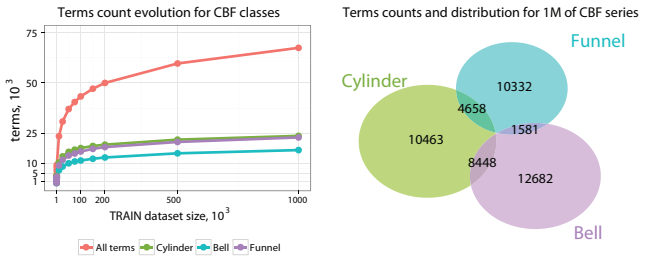

Fig. 4. Left panel: illustration of dictionaries size evolution for CBF with increasingly large training set size. Right panel: distribution of SAX terms in CBF corpus for training set of one million series of each class.

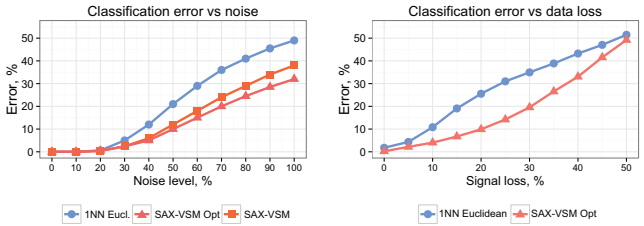

Fig. 5. Classification performance with added noise (left panel; the random noise level varies up to 100% of the signal value, and with a signal loss (right panel). *SAX-VSM Opt* curves correspond to results obtained with "optimized" for each case SAX parameters (we re-trained a classifier).

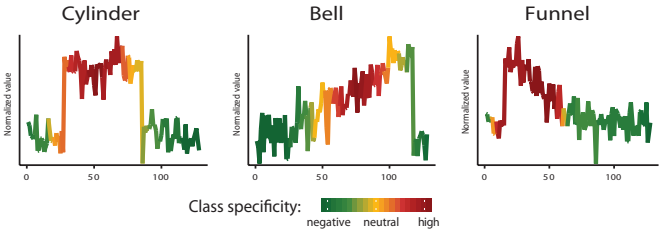

Fig. 6. An example of the heatmap-like visualization of subsequence "importance" to a class identification. Here, for three CBF time series from a training set, a color value of each point was obtained by combining tf∗idf weights of all patterns which cover the point. If a pattern was found in a SAX-VSM-built dictionary corresponding to the time-series class, we added its weight, if, however, a pattern was found in another dictionary - we subtracted its weight. Highlighted by the visualization features corresponding to a sudden rise, plateau, and a sudden drop in Cylinder; increasing trend in Bell; and to a sudden rise followed by a gradual drop in Funnel, align exactly with the design of these classes [30].

C. Robustness to noise

In our experimentation with many data sets, we observed, that the growth of a dimensionality of *tf∗idf* weight vectors continuously follows the growth of a training set size, which indicates that SAX-VSM is actively learning from class variability. This observation, and the fact that a weight of each of the overlapping SAX words is contributing only a small fraction to a final similarity value, prompted an idea that SAX-VSM classifier might be robust to the noise and to the partial loss of a signal in test time series. Intuitively, in such a case, the cosine similarity between high dimensional weight vectors might not degrade significantly enough to cause a misclassification.

While we plan to perform more exploration, current experimentation with CBF data set revealed promising results. In one series of experiments, by fixing a training set size to two hundred fifty time series, we varied the standard deviation of Gaussian noise in CBF model (whose default value is about 17% of a signal level). We found, that SAX-VSM increasingly outperformed 1NN Euclidean classifier with the growth of a noise level (Fig.5 Left). Further improvement of SAX-VSM performance was achieved by fine tuning of smoothing - through a gradual increase of the size of SAX sliding window proportionally to the growth of a noise level (Fig.5 Left, *SAX-VSM Opt* curve).

In another series of experiments, we randomly replaced up to fifty percent of a span of an unlabeled time series with a random noise. Again, SAX-VSM performed consistently better than 1NN Euclidean classifier regardless of a training set size, which we varied from five to one thousand. The *SAX-VSM Opt* curve at Fig.5 (Right) depicts the case with fifty training series when the sliding window size was decreased inversely proportionally to the growth of a signal loss.

D. Interpretable classification

While the classification performance results in previous sections show that SAX-VSM classifier has a very good potential, its major strength is in the level of allowed interpretability of classification results.

Previously, in original shapelets work [10], [11], it was shown that the resulting decision trees provide interpretable classification and offer an insight into the data specific features. In successive work based on shapelets [12], it was shown that the discovery of multiple shapelets provides even better resolution and intuition into the interpretability of classification. However, as the authors noted, a time cost of multiple shapelets discovery in many class problems could be very significant. Contrary, SAX-VSM extracts and weights all patterns at once, without any added cost. Thus, it could be the only choice for interpretable classification in many class problems.

1) Heatmap-like visualization: Since SAX-VSM builds tf∗idf weight vectors using all subsequences extracted from a training set, it is possible to find out the weight of any arbitrary selected subsequence. This feature enables a novel visualization technique that can be used to gain an immediate insight into the layout of "important" class-characterizing subsequences as shown at Figure 6.

2) Gun Point data set: Following previously mentioned shapelet-based work [10], [12], we used a well-studied *Gun-Point* data set [31] to explore the interpretability of classification results. This data set contains two classes: time-series in *Gun* class correspond to the actors' hands motion when drawing a replicate gun from a hip-mounted holster, pointing

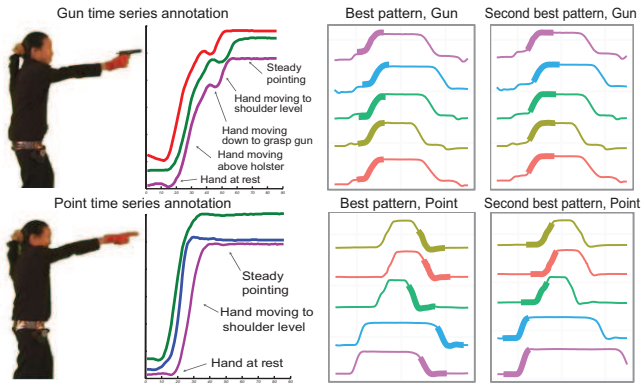

Fig. 7. Best characteristic subsequences (right panels, bold lines) discovered by SAX-VSM in *Gun/Point* data set. Left panel shows actor's stills and time series annotations made by an expert, right panels show locations of characteristic subsequences. Note, that while the upward arm motion found to be more "important" in *Gun* class (gun retrieval and aiming), the downward arm motion better characterizes *Point* class (an "overshoot" phenomena in propless arm return). This result aligns with previous work [10] and [12]. (Stills and annotation used with a permission from E. Keogh)

it at a target for a second, and returning the gun to the holster; time-series in *Point* class correspond to the actors hands motion when pretending of drawing a gun - the actors point their index fingers to a target for about a second, and then return their hands to their sides.

Similarly to previously reported results [10], [12], SAX-VSM was able to capture all distinguishing features as shown at the Figure 7. The most weighted by SAX-VSM patterns in *Gun* class corresponds to fine extra movements required to lift and aim the prop. The most weighted SAX pattern in *Point* class corresponds to the "overshoot" phenomena which is causing the dip in the time series. Also, similarly to the original work [31], SAX-VSM highlighted as second to the best patterns in *Point* class the lack of distinguishing subtle extra movements required for lifting a hand above a holster and reaching down for the gun.

3) OSU Leaf data set: According to the original data source, Ashid Grandhi [32], with the current growth of digitized data, there is a huge demand for automatic management and retrieval of various images. The *OSULeaf* data set consist of curves obtained by color image segmentation and boundary extraction (in the anti-clockwise direction) from digitized leaf images of six classes: *Acer Circinatum, Acer Glabrum, Acer Macrophyllum, Acer Negundo, Quercus Garryana and Quercus Kelloggii*. The authors were able to solve the problem of leaf boundary curves classification by use of DTW, achieving 61% of classification accuracy. However, as we pointed above, DTW provided a very little information about why it succeeded of failed.

In contrast, SAX-VSM application yielded a set of class-specific characteristic patterns for each of six leaves classes from *OSULeaf* data set. These characteristic patterns closely match known techniques of leaves classification based on leaf shape and margin [33]. Highlighted by SAX-CSM features include the slightly lobed shape and acute tips of Acer Circinatum leaves, serrated blade of Acer Glabrum leaves, the acuminate tip and characteristic serration of in Acer Macrophyllum leaves, pinnately compound leaves arrangement of Acer Negundo, the incised leaf margin of Quercus Kelloggii, and a lobed leaf structure of Quercus Garryana. Figure 8 shows a subset of these characteristic patterns and original leaf images with highlighted corresponding features.

4) Coffee data set: Another illustration of interpretable classification with SAX-VSM is based on the analysis of its performance on Coffee dataset [34]. The curves in this dataset correspond to spectra obtained with diffuse reflection infrared Fourier transform (DRIFT) and truncated to 286 data points in the region 800-1900 cm^{-1}. The two top-ranked by SAX-VSM subsequences in both datasets correpond to spectrogram intervals of Chlorogenic acid (best) and Caffeine (second to best). These two chemical compounds are known to be responsible for the flavor differences in Arabica and Robusta coffees; moreover, these spectrogram intervals were reported as discriminative when used in PCA-based technique by the authors of the original work [34].

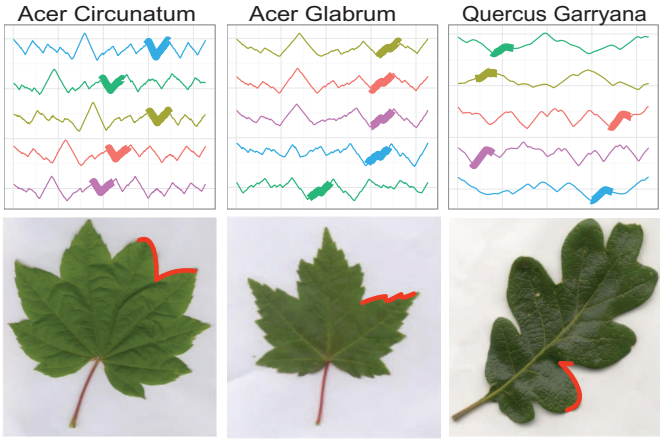

Fig. 8. Best characteristic subsequences (top panels, bold lines) discovered by SAX-VSM in *OSULeaf data set*. These patterns align with well known in botany discrimination techniques by lobe shapes, serrations, and leaf tip types [33].

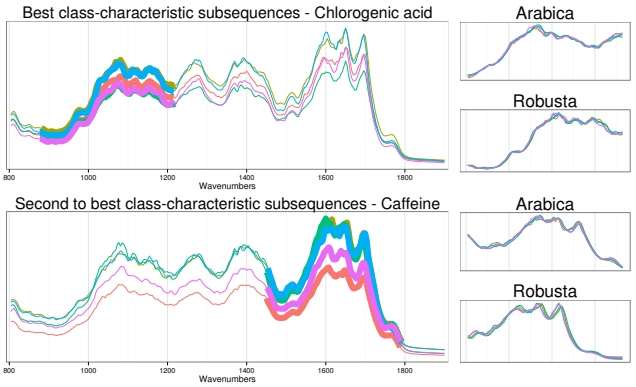

Fig. 9. Best characteristic subsequences (left panels, bold lines) discovered by SAX-VSM in Coffee data set. Right panels show zoom-in view on these subsequences in Arabica and Robusta spectrograms. These discriminative subsequences correspond to chlorogenic acid (best subsequence) and to caffeine (second to best) regions of spectra. This result aligns with the original work based on PCA [34] exactly.

VI. Clustering

Clustering is a common tool used for data partitioning, visualization, exploration, and serves as an important subroutine in many data mining algorithms. Typically, clustering algorithms are built upon a distance function, and the overall performance of an algorithm is highly dependent on a performance of the chosen function. Thus, an experimental evaluation of the proposed technique in clustering provides an additional perspective on its performance and applicability beyond the classification.

A. Hierarchical clustering

Probably, one of the most used clustering algorithms is hierarchical clustering which requires no parameters to be specified [35]. It computes pairwise distances between all objects and produces a nested hierarchy of clusters offering a great data visualization power.

Previously, it was shown that the bag-of-patterns time series representation and Euclidean distance provide a superior

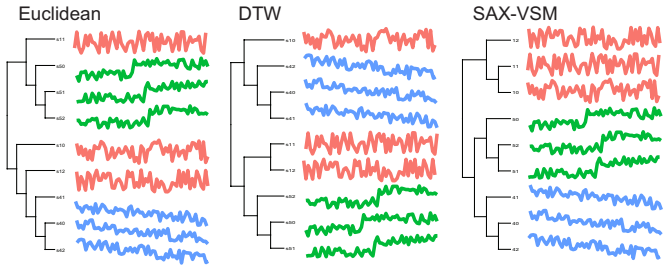

Fig. 10. An comparison of hierarchical clustering application to a subset of three *SyntheticControl* classes: *Normal*, *Decreasing trend*, and *Upward shift*. Euclidean distance, Dynamic time warping, SAX-VSM and Complete linkage were used to generate these plots. Only SAX-VSM was able to partition series properly.

clustering performance [8]. For comparison, we performed similar experiments which differ in time series representation and distance metric - we relied on $tf*idf$ weight vectors and cosine similarity. Affirming the previous work, we found, that the combination of SAX and Vector space model outperforms classical shape-based distance metrics. For example, figure 10 depicts the result of hierarchical clustering of a subset of *SyntheticControl* data. As one can see, SAX-VSM is superior in clustering performance to Euclidean and DTW distance metrics in this particular setup - it produced a hierarchy which properly partitions the data set into three branches.

B. k-Means clustering

Another popular choice for data partitioning is k-Means clustering algorithm [36]. The basic intuition behind this algorithm is that through the iterative reassignment of objects into different clusters the intra-cluster distance is minimized. As was shown, k-Means algorithm scales much better than hierarchical partitioning techniques [37]. Fortunately, this clustering technique is well studied in IR field. Previously, in [38], the authors extensively examined seven different criterion functions for partitional document clustering and found, that k-prototypes partitioning with cosine dissimilarity delivers an excellent performance.

Following this work, we implemented a similar to [39] *spherical k-means algorithm* and found, that algorithm converges quickly and delivers a satisfactory partitioning on short synthetic data sets. Further, we evaluated our technique on the long time series from PhysioNet archive [40]. We extracted two hundred fifty series corresponding to five vital signals: two ECG leads (aVR and II), and RESP, PLETH, and CO2 waves, trimming them to 2'048 points. Similarly to [8], we run a reference k-Means algorithm implementation based on Euclidean distance, which achieved the maximum clustering quality of 0.39, when measured as proposed in [41] on the best clustering (the one with the smallest objective function in 10 runs). SAX-VSM spherical k-Means implementation outperformed the reference technique yielding clusters with the quality of 0.67 (on 10 runs with SAX parameters set to W=33, P=8, A=6).

VII. Conclusion and Future Work

In this paper, we have proposed a novel interpretable technique for time series classification based on characteristic patterns discovery. We have shown, that our approach is competitive with, or superior to, other techniques on a variety of classic data mining problems. In addition, we described several advantages of SAX-VSM over existing structure-based similarity measures, emphasizing its capacity to discover and rank short subsequences by their class characterization power.

The current limitations of our SAX-VSM implementation suggest a number of future work directions. First of all, while Vector space model naturally supports processing of bags of words composed of terms of variable length, our current "stable" implementation lacks this capacity. Inspired by the recently reported superior performance of multi-shapelets based classifiers [12], we prioritize this development. Secondly, as mentioned before, DIRECT optimization it is designed for a function of a real variable. By using rounding in our implementation, we have observed DIRECT iteratively sampling redundant locations in suboptimal neighborhood, thus, a more appropriate optimization scheme is needed. Finally, we are designing and experimenting with an extension of SAX-VSM to multidimensional time series. Currently we are evaluating two candidate implementations: the first is based on a single bag of words accommodating all dimensions for a class (by prefixing SAX words extracted from different dimensions); while the second is based on the use of a single bag of words per each of dimensions. The preliminary results on synthetic data sets look promising and we expect to report our finding soon.

References

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

[2] Xing, Z., Pei, J., Keogh, E.: A brief survey on sequence classification. SIGKDD Explor. Newsl. 12, 1, 40-48, (2010)

[3] Keogh, E., Kasetty, S.: On the need for Time Series Data Mining Benchmarks: a survey and empirical demonstration. In Proc. ACM DMKD, 102–111 (2002)

[4] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. In Proc. the 34th VLDB, 1542–1552 (2008)

[5] Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.: Fast time series classification using numerosity reduction. In Proc. ICML, 1033–1040 (2006)

[6] Chen, Y., Chen, K., Nascimento, M.: Effective and Efficient Shape-Based Pattern Detection over Streaming Time Series. IEEE TKDE, 265–278 (2012)

[7] Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search In Sequence Databases. In Proc. FODO, 69–84 (1993)

[8] Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. J. Intell. Inf. Syst. 39, 2, 287–315 (2012)

[9] Keogh, E.: Exact indexing of dynamic time warping. In Proc. VLDB, 406–417 (2002)

[10] Ye, L., Keogh, E.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min. Knowl. Discov., 149-182, (2011)

[11] Mueen, A., Keogh, E., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In Proc. 17th ACM SIGKDD, 1154–1162 (2011)

[12] Lines, J., Davis, L., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In Proc. 18th ACM SIGKDD, 289–297 (2012)
[13] Salzberg, S.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Min. Knowl. Discov., 1, 317–328 (1997)
[14] Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Min. Knowl. Discov., 107–144 (2007)
[15] Salton, G., Wong, A., Yang., C.: A vector space model for automatic indexing. Commun. ACM 18, 11, 613–620 (1975)
[16] Björkman, M., Holmström, K.: Global Optimization Using the DIRECT Algorithm in Matlab. Advanced Modeling and Optimization, 1(2), 17–37 (1999)
[17] Keogh, E., Lin, J., Fu, A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In Proc. ICDM. 226–233 (2005)
[18] Rakthanamanon, T., Keogh, E.: Fast-Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In Proc. SDM (2013)
[19] Goldin D., Kanellakis, P.: On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. In Proc CP. 137–153 (1995)
[20] Keogh, E., Pazzani, M.: A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases. In Proc. PADKK, 122–133 (2000)
[21] Larsen, R., Marx, M.: An Introduction to Mathematical Statistics and Its Applications, (3rd Ed.), Prentice Hall (2000)
[22] Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In Proc. 9th ACM SIGKDD (2003)
[23] Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, Cambridge University Press (2008)
[24] Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In Proc. 8th DMKD. 2–11 (2003)
[25] Hamming, R.: Error detecting and error correcting codes. Bell System Technical Journal 29, pp. 147–160 (1950)
[26] Jones, D., Perttunen, C., Stuckman, B.: Lipschitzian Optimization without Lipschitz Constant. J. Optim. Theory Appl. 79, 1 (1993)
[27] Keogh, E., Zhu, Q., Hu, B., Hao, Y., Xi, X., Wei, L., Ratanamahatana, C.: The UCR Time Series Classification/Clustering Homepage: http://www.cs.ucr.edu/~eamonn/time_series_data/
[28] WCCI, Ford classification challenge, http://home.comcast.net/~nn_classification/
[29] Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Trans. on Evo. Comp. 1 no. 1, 67–82 (1997)
[30] Saito, N: Local feature extraction and its application using a library of bases. PhD thesis, Yale University (1994)
[31] Ratanamahatana, C., Keogh, E.: Making time-series classification more accurate using learned constraints. In SDM '04 (2004)
[32] Gandhi, A.: Content-Based Image Retrieval: Plant Species Identification. MS thesis, Oregon State University (2002)
[33] Dirr, M.: Manual of Woody Landscape Plants: Their Identification, Ornamental Characteristics, Culture, Propogation and Uses. Stipes Pub Llc, ed. 6 Revised (2009)
[34] Briandet, R., Kemsley, E., Wilson, R.: Discrimination of Arabica and Robusta in Instant Coffee by Fourier Transform Infrared Spectroscopy and Chemometrics. J. Agric. Food Chem, 44, 170–174 (1996)
[35] Johnson, C.: Hierarchical clustering schemes. Psychometrika, 32(3), 241–254 (1967)
[36] MacQueen, J.: Some methods for classification and analysis of multivariate observations Proc. of 5th Berkeley Symp. On Math. Stat. and Prob., 1, 281–296 (1967)
[37] Bradley, P., Fayyad, U., Reina, C.: Scaling clustering algorithms to large databases. In Proc. KDD, 9–15 (1998)
[38] Zhao, Y., Karypis, G.: Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. Mach. Learn., 55, 311–331 (2004)
[39] Dhillon, I., Modha, D.: Concept Decompositions for Large Sparse Text Data Using Clustering. Mach. Learn., 42(1), 143–175 (2001)
[40] Goldberger, A., Amaral, L., Glass, L., et al.: PhysioBank, PhysioToolkit, and PhysioNet: Circulation. Discovery 101(23), 1(3), 215–220 (1997)
[41] Gavrilov, M., Anguelov, D., Indyk, P., Motwani, R.: Mining the stock market: which measure is best? In Proc. 6th KDD. 487–496 (2000)