2013 IEEE 13th International Conference on Data Mining

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
senin@hawaii.edu

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

***Abstract*—In this paper, we propose a novel method for discovering characteristic patterns in a time series called SAX-VSM. This method is based on two existing techniques - Symbolic Aggregate approXimation and Vector Space Model. SAX-VSM automatically discovers and ranks time series patterns by their importance to the class, which not only facilitates well-performing classification procedure, 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.**

***Index Terms*—time series analysis, classification algorithms**

I. Introduction

Time series classification is an increasingly popular area of research, providing solutions to a wide range of fields, including data mining, image and motion recognition, environmental sciences, health care, and chemometrics. Within the last decade, 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 the nearest-neighbor algorithm (1NN), that can be easily tuned for a particular problem by choosing either a distance measure, an approximation technique, or smoothing [1]. The 1NN classifier is simple, accurate and robust, depends on a very few parameters and requires no training [1], [2], [3]. However, the 1NN technique has a number of significant disadvantages, where the major shortcoming is the inability to offer any insight into the classification results. Another limitation is its need for a significantly large training set representing a within-class variance in order to achieve desired accuracy. Finally, while having trivial initialization, 1NN classification is computationally expensive. Thus, the demand for an efficient and interpretable classification technique capable of processing of large data volumes remains.

In this work, we propose an alternative to 1NN algorithm that addresses aforementioned limitations - it provides a superior interpretability, learns efficiently from a small training set, and has a low classification computational complexity.

The paper is structured as follows: Section II 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. We conclude and discuss future work in Section VI.

II. Prior and related work

Almost all of the existing techniques for time series classification can be divided in two major categories [4]. The first category includes techniques based on shape-based similarity metrics where distance is measured directly between time series points. Classical examples from this category is 1NN classifier built upon Euclidean distance [5] and DTW [6]. The second category consists of classification techniques based on structural similarity metrics, which employ a high-level representations of time series based on their global or local features. Examples from this category include classifiers based on time series representation obtained with DFT [7] or Bag-Of-Patterns [8]. The development of these distinct categories can be explained by the difference in their performance: while shape-based similarity methods are virtually unbeatable on short pre-processed time series [2], they usually fail on long and noisy data, where structure-based solutions demonstrate a superior performance [8].

Two techniques, relevant to our work, were recently proposed as possible alternatives to these two categories. The first is the Time Series Shapelet technique which features a superior interpretability and a compactness of delivered solution [9]. 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. 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 normalized Euclidean distance. This exact technique, potentially, combines the superior precision of exact shape-based similarity methods, and the high-throughput classification capacity of feature-based approximate techniques. However, while demonstrating a superior interpretability, robustness, and similar to 1NN algorithm performance, shapelets-based technique is computationally expensive, $O(n^2m^3)$, where n is a number of objects and m is the length of a longest time series, making its adoption for many-class classification problems difficult [10]. While the better solution was recently proposed ($O(nm^2)$), it is an approximate solution based on indexing [11].

The second technique with interpretable results is 1NN

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.52

classifier built upon the Bag-Of-Patterns (BOP) representation of time series [8], which is equated to an Information Retrieval (IR) "bag of words" concept and is obtained by extraction, transformation with Symbolic Aggregate approXimation (SAX) [12], and counting the frequencies of short overlapping subsequences (patterns) along the time series. By applying this procedure to a training set, the algorithm converts the data into the vector space, where the original time series are represented by a pattern (SAX word) occurrence frequency vector. These vectors are classified with 1NN classifier built with Euclidean distance or Cosine similarity applied to raw frequencies or their *tf∗idf* weighting. It was shown that BOP has several advantages: its complexity is linear ($O(nm)$), it is rotation-invariant and considers local and global structures simultaneously, and it provides an insight into patterns distribution through frequency histograms. The authors concluded, that the best classification accuracy of BOP-represented time series is achieved by using 1NN classifier based on Euclidean distance.

Our algorithm has similarities to the aforementioned techniques. Similar to shapelets-based approaches, our algorithm looks for time series subsequences which are characteristic representatives of a class, that enables a superior interpretability. However, instead of recursive search for class-discriminating shapelet, 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 training time series into bags of SAX words and uses *tf∗idf* weighting and Cosine similarity. Nonetheless, instead of building n bags for each of the training time series, our algorithm builds a *single bag of words for each of the classes*, that effectively provides a compact solution of N weight vectors (N is the number of classes, typically $N << n$), and a fast classification time of $O(m)$.

We will show these distinct features - the generalization of the class' patterns with a single bag and their weighting - allow SAX-VSM to achieve a high classification accuracy and to provide an exceptional interpretability.

III. Background

SAX-VSM is based on two well-known techniques. The first technique is Symbolic Aggregate approXimation, which is a high-level symbolic representation of time series [12]. The second technique is the classic Vector Space Model based on *tf∗idf* weighting scheme [13]. Using SAX, our algorithm transforms real-valued time series into combined collections of SAX words. Next, by using *tf∗idf* weighting, it transforms these collections into class-characteristic weight vectors, which, in turn, are used in classification built upon Cosine similarity.

SAX, however, requires two parameters to be provided as an input and no efficient solution for their selection exists to the best of our knowledge. We address this issue by using an optimization scheme based on the dividing rectangles (DIRECT) algorithm that does not require any input [14].

A. Symbolic Aggregate approXimation (SAX)

Symbolic representation of time series, once introduced, has attracted much attention by enabling the application of numerous string-processing algorithms, bioinformatics tools, and text mining techniques to time series [12]. The method provides a significant reduction of the time series dimensionality and a low-bounding to Euclidean distance metrics, which guarantees no false dismissal [15]. These properties are often leveraged by other techniques that embed SAX representation for indexing and approximation [11].

Configured by two parameters, a desired word size w and an alphabet size α, 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. At the first step of the algorithm, T is z-normalized (to unit of standard deviation) [16]. At the second step, a dimensionality of the normalized time series is reduced from n to w by obtaining its Piecewise Aggregate Approximation (PAA) [17]. For this, the normalized time series is divided into w equal-sized segments and mean values for points within each segment are computed. The sequence of these values forms PAA approximation of T. Finally, each of w PAA coefficients is converted into a letter of an alphabet α using the lookup table which defines a set of breakpoints that divide the normalized time series values distribution space into α equal-sized regions (as in the original SAX work [12], we assume Gaussian distribution).

B. Bag of words representation of time series

Following its introduction, SAX was shown to be an efficient tool for solving problems of finding time series motifs and discords [18]. The authors employed a sliding window-based subsequence extraction technique and augmented data structures in order to build SAX words "vocabularies". By analyzing words frequencies, they were able to capture frequent and rare SAX words representing motif and discord subsequences. The same technique, based on the combination of sliding window and SAX, was used in numerous works, most notably in Shapelet [11] and BOP -based classifiers [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 overlapping 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 the vector space model exactly as it is known in Information Retrieval [13]. Similarly, we define and use the following expressions: *term* - a single SAX word, *bag of words* - an unordered collection of SAX words, *corpus* - a set of bags, and *weight matrix* - a matrix defining weights of all words in a corpus.

Given a training set, SAX-VSM builds a bag of SAX words for each of the classes by processing each time series

with a sliding window and SAX. Bags are combined into a corpus, which is built as a *term frequency matrix*, whose rows correspond to the set of all SAX words (terms) found in *all classes*, whereas each column denotes a class of the training set. Each element of this matrix is an observed frequency of a term in a class. Because SAX words extracted from the time series of one class are often not found in others, as shown in Section V-B, this matrix is usually sparse.

Next, SAX-VSM applies *tf∗idf* weighting scheme for each element of this matrix to transform a frequency value into a weight coefficient. The *tf∗idf* weight for a term t 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 [19]:

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

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

The inverse document frequency we compute as usual [19]:

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

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

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

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

for all cases where $\mathrm{f}_{t,d} > 0$ and $\mathrm{df}_t > 0$, or zero otherwise.

Once all frequency values are computed, term frequency matrix becomes the term weight matrix, whose columns used as *class' term weight vectors* that facilitate the classification using Cosine similarity.

For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ Cosine similarity is based on their inner product and defined as

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

IV. SAX-VSM CLASSIFICATION ALGORITHM

As many other classification techniques, SAX-VSM consists of two phases - training and classification.

A. Training phase

The training starts by transforming the labeled time series into SAX representation configured by three parameters: the sliding window length (*W*), the number of PAA segments per window (*P*), and SAX alphabet size (*A*). Each of subsequences extracted with overlapping sliding window is normalized (Sec. III-A) before being processed with PAA. However, if the standard deviation value falls below a fixed threshold, the normalization is not applied in order to avoid over-amplification of a background noise [12].

By applying this procedure to all time series from N training classes, algorithm builds a corpus of N bags, to which it applies *tf∗idf* weighting and outputs N real-valued weight vectors of equal length representing training classes.

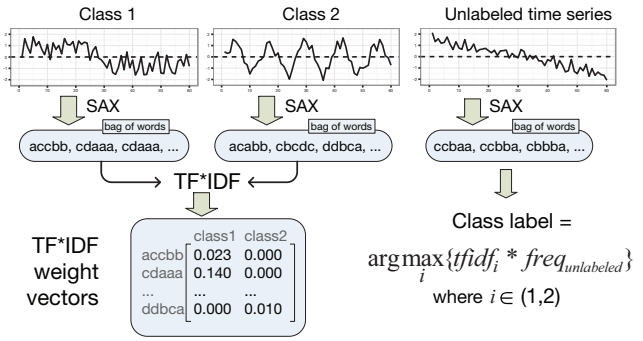

Fig. 1: An overview of SAX-VSM algorithm: at first, labeled time series are converted into bags of words using SAX; secondly, *tf∗idf* statistics is computed resulting in a single weight vector per training class. For classification, an unlabeled time series is converted into a term frequency vector and assigned a label of a weight vector which yields a maximal cosine similarity value. This is *ltc.nnn* weighting schema in SMART notation [19].

Because the whole training set must be processed, training of SAX-VSM classifier is computationally expensive ($O(nm)$). However, there is no need to maintain an index of training time series, or to keep any of them in the memory at the runtime; the algorithm simply iterates over all training time series incrementally building bags of SAX words. Once built and weighted with *tf∗idf*, the corpus is discarded - only a resulting set of N real-valued weight vectors is retained for classification.

B. Classification

In order to classify an unlabeled time series, SAX-VSM transforms it into a terms frequency vector using exactly the same sliding window technique and SAX parameters that were used for training. It computes cosine similarity values between this term 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

As shown, SAX-VSM requires three parameters to be specified upfront. In order to optimize their selection using only a training data set, we propose a solution based on a common cross-validation and DIRECT optimization scheme [20]. Since DIRECT is designed to search for global minima of a real valued function over a bound constrained domain, we use the rounding of a reported solution values to the nearest integer.

DIRECT iteratively performs two procedures - partitioning the search domain and identifying potentially optimal hyper-rectangles. In our case, it begins by scaling the search domain to a 3-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 the error function converges. For brevity, we omit the detailed explanation of the

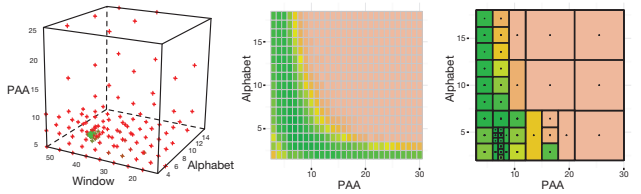

Fig. 2: Parameters optimization with DIRECT for *SyntheticControl* dataset. Left panel shows all points sampled by DIRECT in the space $PAA * Window * Alphabet$, red points correspond to high error values while green points correspond to low error values in cross-validation experiments. Note the green points concentration at W=42. Middle panel shows the classification error heat map obtained by a complete scan of all 432 points of the hypercube slice when W=42. Right panel shows the classification error heat map of the same slice when the parameters search optimized by DIRECT, the optimal solution (P=8,A=4) was found by sampling of 43 points.

algorithm, and refer the to [14] for additional details. Figure 2 illustrates the application of leave-one-out cross-validation and DIRECT to *SyntheticControl* data set, in this case, algorithm converged after sampling just 130 out of 13'860 possible parameters combinations (>100x speedup).

D. Intuition behind SAX-VSM

First, by combining ***all*** SAX words extracted from ***all*** time series of single class into a ***single bag*** of words, SAX-VSM manages to capture and to "generalize" with PAA and SAX observed intraclass variability from a small training set.

Secondly, by normalizing time series subsequences and by discarding their original ordering, SAX-VSM is capable to capture and to recognize characteristic subsequences in distorted and corrupted by noise or signal loss time series.

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

Ultimately, the algorithm compares a set of subsequences extracted from an unlabeled time series with a weighted set of all characteristic subsequences representing the whole of a training class. Thus, an unknown time series is classified by its similarity not to a given number of "neighbors" (as in kNN or BOP classifiers), or to a fixed number of characteristic features (as in shapelet-based classifiers), but by the combined similarity of its subsequences to all known discriminative patterns found in a whole class.

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. We present a range of experiments assessing its performance and showing its ability to provide an insight into classification results.

A. Analysis of the classification accuracy

We evaluated our approach on 45 datasets whose majority was taken from benchmark data disseminated through UCR repository [21]. While all the details are available at the project's homepage [22], Table I compares the classification accuracy of SAX-VSM with previously published performance results of four competing classifiers: two state-of-the-art 1NN classifiers based on Euclidean distance and DTW, the classifier based on recently proposed Fast-Shapelets technique [11], and the classifier based on BOP [8]. We selected these particular techniques in order to position SAX-VSM in terms of classification accuracy and interpretability.

TABLE I: **Classifiers error rates comparison.**

Dataset	Num. of classes	1NN-Euclidean	1NN-DTW	Fast Shapelets	Bag Of Patterns	SAX-VSM
Adiac	37	0.389	0.391	0.514	0.432	**0.381**
Beef	5	0.467	0.467	0.447	0.433	**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	0.027	**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**

In our evaluation, we followed a train/test data split as provided by UCR. Train data were used in cross-validation experiments for optimization of SAX parameters using DIRECT. Once selected, optimal parameters were used to assess SAX-VSM classification accuracy on test data which is reported in the last column of Table I.

B. Scalability analysis

For synthetic datasets, it is possible to create as many time series instances as one needs for experimentation. We used the CBF [23] domain to investigate and assess the performance of SAX-VSM on increasingly large datasets.

In one series of experiments, we varied a training set size from 10 to 10^3, while the test set size remained fixed to 10^4 instances. For small training sets, SAX-VSM was found to be significantly more accurate than 1NN Euclidean classifier, but by the time we had more than 500 time series in a training set, there was no significant difference in accuracy (Fig. 3, left). As per the runtime 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. Note that SAX-VSM allows to perform training off-line and load weight vectors when needed - in this scenario, it performs classification significantly faster than 1NN Euclidean classifier (Fig. 3, center).

In another series of experiments we investigated the scalability of our algorithm with unrealistic training set sizes - up to 10^9 of instances for each of CBF classes. As expected, with the growth of a training set size, the growth curve of a total number of distinct SAX words for each class' dictionary showed significant saturation (similar to logarithmic curve)

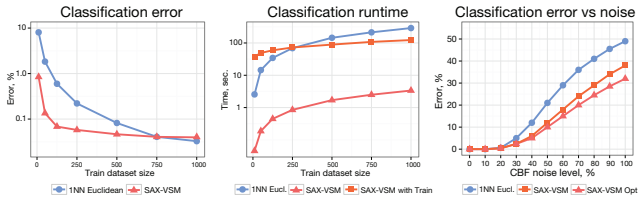

Fig. 3: Comparison of classification precision and run time of SAX-VSM and 1NN Euclidean classifier on CBF data. Left: SAX-VSM performs significantly better with limited amount of training samples. Center: while SAX-VSM is faster in time series classification, its performance is comparable to 1NN Euclidean when training time is accounted for. Right: SAX-VSM increasingly outperforms 1NN Euclidean with noise level growth (the random noise level grows up to 100% of CBF signal value)

peaking at about 10% of all possible words for selected PAA and alphabet sizes. This result reflects SAX-VSM ability to learn efficiently from large datasets: while SAX smoothing limits the generation of new words corresponding to relatively similar sub-sequences, the ***idf*** factor of the weighting schema (Equation 2) efficiently prunes SAX words (patterns) that are losing their discriminative power, i.e. those which appear in all classes.

C. Robustness to noise

Since the weight of each of the overlapping SAX words is contributing only a small fraction to a final similarity value, we hypothesized that SAX-VSM classifier might be robust to the noise and to the partial loss of a signal in test time series. Intuitively, in this case the cosine similarity between high dimensional weight vectors might not degrade significantly enough to cause a misclassification.

We investigated this hypothesis using CBF data. By fixing a training set size to 250 time series, we varied the standard deviation of Gaussian noise in CBF model. SAX-VSM outperformed 1NN Euclidean classifier with the growth of a noise level confirming our hypothesis (Fig.3, right). Further improvement of SAX-VSM performance was achieved by fine tuning of smoothing through a gradual increase of the SAX sliding window size proportionally to the growth of the noise level (*SAX-VSM Opt* curve, Fig.3 right).

D. Interpretable classification

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

Shapelet-based decision trees provide interpretable classification and offer insight into underlying data features [9]. Later, it was shown that the discovery of multiple shapelets provides even better resolution and intuition into the interpretability of classification [10]. However, as the authors noted, the time cost of multiple shapelets discovery in many class problems could be significant. In contrast, 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. Here, we show a few examples in which we exploit the subsequence weighting provided by our technique.

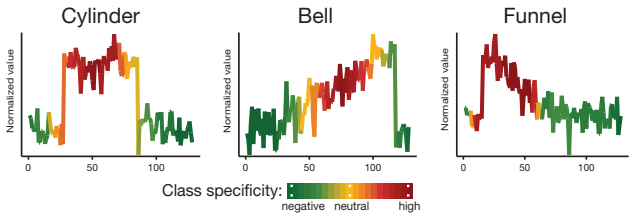

Fig. 4: An example of the heat map-like visualization of subsequence "importance" to a class identification. Color value of each point was obtained by combining *tf∗idf* weights of all patterns which cover the point. Highlighted by the visualization features correspond to a sudden rise, a plateau, and a sudden drop in Cylinder; to a gradual increase in Bell; and to a sudden rise followed by a gradual decline in Funnel, align exactly with CBF design [23].

1) Heatmap-like visualization: Since SAX-VSM outputs tf∗idf weight vectors of all subsequences extracted from a class, it is possible to find the weight of any arbitrary selected subsequence. This feature enables a novel heat map-like visualization technique that provides an immediate insight into the layout of "important" class-characterizing subsequences as shown in Figure 4.

2) Gun Point dataset: Following previous shapelet-based work [9] [10], we used a well-studied *GunPoint* dataset [24] to explore the interpretability of classification results. The class *Gun* of this dataset corresponds to the actors' hands motion when drawing a replicate gun from a hip-mounted holster, pointing it at a target for a second, and returning the gun to the holster; class *Point* 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, SAX-VSM was able to capture all distinguishing features as shown in Figure 5. The top weighted by SAX-VSM patterns in *Gun* class corresponds to fine movements required to lift and aim the prop. The top weighted SAX pattern in *Point* class corresponds to the "overshoot" phenomena, causing the dip in the time series [24], while the second to best pattern captures the lack of movements required for lifting a hand above a holster and reaching down for the prop.

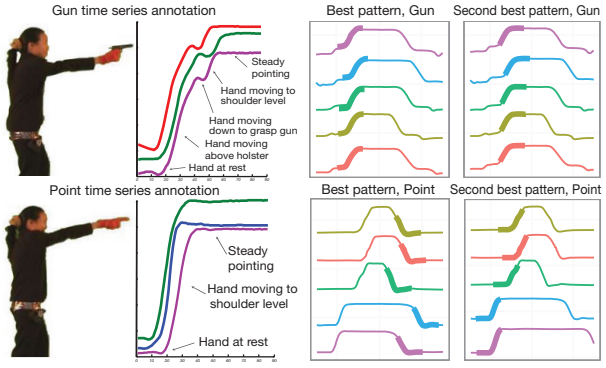

Fig. 5: Best characteristic subsequences (right panels, bold lines) discovered by SAX-VSM in *Gun/Point* dataset. Left panels show actor's stills and time series annotations made by an expert, right panels show locations of characteristic subsequences. Discovered patterns align exactly with previous work [9] [10]. (Stills and annotation used with a permission from E.Keogh)

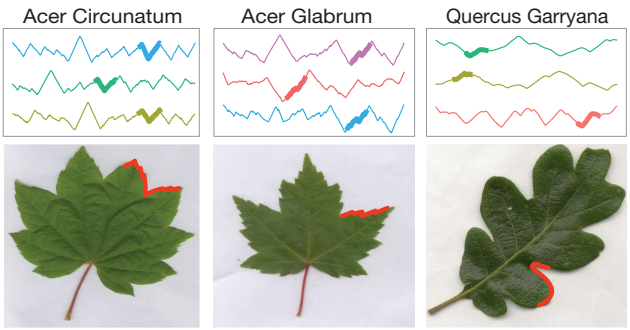

Fig. 6: Example of best characteristic subsequences (top panels, bold lines) discovered by SAX-VSM in *OSULeaf dataset*. Corresponding patterns: the slightly lobed shape and acute leaf tips of Acer Circinatum, the coarsely serrated leaf margins of Acer Glabrum, and the pinnately lobed leaf structure of Quercus Garryana align exactly with known in botany discrimination techniques [26].

3) OSU Leaf dataset: The *OSULeaf* dataset consist of curves obtained by color image segmentation and boundary extraction from digitized leaf images of six classes [25]. The author was able to solve the problem of leaf boundary curves classification with DTW, achieving 61% of classification accuracy. However, DTW provided little information about why it succeeded or failed, whereas SAX-VSM application yielded a set of class-specific characteristic patterns for each of the six classes which match known techniques for leaves classification based on their shape [26]. Figure 6 shows examples of best characteristic patterns of three classes. Our algorithm achieved an accuracy of 89%.

4) Coffee dataset: Similarly to the original work based on PCA [27], SAX-VSM highlighted intervals corresponding to Chlorogenic acid (best) and Caffeine (second to best) in both classes of Coffee spectrograms. Both chemical compounds are not only known to be responsible for the flavor differences in Arabica and Robusta coffees, but were previously proposed for industrial quality analysis of instant coffees [27].

VI. Conclusion and Future Work

We propose a novel interpretable technique for time series classification based on characteristic patterns discovery. We demonstrated that our approach is competitive with, or superior to, other techniques on a set 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. Finally, we outlined an efficient solution for SAX parameters selection.

For our future work, inspired by recently reported superior performance of multi-shapelet based classifiers [10], we prioritize modification of our algorithm for words of variable length. In addition, we explore SAX-VSM applicability for multidimensional time series.

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. DMKD, 26, 2, 275–309 (2013)
[2] Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. DMKD, 7, 4, (2003)

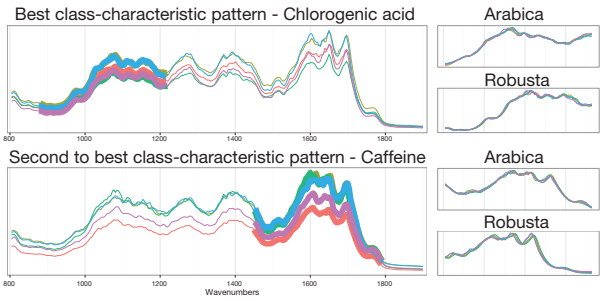

Fig. 7: Best characteristic subsequences (left panels, bold lines) discovered by SAX-VSM in Coffee dataset. Right panels show zoom-in view on these subsequences in Arabica and Robusta spectrograms. These patterns correspond to chlorogenic acid (best subsequence) and to caffeine (second to best) regions of spectra. This result aligns with the original work based on PCA [27] exactly.

[3] Salzberg, S.: On comparing classifiers: Pitfalls to avoid and a recommended approach. DMKD, 1, 317–328 (1997)
[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. 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 (2006)
[6] Sakoe, H. Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. on Acoustics, Speech and Signal Processing, 1, 43–49 (1978)
[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] Ye, L., Keogh, E.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. DMKD, 22, 149–182 (2011)
[10] Lines, J., Davis, L., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In Proc. 18th ACM SIGKDD, 289–297 (2012)
[11] Rakthanamanon, T., Keogh, E.: Fast-Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In Proc. SDM (2013)
[12] Patel, P., Keogh, E.,, Lin, J., Lonardi, S.: Mining Motifs in Massive Time Series Databases. In Proc. ICDM (2002)
[13] Salton, G., Wong, A., Yang., C.: A vector space model for automatic indexing. Commun. ACM 18, 11, 613–620 (1975)
[14] Björkman, M., Holmström, K.: Global Optimization Using the DIRECT Algorithm in Matlab. Adv. Modeling and Optimization, 1, 17-37 (1999)
[15] Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. DMKD, 15, 2, 107–144 (2007)
[16] Goldin D., Kanellakis, P.: On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. In Proc CP. 137–153 (1995)
[17] Keogh, E., Pazzani, M.: A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases. In Proc. PAKDD, 122–133 (2000)
[18] Keogh, E., Lin, J., Fu, A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In Proc. ICDM. 226–233 (2005)
[19] Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, Cambridge University Press (2008)
[20] Jones, D., Perttunen, C., Stuckman, B.: Lipschitzian Optimization without Lipschitz Constant. J. Optim. Theory Appl. 79, 1 (1993)
[21] 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/
[22] Paper authors. Supporting webpage: https://code.google.com/p/jmotif/
[23] Saito, N: Local feature extraction and its application using a library of bases. PhD thesis, Yale University (1994)
[24] Ratanamahatana, C., Keogh, E.: Making time-series classification more accurate using learned constraints. In SDM '04 (2004)
[25] Gandhi, A.: Content-Based Image Retrieval: Plant Species Identification. MS thesis, Oregon State University (2002)
[26] Dirr, M.: Manual of Woody Landscape Plants: Their Identification, Ornamental Characteristics, Culture, Propogation and Uses. Stipes Pub Llc, ed. 6 Revised (2009)
[27] 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)