Symbolic representation for time series

Sylvain W. Combettes, Charles Truong, Laurent Oudre
Université Paris Saclay, Université Paris Cité, ENS Paris Saclay, CNRS, SSA, INSERM, Centre Borelli
Gif-sur-Yvette, France
{firstname.name}@ens-paris-saclay.fr

***Abstract*—This study proposes a novel symbolic representation method for time series data called ASTRIDE. Unlike conventional symbolization techniques, our approach exhibits adaptability in two critical phases: firstly, during the temporal segmentation process, where it dynamically detects change-points in the signals, and secondly, in the sample quantization step, where it leverages quantiles. Additionally, we develop a data-driven edit distance measure for assessing the similarity of our symbolic representations. We demonstrate the performance of our representation compared to standard symbolizations on classification tasks. Our algorithm is evaluated on 86 univariate time series data sets with equal length sourced from the UCR Time Series Classification Archive. An open source GitHub repository is made available to reproduce the experiments in Python.**

***Index Terms*—Symbolic representation, time series, change-point detection, classification**

I. Introduction

Over the past decades, the increasing amount of available time series data has led to a rising interest in time series data mining. In many applications, the collected data take the form of complex time series which can be multivariate, multimodal, or noisy. A fundamental issue is to adopt an actionable representation that takes into account temporal information. In this regard, symbolic representations constitute a tool of choice [1]. Symbolic representations of time series are used for data mining tasks such as classification [1], [2], clustering [1], indexing [1], and anomaly detection [3], [4]. The domain applications include finance [5] or healthcare [6].

Briefly, most symbolization techniques follow two steps: a segmentation step where a real-valued signal $\boldsymbol{y} = (y_1, \dots, y_n)$ of length n is split into w segments, then a quantization step where each segment is mapped to a discrete value $\hat{y}_i$ taken from a set $\{\mathtt{a}_1, \dots, \mathtt{a}_A\}$ of A symbols. The resulting symbolic representation is the discrete-valued signal (or *symbolic sequence*) $\hat{\boldsymbol{y}} = (\hat{y}_1, \dots, \hat{y}_w)$. The set of symbols $\{\mathtt{a}_1, \dots, \mathtt{a}_A\}$ is usually called an *alphabet* or *dictionary*, and A is the *alphabet size*; the length w of the symbolic representation is called the *word length*. While there exist many high-level representations for time series, the two main advantages of symbolic representations are reduced memory usage, and often a better score on data mining tasks thanks to the smoothing effect induced by compression [1].

Related work: One of the most popular symbolic representations for time series is *Symbolic Aggregate approXimation (SAX)* [1]. In SAX, each signal is centered and scaled to unit variance, then split into w segments of equal length. Next, the means of all segments are grouped together in bins and each segment is represented by the bin where its mean falls into. The bin boundaries are chosen so that all symbols are equiprobable under the assumption that the means follow a standard Gaussian distribution. A SAX transformation of a signal taken from the UCR Time Series Classification Archive [7] is shown in Figure 1. However, uniform segmentation has flaws. As can be seen on the SAX representation of Figure 1, the two peaks around timestamps 280 and 330 are not detected. Uniform segmentation does not depend on the specific signal or data set at hand, but only on the input word length w. As for quantization, the Gaussian assumption of SAX can be inappropriate for some data sets. SAX considers that the symbols obtained after quantization will be equiprobable because all normalized time series follow a Gaussian distribution. While normalized time series that are independent and identically distributed do tend to follow a Gaussian distribution, this is not the case for the means per segment [8].

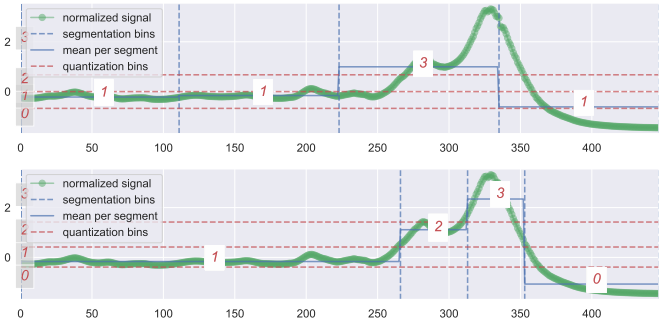

Fig. 1. Example of a SAX (top) and our method ASTRIDE (bottom) representations of a signal. The resulting symbolic sequence is `1131` for SAX, and `1230` for ASTRIDE.

Since the introduction of SAX, many variants and symbolization techniques have been proposed. Some variants focus on the feature(s) per segment. *ENhanced SAX (EN-SAX)* [5] builds a vector for each segment with the minimum, maximum, and mean values. These vectors are then clustered and a symbol is attributed to each cluster. *1d-SAX* [9] represents two features with only one symbol per segment. It uses linear regression to compute the mean and the slope of each segment then discretizes the mean and the slope separately using the same Gaussian assumption as in SAX. The final segment symbol is the combination of the mean symbol and the

S. Combettes is supported by the IDAML chair (ENS Paris-Saclay) and UDOPIA (ANR-20-THIA-0013-01). C. Truong is funded by the PhLAMES chair (ENS Paris-Saclay).

slope symbol. *Complexity-invariant SAX (CSAX)* [10] uses the symbolized mean and the real-valued complexity estimate, and thus holds two values per segment. The complexity estimate corresponds to the L_2-norm of the finite differences vector and quantifies a notion of dispersion.

Some symbolization procedures perform a non-uniform segmentation in order to better adjust to the signal. For instance, in *Adaptive Segmentation Based Symbolic Representations (SBSR)* [11], segment lengths adapt to the shape of the signal. *Adaptive SAX based on the Sum of Absolute Errors (ASAX_SAE)* [12] detects changes in the mean: it uses a bottom-up approach to reduce the approximation error of the PAA representation. *Adaptive SAX based on ENtropy (ASAX_EN)* [12] focuses on entropy and uses a top-down approach to find informative segments with high entropy.

Another category of symbolic representations use an adaptive quantization step in order to relax the Gaussian assumption on the data. *Genetic Algorithms-based SAX (GASAX)* [13] works as SAX, but the bin boundaries are determined through a genetic algorithm. *Adaptive SAX (aSAX)* [3] uses a uniform segmentation and K-means clustering for the quantization. *Adaptive Brownian Bridge-based Aggregation (ABBA)* [4] is adaptive for both the segmentation and quantization steps. For the segmentation, an adaptive piecewise linear continuous approximation of the signal is used. Each linear piece is chosen given a user-specified tolerance tol: when the value of tol increases, the resulting number of segments w decreases. The quantization step consists of a K-means clustering of the increments over the segment and the segment lengths. No distance is proposed on ABBA's symbolic sequences.

Rather than using a predefined distribution, some methods have tried to estimate it in a data-adaptive fashion. *Symbolic Fourier Approximation (SFA)* [14] is based on the Discrete Fourier transform (DFT). First, SFA selects the w Fourier coefficients of the lowest frequencies, and second, uses a procedure called Multiple Coefficient Binning (MCB) to quantize them. No distance on SFA's symbolic representations is introduced. *Distribution-Wise SAX (dwSAX)* [15] tackles non-Gaussian distributions. dwSAX estimates a data distribution of the PAA values using Kernel Density Estimation (KDE). KDE requires the choice of a kernel function and a bandwidth parameter. After KDE, dwSAX finds the quantization bins using the Probability Density Function (PDF) so that they create equal-sized areas under the curve. An improved version called *edwSAX* [16] has been proposed by the same authors.

Contributions: We describe a symbolic representation with adaptive segmentation and quantization, denoted *ASTRIDE (Adaptive Symbolization for Time seRIes DatabasEs)* as well as a compatible distance measure. Instead of using uniform segmentation, ASTRIDE performs adaptive segmentation, a.k.a. change-point detection [17], in order to capture salient events. Moreover, ASTRIDE does not rely on the Gaussian assumption for the quantization. Altogether, our approach circumvents the limitations of the methods described above. We also developed the *Dynamic General Edit Distance (D-GED)*, a new distance measure on symbolic representations which is based on the general edit distance.

II. Method

In this section, we decribe step-by-step the symbolization procedure (ASTRIDE) as well as the distance measure (D-GED). In the following, we consider as input a data set of N univariate time series with equal lengths n.

A. Adaptive segmentation step

As a preprocessing step, all times series in the data set are centered and scaled to unit variance. Then, the N signals of length n are segmented. To that end, all signals are stacked, producing a single multivariate signal of length n and dimension N. ASTRIDE applies multivariate change-points detection with a fixed number of segments on this high-dimensional signal. When w segments are chosen, the segmentation provides $w-1$ change-points that are the same for each univariate signal. Since the change-points are common to all (univariate) signals, this allows ASTRIDE to be memory-efficient. The lengths of each resulting symbolic sequence are the same (equal to w). For a given multivariate signal $\boldsymbol{y} = (y_1, \ldots, y_n)$ with n samples, change-point detection finds the $w-1$ unknown instants $t_1^* < t_2^* < \ldots < t_{w-1}^*$ where some characteristics (here, the mean) of $\boldsymbol{y}$ change abruptly. A recent review of such methods is given in [17]. In the context of ASTRIDE, the number of changes $w-1$ is chosen by the user: it is the desired number of regimes, meaning the length of the resulting symbolic sequences. The change-point algorithm estimates $\hat{t}_1, \ldots, \hat{t}_{w-1}$ which are the minimizers of a discrete optimization problem:

$$\left(\hat{t}_1, \ldots, \hat{t}_{w-1}\right) = \underset{(w, t_1, \ldots, t_{w-1})}{\arg\min} \sum_{k=0}^{w+1} \sum_{t=t_k}^{t_{k+1}-1} \|y_t - \bar{y}_{t_k:t_{k+1}}\|^2, \tag{1}$$

where $\bar{y}_{t_k:t_{k+1}}$ is the empirical mean of $\{y_{t_k}, \ldots, y_{t_{k+1}-1}\}$. By convention, $t_0 = 0$ and $t_w = n$. Formulation (1) seeks to reduce the error between the original signal and the best piecewise constant approximation. This problem is solved using dynamic programming which has a time complexity of $\mathcal{O}\left(Nwn^2\right)$ where N is the number of signals in the data set.

Figure 1 displays an example of an ASTRIDE representation of a signal, along with the SAX representation (for the same parameters w and A). Visually, compared to uniform segmentation, adaptive segmentation leads to more meaningful segments. For example, it detects that one segment is sufficient to approximate the signal from timestamp 0 to 250 and that there is a peak around timestamp 280 and another one around timestamp 330. It shows the importance of our adaptive segmentation scheme.

B. Adaptive quantization step

After segmentation, the means of all segments are computed and grouped into bins based on the empirical quantiles. Each segment is then symbolized by the bin it belongs to. This quantization step is similar to the MCB (Multiple Coefficient Binning) procedure of SFA [14]. Since the segments found

during the segmentation step correspond to mean shifts, it is reasonable to represent each segment by its mean value. The $A-1$ quantiles are calculated on the means of all segments of all signals in the data set, leading to A symbols. The time complexity of the quantization step (computing the means, the quantiles, and applying the binning) is $\mathcal{O}(Nw)$, where N is the number of signals in the data set. By design, all symbols are equiprobable. Figure 1 shows an example of an ASTRIDE representation. Compared to SAX, the bins of ASTRIDE represent the quantiles of the means per segment and are quite different from the ones of SAX. Recall that ASTRIDE is fitted on the whole training set and not on the displayed signal only.

C. The D-GED distance measure

We introduce Dynamic General Edit Distance (D-GED), a novel distance measure on symbolic representations. D-GED is compatible with symbolic sequences of equal or varying lengths. The distance measure D-GED is based on the general edit distance also known as the Levenshtein distance [18]. For two strings, the general edit distance is the minimal cost of a sequence of operations (insertions, deletions, and substitutions) that transform one string into the other. Note that D-GED is not necessarily a metric.

D-GED sets the operation costs of the general edit distance so that they incorporate the distance between individual symbols as follows. The substitution cost $\mathrm{sub}(\mathtt{a},\mathtt{b})$ for individual symbols `a` and `b` is the Euclidean distance between the mean $\mu_{\mathtt{a}}$ of the mean values attributed to symbol `a` and the mean $\mu_{\mathtt{b}}$ of the mean values attributed to symbol `b`:

$$\mathrm{sub}(\mathtt{a},\mathtt{b}) = \|\mu_{\mathtt{a}} - \mu_{\mathtt{b}}\|_2 \,. \tag{2}$$

For all characters, the insertion and deletion costs are set to $\mathrm{sub}_{\max}$, where $\mathrm{sub}_{\max}$ is the maximum value of the modified substitute costs given in (2). For the substitution cost, the intuition is that if symbols `a` and `b` are "very different", then the difference between $\mu_{\mathtt{a}}$ and $\mu_{\mathtt{b}}$ will be wider, and substituting them will have a larger cost in D-GED. Note that, by setting the insertion and deletion costs to $\mathrm{sub}_{\max}$, D-GED favors substitutions over insertions and deletions. The worst-case complexity to compute the D-GED of two symbolic sequences of lengths w_1 and w_2 is $\mathcal{O}(w_1 w_2)$.

The D-GED measure is not applied directly to the symbolic representation but to a replicated version. Indeed, when a method uses a non-uniform segmentation, the segments can have different lengths. Without taking into account the varying segment sizes, D-GED would compare (substitute or delete/insert) symbols corresponding to segments of different lengths. To prevent ASTRIDE from losing this information, we propose the following procedure. Denote by $\ell_1, \ldots, \ell_w$ the segment lengths obtained with our adaptive segmentation. By design, they are the same for all signals in the data set. Each segment length is divided by the minimum of all segment lengths and rounded to its nearest integer to obtain the normalized segment lengths $\widehat{\ell}_1, \ldots, \widehat{\ell}_w$. Then, the symbolic sequences are modified by replicating the symbol of the first segment $\widehat{\ell}_1$ times, then the symbol of the second segment $\widehat{\ell}_2$ times, etc. Finally, the D-GED measure between these replicated symbolic sequences is computed. As an example, consider the symbolic sequence from ASTRIDE depicted in Figure 1. The symbolic sequence without incorporating information about the segment lengths is `1230`. The segment lengths are $(266, 47, 40, 95)$ before normalization. The smallest segment has length 40 samples and, as a result, the normalized segment lengths are $(7, 1, 1, 2)$.

III. Experimental results

A. Experimental setup

Our method ASTRIDE is compared to SAX, 1d-SAX, and CSAX. One-Nearest Neighbor (1-NN) classification is used to compare the quality of both the symbolizations and the distance measures, as often done in the literature [19]. For ASTRIDE, the change-points and the quantization bins are learned on the training set. The adaptive segmentation step of ASTRIDE is implemented with the `ruptures` Python package [17]. The general edit distance in D-GED uses the `weighted-levenshtein` Python package[1]. SAX and 1d-SAX are implemented in the `tslearn` Python package [20]. CSAX is implemented from scratch.

Our comparison is limited to classification techniques directly based on symbolizations since our objective is to evaluate the relevance of this step itself and not to achieve the state-of-the-art time series classification. Hence, we exclude classifiers that are built on top of symbolic representations: bag-of-words and ensemble-based algorithms. More details on these techniques can be found in a recent review [19].

SAX, 1d-SAX, CSAX, and ASTRIDE all hold the following hyperparameters: the word length w and the alphabet size A. For all methods, we fix $A = 9$ as done in [4]. For 1d-SAX, it corresponds to $A_{\text{mean}} = 3$ and $A_{\text{slope}} = 3$. For fair comparison between methods, the value of w depends upon a fixed normalized space complexity, as some methods have a higher storage cost than others. The normalized space complexity is defined as the total number of bits to store all the symbolic sequences in the dataset, divided by the total number of time series in the dataset times the number of samples in a time series, for normalization purposes. Table I provides the normalized space complexities for each method. Note that, for ASTRIDE, contrary to SAX, 1d-SAX, and CSAX, the normalized space complexity of a dataset differs from the space complexity of a single symbolic sequence, because the change-points are shared across signals and thus memory-efficient. In our experiments, the normalized space complexity nsc is chosen in $\{0.8, 0.9, 1, 1.1, 1.2\}$, and, with $A = 9$ being fixed, this leads to different values of w per method and per dataset.

The evaluation metric for the classification is the test accuracy: percentage of correctly classified signals. The larger the accuracy for a given normalized space complexity and alphabet size, the better the symbolic representation. We use

[1] https://github.com/infoscout/weighted-levenshtein

TABLE I

NORMALIZED SPACE COMPLEXITIES FOR EACH SYMBOLIZATION METHOD, WITH $r = 64$ BITS THE NUMBER OF BITS TO STORE A REAL VALUE, N THE TOTAL NUMBER OF TIME SERIES IN THE DATASET, AND n THE NUMBER OF SAMPLES IN A TIME SERIES.

Method	Normalized space complexity
SAX	$\frac{w\lceil \log_2(A) \rceil}{n}$
1d-SAX	$\frac{w\lceil \log_2(A) \rceil}{n}$
CSAX	$\frac{w(\lceil \log_2(A) \rceil + r)}{n}$
ASTRIDE	$\frac{w(N\lceil \log_2(A) \rceil + r)}{Nn}$

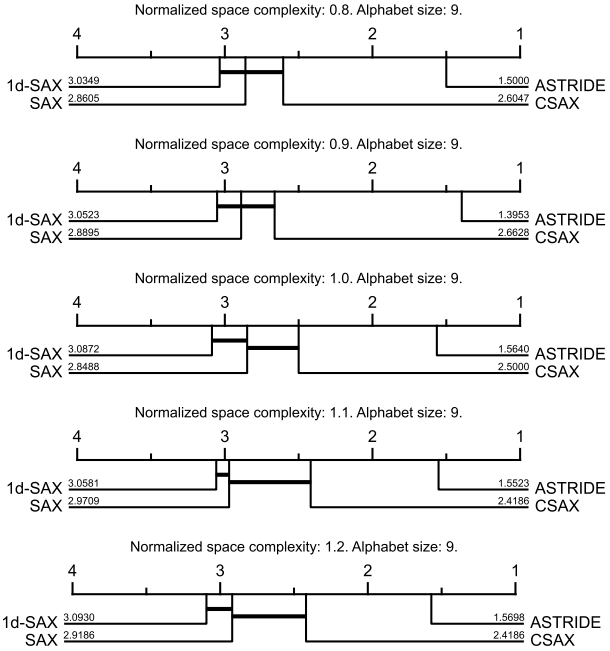

Fig. 2. Critical difference diagrams showing the pairwise statistical difference comparison of ASTRIDE and some popular symbolic representations on 86 datasets from the UCR archive.

statistical tests to compare the symbolizations methods. More precisely, we display critical difference diagrams[2] based on the Wilcoxon-Holm method to compare the test accuracies, as done in [21] to compare classifiers.

The scope of our comparisons concerns 86 data sets of the UCR Times Series Classification Archive [7] that have equal-length univariate signals. A Python implementation of ASTRIDE can be found in a GitHub repository[3] to reproduce the experiments.

B. Results

a) Comparing the methods: Figure 2 displays the results of our benchmark through critical difference diagrams that order classifiers by rank and where thick horizontal lines group into cliques sets of classifiers that are not significantly different. For all considered normalized space complexities, ASTRIDE is the best symbolization on average over the

[2]https://github.com/hfawaz/cd-diagram

[3]https://github.com/sylvaincom/symb-rep

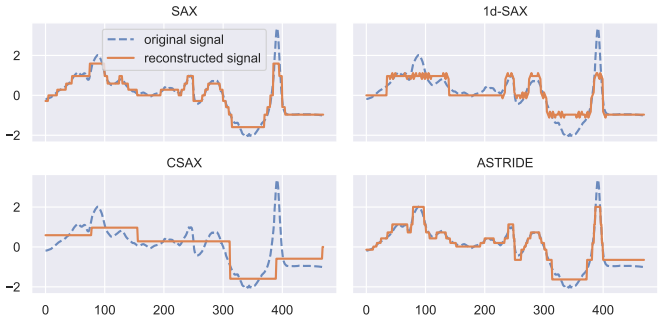

Fig. 3. Example of symbolization of a single signal from the Beef data set (UCR archive) of length $n = 470$ for several methods, with $A = 9$ and $nsc = 0.8$. For CSAX, the scaled complexity estimate values are: 0.005, 0.006, 0.003, 0.010, 0.016, and 0.017. Note that ASTRIDE is fitted on the whole training set (and not on the displayed signal only).

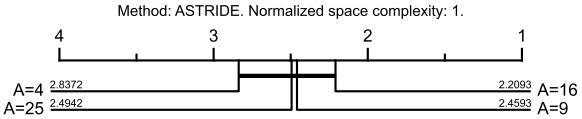

Fig. 4. Critical difference diagrams showing the influence of ASTRIDE's alphabet size on 86 datasets from the univariate UCR archive, for $nsc = 1$.

considered datasets. ASTRIDE performs better than SAX, 1d-SAX, and CSAX on the classification task. This shows that the proposed adaptive symbolization process, combined with the D-GED distance measure, is relevant in this classification context. For $nsc \in \{0.8, 0.9\}$, SAX, 1d-SAX, and CSAX are not significantly different as they are grouped into cliques. For $nsc \in \{1, 1.1, 1.2\}$, SAX and 1d-SAX, as well as 1d-SAX and CSAX are not significantly different. The same experiment was conducted for $A \in \{4, 16, 25\}$ and $nsc \in \{0.8, 0.9, 1\}$ on the 86 datasets (not shown here for lack of space) and leads to the same conclusion: ASTRIDE performs better.

An example of a symbolization of a single time series is given in Figure 3. The plotted symbolization corresponds to the reconstructed signal: approximating the original time series from its symbolized version. For SAX and 1d-SAX, the sample values on each segment of the reconstructed signal are based on the Gaussian bins, as done in `tslearn` [20][4][5]. For CSAX, it corresponds to the same reconstruction as the one done in SAX (with its corresponding word length), and we provide the real values of the scaled complexity estimate per segment separately. For ASTRIDE, it corresponds to the average of the mean values of the attributed symbols. As can be seen in Figure 3, CSAX is allowed less segments than SAX, 1d-SAX, and ASTRIDE, due to its additional real-value per segment. For ASTRIDE, we can see that the segmentation phase allows us to focus on the phenomenon of interest in the signal, thus to devote more memory to the encoding of salient events.

b) Influence of the parameters: For each method, we look into the influence of the normalized space complexity nsc ($nsc \in \{0.8, 0.9, 1\}$ with a fixed $A = 9$), and of the

[4]https://tslearn.readthedocs.io/en/stable/gen_modules/piecewise/tslearn.piecewise.SymbolicAggregateApproximation.html

[5]https://tslearn.readthedocs.io/en/stable/gen_modules/piecewise/tslearn.piecewise.OneD_SymbolicAggregateApproximation.html

TABLE II
PROCESSING TIMES ON THE SYMBOLIZATION AND 1-NN CLASSIFICATION ON THE ECG200 DATA SET COMPOSED OF 100 TRAINING SIGNALS AND 100 TEST SIGNALS OF LENGTH $n = 96$, WITH $w = 10$ AND $A = 9$.

Method	Symbolization (s)	1-NN classification (s)
SAX	0.02	0.11
1d-SAX	0.41	0.21
CSAX	0.58	0.25
ASTRIDE	0.29	0.17

alphabet size A ($A \in \{4, 9, 16, 25\}$ with a fixed $nsc = 1$). For ASTRIDE, the results of the influence of A are plotted on Figure 4: there is no value of A that strikes out. For $A = 9$ and $nsc \in \{0.8, 0.9, 1\}$, the null hypothesis (no difference in performance) over the entire classifiers (corresponding to each value of nsc) cannot be rejected. There is no value of A or nsc that leads to a significantly better performance (among the studied values), hence ASTRIDE seems to be quite robust to the value of its parameters. In the following, for SAX, 1d-SAX, and CSAX, we only report the significant differences. For SAX and for a fixed $nsc = 1$: the larger the value of A, the better the performance. For 1d-SAX and for a fixed $nsc = 1$: $A = 16$ or $A = 25$ perform better than $A = 4$ or $A = 9$. For CSAX and for a fixed $A = 9$, $nsc = 1$ performs better than $nsc = 0.8$ or $nsc = 0.9$.

C. Computational complexity

The processing times of the different methods are compared on the 1-NN classification task applied to the ECG200 data set, and are reported in Table II. We ran the experiments using Python 3.10.6 on a laptop under macOS 13.0.1 with Apple M1 Chip 8-Core CPU. Two durations are reported: the time to compute the symbolization for all time series in the data set, and the time to perform the actual 1-NN classification from the symbolized time series. As expected, ASTRIDE symbolization is more time-consuming than the non-adaptive ones (SAX for example).

IV. CONCLUSION

In this work, we have introduced a new symbolic representation of time series, ASTRIDE, as well as a novel distance measure on symbolic sequences, D-GED. Our experiments showed the quality of our symbolic representations on a classification task. ASTRIDE could also be employed as an intermediate step in classifiers such as BOSS [2] that uses SFA in the symbolization step, and involves overlapping sliding windows that increase the time and space complexities. ASTRIDE could also be employed for more advanced data mining tasks, such as pattern mining or anomaly detection. A multivariate extension, called d_{symb} [22], has recently been published by the authors.

REFERENCES

[1] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, 2007.

[2] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, pp. 1505–1530, 2014.

[3] N. D. Pham, Q. L. Le, and T. K. Dang, "Hot asax: A novel adaptive symbolic representation for time series discords discovery," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2010, pp. 113–121.

[4] S. Elsworth and S. Güttel, "Abba: adaptive brownian bridge-based symbolic aggregation of time series," *Data Mining and Knowledge Discovery*, vol. 34, pp. 1175–1200, 2020.

[5] P. M. Barnaghi, A. A. Bakar, and Z. A. Othman, "Enhanced symbolic aggregate approximation method for financial time series data representation," in *2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012)*. IEEE, 2012, pp. 790–795.

[6] A. Sant'Anna and N. Wickström, "A symbol-based approach to gait analysis from acceleration signals: Identification and detection of gait events and a new measure of gait symmetry," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1180–1187, 2010.

[7] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[8] M. Butler and D. Kazakov, "Sax discretization does not guarantee equiprobable symbols," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 1162–1166, 2015.

[9] S. Malinowski, T. Guyet, R. Quiniou, and R. Tavenard, "1d-sax: A novel symbolic representation for time series," in *International Symposium on Intelligent Data Analysis*. Springer, 2013, pp. 273–284.

[10] X.-M. T. Le, T. M. Tran, and H. T. Nguyen, "An improvement of sax representation for time series by using complexity invariance," *Intell. Data Anal.*, vol. 24, pp. 625–641, 2020.

[11] B. Hugueney, "Adaptive segmentation-based symbolic representations of time series for better modeling and lower bounding distance measures," in *Knowledge Discovery in Databases: PKDD 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 545–552.

[12] L. Djebour, R. Akbarinia, and F. Masseglia, *Variable-Size Segmentation for Time Series Representation*. Springer Berlin Heidelberg, 2023, pp. 34–65.

[13] M. M. Muhammad Fuad, "Genetic algorithms-based symbolic aggregate approximation," in *Data Warehousing and Knowledge Discovery*, A. Cuzzocrea and U. Dayal, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 105–116.

[14] P. Schäfer and M. Högqvist, "Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets," in *Proceedings of the 15th International Conference on Extending Database Technology*, ser. EDBT '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 516–527.

[15] M. Kloska and V. Rozinajova, "Distribution-wise symbolic aggregate approximation (dwsax)," in *Intelligent Data Engineering and Automated Learning – IDEAL 2020*. Cham: Springer International Publishing, 2020, pp. 304–315.

[16] V. R. Matej Kloska, *Towards Symbolic Time Series Representation Improved by Kernel Density Estimators*. Springer Berlin Heidelberg, 2021, pp. 25–45.

[17] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, Feb 2020.

[18] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10. Soviet Union, 1966, pp. 707–710.

[19] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: a review and experimental evaluation of recent time series classification algorithms," *Data Mining and Knowledge Discovery*, 2024.

[20] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020.

[21] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[22] S. W. Combettes, C. Truong, and L. Oudre, "An interpretable distance measure for multivariate non-stationary physiological signals," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2023, pp. 533–539.