Scalable Classification of Repetitive Time Series Through Frequencies of Local Polynomials

Josif Grabocka, Martin Wistuba, Lars Schmidt-Thieme
Information Systems and Machine Learning Lab
University of Hildesheim, Germany
Emails: {josif,wistuba,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract—Time-series classification has attracted considerable research attention due to the various domains where time-series data are observed, ranging from medicine to econometrics. Traditionally, the focus of time-series classification has been on short time-series data composed of a unique pattern with intra-class pattern distortions and variations, while recently there have been attempts to focus on longer, repetitive series composed of repeating local patterns. The primary contribution of this study relies on presenting a novel method which can detect local patterns in repetitive time-series via fitting local polynomial functions of arbitrary degrees. We express the repetitiveness degrees of time-series datasets via a novel measure. Furthermore, our method approximates local polynomials in linear time and ensures an overall linear running time complexity. The coefficients of the polynomial functions are converted to symbolic words via equivolume discretizations of the coefficients' distributions. The symbolic polynomial words enable the detection of similar local patterns by assigning the same words to similar polynomials. Moreover, a histogram of the frequencies of the words is constructed from each time-series' bag of words. Each row of the histogram enables a new representation for the series and symbolizes the occurrence of local patterns and their frequencies. In an experimental comparison against state of the art baselines on repetitive datasets, our method exhibits significant improvements in terms of prediction accuracy.

1 Introduction

Classification of time series is an important domain of machine learning due to the widespread occurrence of time-series data in real-life applications. Measurements conducted in time are frequently encountered in diverse domains ranging from medicine and econometrics up to astronomy. Therefore, time series has attracted considerable research interest in the last decade and a myriad of classification methods have been introduced.

Most of the existing literature on time-series classification focuses on classifying short time series, that is series which mainly incorporate a single long pattern. The research problem within this family of time-series data is the detection of pattern distortions and other types of intra-class pattern variations. Among other successful techniques in this category, the nearest neighbor classifier equipped with a similarity metric called Dynamic Time Warping (DTW) has been shown to perform well in a large number of datasets [1].

Nevertheless, few studies [2], [3], [4] have been dedicated towards the classification of time-series data which is repetitive and composed of many repeating local patterns. Repetitiveness has not been explicitly analyzed in the context of classifying time series. The degree of time-series repetitiveness in a dataset has been identified by common sense and visual inspections. We propose a novel repetitiveness measure that can objectively identify the repetitiveness score of a dataset. Using such an objective repetitiveness measure, we can successfully switch between recommending traditional methods for non-repetitive datasets, and, on the other hand, recommend our novel method for repetitive data.

Furthermore, this paper presents a novel method to classify repetitive time series composed of local patterns occurring in an unordered fashion and by varying frequencies. In this paper we extract frequencies of repeating patterns as a new series representation. Figure 1 provides a toy clustering illustration, in order to demonstrate the efficiency of the similarity using frequency representation.

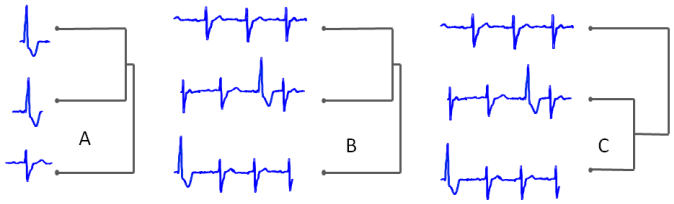


Fig. 1. Three non-repetitive patterns in A and three repetitive series in B and C. A and B use the Euclidean measure and C our proposed method.

For instance, series in sub-plot A are non-repetitive and therefore similarity measures like Euclidan distance are accurate and hard to beat. However, in the B & C sub-plots, the series are repetitive and composed of three

normal heart bits (top) or two normal beats plus one PVC (middle, bottom). Euclidean distance fails to detect similar series, as shown in B, because (i) the position of the PVC pattern varies and (ii) the number of beats varies. On the other hand, we can define a new representation as the frequencies of local patterns we can convert the series into frequencies as $\{(3\ 0), (3\ 1), (3\ 1)\}$, where the first index denotes the frequency of normal beats and the second the frequency of PVC. An L2 distance over the frequency representation yields the correct similarity pairings in C.

As will be detailed in Section 5.2 we propose a fast technique to fit sliding window content which has a linear run-time complexity. Our principle relies on detecting local polynomial patterns which are extracted in a sliding window approach, hence fitting one polynomial to each sliding window segment. Once the polynomial coefficients of each sliding window content are computed, we convert those coefficients into symbolic forms (i.e. alphabet words). The motivation for calling the method Symbolic Polynomial arises from that procedure. Such discretization of polynomial coefficients, in the form of words, allows the detection of similar patterns by converting close coefficient values into the same literal word. In addition, the words computed from the time series allow the construction of a dictionary and a histogram of word frequencies, which enables an efficient representation of local patterns.

We utilize an equivolume discretization of the distributions of the polynomial coefficients to compute the symbolic words, as will be explained in Section 5.3. Threshold values are computed to separate the distribution into equal volumes and each volume is assigned one alphabet letter. Consequently, each polynomial coefficient is assigned to the region its value belongs to, and is replaced by the region's character. Ultimately, the word of a polynomial is the concatenation of the characters of each polynomial coefficient merged together. The words of each time series are then stored in a separate 'bag'. A dictionary is constructed with each word appearing at least once in the dataset and a histogram is initialized with each row representing a time series and each column one of the words in the dictionary. Finally, the respective frequencies of words are updated for each time series and the rows of the histogram are the new representation of the original time series. Such a representation offers a powerful mean to reflect which patterns (i.e. symbolic polynomial words) and how often they occur in a series (i.e. the frequency value in each histogram cell).

The technical novelty of our method, compared to state-of-art approaches [3], [4] which utilize constant functions to express local patterns of series, relies on offering an expressive technique to represent patterns as polynomials of arbitrary degrees. Furthermore, we present a fitting algorithm which can compute the polynomial coefficients for a sliding window segment in linear time, therefore our method offers superior expressiveness without compromising run-time complexity.

Our experimental evaluation is composed on two parts and detailed in Section 6.5. Initially we analyze a large pool of 47 time-series datasets for identifying data having highly repetitive characteristics. Then we conduct experiments on prediction accuracy and run-time against state of the art baselines in the repetitive datasets. Our method outperforms the state of the art in all repetitive datasets with a statistically significance margin the majority of cases. We add experiments regarding the running time of the method and we show that our linear running time method is practically fast and feasible.

2 Related Work

2.1 Time-Series Representations

In order to understand the regularities embedded inside time-series, a large number of researchers have invested efforts into deriving and discovering time series representations. The ultimate target of representation methods is to encapsulate the regularities of time-series patterns by omitting the intrinsic noise. Discrete Fourier transforms have attempted to represent repeating series structures as a sum of sinusoidal signals [5]. Similarly, wavelet transformations approximate a time-series via orthonormal representations in the form of wavelets [6]. However, such representations perform best under the assumption that series contain frequently repeating regularities and little noise which is not strictly the case in real-life applications. Singular Value Decomposition is a dimensionality reduction technique which has also been applied to extract latent dimensionality information of a series [7], while supervised decomposition techniques have aimed at incorporating class information into the low-rank data learning [8].

In addition to those approaches, researchers have been also focused on preserving the original form of the time series without transforming them to different representations. Nevertheless, the large number of measurement points negatively influence the run-time of algorithms. Attempts to shorten time series by preserving their structure started by linearly averaging chunks of series points. Those chunks are converted to a single mean value and the concatenation of means create a short form known as a Piecewise Constant Approximation [9]. A more sophisticated technique operates by converting the mean values into symbolic form into a method called Symbolic Aggregate Approximation, denoted shortly as SAX [10], [11]. SAX enables the conversion of time-series values into a sequence of symbols and offers the possibility to semantically interpret series segments. Further sophistication of lower bounding techniques have advanced the representation method towards efficient indexing and searching [12], enabling large scale mining of time series [13]. Nonlinear approximations of the series segments have also been proposed. For instance least squares approximation of time series via orthogonal polynomials

have been proposed for segmentation purposes in a hybrid sliding/growing window scenario [14]. Throughout this paper we will propose a novel representation technique based on the utilization of polynomial functions of an arbitrary degree to approximate sliding windows of a time series. Our method brings novelty in converting the coefficients into literal representations, while the ultimate form is the frequency of the literal words constructed per each sliding window.

2.2 Time-Series Similarity Metrics

The time-series community has invested considerable efforts in understanding the notion of similarity among series. Time series patterns exhibit high degrees of intra and inter class variation, which is found in forms of noisy distortions, phase delays, frequency differences and signal scalings. Therefore, accurate metrics to evaluate the distance among two series play a crucial role in terms of clustering and classification accuracy. Euclidean distance, commonly known as the L_2 distance between vectors, is a fast metric which compares the distance values of every pair of points from two series, belonging to the same time stamp index. Despite being a fast metric of linear run-time complexity, the Euclidean distance is not directly designed to detect pattern variations. A popular metric called Dynamic Time Warping (DTW) overcomes the deficiencies of the Euclidean distance by allowing the detection of relative time indexes belonging to similar series regions. DTW achieves highly competitive classification accuracies and is regarded as a strong baseline [1]. Even though DTW is slow in the original formulation having a quadratic run-time complexity, still recent techniques involving early pruning and lower bounding have utilized DTW for fast large scale search [15].

Other techniques have put emphasis on the need to apply edit distance penalties for assessing the similarity between time series [16], [17]. Such methods are inspired by the edit distance principle of strings which counts the number of atomic operations needed to convert a string to the other. In the context of time series the analogy is extended to the sum of necessary value changes needed for an alignment. Other approaches have put emphasis on detecting the longest common subsequence of series, believing in the assumption that time series have a fingerprint segment which is the most determinant with respect to classification [18]. Detection of similarities in a streaming time-series scenario motivated attempts to handle scaling and shifting in the temporal and amplitude aspects [19].

2.3 Time-Series Classification

Classifying Non-repetitive Time Series

Classification of non-repetitive (classical, or general) time series has gathered considerable attraction in the literature. Among the initial pioneer methods and still one of the best performing ones is the nearest neighbor classifier accompanied by the DTW distance metrics, which constitute a hard-to-beat baseline [1]. Other powerful nonlinear classifiers like Support Vector Machines have been tweaked to operate over time series, partially because originally the kernel functions are not designed for invariant pattern detection and partially because DTW is not a positive semi-definite kernel [20]. Therefore the creation of positive semi-definite kernels like the Gaussian elastic metric kernel arose [21]. Another approach proposed to handle variations by inflating the training set and creating new distorted instances from the original ones [22].

Classifying Repetitive Time Series

The classification of repetitive time series focuses on long signals which are composed of one or more types of patterns appearing in unpredicted order and frequencies. Principally, the classification of those series has been mainly conducted by detecting the inner patterns and computing statistics over them. Few relevant studies have worked on repetitive data implicitly, because the notion of repetitiveness and its characteristics have not been exploited so far. For instance, underlying series patterns have been expressed as the motifs and the difference between the motif frequencies has been utilized [2]. Other approaches have explored the conversion of each sliding window segment into a literal word constructed by piecewise constant approximations and the SAX method [3], [4]. The words belonging to each time series are gathered in a 'bag' and a histogram of the words is constructed. The histogram vectors are the new representations of the time series. Such a technique has been shown to be rotation-invariant, because the occurrence of a pattern is not related to its position [4]. In contrast to the existing work, our novel study introduces an expressive histogram formulation based on literal words build from local pattern detection via polynomial approximations. Our model ensures scalability by computing in linear run-time.

3 Definitions and Notations

Alphabet

An alphabet is an ordered set of distinct symbols and is denoted by Σ. The number of symbols in an alphabet is called the size of the alphabet and denoted by $\alpha = |\Sigma|$. For illutration purposes we will utilize the Latin variant for the English language composed of the set of character symbols $\Sigma = (A, B, C, \ldots, Y, Z)$.

Word

A word $w \in \Sigma^*$ from an alphabet is defined as a sequence of symbols, therefore one sequence out of the set of possible sequences of arbitrary length l, defined as the Kleene star $\Sigma^* := \cup_{l=0}^{\infty} \Sigma^l$. For instance CACB is a word from the Latin alphabet having length four.

Polynomial

A polynomial of degree d having coefficients $\beta \in \mathbb{R}^{d+1}$, is defined as a sum of terms known as monomials. Each monomial is a multiplication of a coefficient times the a power of the predictor value $X \in \mathbb{R}^N$, as shown in Equation 1. The polynomial can also be written as a linear dot product in case we introduce a new predictor variable $Z \in \mathbb{R}^{N\times(d+1)}$ which is composed of all the powers of the original predictor variable X.

$$\hat{Y} = \sum_{j=0}^{d} \beta_j X^j \quad = \quad Z\beta, \qquad (1)$$
$$\text{where } Z \quad := \quad \left(X^0, X^1, X^2, \ldots, X^d\right)$$

Time Series

A time series of length N is an ordered sequence of numerical values and denoted by $S \in \mathbb{R}^N$. The special characteristics of time-series data compared to plain vector instances is the high degree of correlation that close-by values have in the sequence. A time-series dataset containing M instances is denoted as $T \in \mathbb{R}^{M\times N}$, assuming time series of a dataset have the same length.

Sliding Window Segment

A sliding window segment is an uninterrupted subsequence of S having length n denoted by $S_{t,n} \in \mathbb{R}^n$. The time index t represents the starting point of the series, while the index n the length of the sliding window, i.e. $S_{t,n} = [S_t, S_{t+1}, S_{t+2}, \ldots, S_{t+n-1}]$. The total number of sliding window segments of size n for a series of length N is $N-n$, in case we slide the window by incrementing the start index t by one index at a time.

4 Measuring Repetitiveness

Repetitive time series are characterized by patterns which periodically repeat. Therefore, if we cut the time series into non-overlapping sliding window segments, then each segment will be similar to many others. On the other hand, non-repetitive time series don't have repeating patterns, which means an arbitrary segment is likely different to the others. Subsequently, we define the repetitiveness measure as the average elastic distance among all pairs of non-overlapping segments in a series. Equation 2 presents the repetitiveness measure for a time-series dataset T, given a sliding window size n. The measure computes the average DTW similarity of each pair of the $\frac{M}{n}$-many segments, while the per-series scores are aggregated into the dataset repetitiveness measure. The correction factor $c = \frac{1}{N\frac{1}{2}\frac{M}{n}\left(\frac{M}{n}-1\right)n} = \frac{2n}{NM(M-n)}$, enables the metric to be invariant to the number of time-series, their length (number of segment pairs) and the size of the sliding window. The optimal repetitiveness measure, denoted $D(T)^*$ and Defined in Equation 3, is the smallest distance of the varying sliding window sizes. Further discussions on the relation of the measure to real-life datasets are elaborated in the experimental setup, Section 6.

$$D(T,n) = c\sum_{i=1}^{N}\sum_{j=1}^{\frac{M}{n}-1}\sum_{k=j+1}^{\frac{M}{n}} DTW(S^{(i)}_{jn,n}, S^{(i)}_{kn,n}) \qquad (2)$$
$$D(T)^* = \min_{n\in\mathbb{N}} D(T,n) \qquad (3)$$

Figure 2 shows time-series instances from three different datasets having various repetitiveness. As can be observed, the most repetitive series has the lowest D value. For the sake of illustration quality, only the first 1000 points of the RATBP instances are shown.

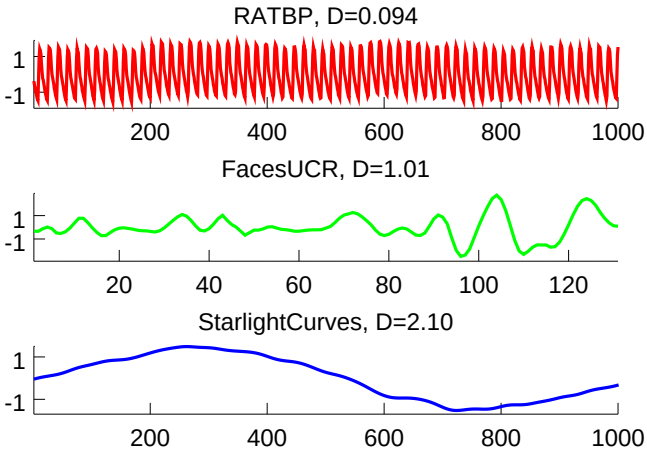


Fig. 2. Time-series having different repetitiveness

5 Proposed Method: Frequencies of Local Polynomials

5.1 Principle

The principle proposed in this study is to detect local patterns in a repetitive time series via computing local polynomials. The polynomials offer a superior mean to detect local patterns compared to constant or linear models, because they can perceive information like the curvature of a sub-series. Furthermore, in case of reasonably sized sliding windows the polynomials can approximate the underlying series segment without over-fitting. In this paper, we demonstrate that the polynomial fitting for the sliding window scenario can be computed in linear run-time. Once the local polynomials are computed, we propose a novel way to utilize the polynomial coefficients for computing the frequencies of the patterns. The polynomial coefficients are converted to alphabet words via an equivolume discretization approach. Such a conversion from real valued coefficients to short symbolic words allows for the translation of similar polynomials to the same word, therefore similar patterns can be

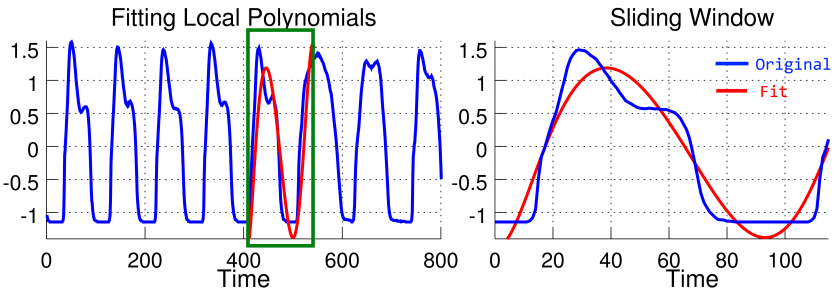


Fig. 3. Fitting a local polynomial of degree 8 to the sliding window region indicated by a green box. The plot on the right shows a scaled up version of the polynomial fit with coefficients $\beta = [8.4 \times 10^{-15}, -5.7 \times 10^{-12}, 1.6 \times 10^{-9}, -2.4 \times 10^{-7}, 2.15 \times 10^{-5}, -0.0011, 0.034, -0.36, -2.8]$ sorted from the highest monomial degree to the lowest. The time series on the left plot is a segment from the GAITPD dataset.

detected. We call such words symbolic polynomials. The words belonging to the time series are collected in a large 'bag' of words, (implemented as a list), then a histogram is created by summing up the frequency of occurrence for each words. Each row of a histogram encapsulates the word frequencies of time series, (i.e. frequencies of local patterns). A histogram row is the new representation of the time series and is used as a vector instance for classification.

5.2 Local Polynomial Fitting

Our method operates by sliding a window throughout a time-series and computing the polynomial coefficients in that sliding window segment. The segment of time series inside the sliding window is normalized before being approximated to a mean 0 and deviation of 1. The incremental step for sliding a window is one, so that every subsequence is scanned. Computing the coefficients of a polynomial regression is conducted by minimizing the least squares error between the polynomial estimate and the true values of the sub-series. The objective function is denoted by L and is shown in Equation 4. The task is to fit a polynomial to approximate the real values Y of the time series window of length n, previously denoted as $S_{t,n}$.

$$\begin{aligned} L(Y, \hat{Y}) &= ||Y - Z\beta||^2 \\ Y &:= [S_t, S_{t+1}, S_{t+2}, \dots S_{t+n-1}] \end{aligned} \tag{4}$$

Initially, the predictors are the time indexes $X = [0, 1, \dots, n-1]$ and they are converted to the linear regression form by introducing a variable $Z \in \mathbb{R}^{n \times (d+1)}$ as shown below in Equation 5.

$$Z = \begin{pmatrix} 0^0 & 0^1 & \dots & 0^d \\ 1^0 & 1^1 & \dots & 1^d \\ \vdots & \vdots & \dots & \vdots \\ (n-2)^0 & (n-2)^1 & \dots & (n-2)^d \\ (n-1)^0 & (n-1)^1 & \dots & (n-1)^d \end{pmatrix} \tag{5}$$

The solution of the least square system is conducted by solving the first derivative with respect the polynomial coefficients β as presented in Equation 6.

$$\frac{\partial L(Y, \hat{Y})}{\partial \beta} = 0 \quad \text{leads to} \quad \beta = \left(Z^T Z\right)^{-1} Z^T Y \tag{6}$$

A typical solution of a polynomial fitting is provided in Figure 3. On the left plot we see an instantiation of a sliding window fitting. The sliding window of size 120 is shown in the left plot, while the fitting of the segment inside the sliding window segment is scaled up on the right plot. Please note that inside the sliding window the time is set relative to the sliding window frame from 0 to 119. The series of Figure 3 is a segment from the GAITPD dataset.

Since the relative time inside each sliding window is between 0 and $n-1$, the predictors Z are the same for all the sliding windows of all time series. Consequently, we can pre-compute the term $P = \left(Z^T Z\right)^{-1} Z^T$ in the beginning of the program and use the projection matrix P to compute the polynomial coefficients β of the local segment Y as $\beta = PY$. Algorithm 1 describes the steps needed to compute all the polynomial coefficients of the sliding windows (starting at t) of every time series (indexed by i) in the dataset. For every time series we collect all the polynomial coefficients in a bag, denoted as $\Phi^{(i)}$. The outcome of the fitting process are the bags of all time series Φ. Please note that the complexity of fitting a polynomial to a sliding window is linear and the overall algorithm has a complexity of $O(M \cdot d \cdot n \cdot N)$, which considering $d << N, d << M$ and $n << N$, means linear run-time complexity in terms of N and M, that is $O(M \cdot N)$.

5.3 Converting Coefficients To Symbolic Words

The next step of our study is to convert the computed polynomial coefficients Φ from Algorithm 1 into words. The principle of conversion is to transform each of the

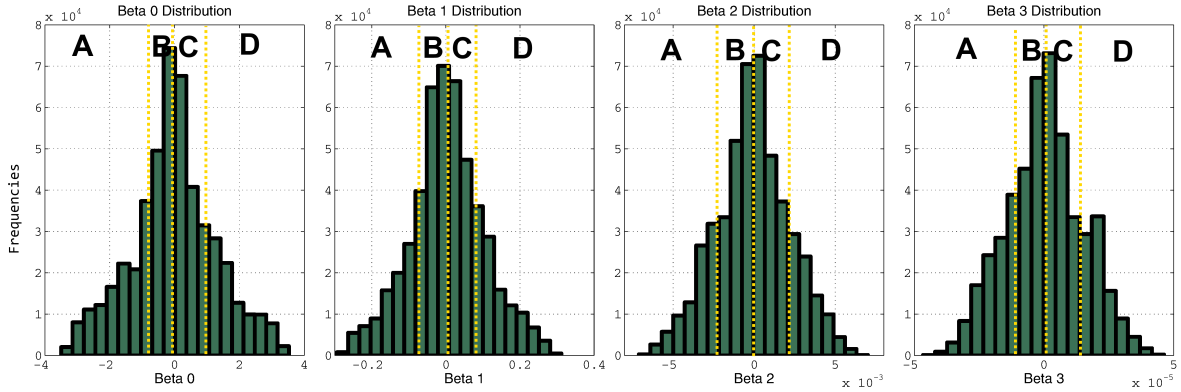


Fig. 4. Equivolume Discretization of Polynomial Coefficients. The illustration depicts the histogram distributions of a third degree polynomial fit over the sliding windows of the RATBP dataset. Each plot shows the values of a polynomial coefficient versus the frequencies. The alphabet size is four corresponding to the set $\{A, B, C, D\}$. The quantile threshold points are shown by dashed yellow lines.

Algorithm 1 Polynomial Fitting of a Time-Series Dataset

Require: Dataset $T \in \mathbb{R}^{M \times N}$, Sliding window size n, Polynomial degree d
Ensure: $\Phi \in \mathbb{R}^{M \times (N-n) \times (d+1)}$

$P \leftarrow \left(Z^T Z\right)^{-1} Z^T$
for $i \in \{1 \dots M\}$ **do**
  $\Phi^{(i)} \leftarrow \emptyset$
  for $t \in \{1 \dots N-n\}$ **do**
    $Y \leftarrow [S_t^{(i)}, S_{t+1}^{(i)}, \dots S_{t+n-1}^{(i)}]$
    $\beta \leftarrow PY$
    $\Phi^{(i)} \leftarrow \Phi^{(i)} \cup \{\beta\}$
  end for
end for
return $(\Phi^{(i)})_{i=1,\dots,M}$

$d+1$ coefficient of every β of Φ to one symbol. Therefore, the extracted words have lengths of $d+1$ symbols. For each of the β values of the polynomial coefficients we construct the histogram distribution and divide it into regions of equal volume as shown in Figure 4. In the image we have divided the histogram into as many regions as the alphabet size ($\alpha = 4$) we would like to utilize. Such a process is called an equivolume discretization. The thresholds between the regions are named quantile points and are defined in the figure as yellow lines. Dividing the histogram into α many regions is equivalent to sorting the coefficient values and choosing the threshold values corresponding to indexes multiple of $\frac{1}{\alpha}$. For instance, dividing the histogram into 4 regions for an alphabet of size 4 requires thresholds values corresponding to indexes at $\frac{1}{4}, \frac{2}{4}, \frac{3}{4}$ of the total number of values, which means that the each region has 25% of the values. Formally, let us define a sorted list of the j-th coefficient values regarding all window segments as $B^j \leftarrow \text{sort}\left(\{\beta_j \mid \beta \in \Phi^{(i)}, i = 1, \dots, M\}\right)$ and let the size of this sorted list be $s^j \leftarrow |B^j|$. Then the $(\alpha - 1)$ many threshold values are defined as $\mu_k^j \leftarrow B^j_{\lfloor s^j \frac{k}{\alpha} \rfloor}, \forall\, k \in \{1, \dots \alpha - 1\}$ and $\mu_\alpha^j \leftarrow \infty$.

Algorithm 2 Convert Polynomial Coefficients to Words

Require: Polynomial Coefficients Φ, Alphabet Size α
Ensure: $W \in \mathbb{R}^{M \times (N-n) \times (d+1)}$

{Compute the thresholds}
for $j \in \{0 \dots d\}$ **do**
  $B^j \leftarrow \text{sort}\left(\{\beta_j \mid \beta \in \Phi^{(i)}, i = 1, \dots, M\}\right)$
  $s^j \leftarrow |B^j|$
  $\mu_\alpha^j \leftarrow \infty$
  for $k \in \{1 \dots \alpha - 1\}$ **do**
    $\mu_k^j \leftarrow B^j_{\lfloor s^j \frac{k}{\alpha} \rfloor}$
  end for
end for
{Convert the coefficients to words}
$\Sigma \leftarrow \{A, B, \dots, Y, Z\}$
for $i \in \{1 \dots M\}$ **do**
  $W^{(i)} \leftarrow \emptyset$
  for $\beta \in \Phi^{(i)}$ **do**
    $w \leftarrow \emptyset$
    for $j \in \{0 \dots d\}$ **do**
      $k \leftarrow \text{argmax}_{k \in \{1,\dots,\alpha\}}\, \beta_j < \mu_k^j$
      $w \leftarrow w \circ \Sigma_k$
    end for
    $W^{(i)} \leftarrow W^{(i)} \cup \{w\}$
  end for
end for
return $(W^{(i)})_{i=1,\dots,M}$

Algorithm 2 describes the conversion of polynomial coefficients to symbolic form, i.e. words. The first phase computes the threshold values μ_k^j to discretize the distribution of each coefficient in an equivolume fashion.

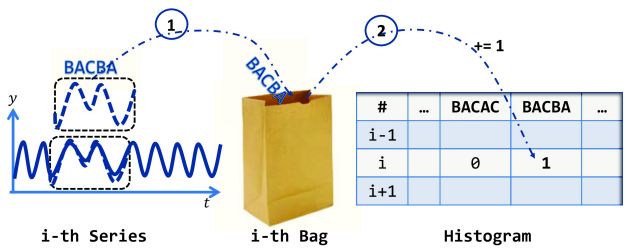


Fig. 5. Polynomial Words Collection and Histogram Population. In the first step all the words of a series are stored in one 'bag' per each time series. Then a histogram is initialized with a column of zeros for each word that occurs in any bag at least once. During the second step the word frequencies in the histogram are incremented upon each word found in a bag.

The second phase processes all the coefficients β of time series sliding windows and converts each individual coefficient to a character c, depending on the position of the β values with respect to the threshold values. The concatenation operator is denoted by the symbol $\circ$. The characters are concatenated into words w and stored in bags of words W. The complexity of this algorithm is also linear in terms of N and M. In case a linear search is used for finding the symbol index k, then the complexity is $O(M \cdot N \cdot \alpha)$, while a binary search reduces the complexity to $O(M \cdot N \cdot \log(\alpha))$. Yet, please note that in practice $\alpha << N$ and $\alpha << M$, therefore the complexity is translated to $O(M \cdot N)$.

5.4 Populating the Histogram

Once we have converted our polynomial coefficients and converted them to words, the next step is to convert the words into a histogram of word frequencies, as depicted in Figure 5. The steps of the histogram population are clarified by Algorithm 3. The first step is to build a dictionary D, which is a set of each word that appears in any time series at least once. Then we create a histogram H with as many rows as time-series and as many columns as there are words in the dictionary. The initial values of the histogram cells are 0. Each cell indicate a positive integer which semantically represent how many times does a word (column index) appear in a time series (row index). The algorithm iterates over all the words of a series and increases the frequency of occurrence of that word in the histogram.

Once the histogram is populated, then each row of the histogram denotes a vector containing the frequencies of the dictionary words for the respective time series. Practically the row represent what local patterns (i.e. words) exist in a series and how often they appear. For instance Figure 6 presents instances from the GAITPD dataset belonging to two types of patients in a binary classification task, healthy patients (blue) and Parkinson's Disease patients (red). In the left plot we show the original time series while on the right plot the histogram rows containing the polynomial words versus their frequencies. The parameters leading to the histogram for the GAITPD dataset are $n = 100, \alpha = 4, d = 7$. As can be inspected the original time-series offer little direct opportunity to distinguish one class from the other and the series look alike. Moreover the Euclidean distance of adjacent series in the figure show that the Euclidean classifier would mistakenly classify the third instance of the blue class. In contrast the histograms are much more informative and it is possible to observe frequencies of local patterns which allow the discrimination of one class from the other. A complete distance matrix between blue B and red R instances is shown in Table 1. As can be seen our histogram representations result in perfect accuracy in terms of nearest neighbor classification (**bold**), while the original series result in 2 errors.

Algorithm 3 Populate the Histogram

Require: Word bags W
Ensure: Histogram H
{Build the dictionary}
Ordered set dictionary $D \leftarrow \emptyset$
for $w \in W^{(i)}, \forall i \in \{1 \dots M\}$ **do**
  if $w \notin D$ **then**
    $D \leftarrow D \cup \{w\}$
  end if
end for
{Build the histogram}
$H \leftarrow \{0\}_{i=1,\dots,M\ j=1,\dots,|D|}$
for $w \in W^{(i)}, \forall i \in \{1 \dots M\}$ **do**
  Find j with $D_j = w$
  $H_{i,j} \leftarrow H_{i,j} + 1$
end for
return H

TABLE 1
Distances: Time-series (left), Histogram (right)

	B_1	B_2	B_3	R_1	R_2	R_3
B_1	-	1.6	**1.28**	1.4	1.29	1.4
B_2	1.6	-	1.4	1.5	1.4	**1.31**
B_3	1.28	1.4	-	1.4	**1.27**	1.4
R_1	1.4	1.5	1.4	-	**1.22**	1.4
R_2	1.29	1.4	1.27	**1.22**	-	1.26
R_3	1.4	1.31	1.4	1.4	**1.26**	-

	B_1	B_2	B_3	R_1	R_2	R_3
B_1	-	**1.7**	2.3	2.9	2.6	2.5
B_2	**1.7**	-	1.9	2.6	2.3	2.3
B_3	2.3	**1.9**	-	3.0	2.6	2.8
R_1	2.9	2.6	3.0	-	1.6	**1.4**
R_2	2.6	2.3	2.6	**1.6**	-	1.8
R_3	2.5	2.3	2.8	**1.4**	1.8	-

5.5 On the Importance of Linear Running Times

Repetitive time series are usually long, as is shown Section 6, therefore linear running times of algorithms are a must for computational feasibility. Our method ensures linearity in terms of running time by having an algorithmic complexity of $O(M \cdot N)$, which practically means that only a single pass over the data is needed. In order to give an illustration on the tyranny of non-linear running times, our method computes in 18.7 mins on the GAITPD dataset, while the full window DTWNN (quadratic complexity) requires 5166.6 mins or 3.9 days.

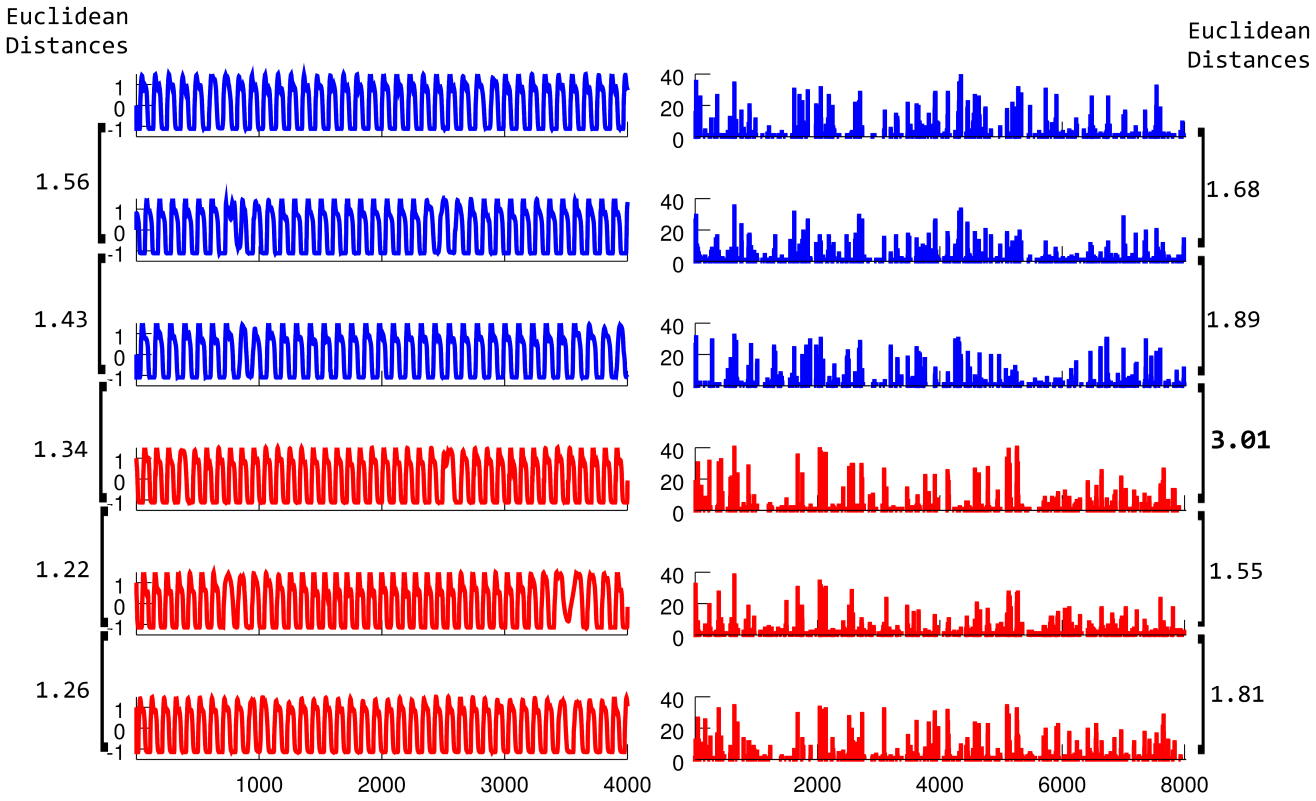


Fig. 6. GAITPD Dataset: Time series describing the gait in a few, randomly selected, control patients (blue) and Parkinson's disease patients (red). The original time series (x-axis is time and y-axis the normalized value) are displayed on the left column, while the respective histogram of each time series is shown on the right. The x-axis of the right plot represent 8026 words of the dictionary while the y-axis is the frequency of each word. The distances on the right show that histograms among a class have lower Euclidean distances, while the distance between the two histograms belonging to different classes is much higher, at a value of 3.01.

5.6 Classifier Selection

Our method produces a new time-series representation and is not bound against a specific classifier. However, the classifier that we are going to use for experimentation is the nearest neighbor method, which is a both a strong classifier in time-series classification [1] and is also used by state-of-art methods [4]. After converting the original time series into pattern frequency representations, in the form of rows of a histogram matrix, then each row will be treated as a vector instance. The nearest neighbor will utilize the Euclidean distance to compute the difference between histogram rows.

6 Experimental Setup

6.1 Selecting Repetitive Datasets

The UCR[1] collection of time-series datasets is a popular source for the time-series community. However the UCR collection is known to have "atomic" (single non-repetitive pattern) [23] time series datasets and not repetitive ones. In pursue of additional datasets exhibiting repetitiveness, we searched the Physionet [24] repository of medical signals and found seven labeled datasets (ECG2, RATBP, NESFDB, GAITPD, BIDMC, UMW, MVT). Another datasets (PAMAP) on time-series are utilized in a recent study [23]. Table 2 represents the repetitiveness scores of all the datasets (the smaller the more repetitive). The scores are computed using the repetitiveness measure of Section 4 and shown under the D column. We tried different sliding window sizes ($n \in \{5\%, 10\%, 15\%, 20\%\}$ of N) and selected the size which resulted in the best repetitiveness, i.e. smaller D. The time series of each dataset were normalized before computing the repetitiveness score. None of the UCR collection datasets have high repetitiveness according to our objective measure. Such a finding is in accordance to previous beliefs, given that the UCR collection contains "atomic" patterns [23], and further indications that those series are "short" [4]. Moreover, two of the added datasets PAMAP and NESFDB resulted to be even

1. www.cs.ucr.edu/~eamonn/time_series_data

TABLE 2
Repetitiveness Scores of 51 Time-Series Datasets

D	Dataset	Length	D	Dataset	Length	D	Dataset	Length	D	Dataset	Length
2.12	TwoLeadECG	82	1.95	ECGF.	136	1.56	Cricket_X	300	1.08	OSULeaf	427
2.09	Trace	275	1.92	Lighting7	319	1.56	Cricket_Z	300	1.02	FaceAll	131
2.09	StarL.	1024	1.82	Lighting2	637	1.53	Cricket_Y	300	0.98	FaceFour	350
2.08	Gun.	150	1.79	yoga	426	1.53	Haptics	1092	0.93	PAMAP	2000
2.08	Medical.	99	1.77	Symbols	398	1.52	CBF	128	0.92	MALLAT	1024
2.07	uWaveY	315	1.77	Adiac	176	1.41	OliveOil	570	0.91	FacesUCR	131
2.06	Inline.	1882	1.72	CinC.	1639	1.41	NESFDB	1800	**0.77**	**MVT**	**1021**
2.06	uWaveZ	315	1.71	Words.	270	1.38	Beef	470	**0.56**	**ECG2**	**2048**
2.03	wafer	152	1.71	Fish	463	1.37	synthetic	60	**0.54**	**UMW**	**1200**
2.01	uWaveX	315	1.66	Diatom.	345	1.32	Two_P.	128	**0.19**	**BIDMC**	**15000**
2.00	ItalyP.	24	1.64	Sony	70	1.26	Coffee	286	**0.15**	**ratbp**	**2000**
1.99	ECG200	96	1.60	50words	270	1.24	Chlorine.	166	**0.11**	**gaitpd**	**4000**
1.96	Mote.	84	1.58	Swedish.	128	1.22	SonyII	65			

less repetitive than some datasets of the UCR collection. Therefore, we are finally left with six highly repetitive datasets for analysis, namely ECG2, GAITPD, RATBP, BIDMC, UMW and MVT. Therefore, those datasets will be the testbed of our further experiments. Please note that repetitiveness is not directly related to the length of the time series, but rather to the amount of repetitive patterns inside them.

6.2 Descriptions of Highly Repetitive Datasets

TABLE 3
Statistics of Highly Repetitive Datasets

Dataset	Instances	Length	Classes
ECG2	250	2048	5
GAITPD	1552	4000	2
RATBP	180	2000	2
BIDMC	300	15000	15
UMW	60	1200	6
MVT	268	1021	3

All the experiments are based on the highly repetitive datasets retrieved from our objective repetitiveness ranking. Three of them are retrieved from Physionet, a repository of complex physiological signals primarily from the health care domain [24]. ECG2, BIDMC and MVT represents ECG recordings. GAITPD and UMV relate to the analysis gait cycles. The last dataset, named RATBP, represent the blood pressure recordings of mice.

The statistics of each dataset in terms of the number of instances, the length of each time series and the number of classes is summarized in Table 3. Please note that all the instances within one dataset have the same length, for a couple of reasons: (i) respecting the source formats (ECG2, BIDMC, MVT, all UCR collection), (ii) practicality in pre-processing (RATBP, GAITPD, NESFDB, UMW), and (iii) various traditional baselines like Euclidean-based nearest neighbor cannot trivially operate on variable series lengths. However, our histogram representation is very easily extensible to time-series of different sizes by normalizing/dividing the pattern frequencies of a particular series by its length.

6.3 Baselines

For notational sake, let us name our method as SymPol, meaning **Sym**bolic **Pol**ynomials, and refer to our method with the abbreviation form in the remaining sections. In order to evaluate the performance of SymPol, we compare against the following three baselines.

1) **BSAX** refers to the method of constructing bags of SAX words from time series through a sliding window approach. The words occurring in the bags are used to populate a histogram of frequencies [4]. A nearest neighbor method is applied to classify the histogram instances by treating the histogram rows as the new time-series representation. Comparing against this classifier will give chance to understand the benefit of polynomial approximation compared to constant models and will provide evidences on the state-of-art quality of the results.
2) **ENN** is the classical nearest neighbor classifier with the Euclidean L_2 loss as the distance metric. It operates over the whole time series, without segmenting the series for local patterns. The comparison against the plain nearest neighbor will show whether the detection of local patterns has more advantage than comparing the whole long series.
3) **DTWNN** differs from the Euclidean nearest neighbor classifier in defining a new distance metric for the comparison of two time series and performs well in time-series classification [1]. Dynamic Time Warping (DTW) operates by creating a matrix with all the possible warping paths, (i.e. alignment of pairs of indexes from two series), and selects the warping alignment with the smallest overall possible distance. DTW compares a full series without segmentations similarly to the Euclidean version of the nearest neighbor. Such comparison will both identify the benefits of the segmentation and also the benefits of local polynomials against global warping alignments.

TABLE 4
Error Rate Results

Dataset	SymPol		BSAX		ENN		DTWNN	
	μ (mean)	σ (st.dev.)	μ (mean)	σ (st.dev.)	μ (mean)	σ (st.dev.)	μ (mean)	σ (st.dev.)
ECG2	**0.0000**	0.0000	0.0080	0.0098	0.5480	0.0240	0.2120	0.0160
GAITPD	**0.0238**	0.0083	0.0548	0.0120	0.3924	0.0211	0.2468	0.0206
RATBP	**0.1333**	0.0272	0.1889	0.0111	0.4389	0.0272	0.3333	0.0994
BIDMC	**0.0000**	0.0000	**0.0000**	0.0000	0.6467	0.0356	0.2433	0.0226
UMW	**0.3167**	0.0372	0.4500	0.0745	0.7167	0.1247	0.6333	0.0667
MVT	0.4628	0.0535	0.4850	0.0169	0.2724	0.0303	**0.1982**	**0.0513**

TABLE 6
Run Time Results (seconds)

Dataset	SymPol		BSAX		ENN		DTWNN	
	μ (mean)	σ (st.dev.)	μ (mean)	σ (st.dev.)	μ (mean)	σ (st.dev.)	μ (mean)	σ (st.dev.)
ECG2	4.1	0.1	3.3	3.8	2.0	0.2	1651.5	52.9
GAITPD	1124.0	807.4	535.0	198.2	91.7	0.9	337K	20K
RATBP	27.2	36.7	1.8	0.8	0.7	0.0	1124.1	17.2
BIDMC	54.4	2.5	17.8	0.8	9.5	0.4	218K	1.4K
UMW	1.4	0.3	0.9	0.9	0.07	0.00	53.2	1.7
MVT	3.2	1.1	5.0	3.8	0.06	0.00	722.1	15.2

TABLE 5
Hyperparameter Search Results

Dataset	SymPol			BSAX		
	n	α	d	n	$\lvert w\rvert$	α
ECG2	100	4	3,4	100	4,6	4,6
GAITPD	100	4,6	7,8	100	6,7	6,8
RATBP	100	4,6	4,5,6,7	100	4,6,7	4,6,8
BIDMC	100	4	2	100	4	3
UMW	100,200	4,6,8	2,5	100,200,300	3,4,5,6	4,6,8
MVT	50,100,300	4,6,8	2,3,4	200,300	3,4,6	6,8

6.4 Reproducibility

The authors are devoted to promote full reproducibility, therefore the source code and all the datasets used in this paper are publicly available unconditionally [2].

Two different types of experiments were conducted in our study. The first empirical evidence focuses on the accuracy of our method with respect to classification of time series. The second experiment will analyze the computational run time of the methods. All the experiments were computed in a **five folds cross-validation** experimental setup. The time-series instances of each dataset were divided into 5 sets. In a circular fashion (repeated five times) each different set was once selected as the testing set, while the remaining four were used for training. Among the four sets used for training, one of them was selected as a validation set and the remaining three left as training. As a summary, all the combination of parameters were evaluated on the validation set and learned on the three training set, while the parameter values giving the smallest errors on the validation were selected. Those parameter values were finally evaluated over the testing set (learning from the three training sets) to report the final error rate.

2. http://fs.ismll.de/publicspace/SymbolicPolynomials/

A grid search mechanism was selected for searching the hyperparameter values. Our method SymPol requires the tuning of three parameters, the size of the sliding window n, the size of the alphabet α and the degree of the polynomials d. The size of the sliding window was selected among the range of $n \in \{50, 100, 200, 300, 400\}$, while the size of the alphabet was picked from $\alpha \in \{4, 6, 8\}$. Lastly the degree of the polynomial was picked to be one of $d \in \{1, 2, 3, 4, 5, 6, 7, 8\}$.

Similarly, the baseline named BSAX also requires the fitting of three hyperparameters. The length of a SAX word, denoted $|w|$, was selected from the range of $\{2, 3, 4, 5, 6, 7, 8, 9\}$, while the size of the alphabet was selected among the values $\{4, 6, 8\}$. The size of the sliding window is selected from a range of $\{50, 100, 200, 300, 400\}$, however those values were rounded to fit the length of a SAX word. For instance if the length of a SAX word is 3, then the size of the sliding window was rounded from 100 to 102 in order for the sliding window to be equally divisible into three chunks. The hyperparameter values found in our experiments are shown in Table 5, with ranges of multiple values due to different parameter searches per each different validation set. The datasets are normalized before usage, which is recommended in the realm of time series [15].

6.5 Results

The classification accuracy results of our experiments are presented in Table 4. For our method SymPol and all the baselines we show the mean and the standard deviation of the five fold cross-validation experiments as described in the setup section. The smallest error rate is highlighted in bold.

As can be clearly seen our method demonstrates a superiority in the majority of the datasets. SymPol wins in four datasets, namely ECG2, GAITPD, RATBP, UMW

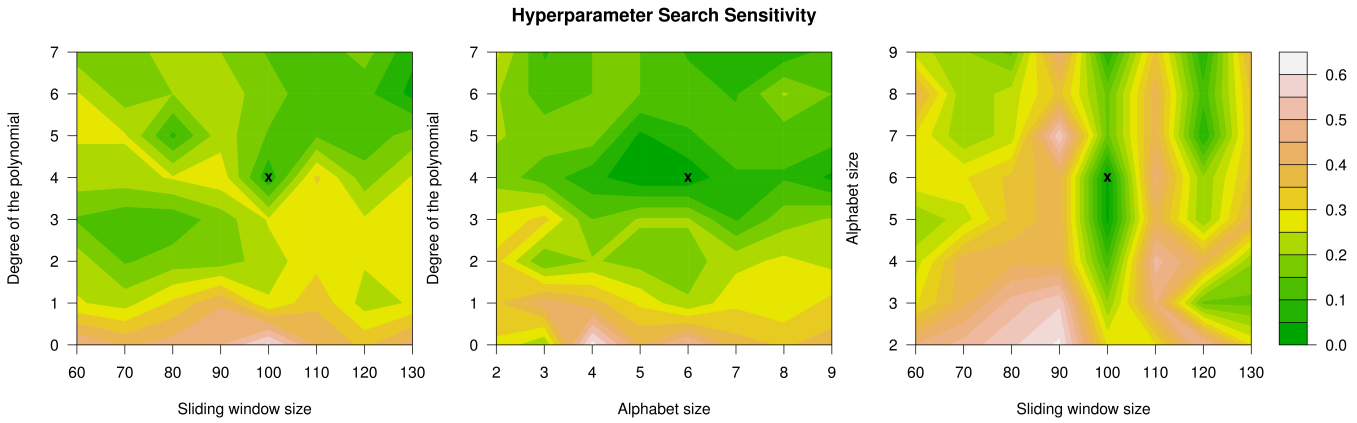


Fig. 7. Hyperparameter Search Sensitivity. Parameter search for one fold of the RATBP dataset resulting in the optimal values $n = 100, \alpha = 6, d = 4$. In the two dimensional illustration the third parameter (z-axis is invisible) is fixed to the optimal value.

TABLE 7
Statistical Significance - T-Test (p values)

Dataset	SymPol - BSAX	SymPol - ENN	SymPol - DTWNN
ECG2	0.1411	$5.8403 \cdot 10^{-11}$	$4.4212 \cdot 10^{-9}$
GAITPD	0.0054	$2.4784 \cdot 10^{-7}$	$5.8147 \cdot 10^{-6}$
RATBP	0.0028	$8.7033 \cdot 10^{-10}$	$3.9227 \cdot 10^{-8}$
BIDMC	-	$3.6036 \cdot 10^{-10}$	$2.2838 \cdot 10^{-8}$
UMW	0.0072	$2.6034 \cdot 10^{-4}$	$2.8225 \cdot 10^{-5}$
MVT	0.4015	$1.4826 \cdot 10^{-4}$	$6.6441 \cdot 10^{-5}$

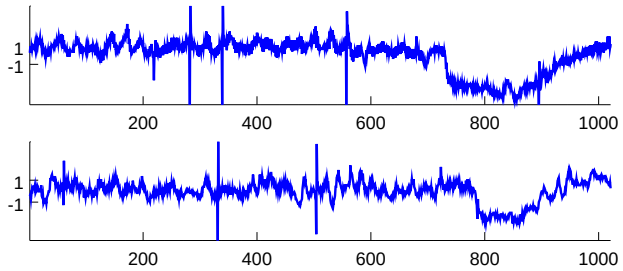


Fig. 8. Series of MR type (same class) in the MVT dataset

while co-sharing a win in BIDMC and loosing to DTW once in MVT. Our method performs perfectly in the ECG2 dataset by having 100% classification accuracy. In addition, SymPol reduces the error on the GAITPD dataset by 57% with respect the closest baseline, while on the RATBP dataset the error is reduced by 29%. BIDMC is a trivial dataset where both BSAX and SymPol have 100% accuracies. In addition we ran a statistical significance test in order to validate the results. Table 7 presents the p-values of a two tails T-Test, where results are statistically significant with a confidence of 95% for $p < 0.05$. Each cell measures the p-value of SymPol against a baseline. The p-value for BIDMC, comparing SymPol vs BSAX, is not defined because of division by zero, since both methods have zero means and zero standard deviations. Our method is statistically significant in 13 out of 18 cases, concretely 3/6 against BSAX, 5/6 against ENN, 5/6 against DTWNN. Please note that ECG2 and BIDMC are trivial datasets, where the baseline (BSAX) has a quasi-perfect score. In a trivial dataset, a method cannot outperform the baseline and the best it can do is to have 0 % error as well, which is what SymPol achieved. Finally we would like to explain the only dataset (MVT) where our method is loosing to DTWNN. As Figure 8 illustrates the MVT series are locally repetitive, but still have a global structure where DTW is hard to beat.

The second type of results represent the running times of the algorithms and is shown in Table 6. As can be clearly seen the Euclidean distance on the original dataset is the fastest method, which is a natural behavior because no processing is done over series to extract histograms. The BSAX method is the next in terms of speed due to the computational advantage of the constant model which requires only one pass over the data. SymPol is positively positioned in terms of run time. As already analyzed before, the algorithmic complexity is comparable to the BSAX except for an additional constant, which is the polynomial degree. In datasets like ECG2 and GAITPD the execution times are bigger by a small constant factor of two. The runtime constant in the RATBP dataset is higher because the hyperparameter search resulted to require a degree of 7. As a summary, we can clearly see that the method is practically very fast in terms of run time and is close even to techniques that use a constant model to fit local patterns. SymPol approximates polynomials of arbitrary degrees, instead of simple averages as BSAX, yet it does it in a competitive linear running time.

6.6 Sensitivity of Parameters

As presented in Section 6.4 our hyperparameter search technique is the grid search, where we scan for all the possible combination of one parameter's values to all

the possible values of other parameters. Figure 7 shows the sensitivity of SymPol's prediction accuracies against changes in the parameters. As can be easily deduced the error rate is nonlinear with respect to the parameter values of the method. Therefore, a grid search mechanism is practically suitable, because gradient based methods would have resulted in local optima while nonlinear optimization techniques would require much more computations than the grid. As can be seen in the plot, the grid search could successfully detect the global optimum in the region denoted by a mark.

7 Conclusion

In this study we presented a novel method to classify repetitive time series, which are composed of repeating local patterns. Local polynomial approximations are computed in a sliding window approach for each normalized segment under the sliding window. The computed polynomial coefficients are converted to symbolic forms (i.e. literal words) via an equivolume discretization procedure. Thresholds for the distribution of the values of each coefficient are determined to split the coefficient's histogram into equal regions and each region is assigned an alphabet symbol. In a second step all the polynomial coefficients are transformed into characters by locating them within the threshold values of the histogram and assigning the region symbol. The final literal representation of a polynomial is a word composed of the concatenation of each coefficient's character, in the order of the coefficient's monomial degrees. Once the bags of words are computed, a histogram is populated with the frequencies of each word in a time series. We presented a linear time technique to compute the polynomial approximation of a sliding window segment, while the overall method has a run time complexity which is linear in terms of the series points.

The classification accuracy of the nearest neighbor method utilizing the histogram rows that our method computed was compared against the performance of three baselines. Our method was the winning method in all the experiments, while achieving a statistically significant margin in the majority. Furthermore, empirical results demonstrate that our method has a practically feasible running time performance, comparable even to the fastest methods which require a single scan over the time series.

Acknowledgment

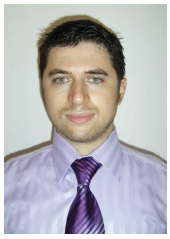
Funded by the Seventh Framework Programme of the European Commission, through project REDUCTION (# 288254). www.reduction-project.eu

The authors would like to express their gratitude to Prof. Eamon Keogh (University of California, Riverside), for his precious advisory assistance.

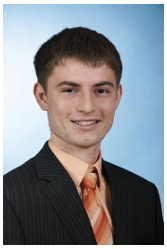
Note

A document containing an early version of the contents from this paper have been filed to the ArXiv file repository [25].

Authors



Josif Grabocka is a doctoral candidate affiliated with the Information Systems and Machine Learning Lab belonging to the Institute of Computer Science at University of Hildesheim, Germany. He obtained his Diploma in Computer Engineering at the Middle East Technical University of Ankara, Turkey in 2007 and received a Master degree in Applied Artificial Intelligence from the University of Dalarna, Sweden in 2010. The primary research interests are Data Mining and Machine Learning. He has published articles in recognized conferences on the field, such as ECML and AAAI.



Martin Wistuba is a doctoral candidate in the Information Systems and Machine Learning Lab, University of Hildesheim, Germany. He received his Master degree in Computer Science from the University of Paderborn, Germany in 2012. The primary research interests are Data Mining and Machine Learning. He has conducted research on applying factorization models to Artificial Intelligence domains.

Lars Schmidt-Thieme is a full professor leading the ISML Lab, University of Hildesheim, Germany. He obtained his Diploma in Mathematics at the University of Heidelberg in 1999 and received his PhD at the Department of Economics and Business Engineering at the University of Karlsruhe, Germany, in 2003. From 2003 to 2006 he was a professor at the Institute for Computer Science at the University of Freiburg, Germany. His main research interests are Machine Learning and Data Mining, especially classification and regression problems as well as pattern mining for complex data. He has published articles in top international conferences proceedings (e.g. IEEE ICDM, ACM KDD, UAI) and journals. He is member of program committees of international top conferences (e.g. ACM KDD, SIAM SDM, ECML) as well as executive board member of the German Classification Society.

REFERENCES

[1] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1542–1552, Aug. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1454159.1454226

[2] K. Buza and L. Schmidt-Thieme, "Motif-based classification of time series with bayesian networks and svms," in *GfKl*, ser. Studies in Classification, Data Analysis, and Knowledge Organization, A. Fink, B. Lausen, W. Seidel, and A. Ultsch, Eds. Springer, 2008, pp. 105–114.

[3] J. Lin and Y. Li, "Finding structural similarity in time series data using bag-of-patterns representation," in *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, ser. SSDBM 2009. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 461–477. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02279-1_33

[4] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *J. Intell. Inf. Syst.*, vol. 39, no. 2, pp. 287–315, 2012.

[5] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '94. New York, NY, USA: ACM, 1994, pp. 419–429. [Online]. Available: http://doi.acm.org/10.1145/191839.191925

[6] K.-P. Chan and A.-C. Fu, "Efficient time series matching by wavelets," in *15th International Conference on Data Engineering, Proceedings 1999*, 1999, pp. 126–133.

[7] J. A. Cadzow, B. Baseghi, and T. Hsu, "Singular-value decomposition approach to time series modelling," *Communications, Radar and Signal Processing, IEE Proceedings F*, vol. 130, no. 3, pp. 202–210, 1983.

[8] J. Grabocka, A. Nanopoulos, and L. Schmidt-Thieme, "Classification of sparse time series via supervised matrix factorization," in *AAAI*, J. Hoffmann and B. Selman, Eds. AAAI Press, 2012.

[9] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, 2001.

[10] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Min. Knowl. Discov.*, vol. 15, no. 2, pp. 107–144, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1007/s10618-007-0064-z

[11] L. Wei, E. Keogh, and X. Xi, "Saxually explicit images: Finding unusual shapes," in *Proceedings of the Sixth International Conference on Data Mining*, ser. ICDM '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 711–720. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2006.138

[12] J. Shieh and E. Keogh, "isax: indexing and mining terabyte sized time series," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 623–631. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401966

[13] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "isax 2.0: Indexing and mining one billion time series," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ser. ICDM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 58–67. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2010.124

[14] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online segmentation of time series based on polynomial least-squares approximations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2232–2245, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2010.44

[15] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD 2012. New York, NY, USA: ACM, 2012, pp. 262–270. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339576

[16] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 792–803. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316758

[17] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '05. New York, NY, USA: ACM, 2005, pp. 491–502. [Online]. Available: http://doi.acm.org/10.1145/1066157.1066213

[18] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 673–684.

[19] Y. Chen, M. Nascimento, B.-C. Ooi, and A. Tung, "Spade: On shape-based pattern detection in streaming time series," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, 2007, pp. 786–795.

[20] S. Gudmundsson, T. P. Runarsson, and S. Sigurdsson, "Support vector machines and dynamic time warping for time series," in *IJCNN*. IEEE, 2008, pp. 2772–2776.

[21] D. Zhang, W. Zuo, D. Zhang, and H. Zhang, "Time series classification using support vector machine with gaussian elastic metric kernel," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, 2010, pp. 29–32.

[22] J. Grabocka, A. Nanopoulos, and L. Schmidt-Thieme, "Invariant time-series classification," in *ECML/PKDD (2)*, ser. Lecture Notes in Computer Science, P. A. Flach, T. D. Bie, and N. Cristianini, Eds., vol. 7524. Springer, 2012, pp. 725–740.

[23] Y. Chen, B. Hu, and E. J. Keogh, "Time series classification under more realistic assumptions," in *SDM*. SIAM, 2013, pp. 578–586.

[24] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13), circulation Electronic Pages: http://circ.ahajournals.org/cgi/content/full/101/23/e215 PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[25] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Time-Series Classification Through Histograms of Symbolic Polynomials," *ArXiv e-prints*, Jul. 2013.