# Automatically Estimating iSAX Parameters

Nuno C. Castro and Paulo J. Azevedo
HASLab / INESC TEC
Department of Informatics
University of Minho, Portugal
{*castro,pja*}@*di.uminho.pt*

**Abstract.** The Symbolic Aggregate Approximation (iSAX) is widely used in time series data mining. Its popularity arises from the fact that it largely reduces time series size, it is symbolic, allows lower bounding and is space efficient. However, it requires setting two parameters: the symbolic length and alphabet size, which limits the applicability of the technique. The optimal parameter values are highly application dependent. Typically, they are either set to a fixed value or experimentally probed for the best configuration. In this work we propose an approach to automatically estimate iSAX's parameters. The approach – AutoiSAX – not only discovers the best parameter setting for each time series in the database, but also finds the alphabet size for each iSAX symbol within the same word. It is based on simple and intuitive ideas from time series complexity and statistics. The technique can be smoothly embedded in existing data mining tasks as an efficient sub-routine. We analyze its impact in visualization interpretability, classification accuracy and motif mining. Our contribution aims to make iSAX a more general approach as it evolves towards a parameter-free method.

**Keywords.** Time Series, Data Mining, Representation, iSAX, Parameters.

## 1. Introduction

The large dimension of time series databases has generated the need for scalable algorithms to mine these data, including indexing [Agrawal et al., 1995, Camerra et al., 2010, Faloutsos et al., 1994], classification [Geurts, 2001], clustering [Kalpakis et al., 2001], anomaly detection [Dasgupta and Forrest, 1996], and motif discovery [Castro and Azevedo, 2010, Chiu et al., 2003, Mueen et al., 2009]. The key to the efficiency of these algorithms is to select a suitable representation of the data [Ding et al., 2008, Lin et al., 2003]. Typically the data do not fit into main memory and disk I/O remains very inefficient [Lin et al., 2007], making the task of mining massive datasets difficult. Time series representations reduce the large size of the data while maintaining its overall characteristics [Ding et al., 2008]. From the plethora of existing time series representations [Faloutsos et al., 1994, Chan and Fu, 1999, Keogh et al., 2001, Chakrabarti et al., 2002, Korn et al., 1997, Ding et al., 2008], the Symbolic Aggregate Approximation (iSAX) [Shieh and Keogh, 2008, Lin et al., 2003, Lin et al., 2007] has been widely used in the time series data mining community [Lin et al., 2007] as it is symbolic, reduces the size of the time series, allows lower bounding, is space efficient, and robust to noise and missing values. In practice, iSAX converts a time series of length $n$ to a symbolic sequence of $l$ symbols from an alphabet of size $a$. First, it divides the original series in $l$ segments of size $n/l$. Then, it splits the amplitude of the series into equiprobable intervals, according to the Gaussian distribution, by using a set of breakpoints (Table 1). Finally, each segment is averaged and assigned a symbol according to the interval where the segment's average is. This process is shown in Fig. 1.

It can easily be observed that the best number of segments ($l$) and the segment's alphabet size ($a$) is specific to each time series in the database. Nevertheless, a more generic approach has been practiced by iSAX adopters: both parameters are typically set to a fixed value (e.g. 8) [Ding et al., 2008, Lin et al., 2007]. This means that the recipe is to convert each time series to a length 8 iSAX word, regardless of its origin, size and behavior. Each symbol should be selected from an alphabet of 8 possible characters. This recipe does not generalize well, as time series (and databases) have their own intrinsic dimensionality and cardinality. The crucial step of finding the best parameter setting still remains a "black art" [Hu et al., 2011].

The effect of using a fixed representation is shown in Figure 2 where a time series from the power demand dataset (available at [Castro and Azevedo, 2011]) is shown.

To get an intuition on how much information iSAX is retaining from the original time series, the original time series is removed from the bottom part of the image and only the representation is shown. It can be observed that for this particular example, using fixed iSAX (bottom) does not provide a good visual intuition on the original time series' shape. The image highlights the fact that a fixed length and alphabet size should not be used, as one cannot expect that every time series nicely matches a fixed parameter setting. An automatic approach to automatically derive a good parameter setting for each time series is required.

The current alternative to setting fixed parameters is to experimentally re-run the algorithm until the best parameter setting is found. In large datasets, to test all possible combinations of parameters is simply unfeasible [Castro and Azevedo, 2011]. For example, building an index with 1000000 time series can take as much as 6 days [Camerra et al., 2010]. Also, in unsupervised data mining tasks (e.g. clustering) it may not be possible to evaluate what the best parameter setting is, even by testing all possible configurations. Furthermore, to use large alphabet sizes is not space-efficient. The only solution is to use a fixed parameter setting regardless of the dataset, which is not a correct solution for all time series, as we have highlighted.

In this work, we introduce a novel technique to automatically derive parameters for the iSAX representation. This technique, referred as AutoiSAX, finds the symbolic length that best describes each time series in the database. Furthermore, it also finds the alphabet size for each iSAX symbol individually, as in iSAX each symbol can have a different alphabet size (it is this feature that allows iSAX to create time series indexes). Different sections of the series can behave differently (Figure 2). Similarly to the symbolic length parameter, using the same alphabets for all the symbols in the same series can lead to incorrect results.

The technique is based on two observations.

Regarding the symbolic length parameter, a more complex time series should need a larger number of iSAX symbols to capture its larger number of "peaks and valleys". A simpler time series may only need a handful of symbols to be represented (e.g. a constant time series would only require one symbol).

To explain the intuition for the alphabet parameter, one can think of the smallest rectangle that can visually enclose a time series section. A section with a large standard deviation requires a larger rectangle to contain it. On the other hand, a small rectangle should be enough to enclose a time series section presenting a low standard deviation (almost constant behavior). Larger rectangles correspond to an alphabet with fewer symbols, each one taking a large chunk of the amplitude space. Smaller rectangles to a larger alphabet, each one taking only a small portion of the available amplitude space.

We incorporate these observations into iSAX using two simple techniques. To capture a series complexity the Complexity Estimate (CE) [Batista et al., 2011] is used. It is computed by simply "stretching" the time series into a straight line and then measuring the line's length. It has been experimentally compared to twelve other (more elaborate) measures such as the Kolmogorov complexity and entropy with competitive results [Batista et al., 2011]. Computing this measure is intuitive, requires low time and space complexity, and zero parameters. Standard deviation is used to calculate a segment's variability. Note that zero extra space is required to add this information to the iSAX representation. That magic is to take advantage of the actual parameters we are trying to optimize: symbolic length and alphabet size, to encode the information.

In Figure 3, AutoiSAX is used to derive the representation's parameters. The generated symbolic length captures the series behavior more accurately than the fixed parameter. It also automatically generates a different alphabet size for each symbol, as they clearly have different behaviors. To fit bigger standard deviations' segments larger rectangles are required (coarser resolution). For smaller deviation segments, smaller rectangles suffice (finer resolution). Using AutoiSAX, it is straightforward to perceive the original time series shape by looking only at its representation (bottom). This is one of the primary goals of time series representations. We do not intend to show that larger parameters generate better representations. For example, in the automatic representation there are alphabet sizes and lengths smaller than the fixed ones. Our aim is solely to automatically determine the parameters, regardless of the relative order when compared to the fixed representation.

The main contribution of this work is that it enables automatic efficient parameter generation for iSAX. Up until now, brute-force search was required. It also automatically derives, for the first time in the literature, different alphabet sizes within the same iSAX word. Previously, the use of different intra-word resolutions was limited to indexing, as it is untenable and cumbersome to manually define multiple resolutions for different symbols of the same iSAX word. In practice, our technique fully exploits iSAX capabilities to other mining tasks. Besides the parameter estimation is a major feature per-se, advantages in visualization, classification, motif discovery performance are also obtained.

This section is organized as follows: in section 2 necessary background and notations are introduced. Section 3 analyzes existing SAX extensions and parameter estimation techniques. The AutoiSAX approach is presented in section 4. The experimental analysis of our approach is laid in section 5. In section 6 we discuss the main work breakthroughs and discuss future work.

## 2. Background and Notation

In this section we introduce some notations and useful definitions.

**Definition 1.** A word $W = w_1 w_2 \dots w_l$ is the symbolic representation of a time series $T$, with $w_i \in \sum$.

The symbolic length of word $W$ is $l$. The $\sum$ is the representation alphabet. For example, $\sum = a, b, c, d$. The alphabet size is called the representation cardinality or resolution. The representation of time series S by a generic representation technique $R$ is denoted by $R(T) = W$. For the scope of this work, the matching between two time series $T_1$ and $T_2$ can be defined as follows.

**Definition 2.** Time series $T_1$ and $T_2$ match if $R(T_1) = R(T_2)$.

In this work we explore an approach that has shown to be general and competitive in many domains and problems available in the literature, as experimentally shown in [Ding et al., 2008] – iSAX. For completeness, we also introduce its first version – SAX – in this chapter. As a symbolic approximation, SAX takes two parameters $l$ and $a$, and converts the raw time series $T$ of length $n$ into a sequence of $l$ symbols (word) with an alphabet size $a$. This operation is represented by the following notation: $SAX(T, l, a)$, or simply $SAX(l, a)$, and operates as follows: First, the dimensionality of the time series is reduced by dividing it into $l$ segments (word length) with the same length $n/l$ using the Piecewise Aggregate Approximation (PAA) algorithm. This algorithm assigns to each segment its average value. Then, the amplitude of the time series is divided into $a$ intervals, so that a symbol can be assigned to each interval. To obtain equiprobable intervals, $a - 1$ breakpoints are used. They produce the same area under the Normal curve, as shown in Figure 1. These breakpoints can be obtained by using a statistical table and are shown in Table 1 for alphabet sizes 2, 4, 8, 16 and 32. Finally, symbols are obtained from the intervals. The segments below the smallest breakpoint are assigned the 0 symbol. The segments between the first and second breakpoints the symbol 1, and so forth.

The iSAX representation extends classic SAX by allowing different alphabet sizes within the same word. To avoid ambiguity, the resolution of each symbol needs to be clearly stated in the iSAX word. For example, $W = 2^4, 0^8, 7^8, 9^{16}$. This enhancement enables the creation of a time series index. In practice, the use of this feature has been limited to indexing, as it is both cumbersome and untenable to manually set the resolution of each symbol. The problem is further aggravated by possibly having millions of iSAX words in the database. This is a central feature of our work. As we automatically calculate each symbol's alphabet size, our approach effectively enables to fully exploit iSAX capabilities in other tasks such as classification, motif discovery and visualization. For consistency, and because iSAX subsumes classical SAX, we will only refer iSAX for the remainder of the paper. The exception will be section 3, as we refer works proposed before iSAX was introduced.

## 3. Related Work

The iSAX representation has been widely used in a large range of application domains and data mining tasks [Lin et al., 2007]. Surprisingly, maybe due to the generality of the original approach, a very small number of works attempt to modify the actual representation technique [Lin et al., 2007]. In [Lkhagva et al., 2006], SAX is extended by adding two points to each symbol: the minimum and maximum points of that segment. The reasoning behind is that in some domains, e.g. financial data, the averaging process behind PAA can miss some important information. Adding maxima and minima information to segments yields further insight about the series that just the average simply cannot perceive. Preliminary results show the usefulness of the technique in some domains. Their goal is in principle similar to ours. We implicitly encode segment variability information in the representation. Rather than obtaining it through extreme points, we use the standard deviation. Also, we encode it in the original representation (accessing higher or lower resolutions), rather than using extra space along each symbol. This enables the user to capture more information about the raw time series in a space efficient way. Pham et al. [Pham et al., 2010] attempt to improve SAX by adding a pre-processing step where adaptive, rather than fixed (equiprobable) intervals, are discovered using clustering techniques. Experimental results show that it can outperform iSAX. In [Yankov et al., 2007] a motif discovery algorithm is proposed. The algorithm's parameter setting (including SAX's) is experimentally estimated in a smaller subset of training data. We aim to automatically derive the best parameter setting by using analytical tools, rather than using an iterative approach.To the best of our knowledge, there is no iSAX extension to automatically derive the parameters without experimentally probing the best configuration. An approach to find the best representation, cardinality and dimensionality of time series using the Minimum Description Length (MDL) principle has been introduced in [Hu et al., 2011]. This algorithm can be used to select the best parameter setting for a particular representation. In practice, it calculates the reduced description length – the number of bits required to rebuild the raw time series – for every possible parameter configuration and selects the parameter configuration with the minimum length. The output configuration is the intrinsic cardinality and dimensionality of the time series. This approach is more generic than ours, as it can also detect the best model to use among the DFT, APCA, PLA representations, while also being parameter-free. AutoiSAX is specifically tailored to find the best parameter configuration of the iSAX representation. We analytically compute the best parameter configuration, rather than taking a brute-force approach, resulting into better computational complexity (section 5). Furthermore, our approach yields a specific parameter configuration for each symbol, rather than one single configuration for the entire (possibly long) time series. The per-symbol automatic configuration effectively enables to fully exploit iSAX intra-word multiresolution capability to other applications beyond indexing.

## 4. AutoiSAX

In this section we describe our approach to automatically discover the iSAX symbolic length and alphabet size parameters. The approach is straightforward to interpret as it is based on two very simple concepts. To derive the symbolic length parameter we focus on time series complexity. For the alphabet size we look into the time series standard deviation.

A complex time series can be intuitively defined as having more ”peaks, valleys, and features” than a simpler one [Batista et al., 2011]. Our intuition is that a complex time series needs larger iSAX words, i.e. symbolic length, to better capture their complexity. On the other hand, simpler time series require smaller symbolic lengths. Complexity can be measured using many approaches, including information theory related (e.g. entropy), chaos based, Kolmogorov estimates, etc. In this work an alternative measure is used. The complexity estimate (CE) [Batista et al., 2011] is a simple measure of time series complexity. The idea is to "stretch" the time series into a straight line and measuring the line's length. It is based on the intuition that in a complex series, the distance between every two consecutive points is larger than in a simpler time series. Is it computed by simply square rooting the sum of every two consecutive points' differences:

$$CE(T) = \sqrt{\sum_{i=1}^{n-1} (t_i - t_{i+1})^2}$$

It was empirically compared to several more elaborate measures with surprisingly competitive results [Batista et al., 2011]. Furthermore, it has low space and time complexity, is intuitive and parameter-free. In order to maintain coherence between different length time series, the average per-point complexity estimate (ACE) is computed:

$$ACE(T) = CE(T)/|T|$$

This corresponds to step 1 in Algorithm 1.

The smoothing effect caused by the average calculation of each time series segment in iSAX can miss some features of the raw data [Lkhagva et al., 2006]. For example, extreme points can be absorbed by the mean. To tackle this issue, an interesting approach is to consider the variability of each segment's points regarding the segment's mean. That is, how "spread" the points are with respect to the mean. To capture this intuition the simple standard deviation is selected. A segment with a large standard deviation needs a coarser rectangle to capture its variability towards the mean. Hence it requires a coarser resolution (alphabet size) to fit the original time series behavior. A segment with small standard deviation presents slight variation around the mean. This behavior "fits" in a narrower rectangle. The space-efficiency in our approach arises from using the actual iSAX representation resolution to encode the standard deviation, rather than requiring any extra space. That is, each segment's standard deviation is encoded within the alphabet size parameter.

The AutoiSAX approach is presented in Algorithm 1. The actual parameter estimation consists of steps 2 (symbolic length) and 4 (alphabet size), which are explored separately in algorithms 2 and 3, respectively. For simplicity, the algorithm is hereby presented without any optimizations. First, the time series average complexity (ACE) is computed. The word length parameter is then calculated by framing the ACE between the minimum and maximum word length for the series, 2 and $(|T|)/2$, respectively. Then, a modified PAA algorithm is executed in order to output, besides each segment's mean, its standard deviation. Next, the standard deviation of each segment (pre-symbol) is converted to an alphabet size between the minimum (2) and maximum (32) resolution. In section 5 we show why resolutions larger than 32 are not used in our approach. Finally, each segment's mean is mapped to the corresponding symbol, according to the previously calculated symbol's alphabet size.

**Algorithm 1** AutoiSAX(time series $T$)

```
1: $Compl \leftarrow ACE(T)$
2: $l \leftarrow complexityToLength(Compl)$
3: $paa\_stds \leftarrow PAA'(T, l)$
4: $resolutions \leftarrow stdevsToResolutions(paa\_stds)$
5: $W = mapToSymbols(PAA', resolutions)$
6: **return** $W$
```

The symbolic length parameter is estimated in step 2 by the *complexityToLength* function. It takes a complexity average and maps it into a symbolic length. As the time series is normalized with zero average and one standard deviation, the average complexity ranges between 0 and the maximum complexity. The maximum possible average complexity is experimentally determined to be 0.16. This function simply performs a linear interpolation of the series complexity ($c$) into the symbolic length range between 2 and $(|T|)/2$. The *complexityToLength* function is computed using the following formula:

$$floor\left(\frac{(c-c_0)\times l_1}{c_1-c_0}+l_0\right)$$

$$\text{where } c_0 \leftarrow 0.00001; c_1 \leftarrow \text{maxComplexity}; l_0 \leftarrow 2; l_1 \leftarrow |T|/2$$

The minimum ACE is set to 0.00001 to obtain a 5 digit precision in the calculations. The *stdevsToResolutions* function (step 4) takes a list of segments' standard deviations and maps it to the final word list of resolutions. The mapping is based on a simple interpolation between the minimum (2) and maximum alphabet sizes. The maximum alphabet size is set to 32, as any further increase generates meaningless results in symbolic motif discovery. The intuition of this approach is based on a simple idea taken from the breakpoints table 1. Let $\beta_{zero}$ be the zero crossing breakpoint for a given alphabet size (shown in bold). The minimum resolution difference (MRD) for that resolution is $\beta_{zero} - \beta_{zero-1}$. For example, considering resolution 8, $\beta_{zero} = 4$ and MRD$= 0.16$. Table 2 shows the MRD for the resolutions 2 up to 32.

In practice, the MRD is a symbol's rectangle size for a given resolution. That is, the minimum rectangle where a segment's mean can lie in order to be assigned a given symbol. Figure 4 shows the size of MRD rectangle for the above resolutions.

The larger the resolution, the narrower is the rectangle. As the maximum standard deviation we aim to enclose corresponds to an alphabet size of 2, we can safely assume $maxStd = 2 \times MRD(4)$. Generally speaking, if the segment's standard deviation is smaller than a resolution's minimum rectangle size, then the symbol is assigned to that resolution. Thus, the linear interpolation technique seamlessly interleaves between the target resolutions, according to the obtained standard deviation (Algorithm 2).

The base of every time series data mining task is time series similarity [Ding et al., 2008], which is typically measured as dissimilarity in the form of distance measures. One of the main iSAX features is that it enables the definition of a distance measure (*mindist*) that lower bounds the Euclidean Distance in the original time series. In this work, two same-length time series can generate different length iSAX words. The iSAX

**Algorithm 2** stdevsToResolutions ($paa\_stds[]$)

```
1: for $sd_i$ in $paa\_stds$ do
2:     $s_0 \leftarrow 0; s_1 \leftarrow maxStd$
3:     $a_0 \leftarrow 2; a_1 \leftarrow 32$
4:     $a' = \frac{(sd_i - s_0) \times a_1}{s_1 - s_0} + a_0$
5:     $a = (a_1 - a') + a_0$
6:     $r_i = floor(a)$
7: end for
8: return r
```

mindist function is only defined to compare words with the same length. In order to enable AutoiSAX to be used in classification, motif discovery, etc., we slightly modify the original mindist function to enable the comparison of different length iSAX words. The modified mindist' between $\hat{Q}$ and $\hat{Q}$ is shown in Equation 1.

$$\text{Let } m = min(|\hat{Q}|, |\hat{C}|), M = max(|\hat{Q}|, |\hat{C}|)$$

$$mindist'(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{M} \times \sqrt{\sum_{i=1}^{M} (dist(\hat{q_j}, \hat{c_k}))^2}} \quad (1)$$

The intuition is as follows: one time series is aligned on top of the other for comparison. In case they have the same symbolic length, each symbol compares to the one to which it is aligned. If they have different symbolic lengths, first the smallest series is stretched to the size of the largest and then the comparison takes place. This is not a problem since the raw time series have the same size. Also, in practice this is done analytically, rather than performing an actual stretching of the iSAX word. To compare them, an iteration is performed over the largest word: each symbol is compared to the other series' symbol, the one that it is aligned to. Notice that one each symbol of the smallest word may be used multiple times as a comparison. For example, when comparing $(3^8, 4^8, 2^4, 2^2, 5^{16}, 2^2)$ with $(3^{12}, 4^{16}, 5^{16})$ the following comparisons are performed: $3^8$ and $4^8$ compare to $3^{12}$, $2^4$ and $2^2$ to $4^{16}$, $5^{16}$ and $2^2$ to $5^{16}$, as every symbol in the smaller word will represent one or more symbols from the larger word. As observed in this example, it is not required that the word sizes are multiple, as symbol $j$ of one series is compared to its aligned symbol $k$ in the other series. Thus, the admissible lower bound provided in the original iSAX's *mindist* is preserved [Shieh and Keogh, 2008].

## 5. Experimental Analysis

In this section we perform the experimental analysis to show the validity and impact of our approach. The experimental analysis uses different benchmark datasets for execution time (5.2), classification (5.3) and motif mining (5.4). Each dataset is briefly described in the beginning of each subsection. All experiments were executed on a machine with an Intel$^{®}$ Core$^{\mathrm{TM}}$$i5-530$ processor with 4GB of RAM. Unless explicitly stated otherwise, Java implementations compiled with JDK 6 were used. The experimental analysis proceeds as follows. 5.1 analyzes the danger of using too high resolutions. The computational performance of the approach is described in 5.2. In subsection 5.3 the impact of using AutoiSAX in classification is assessed. Finally, in 5.4 we apply our approach to motif mining.

### *5.1. The danger of selecting large resolutions*

In section 1 a simple example from motif mining is shown to explain why one should not select too large resolutions. In this section, we aim to empirically demonstrate why this is the case. First, let us properly motivate the need for such experiment. We have observed that the alphabet size parameter is typically set to very large values. For example, tightness of lower bound (TLB) is widely used to compare representation techniques and their distance measures [Ding et al., 2008, Shieh and Keogh, 2008]. It provides a good approximation of indexing performance, as it is hardware and implementation independent [Esling, P. and Agon, C., 2011]. Intuitively, it calculates how closely a representation's distance measure approximates the Euclidean Distance (ED). It is used for the comparison between different representation techniques (e.g. iSAX to DFT, etc.). In these experiments, the iSAX alphabet size parameter is often hard-coded to 256 [Ding et al., 2008, Shieh and Keogh, 2008]. The authors argue that as iSAX is more space efficient, as it can afford to encode a higher resolution using the same space than, for example, DFT. Indeed this is true. However, setting the alphabet size to 256 different symbols in a normalized series with mean 0 and standard deviation of 1 does not seem to bring any advantage. The minimum interval size corresponding to a 256 resolution is of 0.01. To use such a small interval to distinguish between different iSAX symbols at such fine resolution is similar to using the raw data itself. We claim that using such resolutions is both unpractical and unfair to competing representations. It is expected that by using a resolution so close to the raw data one obtains a much tighter lower bound. It is also straightforward to observe, and has been shown 10 years ago [Lin et al., 2003], that in tasks where the degree of proximity to the raw data is important, higher resolutions will outperform lower ones.

To analyze the impact of resolutions in motif discovery, 52 datasets from a wide variety of sources, aggregated in [Castro and Azevedo, 2011], are selected. For each dataset, we interleave between the resolutions 2, 4, 8, 16, 32, and 64, and record the number of symbolic motifs extracted. The motif extraction algorithm is simply to generate an iSAX word for each time series received as input and counting the number of times each word appears. The MrMotif algorithm [Castro and Azevedo, 2010] is suitable for this task, as it gracefully interleaves between resolutions and outputs the symbolic motifs discovered for each resolutions. In Figure 5, the number of discovered motifs ($N_d$) for each resolution and dataset are shown. For space, 7 datasets are displayed. Results for the remainder of the datasets are similar. The names *eeg* and *EEG* refer to different datasets.

It can be observed that $N_d$ decreases as one increases the alphabet size. By looking closely, very few motifs can be found at resolution 32 or larger. In fact, at resolution 64, either no motifs or only identical motifs can be found. The larger the resolution, the finer are the intervals available for the time series' segments. This makes it hard in practice for two non-identical time series to match. This suggests that using resolutions larger than 32 in approximate motif discovery is of little value, and is therefore meaningless.

## *5.2. Computational Performance*

In this subsection we study the proposal's execution time and compare it to the MDL based parameter estimation approach [Hu et al., 2011]. The authors' MATLAB® code for the competing approach is used. For the sake of fair comparison, we use an AutoiSAX MATLAB® implementation. Ten different sets of increasing size from [Mueen et al., 2009] are used, ranging from 10000 to 100000 time series of length 1024. We use these data for two reasons: they have been used before and results on similar datasets are encouraged; in addition, the size (in the gigabytes) makes them attractive to test any approach. All the time series in each set are concatenated generating ten time series with lengths 10240000 up to 102400000. We apply both techniques to each time series ten times and record the average execution time for each series. Results are presented in Figure 6.

For the 10 million-sized time series, the techniques execution time is as follows. For iSAX: 0.40 seconds, AutoiSAX: 1.45 seconds, MDL: 102s. It can be observed that the MDL approach's execution time increases quadratically with the time series size. The AutoiSAX and iSAX methods increase linearly, as highlighted in Figure 7 in a separate plot.

The MDL is about two orders of magnitude slower than iSAX and about 70 times slower than AutoiSAX, on average. In practice, for estimating the parameters of a single time series taking 1s to convert to the AutoiSAX representation, MDL would take longer than one minute. As an exhaustive search approach, it needs to scan all possible models from lengths 2 up to 64, and the (simplified) resolutions 2, 4, 8, 16, 32 and 64. To calculate one single model costs just slightly higher than AutoiSAX. However, the approach requires that all models are computed, in order to select the best. This effectively makes it perform 372 model calculations instead of just one. To further inspect the impact of MDL approach' execution time, we performed an experiment with a small dataset ("50words") from the UCR classification benchmark [Keogh et al., 2009]. The dataset consists of 450 training and 455 test series of length 270, and 50 classes. The MDL approach was selected for estimating the iSAX parameters for 1-NN classification using the iSAX's *mindist* distance measure. After one hour of execution, the approach had not yet terminated execution. As the approach's execution time makes it untenable to use in real word data mining tasks, we omit it from the remainder of the experimental analysis. The AutoiSAX presents linear complexity having a constant factor larger than the fixed iSAX. Another scan over the original series is required for the complexity (ACE) and standard deviation computation.

## *5.3. Classification Accuracy*

In this section the 1-NN classification experiment performed in the previous subsection is extended. The AutoiSAX and fixed iSAX (using several resolutions) are compared for all datasets in the UCR classification benchmark in terms of classification accuracy. One-NN if often used to argue the suitability of a distance measure or representation [Ding et al., 2008]. Our aim is not to analyze the applicability of AutoiSAX for classification. It has been shown that using 1-NN with the Euclidean Distance directly in the raw data is surprisingly competitive when compared to other distance measures or classification algorithms, and it is also computationally more efficient [Ding et al., 2008]. Rather, we aim to show that AutoiSAX can obtain results as accurate as fixed iSAX, without the hassle of manually optimizing the parameters. The benchmark is composed of 42 datasets with training sets containing 16 to 1000 and testing sets with 28 up to 3582 time series. Each time series length ranges from 24 to 1882, with 2 up to 50 classes. The MATLAB® classification framework provided in the benchmark is used. The single obvious modification is changing the distance measure for the selected representation. A scatter-plot of the AutoiSAX and fixed iSAX pairs of results for each dataset are displayed in Figure 8. Thus, one point represents the classification accuracy of the pair (AutoiSAX,iSAX) for one dataset. Points above the identity function (blue) mean that AutoiSAX outperforms fixed iSAX for that dataset and vice-versa.

In general, the AutoiSAX approach showed better classification accuracy than the fixed one. For the remainder of the datasets, the approaches showed similar accuracy. This suggests that our approach approximates the raw data in a more realistic way, and this fact has repercussions in classification accuracy.

## 5.4. *Motif Discovery*

In this subsection we apply AutoiSAX to the task of motif discovery. We use the same simple algorithm as in 5.1: generating an AutoiSAX word for each time series and counting repetitions. We compare the approach to fixed iSAX with length and resolution 8, and with an average of fixed iSAX resolutions ranging from 2 up to 64. MrMotif is selected as it can seamlessly provide these results. The 52 dataset benchmark (aggregated in [Castro and Azevedo, 2011]) is selected for the experiment. The question we would like to be able to answer is whether the derived motifs are more meaningful, as we are using two further implicit dimensions to filter motifs: complexity and standard deviation. To answer the question the average Euclidean Distance between each match in the dataset is measured. Matching time series should present a small ED for the motif to be interesting. In Figure 9 we show the scatter-plot of the pairs of average intra-motif distance for the approaches. Results are normalized in the $[0, 1]$ interval. Distances are only shown for the datasets motifs were found.

It can be observed that the intra-motif distance of the AutoiSAX approach is smaller than the other two approaches', for most datasets. This is expected, as AutoiSAX adds two other dimensions (besides the average) to further restrict the motifs: complexity and standard deviation. If the motifs size and alphabet size are not previously fixed, their natural length and variability will cause matches to be more accurate. For example, let us consider two iSAX(8,8) words with standard deviations of 0.1 and 1.2. If by chance their mean is the same, they would match with fixed iSAX, but would generate two very different words using AutoiSAX. Using our approach, only very similar time series can match. To investigate whether this also affects the number of discovered motifs, we count the number of motifs returned for each approach and dataset. For the majority of the datasets the number of returned motifs is smaller using AutoiSAX approach. This suggests that using AutoiSAX to extract symbolic motifs can act as a filter to prune spurious motifs. The effect of using our approach in motif mining is shown with a representative example in Figure 10. The iMotifs Visualization Tool [Castro, 2011] is used to visualize the extracted motifs from the projectiles shape dataset from [Yankov et al., 2007] using the fixed iSAX and AutoiSAX approaches.

The first thing to notice is that the number of extracted motifs (enclosed in parentheses) is larger in the fixed iSAX approach. Also, some incorrect motifs are found with this approach. For instance, the motif in Figure 10 at the fourth and fifth symbols is a false discovery. As only the mean is used to match time series, the approach is unable to prune out motifs with a different standard deviation. The AutoiSAX approach exploits more information on the series to obtain fewer and more similar matches. This suggests AutoiSAX can be used as a parameter-free alternative to mine symbolic motifs. Nevertheless, in some domains the validation of the discovered motifs is important and may require validation of a domain expert. When the number of motifs is prohibitively large, automatic evaluation techniques may be required, such as the one proposed in [Castro and Azevedo, 2011].

## 6. Discussion

The major advantage of AutoiSAX is estimating the symbolic length and alphabet size parameters for the iSAX representation. As a consequence of the techniques used to obtain these parameters, two more dimensions become available (for free) for analyzing the represented time series. First, complexity estimation allows to assign more symbols to complex time series. Second, standard deviation computation permits to adjust the representation's resolution. Furthermore, it enables different resolutions within the same symbolic word. This is a major feature, as time series are typically long and present large variations in behavior along their length. Even small time series can have wildly changing behavior, which a single resolution per-series simply cannot capture. This fully exploits iSAX's multiresolution to applications beyond indexing. In this task, interleaving between different resolutions is performed by the indexing algorithm. The resolutions are doubled whenever the count of a time series container surpasses a given threshold [Shieh and Keogh, 2008]. In other mining tasks, such as visualization, classification and motif discovery, this was not previously possible as the only way to use intra-word multiresolution was to manually set the iSAX word's parameters. One potential feature of our approach is that it seems to be more suitable to deal with extreme values than classical iSAX. The standard deviation calculation appears to compensate for the information loss caused by average smoothing. These two dimensions provide further information for visualization, classification and motif discovery algorithms, which are leveraged to obtain more accurate results, as we have seen in section 5. In visualization, one can get a more intuitive grasp at the original series shape just by looking at the representation. This is relevant as domain experts can more effectively visually detect interesting patterns in their data using tools such as the aforementioned iMotifs [Castro, 2011]. One can also obtain better classification accuracy. It seems that this measure is highly correlated with the resolution used. However, too high resolutions can be harmful as we have shown. AutoiSAX appears to provide a nice trade-off between the minimum resolution required to discriminate between the classes and a not too high resolution that can overfit the data. Finally, more interesting motifs can be found as closer matches are retrieved and spurious motifs discarded. The addition of complexity and deviation information in the motif computation seems to impose an important filter in the process. It remains to be investigated whether these motifs are also statistically more significant by using our motifs statistical significance approach, tailored for fixed iSAX [Castro and Azevedo, 2011]. It also remains as future work to research whether a more efficient indexing algorithm can be obtained by mixing AutoiSAX in the original iSAX indexing framework.

Preliminary results with the benchmark datasets [Castro and Azevedo, 2011] show that the Tightness of Lower Bound measure depends highly on the resolution at hand, as expected. Nevertheless, further experimentation is required to reach a firm conclusion whether the measure is appropriate in the current settings. We are also currently investigating whether the application of AutoiSAX to streaming data is trivial or requires more elaboration. Finally, we stress out that almost all these mining techniques can effectively become parameter-free as a consequence of using our approach.

## 7. Conclusion

We have proposed an approach to automatically estimate the parameters of the iSAX time series representation. The novelty of our work is that it enables efficient computation of the symbolic length and alphabet size parameters. Furthermore, it permits to calculate the alphabet size for each symbol within the same time series. The approach, referred as AutoiSAX, is based on two very simple ideas. First, complex time series require larger symbolic lengths to better capture their behavior. Second, as time series segments present different deviations from the mean, they should be represented using different alphabet sizes. Our technique allows to leverage iSAX's intra-word multiresolution, which effectively brings iSAX to other tasks besides time series indexing. We have shown that AutoiSAX improves visualization interpretability, classification accuracy and motif mining results. We also aim to highlight the importance of making iSAX a parameter-free approach, as the large number of researchers and practitioners using it can benefit from it.

## Reproducibility Note

All experiments, source code and datasets are available online at [Castro, 2014].

## References

[Agrawal et al., 1995] Agrawal, R., Psaila, G., Wimmers, E., and Zaït, M. (1995). Querying shapes of histories. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 502–514. Morgan Kaufmann Publishers Inc.

[Batista et al., 2011] Batista, G., Wang, X., and Keogh, E. (2011). A complexity-invariant distance measure for time series. In *SDM-2011: Proceedings of SIAM International Conference on Data Mining, Philadelphia, PA, USA*.

[Camerra et al., 2010] Camerra, A., Palpanas, T., Shieh, J., and Keogh, E. (2010). isax 2.0: Indexing and mining one billion time series. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 58–67. IEEE.

[Castro, 2011] Castro, N. (2011). iMotifs: Interactive Time Series Motif Discovery and Visualization Tool. http://www.di.uminho.pt/∼castro/imotifs.

[Castro, 2014] Castro, N. (2014). Autoisax website. http://www.di.uminho.pt/∼castro/autoisax.

[Castro and Azevedo, 2010] Castro, N. and Azevedo, P. (2010). Multiresolution Motif Discovery in Time Series. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, 2010, Columbus, Ohio, USA*, pages 665–676. SIAM.

[Castro and Azevedo, 2011] Castro, N. and Azevedo, P. (2011). Time Series Motifs Statistical Significance. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2011, 2011, Mesa, Arizona, USA*, pages 687–698. SIAM.

[Chakrabarti et al., 2002] Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. (2002). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228.

[Chan and Fu, 1999] Chan, K.-P. and Fu, A. W.-C. (1999). Efficient time series matching by wavelets. In *ICDE*, pages 126–133.

[Chiu et al., 2003] Chiu, B., Keogh, E., and Lonardi, S. (2003). Probabilistic discovery of time series motifs. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–498, New York, NY, USA. ACM.

[Dasgupta and Forrest, 1996] Dasgupta, D. and Forrest, S. (1996). Novelty detection in time series data using ideas from immunology. In *Proceedings of the International Conference on Intelligent Systems*, pages 82–87.

[Ding et al., 2008] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. (2008). Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552.

[Esling, P. and Agon, C., 2011] Esling, P. and Agon, C. (2011). Time series data mining and analysis. *ACM Computing Surveys (to appear).*

[Faloutsos et al., 1994] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 419–429, New York, NY, USA. ACM.

[Geurts, 2001] Geurts, P. (2001). Pattern extraction for time series classification. *Principles of Data Mining and Knowledge Discovery*, pages 115–127.

[Hu et al., 2011] Hu, B., Rakthanmanon, T., Hao, Y., Evans, S., Lonardi, S., and Keogh, E. (2011). Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1086–1091. IEEE.

[Kalpakis et al., 2001] Kalpakis, K., Gada, D., and Puttagunta, V. (2001). Distance measures for effective clustering of arima time-series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 273–280. IEEE.

[Keogh et al., 2001] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286.

[Keogh et al., 2009] Keogh, E., Xi, X., Wei, L., and Ratanamahatana, C. (2009). Ucr time series classification/clustering page. *Training and testing data sets: Available online: http://www.cs.ucr.edu/eamonn/time series data/(accessed on 18 July 2011).*

[Korn et al., 1997] Korn, F., Jagadish, H. V., and Faloutsos, C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 289–300, New York, NY, USA. ACM.

[Lin et al., 2003] Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, page 11. ACM.

[Lin et al., 2007] Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing SAX: a novel symbolic representation of time series. *Data mining and knowledge discovery*, 15(2):107–144.

[Lkhagva et al., 2006] Lkhagva, B., Suzuki, Y., and Kawagoe, K. (2006). New Time Series Data Representation ESAX for Financial Applications. *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)-Volume 00.*

[Mueen et al., 2009] Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B. (2009). Exact discovery of time series motifs. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2009)*, pages 473–484. Citeseer.

[Pham et al., 2010] Pham, N., Le, Q., and Dang, T. (2010). Two novel adaptive symbolic representations for similarity search in time series databases. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pages 181–187. IEEE.

[Shieh and Keogh, 2008] Shieh, J. and Keogh, E. (2008). iSAX: indexing and mining terabyte sized time series. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631.

[Yankov et al., 2007] Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V. (2007). Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 844–853. ACM.

## Tables

**Table 1.** SAX breakpoints for resolutions from 2 until 32

| $a$ / $\beta_i$ | **2** | **4** | **8** | **16** | **32** |
|---|---|---|---|---|---|
| $\beta_1$ | **0.00** | -0.67 | -1.15 | -1.53 | -1.87 |
| $\beta_2$ | | **0.00** | -0.67 | -1.15 | -1.54 |
| $\beta_3$ | | 0.67 | -0.32 | -0.89 | -1.32 |
| $\beta_4$ | | | **0.00** | -0.67 | -1.16 |
| $\beta_5$ | | | 0.32 | -0.49 | -1.01 |
| $\beta_6$ | | | 0.67 | -0.32 | -0.89 |
| $\beta_7$ | | | 1.15 | -0.16 | -0.78 |
| $\beta_8$ | | | | **0.00** | -0.68 |
| $\beta_9$ | | | | 0.16 | -0.58 |
| … | | | | … | … |
| $\beta_{14}$ | | | | 1.15 | -0.16 |
| $\beta_{15}$ | | | | 1.53 | -0.08 |
| $\beta_{16}$ | | | | | **0.00** |
| $\beta_{17}$ | | | | | 0.07 |
| $\beta_{18}$ | | | | | 0.15 |
| … | | | | | … |
| $\beta_{31}$ | | | | | 1.86 |

**Table 2.** $\beta_{zero}$ and MRD for several resolutions

| a | **2** | **4** | **8** | **16** | **32** |
|---|---|---|---|---|---|
| $\beta_{zero}$ | $\beta_1$ | $\beta_2$ | $\beta_4$ | $\beta_8$ | $\beta_{16}$ |
| MRD | $\infty$ | 0.67 | 0.32 | 0.16 | 0.08 |

## Figures

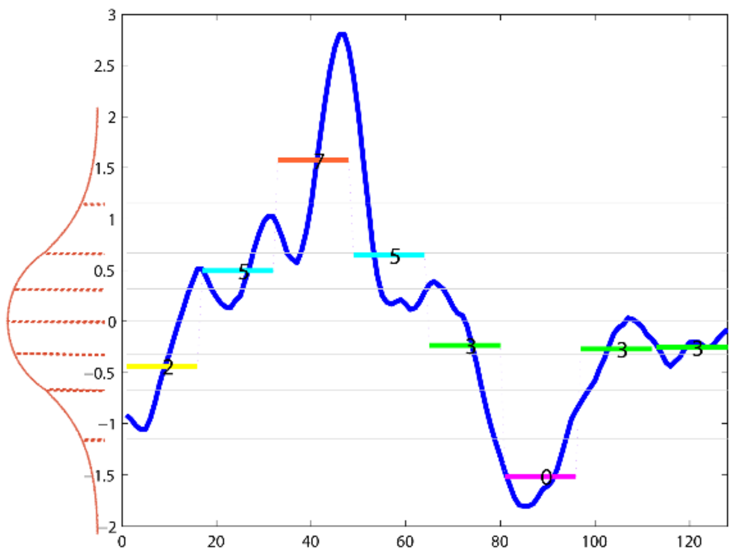

**Figure 1.** Conversion of a time series of length 128 to the length-8 iSAX word $(2, 5, 7, 5, 3, 0, 3, 3)$

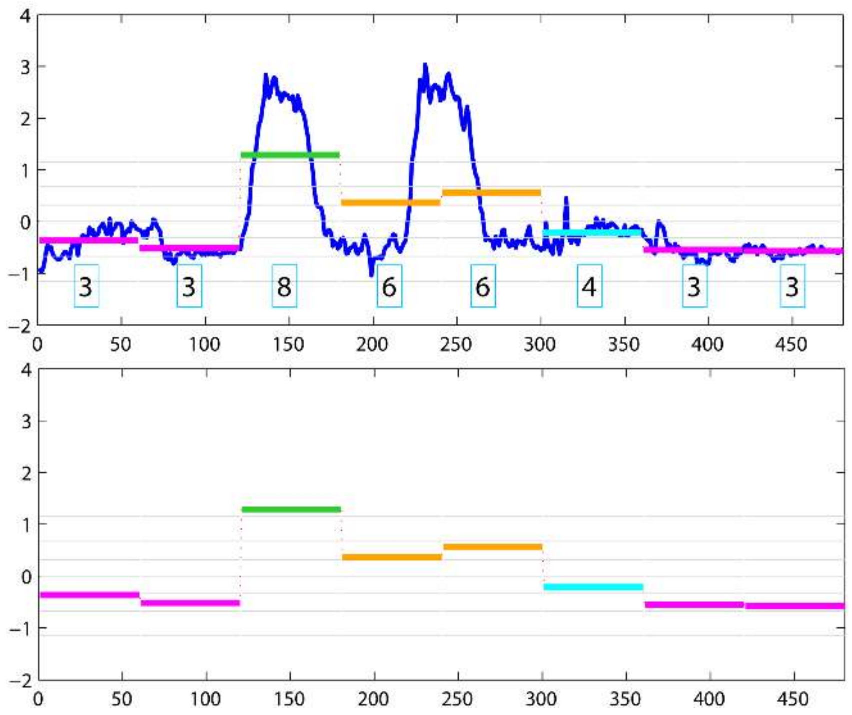

**Figure 2.** *Top:* Time series of five days' electric power consumption and iSAX representation with fixed $l = 8$ and $a = 8$, generating the $(3^8, 3^8, 8^8, 6^8, 6^8, 4^8, 3^8, 3^8)$ word. *Bottom:* Showing only the iSAX representation without the original time series.

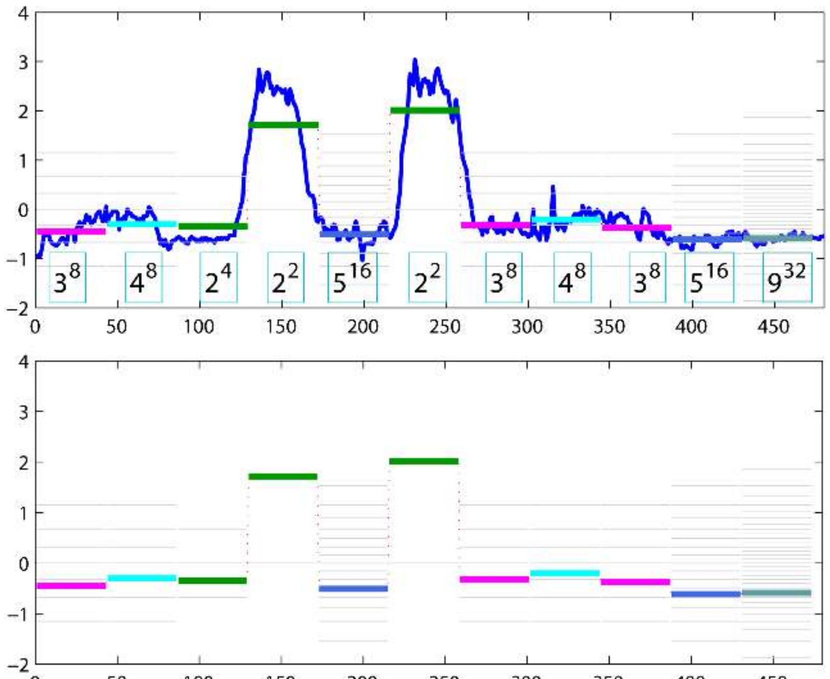

**Figure 3.** Same time series as before, now represented with AutoiSAX. *Top:* The $(3^8, 4^8, 2^4, 2^2, 5^{16}, 2^2, 3^8, 4^8, 5^{16}, 9^{32})$ word of length 11 (top) is generated. *Bottom:* Showing only the iSAX representation without the original time series.

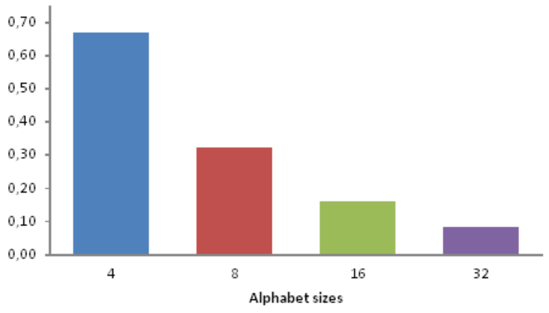

**Figure 4.** Size of the MRD rectangles for several resolutions

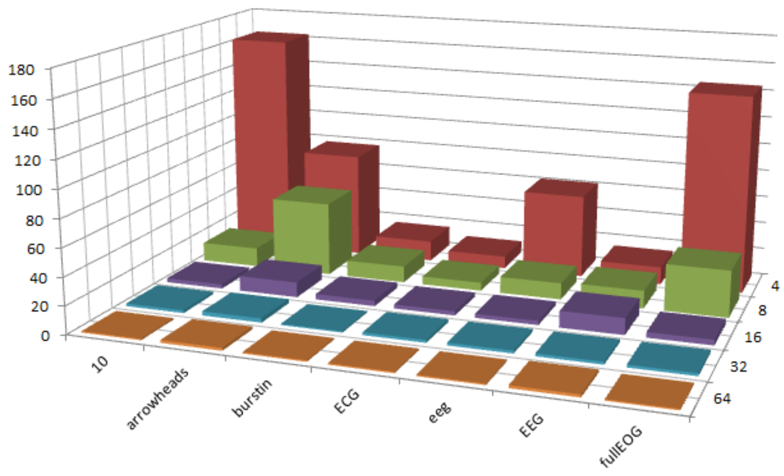

**Figure 5.** Number of symbolic motifs found for several resolutions and datasets.

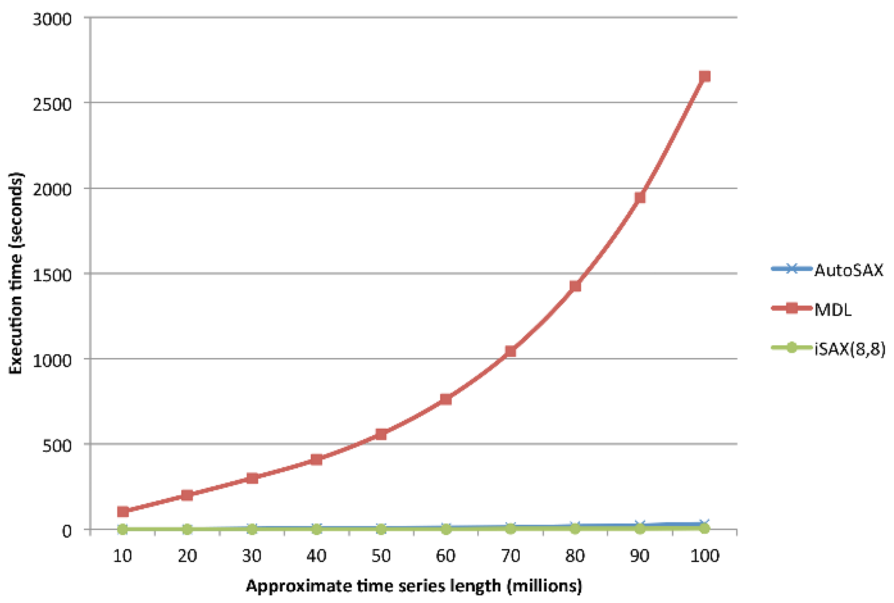

**Figure 6.** Execution time of AutoiSAX, fixed iSAX and MDL approaches in ten increasingly sized time series.

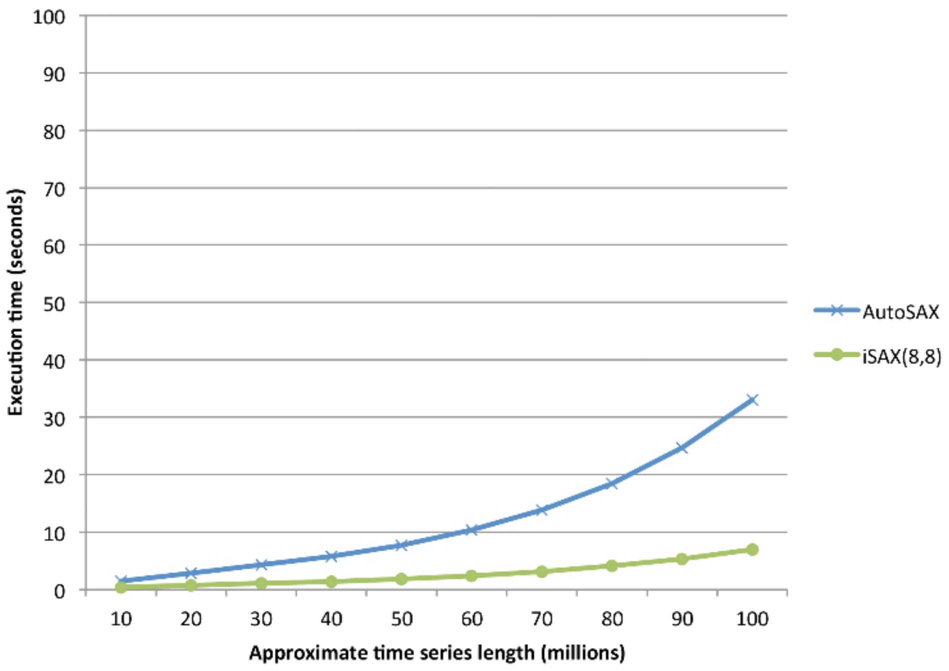

**Figure 7.** Execution time of AutoiSAX and fixed iSAX approaches displayed separately.

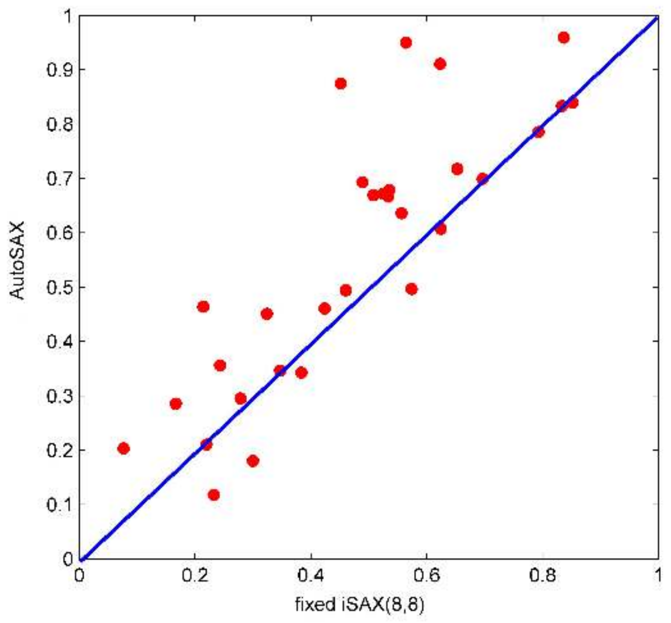

**Figure 8.** Accuracy of 1-NN classification on 42 datasets: auto SAX vs fixed iSAX $(8, 8)$. Above the identity line means AutoiSAX outperforms fixed iSAX.

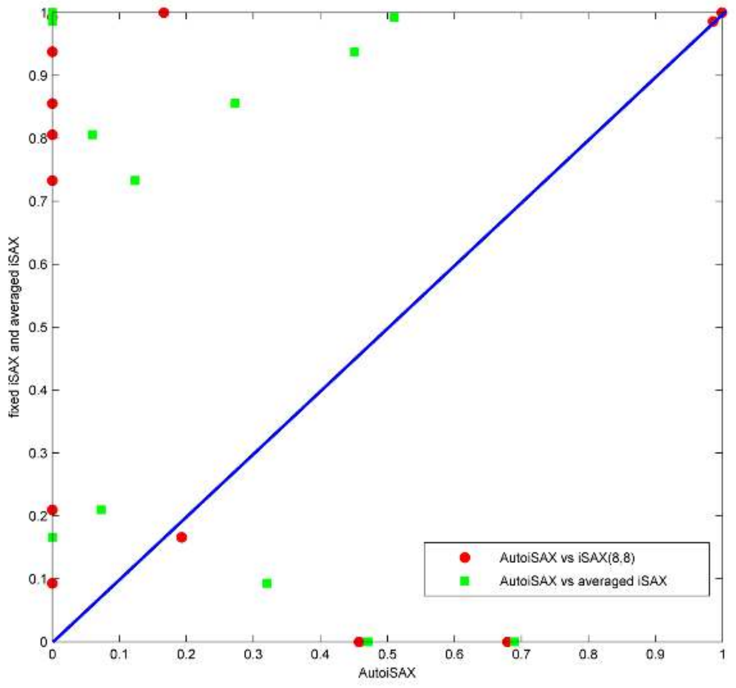

**Figure 9.** Intra-motif normalized distances. Circles: AutoiSAX vs iSAX(8,8). Squares: AutoiSAX vs average iSAX. Above the line the distance is smaller (better).

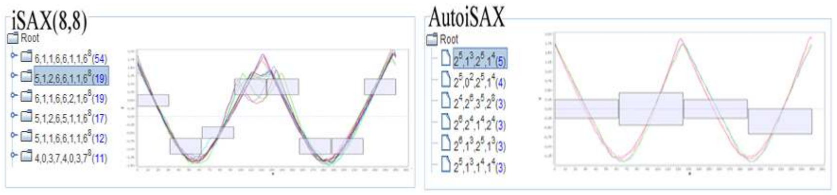

**Figure 10.** Motifs extracted from the projectile shapes dataset using the iSAX$(8, 8)$ (top) and AutoiSAX (bottom) approaches, using the iMotifs Visualization Tool.