Experiencing SAX: a Novel Symbolic Representation of Time Series

JESSICA LIN jessica@ise.gmu.edu
Information and Software Engineering Department, George Mason University, Fairfax, VA 22030

EAMONN KEOGH eamonn@cs.ucr.edu
LI WEI wli@cs.ucr.edu
STEFANO LONARDI stelo@cs.ucr.edu
Computer Science & Engineering Department, University of California – Riverside, Riverside, CA 92521

Abstract. Many high level representations of time series have been proposed for data mining, including Fourier transforms, wavelets, eigenwaves, piecewise polynomial models etc. Many researchers have also considered symbolic representations of time series, noting that such representations would potentiality allow researchers to avail of the wealth of data structures and algorithms from the text processing and bioinformatics communities. While many symbolic representations of time series have been introduced over the past decades, they all suffer from two fatal flaws. Firstly, the dimensionality of the symbolic representation is the same as the original data, and virtually all data mining algorithms scale poorly with dimensionality. Secondly, although distance measures can be defined on the symbolic approaches, these distance measures have little correlation with distance measures defined on the original time series.

In this work we formulate a new symbolic representation of time series. Our representation is unique in that it allows dimensionality/numerosity reduction, and it also allows distance measures to be defined on the symbolic approach that lower bound corresponding distance measures defined on the original series. As we shall demonstrate, this latter feature is particularly exciting because it allows one to run certain data mining algorithms on the efficiently manipulated symbolic representation, while producing identical results to the algorithms that operate on the original data. In particular, we will demonstrate the utility of our representation on various data mining tasks of clustering, classification, query by content, anomaly detection, motif discovery, and visualization.

Keywords
Time Series, Data Mining, Symbolic Representation, Discretize

1. Introduction

Many high level representations of time series have been proposed for data mining. Figure 1 illustrates a hierarchy of all the various time series representations in the literature [3, 11, 20, 24, 27, 30, 36, 53, 54, 63]. One representation that the data mining community has not considered in detail is the discretization of the original data into symbolic strings. At first glance this seems a surprising oversight. There is an enormous wealth of existing algorithms and data structures that allow the efficient manipulations of strings. Such algorithms have received decades of attention in the text retrieval community, and more recent attention from the bioinformatics community [5, 19, 25, 52, 57, 60]. Some simple examples of "tools" that are not defined for real-valued sequences but are defined for symbolic approaches include hashing, Markov models, suffix trees, decision trees etc.

There is, however, a simple explanation for the data mining community's lack of interest in string manipulation as a supporting technique for mining time series. If the data are transformed into virtually any of the other representations depicted in Figure 1, then it is possible to measure the similarity of two time series in that representation space, such that the distance is guaranteed to lower bound the true distance between the time series in the original space[1]. This simple fact is at the core of almost all algorithms in time

[1] The exceptions are *random mappings*, which are only guaranteed to be within an epsilon of the true distance with a certain probability, *trees*, *interpolation* and *natural language*.

series data mining and indexing [20]. However, in spite of the fact that there are dozens of techniques for producing different variants of the symbolic representation [3, 15, 27], there is no known method for calculating the distance in the symbolic space, while providing the lower bounding guarantee.

In addition to allowing the creation of lower bounding distance measures, there is one other highly desirable property of any time series representation, including a symbolic one. Almost all time series datasets are very high dimensional. This is a challenging fact because all non-trivial data mining and indexing algorithms degrade exponentially with dimensionality. For example, above 16-20 dimensions, index structures degrade to sequential scanning [26]. None of the symbolic representations that we are aware of allow dimensionality reduction [3, 15, 27]. There is some reduction in the storage space required, since fewer bits are required for each value, however the intrinsic dimensionality of the symbolic representation is the same as the original data.

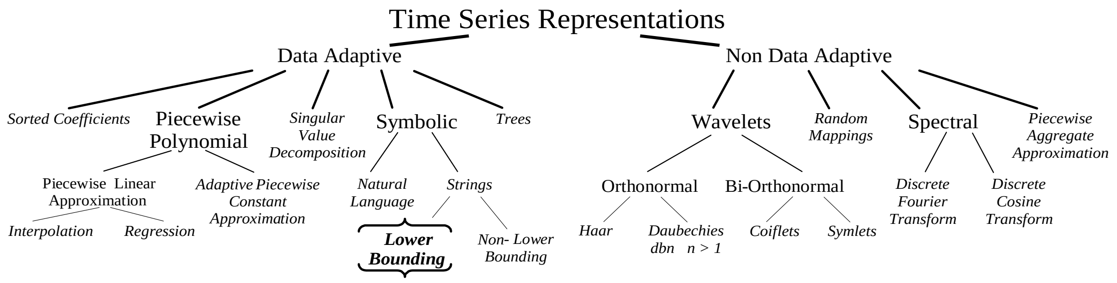


Figure 1: A hierarchy of all the various time series representations in the literature. The leaf nodes refer to the actual representation, and the internal nodes refer to the classification of the approach. The contribution of this paper is to introduce a new representation, the lower bounding symbolic approach

There is no doubt that a new symbolic representation that remedies all the problems mentioned above would be highly desirable. More specifically, the symbolic representation should meet the following criteria: space efficiency, time efficiency (fast indexing), and correctness of answer sets (no false dismissals).

In this work we formally formulate a novel symbolic representation and show its utility on other time series tasks[2]. Our representation is unique in that it allows dimensionality/numerosity reduction, and it also allows distance measures to be defined on the symbolic representation that lower bound corresponding popular distance measures defined on the original data. As we shall demonstrate, the latter feature is particularly exciting because it allows one to run certain data mining algorithms on the efficiently manipulated symbolic representation, while producing identical results to the algorithms that operate on the original data. In particular, we will demonstrate the utility of our representation on the classic data mining tasks of clustering [29], classification [24], indexing [2, 20, 30, 63], and anomaly detection [14, 34, 54].

The rest of this paper is organized as follows. Section 2 briefly discusses background material on time series data mining and related work. Section 3 introduces our novel symbolic approach, and discusses its dimensionality reduction, numerosity reduction and lower bounding abilities. Section 4 contains an experimental evaluation of the symbolic approach on a variety of data mining tasks. Impact of the symbolic approach is also discussed. Finally, Section 5 offers some conclusions and suggestions for future work.

2. Background and Related Work

Time series data mining has attracted enormous attention in the last decade. The review below is necessarily brief; we refer interested readers to [32, 53] for a more in depth review.

[2] A preliminary version of this paper appears in [41].

2.1 Time Series Data Mining Tasks

While making no pretence to be exhaustive, the following list summarizes the areas that have seen the majority of research interest in time series data mining.

- **Indexing:** Given a query time series Q, and some similarity/dissimilarity measure *D*(Q,C), find the most similar time series in database *DB* [2, 11, 20, 30, 63].
- **Clustering:** Find natural groupings of the time series in database *DB* under some similarity/dissimilarity measure *D*(Q,C) [29, 36].
- **Classification:** Given an unlabeled time series Q, assign it to one of two or more predefined classes [24].
- **Summarization:** Given a time series Q containing *n* datapoints where *n* is an extremely large number, create a (possibly graphic) approximation of Q which retains its essential features but fits on a single page, computer screen, executive summary etc [43].
- **Anomaly Detection:** Given a time series Q, and some model of "normal" behavior, find all sections of Q which contain anomalies or "surprising/interesting/unexpected/novel" behavior [14, 34, 54].

Since the datasets encountered by data miners typically don't fit in main memory, and disk I/O tends to be the bottleneck for any data mining task, a simple generic framework for time series data mining has emerged [20]. The basic approach is outlined in Table 1.

Table 1: A generic time series data mining approach

1.	Create an approximation of the data, which will fit in main memory, yet retains the essential features of interest.
2.	Approximately solve the task at hand in main memory.
3.	Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data.

It should be clear that the utility of this framework depends heavily on the quality of the approximation created in Step 1. If the approximation is very faithful to the original data, then the solution obtained in main memory is likely to be the same as, or very close to, the solution we would have obtained on the original data. The handful of disk accesses made in Step 3 to confirm or slightly modify the solution will be inconsequential compared to the number of disk accesses required had we worked on the original data. With this in mind, there has been great interest in approximate representations of time series, which we consider below.

2.2 Time Series Representations

As with most problems in computer science, the suitable choice of representation greatly affects the ease and efficiency of time series data mining. With this in mind, a great number of time series representations have been introduced, including the Discrete Fourier Transform (DFT) [20], the Discrete Wavelet Transform (DWT) [11], Piecewise Linear, and Piecewise Constant models (PAA) [30], (APCA) [24, 30], and Singular Value Decomposition (SVD) [30]. Figure 2 illustrates the most commonly used representations.

Recent work suggests that there is little to choose between the above in terms of indexing power [32], however, the representations have other features that may act as strengths or weaknesses. As a simple example, wavelets have the useful multiresolution property, but are only defined for time series that are an integer power of two in length [11].

One important feature of all the above representations is that they are real valued. This limits the algorithms, data structures and definitions available for them. For example, in anomaly detection we cannot meaningfully define the probability of observing any particular set of wavelet coefficients, since the probability of observing any real number is zero [38]. Such limitations have lead researchers to consider using a symbolic representation of time series.

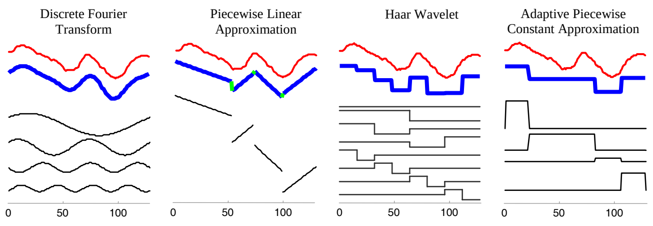


Figure 2: The most common representations for time series data mining. Each can be visualized as an attempt to approximate the signal with a linear combination of basis functions

While there are literally hundreds of papers on discretizing (symbolizing, tokenizing, quantizing) time series [3, 27] (see [15] for an extensive survey), none of the techniques allows a distance measure that lower bounds a distance measure defined on the original time series. For this reason, the generic time series data mining approach illustrated in Table 1 is of little utility, since the approximate solution to problem created in main memory may be arbitrarily dissimilar to the true solution that would have been obtained on the original data. If, however, one had a symbolic approach that allowed lower bounding of the true distance, one could take advantage of the generic time series data mining model, and of a host of other algorithms, definitions and data structures which are only defined for discrete data, including hashing, Markov models, and suffix trees. This is exactly the contribution of this paper. We call our symbolic representation of time series SAX (Symbolic Aggregate approXimation), and define it in the next section.

3. SAX: Our Symbolic Approach

SAX allows a time series of arbitrary length n to be reduced to a string of arbitrary length w, ($w < n$, typically $w << n$). The alphabet size is also an arbitrary integer a, where $a > 2$. Table 2 summarizes the major notation used in this and subsequent sections.

Table 2: A summarization of the notation used in this paper

C	A time series $C = c_1,\ldots,c_n$
$\bar{C}$	A Piecewise Aggregate Approximation of a time series $\bar{C} = \bar{c}_1,\ldots,\bar{c}_w$
$\hat{C}$	A symbol representation of a time series $\hat{C} = \hat{c}_1,\ldots,\hat{c}_w$
w	The number of PAA segments representing time series C
a	Alphabet size (e.g., for the alphabet = {**a**,**b**,**c**}, a = 3)

Our discretization procedure is unique in that it uses an intermediate representation between the raw time series and the symbolic strings. We first transform the data into the Piecewise Aggregate Approximation (PAA) representation and then symbolize the PAA representation into a discrete string. There are two important advantages to doing this:

- **Dimensionality Reduction:** We can use the well-defined and well-documented dimensionality reduction power of PAA [30, 63], and the reduction is automatically carried over to the symbolic representation.
- **Lower Bounding:** Proving that a distance measure between two symbolic strings lower bounds the true distance between the original time series is non-trivial. The key observation that

allowed us to prove lower bounds is to concentrate on proving that the symbolic distance measure lower bounds the *PAA distance measure*. Then we can prove the desired result by transitivity by simply pointing to the existing proofs for the PAA representation itself [31, 63].

We will briefly review the PAA technique before considering the symbolic extension.

3.1 Dimensionality Reduction Via PAA

A time series *C* of length *n* can be represented in a *w*-dimensional space by a vector $\overline{C} = \overline{c}_1, \ldots, \overline{c}_w$. The *i*th element of $\overline{C}$ is calculated by the following equation:

$$\overline{c}_i = \tfrac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \qquad (1)$$

Simply stated, to reduce the time series from *n* dimensions to *w* dimensions, the data is divided into *w* equal sized “frames”. The mean value of the data falling within a frame is calculated and a vector of these values becomes the data-reduced representation. The representation can be visualized as an attempt to approximate the original time series with a linear combination of box basis functions as shown in Figure 3. For simplicity and clarity, we assume that *n* is divisible by *w*. We will relax this assumption in Section 3.5.

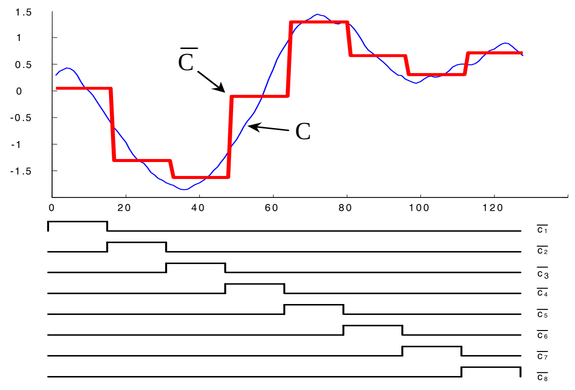


Figure 3: The PAA representation can be visualized as an attempt to model a time series with a linear combination of box basis functions. In this case, a sequence of length 128 is reduced to 8 dimensions

The PAA dimensionality reduction is intuitive and simple, yet has been shown to rival more sophisticated dimensionality reduction techniques like Fourier transforms and wavelets [30, 32, 63].

We normalize each time series to have mean of zero and a standard deviation of one before converting it to the PAA representation, since it is well understood that it is meaningless to compare time series with different offsets and amplitudes [32].

3.2 Discretization

Having transformed a time series database into the PAA we can apply a further transformation to obtain a discrete representation. It is desirable to have a discretization technique that will produce symbols with equiprobability [5, 45]. This is easily achieved since normalized time series have a Gaussian distribution [38]. To illustrate this, we extracted subsequences of length 128 from 8 different time series and plotted normal probability plots of the data as shown in Figure 4. A normal probability plot is a graphical technique that shows if the data is approximately normally distributed [1]: an approximate straight line indicates that the data is approximately normally distributed. As the figure shows, the highly linear nature of the plots suggests that the data is approximately normal. For a large family of the time series data in our disposal, we notice that the Gaussian assumption is indeed true. For the small subset of data where the assumption is not obeyed, the efficiency is slightly deteriorated; however, the correctness of the algorithm is unaffected. The correctness of the algorithm is guaranteed by the lower-bounding property of the distance measure in the symbolic space, which we will explain in the next section.

Given that the normalized time series have highly Gaussian distribution, we can simply determine the “breakpoints” that will produce *a* equal-sized areas under Gaussian curve [38].

Definition 1. *Breakpoints*: breakpoints are a sorted list of numbers $B = \beta_1,\ldots,\beta_{a-1}$ such that the area under a $N(0,1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/a$ (β_0 and β_a are defined as $-\infty$ and ∞, respectively).

These breakpoints may be determined by looking them up in a statistical table. For example, Table 3 gives the breakpoints for values of *a* from 3 to 10.

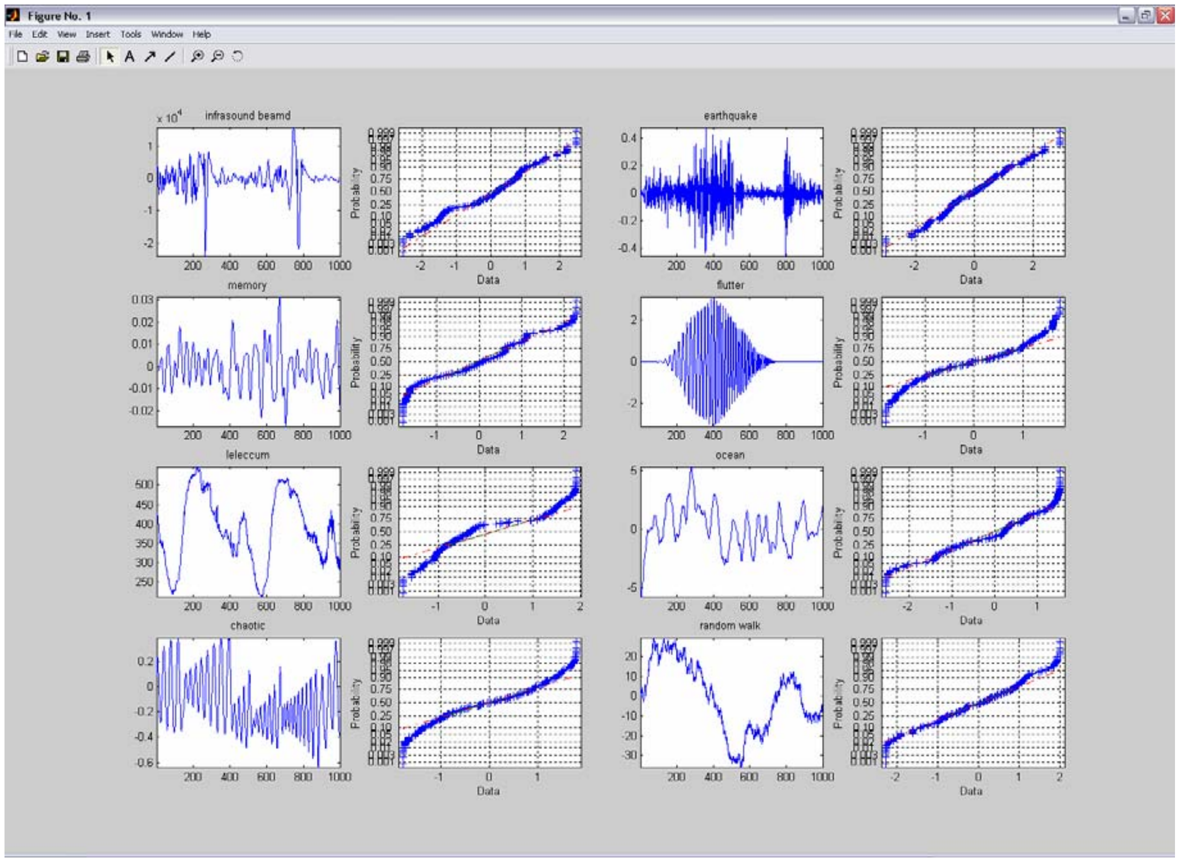


Figure 4: A normal probability plot of the distribution of values from subsequences of length 128 from 8 different datasets. The highly linear nature of the plot strongly suggests that the data came from a Gaussian distribution

Table 3: A lookup table that contains the breakpoints that divide a Gaussian distribution in an arbitrary number (from 3 to 10) of equiprobable regions

β_i \ *a*	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Once the breakpoints have been obtained we can discretize a time series in the following manner. We first obtain a PAA of the time series. All PAA coefficients that are below the smallest breakpoint are mapped to the symbol "**a**", all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol "**b**", etc. Figure 5 illustrates the idea.

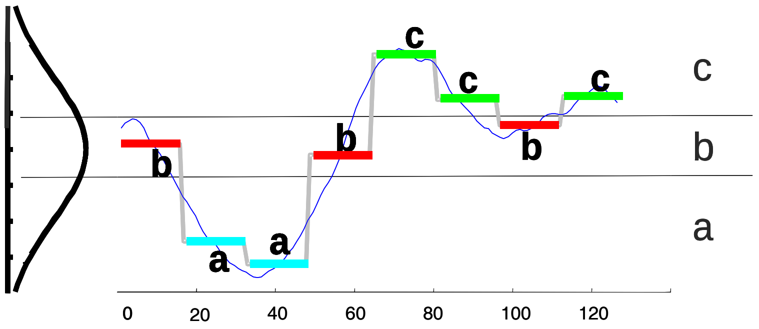


Figure 5: A time series is discretized by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into SAX symbols. In the example above, with n = 128, w = 8 and a = 3, the time series is mapped to the word **baabccbc**

Note that in this example the 3 symbols, "**a**", "**b**" and "**c**" are approximately equiprobable as we desired. We call the concatenation of symbols that represent a subsequence a *word*.

Definition 2. *Word*: A subsequence C of length n can be represented as a *word* $\hat{C} = \hat{c}_1, \ldots, \hat{c}_w$ as follows. Let alpha$_i$ denote the ith element of the alphabet, i.e., alpha$_1$ = **a** and alpha$_2$ = **b**. Then the mapping from a PAA approximation $\overline{C}$ to a word $\hat{C}$ is obtained as follows:

$$\hat{c}_i = alpha_j, \quad iif \quad \beta_{j-1} \le \overline{c}_i < \beta_j \qquad (2)$$

We have now defined our symbolic representation (the PAA representation is merely an intermediate step required to obtain the symbolic representation).

Recently, [6] has empirically and theoretically shown some very promising clustering results for "clipping", that is to say, converting the time series to a binary vector. They demonstrated that discretizing the time series before clustering significantly improves the accuracy in the presence of outliers. We note that "clipping" is actually a special case of SAX, where a = 2.

3.3 Distance Measures

Having introduced the new representation of time series, we can now define a distance measure on it. By far the most common distance measure for time series is the Euclidean distance [32, 52]. Given two time series Q and C of the same length n, Eq. 3 defines their Euclidean distance, and Figure 6A illustrates a visual intuition of the measure.

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \qquad (3)$$

If we transform the original subsequences into PAA representations, $\overline{Q}$ and $\overline{C}$, using Eq. 1, we can then obtain a lower bounding approximation of the Euclidean distance between the original subsequences by:

$$DR(\overline{Q},\overline{C}) \equiv \sqrt{\tfrac{n}{w}} \sqrt{\sum_{i=1}^{w} (\overline{q}_i - \overline{c}_i)^2} \qquad (4)$$

This measure is illustrated in Figure 6B. If we further transform the data into the symbolic representation, we can define a MINDIST function that returns the minimum distance between the original time series of two words:

$$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\tfrac{n}{w}}\sqrt{\sum_{i=1}^{w}\left(dist(\hat{q}_i,\hat{c}_i)\right)^2} \quad (5)$$

The function resembles Eq. 4 except for the fact that the distance between the two PAA coefficients has been replaced with the sub-function *dist*(). The *dist*() function can be implemented using a table lookup as illustrated in Table 4.

Table 4: A lookup table used by the MINDIST function. This table is for an alphabet of cardinality of 4, i.e. *a=4*. The distance between two symbols can be read off by examining the corresponding row and column. For example, *dist*(**a**,**b**) = 0 and *dist*(**a**,**c**) = 0.67.

	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

The value in cell (r,c) for any lookup table can be calculated by the following expression.

$$cell_{r,c} = \begin{cases} 0, & if\ |r-c| \le 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & otherwise \end{cases} \quad (6)$$

For a given value of the alphabet size *a*, the table needs only be calculated once, then stored for fast lookup. The MINDIST function can be visualized is Figure 6C.

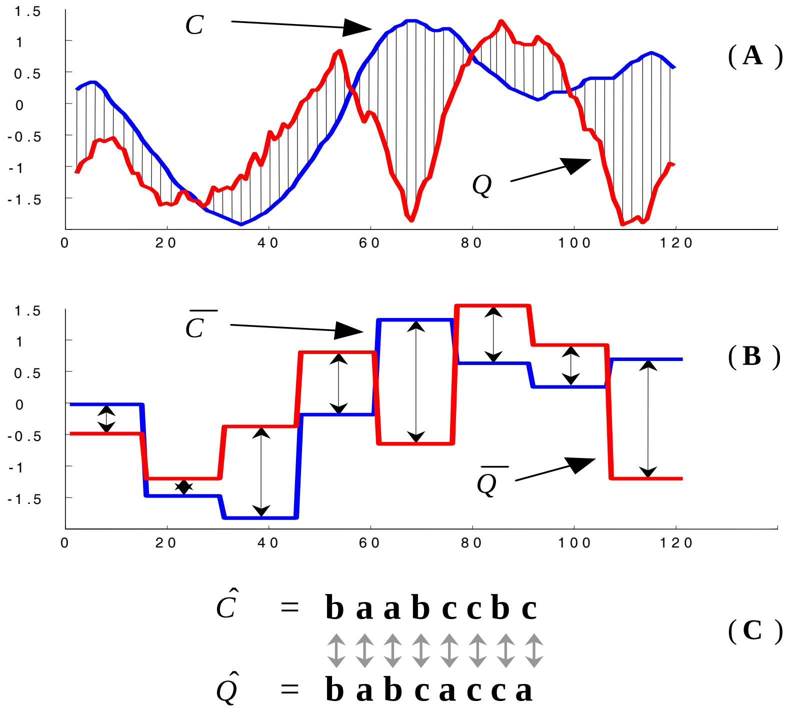


Figure 6: A visual intuition of the three representations discussed in this work, and the distance measures defined on them. **A**) The Euclidean distance between two time series can be visualized as the square root of the sum of the squared differences of each pair of corresponding points. **B**) The distance measure defined for the PAA approximation can be seen as the square root of the sum of the squared differences between each pair of corresponding PAA coefficients, multiplied by the square root of the compression rate. **C**) The distance between two SAX representations of a time series requires looking up the distances between each pair of symbols, squaring them, summing them, taking the square root and finally multiplying by the square root of the compression rate

As mentioned, one of the most important characteristics of SAX is that it provides a lower-bounding distance measure. Below, we show that MINDIST lower-bounds the Euclidean distance in two steps. First, we will show that the PAA distance lower-bounds the Euclidean distance. The proof has appeared in [31] by the current author; for completeness, we repeat the proof here. Next, we will show that MINDIST lower-bounds the PAA distance, which in turn, by transitivity, shows that MINDIST lower-bounds the Euclidean distance.

Step 1: We need to show that the PAA distance lower-bounds the Euclidean distance; that is, $D(Q,C) \geq DR(\overline{Q},\overline{C})$. We will show the proof on the case where there is a single PAA frame (i.e. mapping the time series into one single PAA coefficient). A more generalized proof for N frames can be obtained by applying the single-frame proof on every frame.

Proof: Using the same notations as Eq. (3) and Eq. (4), we want to prove that

$$\sqrt{\sum_{i=1}^{n}\left(q_i - c_i\right)^2} \geq \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(\overline{q_i} - \overline{c_i})^2} \tag{7}$$

Let $\overline{Q}$ and $\overline{C}$ be the means of time series *Q* and *C*, respectively. Since we are considering only the single-frame case, Ineq. (7) can be rewritten as:

$$\sqrt{\sum_{i=1}^{n}\left(q_i - c_i\right)^2} \geq \sqrt{n}\sqrt{(\overline{Q} - \overline{C})^2} \tag{8}$$

Squaring both sides we get

$$\sum_{i=1}^{n}\left(q_i - c_i\right)^2 \geq n\ (\overline{Q} - \overline{C})^2 \tag{9}$$

Each point q_i in *Q* can be represented in term of $\overline{Q}$, i.e. $q_i = \overline{Q} - \Delta q_i$. Same applies to each point c_i in *C*. Thus, Ineq. (9) can be rewritten as:

$$\sum_{i=1}^{n}((\overline{Q} - \Delta q_i) - (\overline{C} - \Delta c_i))^2 \geq n\ (\overline{Q} - \overline{C})^2 \tag{10}$$

Re-arranging the left-hand side we get

$$\sum_{i=1}^{n}((\overline{Q} - \overline{C}) - (\Delta q_i - \Delta c_i))^2 \geq n\ (\overline{Q} - \overline{C})^2 \tag{11}$$

We can expand and rewrite Ineq. (11) as:

$$\sum_{i=1}^{n}((\overline{Q} - \overline{C})^2 - 2(\overline{Q} - \overline{C})(\Delta q_i - \Delta c_i) + (\Delta q_i - \Delta c_i)^2) \geq n\ (\overline{Q} - \overline{C})^2 \tag{12}$$

By distributive law we get:

$$\sum_{i=1}^{n}(\overline{Q}-\overline{C})^2-\sum_{i=1}^{n}2(\overline{Q}-\overline{C})(\Delta q_i-\Delta c_i)+\sum_{i=1}^{n}(\Delta q_i-\Delta c_i)^2 \ge n\,(\overline{Q}-\overline{C})^2 \qquad (13)$$

Or

$$n\,(\overline{Q}-\overline{C})^2-2(\overline{Q}-\overline{C})\sum_{i=1}^{n}(\Delta q_i-\Delta c_i)+\sum_{i=1}^{n}(\Delta q_i-\Delta c_i)^2 \ge n\,(\overline{Q}-\overline{C})^2 \qquad (14)$$

Recall that $q_i=\overline{Q}-\Delta q_i$, which means that $\Delta q_i=\overline{Q}-q_i$, and similarity, $\Delta c_i=\overline{C}-c_i$. Therefore, the summation part of the second term on the left-hand side of the inequality becomes:

$$\begin{aligned}\sum_{i=1}^{n}(\Delta q_i-\Delta c_i)&=\sum_{i=1}^{n}((\overline{Q}-q_i)-(\overline{C}-c_i))\\&=(\sum_{i=1}^{n}\overline{Q}-\sum_{i=1}^{n}q_i)-(\sum_{i=1}^{n}\overline{C}-\sum_{i=1}^{n}c_i)\\&=(n\overline{Q}-\sum_{i=1}^{n}q_i)-(n\overline{C}-\sum_{i=1}^{n}c_i)\\&=(\sum_{i=1}^{n}q_i-\sum_{i=1}^{n}q_i)-(\sum_{i=1}^{n}c_i-\sum_{i=1}^{n}c_i)\\&=0-0\\&=0\end{aligned}$$

Substituting 0 into the second term on the left-hand side, Ineq. (14) becomes:

$$n\,(\overline{Q}-\overline{C})^2-0+\sum_{i=1}^{n}(\Delta q_i-\Delta c_i)^2 \ge n\,(\overline{Q}-\overline{C})^2 \qquad (15)$$

Cancelling out $n(\overline{Q}-\overline{C})^2$ on both sides of the inequality, we get

$$\sum_{i=1}^{n}(\Delta q_i-\Delta c_i)^2 \ge 0 \qquad (16)$$

which always holds true, hence completes the proof.

Step 2: Continuing from Step 1 and using the same methodology, we will now show that MINDIST lower-bounds the PAA distance; that is, we will show that

$$n\,(\overline{Q}-\overline{C})^2 \ge n\,(dist(\hat{Q},\hat{C}))^2 \qquad (17)$$

Let a = 1, b = 2, and so forth, there are two possible scenarios:

Case 1: $|(\hat{Q}-\hat{C})|\le 1$. In other words, the symbols representing the two time series are either the same, or consecutive from the alphabet, e.g. $\hat{Q}=\hat{C}='a', or\ \hat{Q}='a'\ and\ \hat{C}='b'$. From Eq. (6), we know that the

MINDIST is 0 in this case. Therefore, the right-hand side of Ineq. (17) becomes zero, which makes the inequality always hold true.

Case 2: $|(\hat{Q}-\hat{C})|>1$. In other words, the symbols representing the two time series are at least two alphabets apart, e.g. $\hat{Q}='c'\ and\ \hat{C}='a'$. For simplicity, assume $\hat{Q}>\hat{C}$; the case where $\hat{Q}<\hat{C}$ can be proven in similar fashion. According to Eq. (6), $dist(\hat{Q},\hat{C})$ is

$$dist\ (\hat{Q},\hat{C}) = \beta_{\hat{Q}-1} - \beta_{\hat{C}} \tag{18}$$

For the example above, $dist('c','a') = \beta_2 - \beta_1$.

Recall that Eq. (2) states the following:

$$\hat{c}_i = \quad alpha_j\ , \quad iif \quad \beta_{j-1} \le \bar{c}_i < \beta_j$$

So we know that

$$\begin{cases} \beta_{\hat{Q}-1} \le \overline{Q} < \beta_{\hat{Q}} \\ \beta_{\hat{C}-1} \le \overline{C} < \beta_{\hat{C}} \end{cases} \tag{19}$$

Substituting Eq. (18) into Ineq. (17) we get

$$n\ (\overline{Q}-\overline{C})^2 \ge n\ (\beta_{\hat{Q}-1} - \beta_{\hat{C}})^2 \tag{20}$$

which implies that

$$\left|\overline{Q}-\overline{C}\right| \ge \left|\beta_{\hat{Q}-1} - \beta_{\hat{C}}\right| \tag{21}$$

Note that from our assumptions earlier that $|(\hat{Q}-\hat{C})|>1$ and $\hat{Q}>\hat{C}$ (i.e. $\overline{Q}$ is at a “higher” region than $\overline{C}$), we can drop the absolute value notations on both sides:

$$\overline{Q}-\overline{C} \ge \beta_{\hat{Q}-1} - \beta_{\hat{C}} \tag{22}$$

Rearranging the terms we get:

$$\overline{Q} - \beta_{\hat{Q}-1} \ge \overline{C} - \beta_{\hat{C}} \tag{23}$$

which we know always holds true since, from Ineq. (19), we know that

$$\begin{cases} \overline{Q} - \beta_{\hat{Q}-1} \geq 0 \\ \overline{C} - \beta_{\hat{C}} < 0 \end{cases} \tag{24}$$

This completes the proof for $\hat{Q} > \hat{C}$. The case where $\hat{Q} < \hat{C}$ can be proven similarily, and is omitted for brevity.

There is one issue we must address if we are to use a symbolic representation of time series. If we wish to approximate a massive dataset in main memory, the parameters *w* and *a* have to be chosen in such a way that the approximation makes the best use of the primary memory available. There is a clear tradeoff between the parameter *w* controlling the number of approximating elements, and the value *a* controlling the granularity of each approximating element.

It is unfeasible to determine the best tradeoff analytically, since it is highly data dependent. We can however empirically determine the best values with a simple experiment. Since we wish to achieve the tightest possible lower bounds, we can simply estimate the lower bounds over all possible feasible parameters, and choose the best settings.

$$Tightness\ of\ \ Lower\ Bound = \frac{MINDIST(\hat{Q},\hat{C})}{D(Q,C)} \tag{25}$$

We performed such a test with a concatenation of 50 time series databases taken from the UCR time series data mining archive. For every combination of parameters we averaged the result of 100,000 experiments on subsequences of length 256. Figure 7 shows the results.

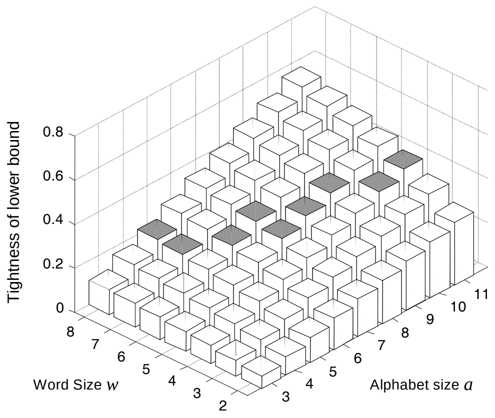


Figure 7: The empirically estimated tightness of lower bounds over the cross product of *a* = [3…11] and *w* = [2…9]. The darker histogram bars illustrate combinations of parameters that require approximately equal space to store every possible word.

The results suggest that using a low value for *a* results in weak bounds. While it’s intuitive that larger alphabet sizes yield better results, there are diminishing returns as *a* increases. If space is an issue, an alphabet size in the range 5 to 8 seems to be a good choice that offers a reasonable balance between space and tightness of lower bound – each alphabet within this range can be represented with just 3 bits. Increasing the alphabet size would require more bits to represent each alphabet.

We end this section with a visual comparison between SAX and the four most used representations in the literature (Figure 8). We can see that SAX preserves the general shape of the original time series. Note that since SAX is a symbolic representation, the alphabets can be stored as bits rather than doubles, which results in a considerable amount of space-saving. Therefore, SAX representation can afford to have higher dimensionality than the other real-valued approaches, while using less or the same amount of space.

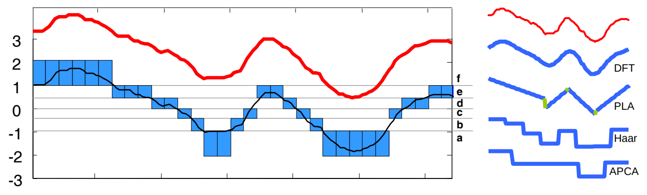


Figure 8: A visual comparison of SAX and the four most common time series data mining representations. A raw time series of length 128 is transformed into the word **ffffffeeeddcbaabceedcbaaaaacddee**. This is a fair comparison since the number of bits in each representation is the same

3.4 Numerosity Reduction

We have seen that, given a single time series, our approach can significantly reduce its dimensionality. In addition, our approach can reduce the numerosity of the data for some applications.

Most applications assume that we have one very long time series *T*, and that manageable *subsequences* of length *n* are extracted by use of a sliding window, then stored in a matrix for further manipulation [11, 20, 30, 63]. Figure 9 illustrates the idea.

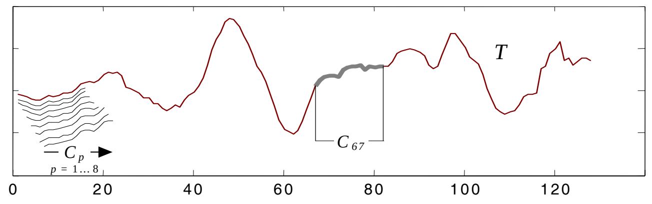


Figure 9: An illustration of the notation introduced in this section: A time series *T* of length 128, the *subsequence* C_{67}, of length $n = 16$, and the first 8 subsequences extracted by a *sliding window*. Note that the sliding windows are overlapping

When performing sliding windows subsequence extraction, with any of the real-valued representations, we must store all $|T| - n + 1$ extracted subsequences (in dimensionality reduced form). However, imagine for a moment that we are using our proposed approach. If the first word extracted is **aabbcc**, and the window is shifted to discover that the second word is also **aabbcc**, we can reasonably decide not to include the second occurrence of the word in sliding windows matrix. If we ever need to retrieve all occurrences of **aabbcc**, we can go to the location pointed to by the first occurrences, and remember to slide to the right, testing to see if the next window is also mapped to the same word. We can stop testing as soon as the word changes. This simple idea is very similar to the run-length-encoding data compression algorithm.

The utility of this optimization depends on the parameters used and the data itself, but it typically yields a numerosity reduction factor of two or three. However, many datasets are characterized by long periods of little or no movement, followed by bursts of activity (seismological data is an obvious example). On these datasets the numerosity reduction factor can be huge. Consider the example shown in Figure 10.

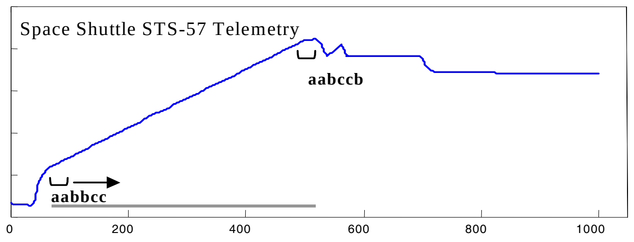


Figure 10: Sliding window extraction on Space Shuttle Telemetry data, with *n* = 32. At time point 61, the extracted word is **aabbcc**, and the next 401 subsequences also map to this word. Only a pointer to the first occurrence must be recorded, thus producing a large reduction in numerosity

There is only one special case we must consider. As we noted in Section 3.1, we normalize each time series (including subsequences) to have a mean of zero and a standard deviation of one. However, if the subsequence contains only one value, the standard deviation is not defined. More troublesome is the case where the subsequence is *almost* constant, perhaps 31 zeros and a single 0.0001. If we normalize this subsequence, the single differing element will have its value exploded to 5.65. This situation occurs quite frequently. For example, the last 200 time units of the data in Figure 10 appear to be constant, but actually contain tiny amounts of noise. If we were to normalize subsequences extracted from this area, the normalization would magnify the noise to large meaningless patterns.

We can easily deal with this problem, if the standard deviation of the sequence before normalization is below an epsilon ε, we simply assign the entire word to the middle-ranged alphabet (e.g. **cccccc** if *a* = 5).

3.5 Relaxation on the Number of Segments

So far we have described SAX with the assumption that the length of the time series is divisible by the number of segments, i.e. *n/w* must be an integer. If *n* is not dividable by *w*, there will be some points in the time series that we are not sure which segment to put them. For example, in Figure 11A, we are dividing 10 data points into 5 segments. And it is obvious that point 1, 2 should be in segment 1; point 3, 4 should be in segment 2; so on and so forth. In Figure 11B, we are dividing 10 data points into 3 segments. It's not clear which segment point 4 should go: segment 1 or segment 2. Same problem holds for point 7. The assumption *n* must be dividable by *w* clearly limits our choices of *w*, and is problematic if *n* is a prime number. Here we show that this needs not be the case and provide a simple solution when *n* is not divisible by *w*.

Instead of putting the whole point into a segment, we can put part of it. For example, in Figure 11B, point 4 contributes its 1/3 to segment 1 and its 2/3 to segment 2, and point 7 contributes its 2/3 to segment 2 and its 1/3 to segment 3. This makes each segment contains exactly 3 1/3 data points and solves the undividable problem. This generalization is implemented in the later version of SAX, as well as some of the applications that utilize SAX.

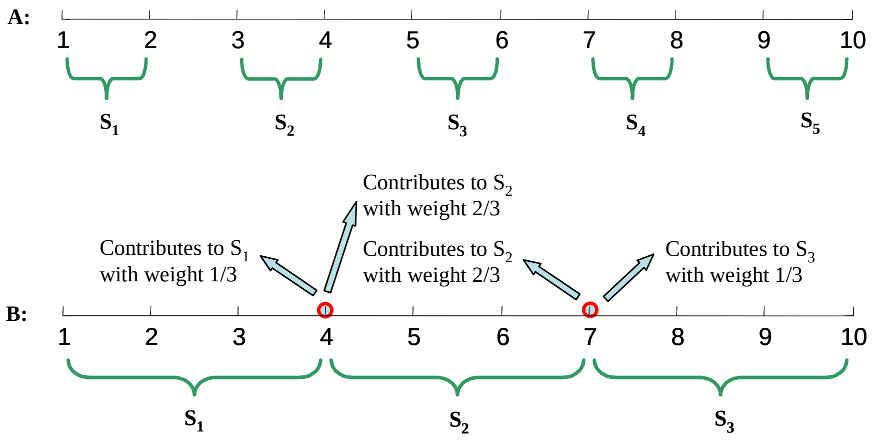

A:
1
2
3
4
5
6
7
8
9
10
S1
S2
S3
S4
S5
B:
Contributes to S1 with weight 1/3
Contributes to S2 with weight 2/3
Contributes to S2 with weight 2/3
Contributes to S3 with weight 1/3
S1
S2
S3

Figure 11: A) 10 data points are divided into 5 segments. **B)** 10 data points are divided into 3 segments. The data points marked with circles contribute to two adjacent segments at the same time

4. Experimental Validation of Our Symbolic Approach

In this section, we perform various data mining tasks using our symbolic approach and compare the results with other well-known existing approaches.

For clustering, classification, and anomaly detection, we compare the results with the classic Euclidean distance, and with other previously proposed symbolic approaches. Note that none of these other approaches use dimensionality reduction. In the next paragraphs we summarize the strawmen representations that we compare ours to. We choose these two approaches since they are typical representatives of approaches in the literature.

André-Jönsson, and Badal [3] proposed the SDA algorithm that computes the changes between values from one instance to the next, and divide the range into user-predefined sections. The disadvantages of this approach are obvious: prior knowledge of the data distribution of the time series is required in order to set the breakpoints; and the discretized time series does not conserve the general shape or distribution of the data values.

Huang and Yu proposed the IMPACTS algorithm, which uses change ratio between one time point to the next time point to discretize the time series [27]. The range of change ratios are then divided into equal-sized sections and mapped into symbols. The time series is converted to a discretized collection of change ratios. As with SAX, the user needs to define the cardinality of symbols.

4.1 Clustering

Clustering is one of the most common data mining tasks, being useful in its own right as an exploratory tool, and also as a sub-routine in more complex algorithms [16, 21, 29]. We consider two clustering algorithms, one of hierarchical clustering, and one of partitional clustering.

4.1.1 Hierarchical Clustering

Comparing hierarchical clusterings is a very good way to compare and contrast similarity measures, since a dendrogram of size N summarizes O(N^2) distance calculations [32]. The evaluation is typically subjective, we simply adjudge which distance measure appears to create the most natural groupings of the data. However, if we know the data labels in advance we can also make objective statements of the quality of the clustering. In Figure 12 we clustered nine time series from the Control Chart dataset, three each from the *decreasing trend*, *upward shift* and *normal* classes.

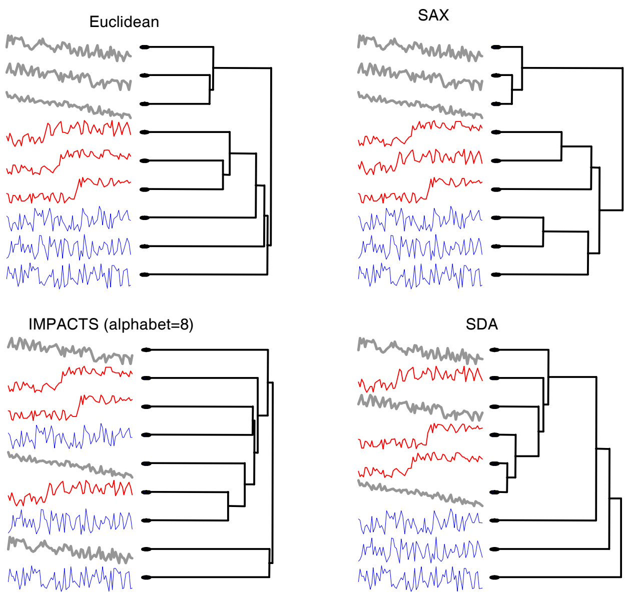


Figure 12: A comparison of the four distance measures' ability to cluster members of the Control Chart dataset. Complete linkage was used as the agglomeration technique

In this case we can objectively state that SAX is superior, since it correctly assigns each class to its own subtree. This is simply a side effect due to the smoothing effect of dimensionality reduction. Therefore, it's not surprising that SAX can sometimes outperform the simple Euclidean distance, especially on noisy data, or data with shifting on the time-axis. This fact is demonstrated in the dendrogram produced by Euclidean distance: the "normal" class, which contains a lot of noise, is not clustered correctly. More generally, we observed that SAX closely mimics Euclidean distance on various datasets.

The reasons that SDA and IMPACTS perform poorly, we observe, are that neither symbolic representation is very descriptive of the general shape of the time series, and that the lack of dimensionality reduction can further distort the results if the data is noisy. What SDA does is essentially differencing the time series, and then discretizing the resulting series. While differencing has been used historically in statistical time series analysis, its purposes to remove some autocorrelation, and to make a time series stationary are not always applicable in determination of similarity in data mining. In addition, although computing the derivatives tells the type of change from one time point to the next time point: sharp increase, slight increase, sharp decrease, etc., this approach doesn't appear very useful since time series data are typically noisy. More specifically, in addition to the overall trends or shapes, there are noises that appear throughout the entire time series. Without any smoothing or dimensionality reduction, these noises are likely to overshadow the actual characteristics of the time series.

To demonstrate why the "decreasing trend" and the "upward shift" classes are indistinguishable by the clustering algorithm for SDA, let's look at what the differenced series look like. Figure 13 shows the original time series and their corresponding series after differencing. It's clear that the differenced series from the same class are not any more similar than those from a different class. As a matter of fact, as we compute the pairwise distances between all 6 differenced series, we realize that the distances are not indicative at all of the classes these data belong. Table 5 and Table 6 show the inter- and the intra-distances between the series (the series from the "decreasing trend" class are denoted as A_i, and the series from the "upward shift" are denoted as B_i).

Interestingly, in [23], the authors show that taking the first derivatives (i.e. differencing) actually worsens the results when compared to using the raw data. They further show that performing piecewise normalization (i.e. normalization on fixed-sized windows rather on the whole series) on the first derivatives improves the results. Our experimental results validate their observations, as SDA does not do any kind of normalization, whereas piecewise normalization is a part of SAX (in the PAA step).

IMPACTS suffers from similar problems as SDA. In addition, it's clear that neither IMPACTS nor SDA can beat simple Euclidean distance, and the discussion above applies to all data mining tasks, since the problems lie in the nature of the representations.

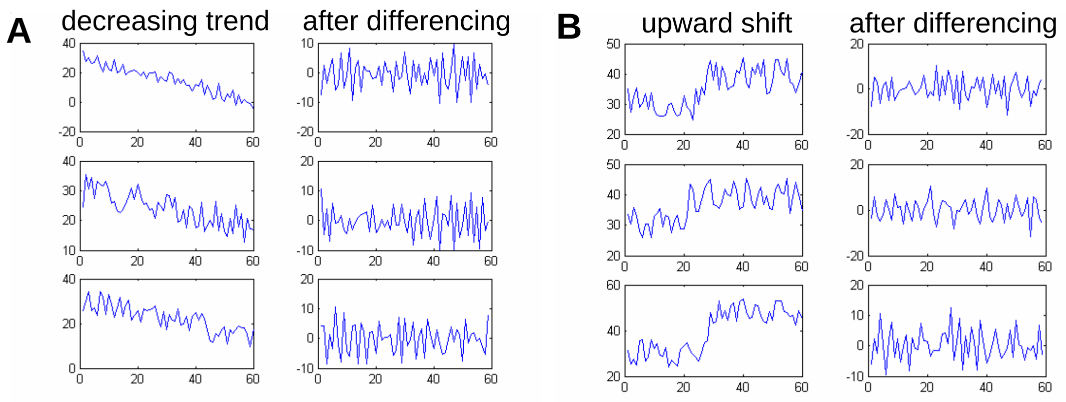


Figure 13: A) Time series from the "decreasing trend" class and the resulting series after differencing. **B)** Time series from the "upward shift" class and the resulting series after differencing.

Table 5: Intra-class distances between the differenced time series from the "decreasing trend" class.

	A1	A2	A3
A1	0	60.59	59.08
A2	60.59	0	57.12
A3	59.08	57.12	0

Table 6: Inter-class distances between the differenced time series from the "decreasing trend" and the "upward shift" classes.

	B1	B2	B3
A1	49.92	46.02	49.21
A2	54.24	58.07	61.38
A3	51.28	49.07	51.72

4.1.2 Partitional Clustering

Although hierarchical clustering is a good sanity check for any proposed distance measure, it has limited utility for data mining because of its poor scalability. The most commonly used data mining clustering algorithm is *k*-means [21], so for completeness we will consider it here. We performed *k*-means on both the original raw data, and our symbolic representation. Figure 14 shows a typical run of *k*-means on a space telemetry dataset. Both algorithms converge after 11 iterations. Since *k*-means algorithm seeks to optimize the objective function, by *minimizing* the sum of squared intra-cluster error, we compare and plot the objective functions for each iteration. The objective function for a given clustering is given by Eq. 26, where x_i is the time series, and c_m is the cluster center of the cluster that x_i belongs to. The smaller the objective function, the more compact (thus better) the clusters.

$$F = \sum_{m=1}^{k} \sum_{i=1}^{N} \left\| x_i - c_m \right\| \qquad (26)$$

The results here are quite unintuitive and surprising: working with an approximation of the data gives better results than working with the original data. Fortunately, a recent paper offers a suggestion as to why this might be so. It has been shown that initializing the clusters centers on a low dimension approximation of the data can improve the quality [16], this is what clustering with SAX implicitly does.

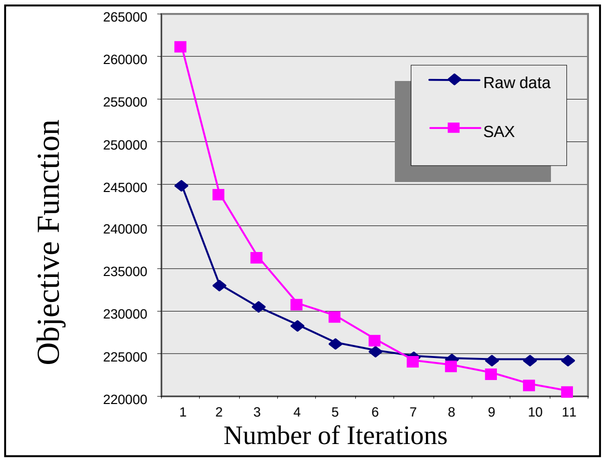


Figure 14: A comparison of the *k*-means clustering algorithm using SAX and using the raw data. The dataset was Space Shuttle telemetry, 1,000 subsequences of length 512. Surprisingly, working with the symbolic approximation produces better results than working with the original data

In 4.4.3 we introduce another distance measure based on SAX. By applying it on clustering, we show that it outperforms the Euclidean distance measure.

4.2 Classification

Classification of time series has attracted much interest from the data mining community. Although special-purpose algorithms have been proposed [36], we will consider only the two most common classification algorithms for brevity, clarity of presentations and to facilitate independent confirmation of our findings.

4.2.1 Nearest Neighbor Classification

To compare different distance measures on 1-nearest-neighbor classification, we use leaving-one-out cross validation. Firstly, we compare SAX with Euclidean distance, IMPACTS, SDA, and LP_{inf}. Two classic synthetic datasets are used: the Cylinder-Bell-Funnel (CBF) dataset has 50 instances of time series for each of the three clusters, and the Control Chart (CC) has 100 instances for each of the six clusters [32].

Since SAX allows dimensionality and alphabet size as user input, and the IMPACTS allows variable alphabet size, we ran the experiments on different combinations of dimensionality reduction and alphabet size. For the other approaches we applied the simple dimensionality reduction technique of skipping data points at a fixed interval. In Figure 15, we show the result with a dimensionality reduction of 4 to 1.

Similar results were observed for other levels of dimensionality reduction. Once again, SAX's ability to beat Euclidean distance is probably due to the smoothing effect of dimensionality reduction, nevertheless this experiment does show the superiority of SAX over the others proposed in the literature.

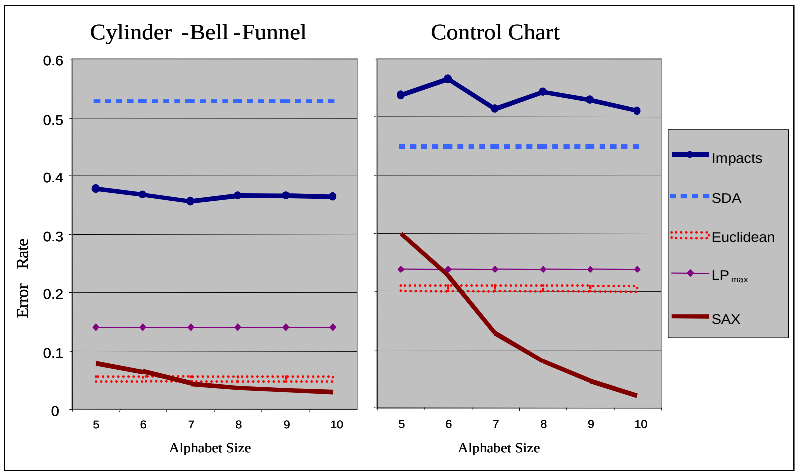


Figure 15: A comparison of five distance measures utility for nearest neighbor classification. We tested different alphabet sizes for SAX and IMPACTS, SDA's alphabet size is fixed at 5

Since both IMPACTS and SDA perform poorly compared to Euclidean distance and SAX, we will exclude them from the rest of the classification experiments. To provide a closer look on how SAX compares to Euclidean distance, we ran an extensive experiment and compared the error rates on 22 datasets (available online at http://www.cs.ucr.edu/~eamonn/time_series_data/). Each dataset is split to training and testing parts. We use the training part to search the best value for SAX parameters *w* (number of SAX words) and *a* (size of the alphabet):

- For *w*, we search from 2 up to *n*/2 (n is the length of the time series). Each time we double the value of *w*.
- For *a*, we search each value between 3 and 10.
- If there is a tie, we use the smaller values.

The compression ratio (last column of next table) is calculated as: $\frac{w \times \lceil \log_2 a \rceil}{n \times 32}$, because for SAX representation we only need $\lceil \log_2 a \rceil$ bits per word, while for the original time series we need 4 bytes (32 bits) for each value.

Then we classify the testing set based on the training set using one nearest neighbor classifier and report the error rate. The results are shown in Table 7. We also summarize the results by plotting the error rates for each dataset as a 2-dimensional point: (EU_error, SAX_error). If a point falls within the lower triangle, then SAX is more accurate than Euclidean distance, and vice versa for the upper triangle. The plot is shown in Figure 16. From this experiment, we can conclude that SAX is competitive with Euclidean distance, but requires far less space.

Table 7: 1-NN comparison between Euclidean Distance and SAX.

Name	Number of Classes	Size of Training Set	Size of Testing Set	Time Series Length	1-NN EU Error	1-NN SAX Error	w	a	Compression Ratio
Synthetic Control	6	300	300	60	0.12	0.02	16	10	3.33%
Gun-Point	2	50	150	150	0.087	0.18	64	10	5.33%
CBF	3	30	900	128	0.148	0.104	32	10	3.13%
Face (all)	14	560	1690	131	0.286	0.330	64	10	6.11%
OSU Leaf	6	200	242	427	0.483	0.467	128	10	3.75%
Swedish Leaf	15	500	625	128	0.211	0.483	32	10	3.13%
50Words	50	450	455	270	0.369	0.341	128	10	5.93%
Trace	4	100	100	275	0.24	0.46	128	10	5.82%
Two Patterns	4	1000	4000	128	0.093	0.081	32	10	3.13%
Wafer	2	1000	6174	152	0.0045	0.0034	64	10	5.26%
Face (four)	4	24	88	350	0.216	0.170	128	10	4.57%
lightning-2	2	60	61	637	0.246	0.213	256	10	5.02%
lightning-7	7	70	73	319	0.425	0.397	128	10	5.02%
ECG	2	100	100	96	0.12	0.12	32	10	4.17%
Adiac	37	390	391	176	0.389	0.890	64	10	4.55%
Yoga	2	300	3000	426	0.170	0.195	128	10	3.76%
Fish	7	175	175	463	0.217	0.474	128	10	3.46%
Plane	7	105	105	144	0.038	0.038	64	10	5.56%
Car	4	60	60	577	0.267	0.333	256	10	5.55%
Beef	5	30	30	470	0.467	0.567	128	10	3.40%
Coffee	2	28	28	286	0.25	0.464	128	10	5.59%
Olive Oil	4	30	30	570	0.133	0.833	256	10	5.61%

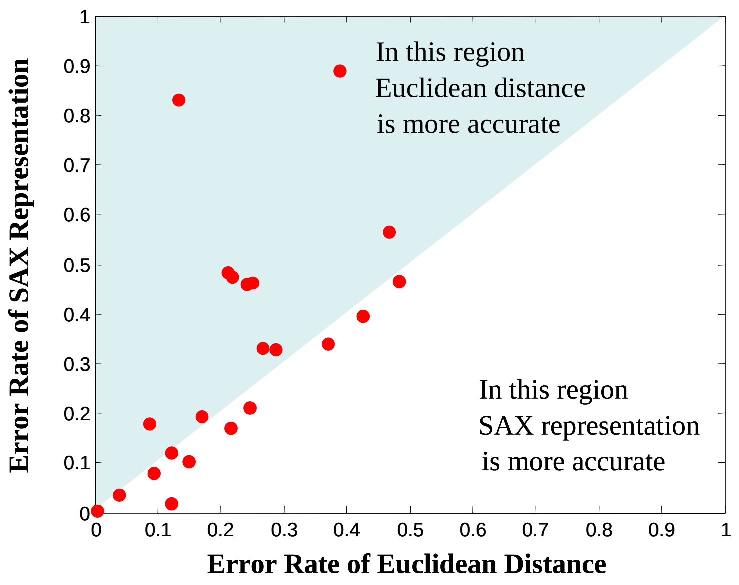


Figure 16: Error rates for SAX and Euclidean distance on 22 datasets. Lower triangle is the region where SAX is more accurate than Euclidean distance, and upper triangle is where Euclidean distance is more accurate than SAX.

4.2.2 Decision Tree Classification

Because of Nearest Neighbor's poor scalability, it is unsuitable for most data mining applications; instead decision trees are the most common choice of classifier. While decision trees are defined for real data, attempting to classify time series using the raw data would clearly be a mistake, since the high dimensionality and noise levels would result in a deep, bushy tree with poor accuracy.

In an attempt to overcome this problem, Geurts [24] suggests representing the time series as a Regression Tree (RT) (this representation is essentially the same as APCA [30], see Figure 2, and training the decision tree directly on this representation. The technique shows great promise.

We compared SAX to the Regression Tree (RT) on two datasets; the results are in Table 8.

Table 8: A comparison of SAX with the specialized Regression Tree approach for decision tree classification. Our approach used an alphabet size of 6, both approaches used a dimensionality of 8

Dataset	SAX	Regression Tree
CC	3.04 ± 1.64	2.78 ± 2.11
CBF	0.97 ± 1.41	1.14 ± 1.02

Note that while our results are competitive with the RT approach, the RT representation is undoubtedly superior in terms of interpretability [24]. Once again, our point is simply that our "black box" approach can be competitive with specialized solutions.

4.3 Query by Content (Indexing)

The majority of work on time series data mining appearing in the literature has addressed the problem of indexing time series for fast retrieval [53]. Indeed, it is in this context that most of the representations enumerated in Figure 1 were introduced [11, 20, 30, 63]. Dozens of papers have introduced techniques to do indexing with a symbolic approach [3, 27], but without exception, the answer set retrieved by these techniques can be very different to the answer set that would be retrieved by the true Euclidean distance. It is only by using a lower bounding technique that one can guarantee retrieving the full answer set, with no false dismissals [20].

To perform query by content, we build an index using SAX, and compare it to an index built using the Haar wavelet approach [11]. Since the datasets we use are large and disk-resident, and the reduced dimensionality could still be potentially high (or at least high enough such that the performance degenerates to sequential scan if R-tree were used [26]), we use Vector Approximation (VA) file as our indexing algorithm. We note, however, that SAX could also be indexed by classic string indexing techniques such as suffix trees.

To compare performance, we measure the percentage of disk I/Os required in order to retrieve the one-nearest neighbor to a randomly extracted query, relative to the number of disk I/Os required for sequential scan. Since it has been forcibly shown that the choice of dataset can make a significant difference in the relative indexing ability of a representation, we tested on more than 50 datasets from the UCR Time Series Data Mining Archive. In Figure 17 we show 4 representative examples. The y-axis shows the index power in terms of the percentage of the data retrieved from the disk, compared to sequential scan. In almost all cases, SAX shows a superior reduction in the number of disk accesses. In addition, SAX does not have the limitation faced by the Haar Wavelet that the data length must be a power of two.

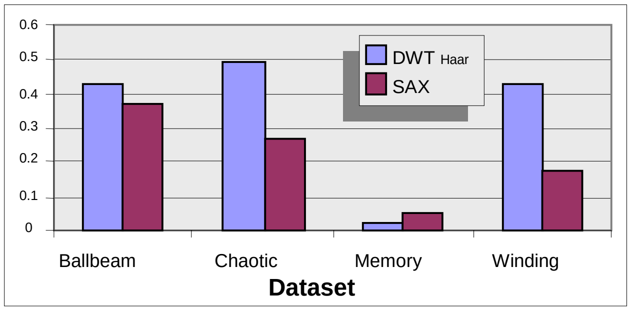


Figure 17: A comparison of indexing ability of wavelets versus SAX. The Y-axis is the percentage of the data that must be retrieved from disk to answer a 1-NN query of length 256, when the dimensionality reduction ratio is 32 to 1 for both approaches

4.4 Taking Advantage of the Discrete Nature of our Representation

In the previous sections we showed examples of how our proposed representation can compete with real-valued representations and the original data. In this section we illustrate examples of data mining algorithms that take explicit advantage of the discrete nature of our representation.

4.4.1 Detecting Novel/Surprising/Anomalous Behavior

A simple idea for detecting anomalous behavior in time series is to examine previously observed normal data and build a model of it. Data obtained in the future can be compared to this model and any lack of conformity can signal an anomaly [14]. In order to achieve this, in [34] we combined a statistically sound scheme with an efficient combinatorial approach. The statistically scheme is based on Markov chains and normalization. Markov chains are used to model the "normal" behavior, which is inferred from the previously observed data. The time- and space-efficiency of the algorithm comes from the use of suffix tree as the main data structure. Each node of the suffix tree represents a pattern. The tree is annotated with a score obtained comparing the support of a pattern observed in the new data with the support recorded in the Markov model. This apparently simple strategy turns out to be very effective in discovering surprising patterns. In the original work we use a simple symbolic approach, similar to IMPACTS [27]; here we revisit the work using SAX.

For completeness, we will compare SAX to two highly referenced anomaly detection algorithms that are defined on real valued representations, the TSA-tree Wavelet based approach of Shahabi et al. [54] and the Immunology (IMM) inspired work of Dasgupta and Forrest [14]. We also include the Markov technique using IMPACTS and SDA in order to discover how much of the difference can be attributed directly to the representation. Figure 18 contains an experiment comparing all 5 techniques.

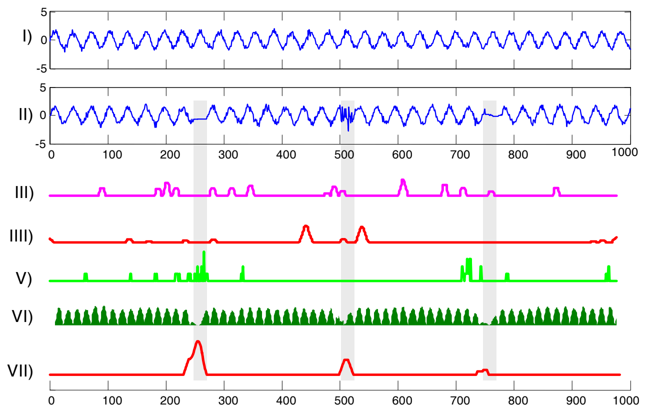


Figure 18: A comparison of five anomaly detection algorithms on the same task. **I)** The training data, a slightly noisy sine wave of length 1,000. **II)** The time series to be examined for anomalies is a noisy sine wave that was created with the same parameters as the training sequence, then an assortment of anomalies were introduced at time periods 250, 500 and 750. **III)** and **IIII)** The Markov Model technique using the IMPACTS and SDA representations did not clearly discover the anomalies, and reported some false alarms. **V)** The IMM anomaly detection algorithm appears to have discovered the first anomaly, but it also reported many false alarms. **VI)** The TSA-Tree approach is unable to detect the anomalies. **VII)** The Markov model-based technique using SAX clearly finds the anomalies, with no false alarms

The results on this simple experiment are impressive. Since suffix trees and Markov models can be used only on discrete data, this offers a motivation for our symbolic approach. While all the other approaches, including the Markov Models using IMPACTS and SDA representations, the Immunology-based anomaly detection approach, and the TSA-Tree approach, did not clearly discover the anomalies and reported some false alarms, the SAX-based Markov Model clearly finds the anomalies with no false alarms.

4.4.2 Motif Discovery

It is well understood in bioinformatics that overrepresented DNA sequences often have biological significance [5, 19, 52]. A substantial body of literature has been devoted to techniques to discover such patterns [25, 57, 60]. In a previous work, we defined the related concept of "time series motif" [43]. Time series motifs are close analogues of their discrete cousins, although the definitions must be augmented to prevent certain degenerate solutions. The naïve algorithm to discover the motifs is quadratic in the length of the time series. In [43], we demonstrated a simple technique to mitigate the quadratic complexity by a large constant factor, nevertheless this time complexity is clearly untenable for most real datasets.

The symbolic nature of SAX offers a unique opportunity to avail of the wealth of bioinformatics research in this area. In particular, recent work by Tompa and Buhler holds great promise [60]. The authors show that many previously unsolvable motif discovery problems can be solved by hashing subsequences into buckets using a random subset of their features as a key, then doing some post-processing search on the hash buckets[3]. They call their algorithm PROJECTION.

We carefully reimplemented the random projection algorithm of Tompa and Buhler, making minor changes in the post-processing step to allow for the fact that although we are hashing random projections of our symbolic representation, we actually wish to discover motifs defined on the original raw data [13]. Figure 19 shows an example of a motif discovered in an industrial dataset [8] using this technique. The patterns found are extremely similar to one another.

[3] Of course, this description greatly understates the contributions of this work. We urge the reader to consult the original paper.

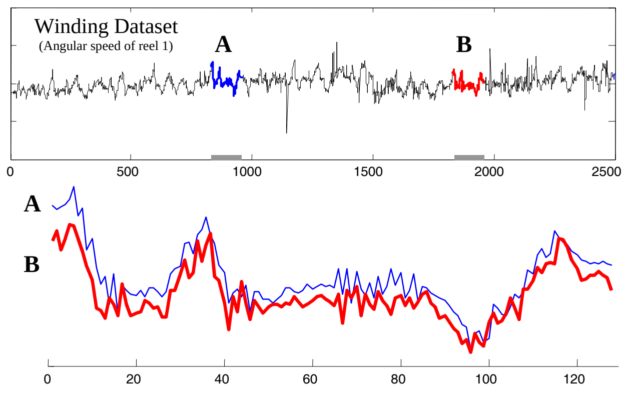


Figure 19: Above, a motif discovered in a complex dataset by the modified PROJECTION algorithm. Below, the motif is best visualized by aligning the two subsequences and "zooming in". The similarity of the two subsequences is striking, and hints at unexpected regularity

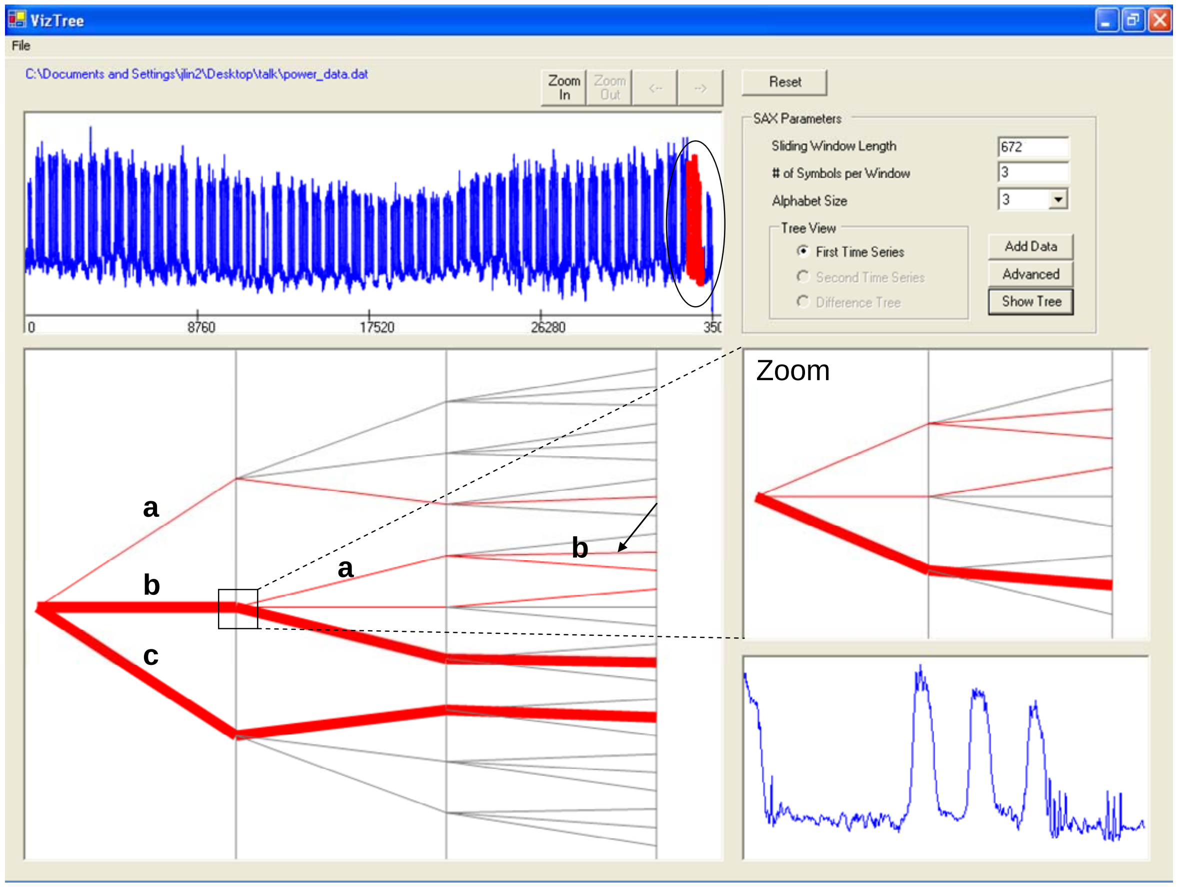
Apart from the attractive scalability of the algorithm, there is another important advantage over other approaches. The PROJECTION algorithm is able to discover motifs even in the presence of noise. Our extension of the algorithm inherits this robustness to noise. We direct interested readers to [13] for more detailed discussion of this algorithm.

4.4.3 Visualization

Data visualization techniques are very important for data analysis, since the human eye has been frequently advocated as the ultimate data-mining tool. However, despite their illustrative nature, which can provide users better understanding of the data and intuitive interpretation of the mining results, there has been surprisingly little work on visualizing large time series datasets. One reason for this lack of interest is that time series data are also usually very massive in size. With limited pixel space and the typically enormous amount of data at hand, it is infeasible to display all the data on the screen at once, much less finding any useful information from the data. How to efficiently organize the data and present them in such a way that is intuitive and comprehensible to human eyes thus remains a great challenge. Ideally, the visualization technique should follow the Visual Information Seeking Mantras, as summarized by Dr. Ben Shneiderman: "*Overview, zoom & filter, details-on-demand.*" In other words, it should be able to provide users the overview or summary of the data, and allows users to further investigate on the interesting patterns highlighted by the tool. To this end, we developed VizTree [42], a time series pattern discovery and visualization system based on augmenting suffix trees. VizTree visually summarizes both the global and local structures of time series data at the same time. In addition, it provides novel interactive solutions to many pattern discovery problems, including the discovery of frequently occurring patterns (motif discovery), surprising patterns (anomaly detection), and query by content. The user interactive paradigm allows users to visually explore the time series, and perform real-time hypotheses testing. Since the use of suffix tree requires that the input data be discrete, SAX is the perfect candidate for discretizing the time series data.

Compared to the existing time series visualization systems in the literature, VizTree is unique in several respects. First, almost all other approaches assume highly periodic time series, whereas VizTree makes no such assumption. Other methods typically require space (both memory space, and pixel space) that grows at least linearly with the length of the time series, making them untenable for mining massive datasets. Finally, VizTree allows us to visualize a much richer set of features, including global summaries of the differences between two time series, locally repeated patterns, anomalies, etc.

In VizTree, patterns are represented in a depth-limited tree structure, in which their frequencies of occurrence are encoded in the thicknesses of branches. The algorithm works by sliding a window across the time series and extracting subsequences of user-defined lengths. The subsequences are then discretized into strings by SAX and inserted into an augmented suffix tree. Each string is regarded as a pattern, and the frequency of occurrence for each pattern is encoded by the thickness of the branch: the thicker the branch, the more frequent the corresponding pattern. Motif discovery and anomaly detection can thus be easily achieved: those that occur frequently can be regarded as motifs, and those that occur rarely can be regarded as anomaly. Figure 20 shows the screenshot of VizTree for anomaly detection on the Dutch power demand dataset. Electricity consumption is recorded every 15 minutes; therefore, for the year of 1997, there are 35,040 data points. The majority of the weeks follow the regular Monday-Friday, 5-working-day pattern, as shown by the thick branches. The thin branches denote the anomalies (in the sense that the electricity consumption is abnormal given the day of the week). Note that in VizTree, we reverse the alphabet ordering so the alphabets now read top-down rather than bottom-up (e.g. '*a*' is now in the topmost branch, rather than in the bottom-most branch). This way, the string better describes the actual shape of the time series – '*a*' denotes the top region, '*b*' the middle region, '*c*' the bottom region. The top right window shows the subtree when we click on the 2nd child of the root node. Clicking on any of the existing branches (in the main or the subtree window) will plot the subsequences represented by them in the bottom right window. The highlighted, circled subsequence is retrieved by clicking on the branch "*bab*." The zoom-in shows why it is an anomaly: it's the beginning of the three-day week during Christmas (Thursday and Friday off). The other thin branches denote other anomalies such as New Year's Day, Good Friday, Queen's Birthday, etc.



C:\Documents and Settings\jlin2\Desktop\talk\power_data.dat

SAX Parameters

Sliding Window Length 672

# of Symbols per Window 3

Alphabet Size 3

Tree View: [✓ First Time Series] Second Time Series Difference Tree

Figure 20: Anomaly detection on power consumption data. The anomaly shown here is a short week during Christmas.

The evaluation for visualization techniques is usually subjective. Although VizTree clearly demonstrates its capability in detecting non-trivial patterns, in [40] we also devise a measure that quantifies the effectiveness of the algorithm. The measure, which we call the "dissimilarity coefficient," describes how dissimilar two time series are, and ranges from 0 to 1. In essence, the coefficient summarizes the difference in "behavior" of each pattern (represented by a string) in two time series. More concretely, for each pattern, we count its respective numbers of occurrences in both time series, and see how much the frequencies differ. We call this measure *support*, which is then weighted by the *confidence*, or the degree of "interestingness" of the pattern. For example, a pattern that occurs 120 times in time series A and 100 times in time series B is probably *less* significant than a pattern that occurs 20 times in A but zero times in B, even though the *support* for both cases is 20.

Subtracting the dissimilarity coefficient from 1 then gives us a novel similarity measure that describes how *similar* two time series are. More details on the (dis)similarity measure can be found in [40]. An important fact about this similarity measure is that, unlike a distance measure that computes point-to-point distances, it captures the global structure of the time series rather than local differences. This time-invariant feature is useful if we are interested in the overall structures of the time series. Figure 21 shows the dendrogram of clustering result using the (dis)similarity coefficient as the distance measure. It clearly demonstrates that the coefficient captures the dissimilarity very well and that all clusters are separated perfectly. Note that it's even able to distinguish the four different sets of heartbeats (from top down, cluster 1, 4, 5, and 6)!

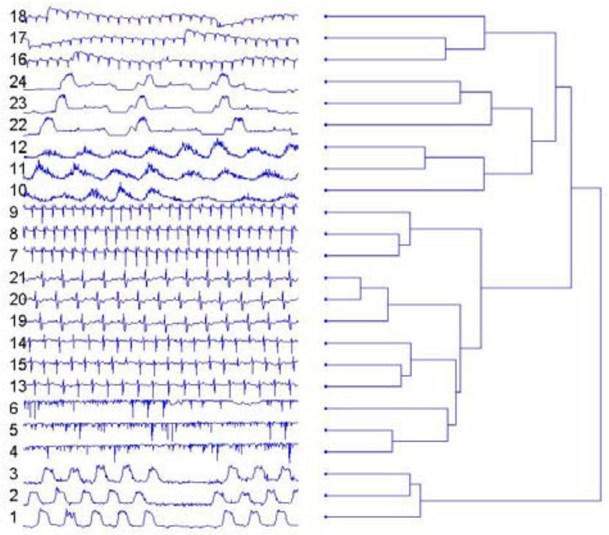


Figure 21: Clustering result using the (dis)similarity coefficient

As a reference, we ran the same clustering algorithm using the widely-used Euclidean distance. The result is shown in Figure 22. Clearly, clustering using our (dis)similarity measure returns superior results.

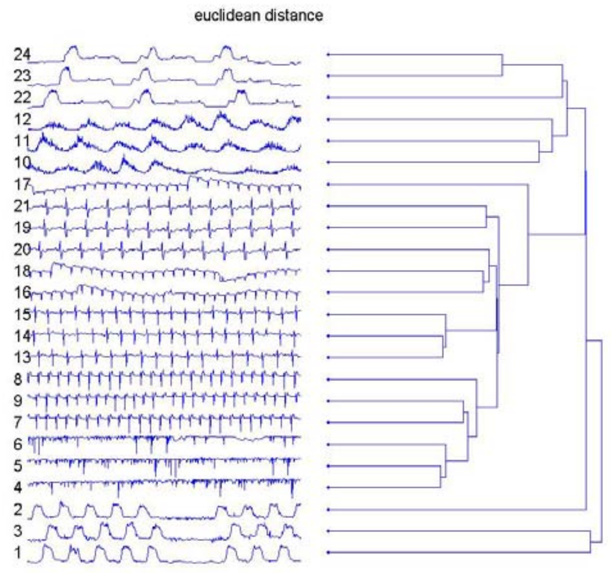


Figure 22: Clustering result using Euclidean distance

So far we discussed how the symbolic nature of SAX makes possible the approaches that were not considered, at least not effectively, by the time series data mining community before, since these approaches require the input be discrete. It sheds some light in offering efficient solutions on various problems from a new direction. The problems listed in this section, however, are just a small subset of examples that show the efficacy of SAX. We have since then proposed more algorithms on anomaly detection (HOT SAX) [33], contrast sets mining [39], visualization by time series bitmaps [37], clustering [51], compression-based distance measures [35], etc., all of which based on SAX. In the next section, we discuss the evident impact of SAX, illustrated by great interests from *other* researchers/users from both the academic and the industrial communities.

4.5 The Impact of SAX

In the relatively short time since its initial introduction, SAX has had a large impact in industry and academia. Below we summarize some of this work, without attempting to be exhaustive. We can broadly classify this work into those who have simply used the original formulation of SAX to solve a particular problem, and those who have attempted to extend or augment SAX in some way.

4.5.1 Applications of SAX

In addition to the applications mentioned in this section, it has been used worldwide in various domains. To name a few, in [4] the authors analyzed complex kinetic mechanisms using a method based on SAX. In [22] the authors consider the problem of analysis of protein unfolding data. After noting that "*a complexity/dimensionality reduction on the data may be necessary and desirable*" the authors consider various alternatives before noting "*We adopted a two step approach called SAX* ." Dr. Amy McGovern of the University of Oklahoma is leading a project on dynamic relational models for improved hazardous weather prediction. In attempting to use propositional models for this task, her group needs a discrete representation of the real-valued metrological data, and they noted in a recent paper "*We are currently using Lin et al.'s SAX approach to creating discrete data from continuous data*" [46]. In [17, 18, 55, 56], the authors use SAX and random projection to discover motifs in telemedicine time series. In [12] the authors convert palmprint to time series, then to SAX, then they do biometric recognition. The authors in [59] use SAX and random projection to mine motion capture data. In [10] SAX is used to find repeated patterns in motion capture data. [49] uses SAX to find rules in time series. In [58] the authors use SAX to find motifs of unspecified length. In [48] SAX is used to find repeated patterns in robot sensors. In [7] the authors use SAX to do spatiotemporal trajectory joins. In [50] the authors use SAX to *"detect multi-headed stealthy attack tools"*.

4.5.2 Extensions to SAX

As noted above, there have been dozens of applications of SAX to diverse problems. However, there has been surprisingly little work to augment or extend the SAX representation itself. We attribute this to the generality of the original framework; it simply works very well for most problems. Nevertheless, recently there have been some SAX extensions, which we consider below.

In [44] the authors augment each SAX symbol by incorporating the minimum and maximum value in the range. Thus each SAX segment contains a triplet of information, rather that a single symbol. Very preliminary results are presented which suggest that this may be useful in some domains. Bernard Hugueney [28] has recently suggested that SAX could be improved by allowing SAX symbols to adaptively represent different length sections of a time series. Just as SAX may be seen as symbolic a symbolic version of the PAA representation [31], Dr. Hugueneys approach may be seen as symbolic a symbolic version of the APCA representation [30]. While the author shows that this does decrease the reconstruction error on some datasets, it is not obvious that the new representation can be used with hashing (as in [43]) or with suffix trees (as in [42]). In contrast to Hugueneys idea of allowing *segment lengths* to be adaptive, recent work by Mörchen and Ultsch has suggested that the *breakpoints* should be adaptive [47]. The authors make a convincing case that in some datasets this may be better than the Gaussian assumption (cf. section 3.2). For example they consider a time series of muscle activity and show that it strongly violates the Gaussian assumption. Note that once the new breakpoints are defined, and tables 3 and 4 are appropriately adjusted, the lower bounding property still holds. Finally, in an interesting new development, Wei et al. have shown techniques to adapt SAX to various problems in 2D shape matching, after modifying algorithms/representations to allow for rotation invariance, which in the SAX representation corresponds to circular shifts [62].

The list of papers using SAX keeps growing at rapid rate. The wide acceptance of SAX by fellow researchers has shown its generality and utilities in diverse domains. There is great potential for adapting and extending SAX on an even broader class of data mining problems.

5. Conclusions and Future Directions

In this work we have formulated the first dimensionality/numerosity reduction, lower bounding symbolic approach in the literature. We have shown that our representation is competitive with, or superior to, other representations on a wide variety of classic data mining problems, and that its discrete nature allows us to tackle emerging tasks such as anomaly detection, motif discovery and visualization.

A host of future directions suggest themselves. There is an enormous wealth of useful definitions, algorithms and data structures in the bioinformatics literature that can be exploited by our representation [5, 19, 25, 45, 52, 57, 60]. It may be possible to create a lower bounding approximation of Dynamic Time Warping [9], by slightly modifying the classic string edit distance. Finally, there may be utility in extending our work to multidimensional and streaming time series [61].

6. References

[1] NIST/SEMATECH e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/

[2] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying Shapes of Histories. In proceedings of the 21st Int'l Conference on Very Large Databases. Zurich, Switzerland. Sept 11-15, 1995. pp. 502-514

[3] H. Andre-Jonsson and D. Badal. Using Signature Files for Querying Time-Series Data. In proceedings of Principles of Data Mining and Knowledge Discovery, 1st European Symposium. Trondheim, Norway. Jun 24-27, 1997. pp. 211-220

[4] I. P. Androulakis. New Approaches for Representing, Analyzing and Visualizing Complex Kinetic Mechanisms. In proceedings of the 15th European Symposium on Computer Aided Process Engineering. Barcelona, Spain. May 29-June 1, 2005.

[5] A. Apostolico, M. E. Bock, and S. Lonardi. Monotony of Surprise in Large-Scale Quest for Unusual Words. In proceedings of the 6th Int'l Conference on Research in Computational Molecular Biology. Washington, D.C. Apr 18-21, 2002. pp. 22-31

[6] A. J. Bagnall and G. Janakec. Clustering Time Series from ARMA Models with Clipped Data. In proceedings of the 10th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Seattle, WA. Aug 22-25, 2004. pp. 49-58

[7] P. Bakalov, M. Hadjieleftherious, and V. J. Tsotras. Time Relaxed Spatiotemporal Trajectory. In proceedings of the ACM International Symposium on Advances in Geographic Information Systems. Bremen, Germany. Nov 4-5, 2005.

[8] T. Bastogne, H. Noura, A. Richard, and J. M. Hittinger. Application of Subspace Methods to the Identification of a Winding Process. In proceedings of the 4th European Control Conference. Brussels, Belgium. 2002.

[9] D. Berndt and J. Clifford, Using Dynamic Time Warping to Find Patterns in Time Series, *the Workshop on Knowledge Discovery in Databases, the 12th Int'l Conference on Artificial Intelligence*. Seattle, WA, 1994, pp. 229-248.

[10] B. Celly and V. B. Zordan. Animated People Textures. In proceedings of the 17th International Conference on Computer Animation and Social Agents. Geneva, Switzerland. July 7-9, 2004.

[11] K. Chan and A. W. Fu. Efficient Time Series Matching by Wavelets. In proceedings of the 15th IEEE Int'l Conference on Data Engineering. Sydney, Australia. Mar 23-26, 1999. pp. 126-133

[12] J. S. Chen, Y. S. Moon, and H. W. Yeung. Palmprint Authentication Using Time Series. In proceedings of the 5th International Conference on Audio- and Video-Based Biometric Person Authentication. Hilton Rye Town, NY. July 20-22, 2005.

[13] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic Discovery of Time Series Motifs. In proceedings of the 9th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Washington DC, USA. Aug 24-27, 2003. pp. 493-498

[14] D. Dasgupta and S. Forrest. Novelty Detection in Time Series Data Using Ideas from Immunology. In proceedings of the 8th Int'l Conference on Intelligent Systems. Denver, CO. Jun 24-26, 1999.

[15] C. S. Daw, C. E. A. Finney, and E. R. Tracy. Symbolic Analysis of Experimental Data. Review of Scientific Instruments. vol. 74. pp. 915-930. 2001.

[16] C. Ding, X. He, H. Zha, and H. Simon. Adaptive Dimension Reduction for Clustering High Dimensional Data. In proceedings of the 2nd IEEE Int'l Conference on Data Mining. Maebashi, Japan. Dec 9-12, 2002. pp. 147-154

[17] F. Duchene and C. Garbay. Apprentissage de Motifs Temporels, Multidimensionnels et Heterogenes: Application a la Telesurveillance Medicale. In proceedings of Conference Francophone sur l'apprentissage Automatique. Nice, France. May 31 - June 3, 2005.

[18] F. Duchene, C. Garbay, and V. Rialle. Mining Heterogeneous Multivariate Time-Series for Learning Meaningful Patterns: Application to Home Health Telecare. Research Report 1070-I, Institue de Informatique et Mathematiques Appliquees de Grenoble. Grenoble, France. 2004.

[19] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids: Cambridge University Press. 1998.

[20] C. Faloutsos, M. Ranganathan, and Y. Manolopulos. Fast Subsequence Matching in Time-Series Databases. SIGMOD Record. vol. 23. pp. 419-429. 1994.

[21] U. Fayyad, C. Reina, and P. Bradley. Initialization of Iterative Refinement Clustering Algorithms. In proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining. New York, NY. Aug 27-31, 1998. pp. 194-198

[22] P. G. Ferreira, P. Azevedo, C. Silva, and R. Brito. Mining Approximate Motifs in Time Series. In proceedings of the 9th International Conference on Discovery Science. Barcelona, Spain. Oct 7-10, 2006.

[23] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the Stock Market: Which Measure is Best? In proceedings of the 6th ACM Int'l Conference on Knowledge Discovery and Data Mining. Boston, MA. Aug 20-23, 2000. pp. 487-496

[24] P. Geurts. Pattern Extraction for Time Series Classification. In proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery. Freiburg, Germany. 2001. pp. 115-127

[25] A. Gionis and H. Mannila. Finding Recurrent Sources in Sequences. In proceedings of the 7th Int'l Conference on Research in Computational Molecular Biology. Berlin, Germany. 2003. pp. 123-130

[26] J. M. Hellerstein, C. H. Papadimitriou, and E. Koutsoupias. Towards an Analysis of Indexing Schemes. In proceedings of the 16th ACM Symposium on Principles of Database Systems. Tucson, AZ. May 12-14, 1997. pp. 249-256

[27] Y. W. Huang and P. S. Yu. Adaptive Query Processing for Time-Series Data. In proceedings of the 5th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. San Diego, CA. Aug 15-18, 1999. pp. 282-286

[28] B. Hugueney. Adaptive Segmentation-Based Symbolic Representation of Time Series for Better Modeling and Lower Bounding Distance Measures. In proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases. Berlin, Germany. Sept 18-22, 2006. pp. 545-552

[29] K. Kalpakis, D. Gada, and V. Puttagunta. Distance Measures for Effective Clustering of ARIMA Time-Series. In proceedings of the 2001 IEEE In'l Conference on Data Mining. San Jose, CA. Nov 29-Dec 2, 2001. pp. 273-280

[30] E. Keogh, K. Chakrabarti, and M. Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In proceedings of ACM SIGMOD Conference on Management of Data. Santa Barbara. May 21-24, 2001. pp. 151-162

[31] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Journal of Knowledge and Information Systems. vol. 3. pp. 263-286. 2001.

[32] E. Keogh and S. Kasetty. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In proceedings of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada. July 23-26, 2002. pp. 102-111

[33] E. Keogh, J. Lin, and A. W. Fu. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In proceedings of the 5th IEEE International Conference on Data Mining. Houston, TX. Nov 27-30, 2005. pp. 226-233

[34] E. Keogh, S. Lonardi, and B. Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space. In proceedings of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada. Jul 23-26, 2002. pp. 550-556

[35] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards Parameter-Free Data Mining. In proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle. Aug 22-25, 2004. pp. 206-215

[36] E. Keogh and M. Pazzani. An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In proceedings of the 4th In'l Conference on Knowledge Discovery and Data Mining. New York, NY. Aug 27-31, 1998. pp. 239-241

[37] N. Kumar, N. Lolla, E. Keogh, S. Lonardi, C. A. Ratanamahatana, and L. Wei. Time Series Bitmaps: A Practical Visualization Tool for Working with Large Time Series Databases. In proceedings of the 2005 SIAM International Conference on Data Mining. Newport Beach, CA. April 21-23, 2005. pp. 531-535

[38] R. J. Larsen and M. L. Marx. An Introduction to Mathematical Statistics and Its Applications. 2nd ed. Englewood, Cliffs, NJ: Prentice Hall. 1986.

[39] J. Lin and E. Keogh. Group SAX: Extending the Notion of Contrast Sets to Time Series and Multimedia Data. In proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases. Berlin, Germany. Sept 18-22, 2006. pp. 284-296

[40] J. Lin, E. Keogh, and S. Lonardi. Visualizing and Discovering Non-Trivial Patterns in Large Time Series Databases. Information Visualization. vol. 4. pp. 61-82. 2005.

[41] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, A Symbolic Representation of Time Series, with Implications for Streaming Algorithms, *Workshop on Research Issues in Data Mining and Knowledge Discovery*, *the 8th ACM SIGMOD*. San Diego, CA, 2003.

[42] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Visually Mining and Monitoring Massive Time Series. In proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, WA. Aug 22-25, 2004. pp. 460-469

[43] J. Lin, E. Keogh, P. Patel, and S. Lonardi, Finding Motifs in Time Series, *the 2nd Workshop on Temporal Data Mining*, *the 8th ACM Int'l Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada, 2002, pp. 53-68.

[44] B. Lkhagva, Y. Suzuki, and K. Kawagoe. New Time Series Data Representation ESAX for Financial Applications. In proceedings of the 22nd International Conference on Data Engineering Workshops. Atlanta, GA. Apr 3-8, 2006. pp. 115

[45] S. Lonardi, Global Detectors of Unusual Words: Design Implementation and Applications to Pattern Discovery in Biosequences, *Department of Computer Sciences*: Purdue University, 2001.

[46] A. McGovern, A. Kruger, D. Rosendahl, and K. Droegemeier. Open Problem: Dynamic Relational Models for Improved Hazardous Weather Prediction. In proceedings of ICML Workshop on Open Problems in Statistical Relational Learning. Pittsburgh, PA. June 29, 2006.

[47] F. Morchen and A. Ultsch. Optimizing Time Series Discretization for Knowledge Discovery. In proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago, IL. Aug 21-24, 2005. pp. 660-665

[48] K. Murakami, Y. Yano, S. Doki, and S. Okuma. Behavior Extraction from a Series of Observed Robot Motion. In proceedings of JSME Conference on Robotics and Mechatronics. Nagoya, Japan. June, 2004.

[49] M. Ohsaki, Y. Sato, H. Yokoi, and T. Yamaguchi, A Rule Discovery Support System for Sequential Medical Data, in the Case Study of a Chronic Hepatitis Dataset, *Discovery Challenge Workshop*, *the 14th European Conference on Machine Learning/the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*. Cavtat-Dubrovnik, Croatia, 2003.

[50] F. Pouget, G. Urvoy-Keller, and M. Dacier. Time Signature to Detect Multi-Headed Stealthy Attack Tools. In proceedings of the 18th Annual FIRST Conference. Baltimore, MD. June 25-30, 2006.

[51] C. A. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi. A Novel Bit Level Time Series Representation with Implications for Similarity Search and Clustering. In proceedings of Advances

in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference. Hanoi Vietnam. May 18-20, 2005. pp. 771-777

[52] G. Reinert, S. Schbath, and M. S. Waterman. Probabilistic and Statistical Properties of Words: An Overview. Journal of Computational Biology. vol. 7. pp. 1-46. 2000.

[53] J. F. Roddick, K. Hornsby, and M. Spiliopoulou, An Updated Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research, *Post-Workshop proceedings of the Int'l Workshop on Temporal, Spatial and Spatio-Temporal Data Mining*. Berlin: Springer, 2001, pp. 147-163.

[54] C. Shahabi, X. Tian, and W. Zhao. TSA-Tree: A Wavelet-Based Approach to Improve the Efficiency of Multi-Level Surprise and Trend Queries. In proceedings of the 12th Int'l Conference on Scientific and Statistical Database Management. Berlin, Germany. Jul 26-28, 2000. pp. 55-68

[55] A. Silvent, C. Carbay, P. Y. Carry, and M. Dojat. Data Information and Knowledge for Medical Scenario Construction. In proceedings of the Intelligent Data Analysis in Medicine and Pharmacology Workshop. Protaras, Cyprus. Oct 19-22, 2003.

[56] A. Silvent, M. Dojat, and C. Garbay. Multi-level Temporal Abstraction for Medical Scenario Construction. International Journal of Adaptive Control and Signal Processing. 2004.

[57] R. Staden. Methods for Discovering Novel Motifs in Nucleic Acid Sequences. Computer Applications in Biosciences. vol. 5. pp. 293-298. 1989.

[58] Y. Tanaka and K. Uehara. Discover Motifs in Multi Dimensional Time-Series Using the Principal Component Analysis and the MDL Principle. In proceedings of the 3rd Int'l Conference on Machine Learning and Data Mining in Pattern Recognition. Leipzig, Germany. Jul 5-7, 2003. pp. 252-265

[59] Y. Tanaka and K. Uehara. Motif Discovery Algorithm from Motion Data. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence. Kanazawa, Japan. June 2-4, 2004.

[60] M. Tompa and J. Buhler. Finding Motifs Using Random Projections. In proceedings of the 5th Int'l Conference on Computational Molecular Biology. Montreal, Canada. Apr 22-25, 2001. pp. 67-74

[61] M. Vlachos, G. Kollios, and G. Gunopulos. Discovering Similar Multidimensional Trajectories. In proceedings of the 18th Int'l Conference on Data Engineering. San Jose, CA. Feb 26-Mar 1, 2002. pp. 673-684

[62] L. Wei, E. Keogh, and X. Xi. SAXually Explicit Images: Finding Unusual Shapes. In proceedings of the 2006 IEEE International Conference on Data Mining. Hong Kong. Dec 18-22, 2006.

[63] B. K. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. In proceedings of the 26th Int'l Conference on Very Large Databases. Cairo, Egypt. Sep 10-14, 2000. pp. 385-394