A Symbolic Representation of Time Series, with Implications for Streaming Algorithms

Jessica Lin Eamonn Keogh Stefano Lonardi Bill Chiu

University of California - Riverside
Computer Science & Engineering Department
Riverside, CA 92521, USA
{*jessica, eamonn, stelo, bill*}*@cs.ucr.edu*

ABSTRACT

The parallel explosions of interest in streaming data, and data mining of time series have had surprisingly little intersection. This is in spite of the fact that time series data are typically streaming data. The main reason for this apparent paradox is the fact that the vast majority of work on streaming data explicitly assumes that the data is discrete, whereas the vast majority of time series data is real valued.

Many researchers have also considered transforming real valued time series into symbolic representations, noting that such representations would potentially allow researchers to avail of the wealth of data structures and algorithms from the text processing and bioinformatics communities, in addition to allowing formerly "batch-only" problems to be tackled by the streaming community. While many symbolic representations of time series have been introduced over the past decades, they all suffer from three 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. Finally, most of these symbolic approaches require one to have access to all the data, before creating the symbolic representation. This last feature explicitly thwarts efforts to use the representations with streaming algorithms.

In this work we introduce 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. Finally, our representation allows the real valued data to be converted in a streaming fashion, with only an infinitesimal time and space overhead.

We will demonstrate the utility of our representation on the classic data mining tasks of clustering, classification, query by content and anomaly detection.

Keywords

Time Series, Data Mining, Data Streams, Symbolic, Discretize

1. INTRODUCTION

The parallel explosions of interest in streaming data [4, 8, 10, 18], and data mining of time series [6, 7, 9, 20, 21, 24, 26, 34] have had surprisingly little intersection. This is in spite of the fact that time series data are typically streaming data, for example, stock value, medical and meteorological data [30]. The main reason for this apparent paradox is the fact that the vast majority of work on streaming data explicitly assumes that the data is discrete, whereas the vast majority of time series data is real valued [23].

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 [2, 7, 14, 16, 20, 22, 25, 30, 31, 35]. 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. In addition to allowing the framing of time series problems as streaming problems, there is an enormous wealth of existing algorithms and data structures that allow the efficient manipulations of symbolic representations. Such algorithms have received decades of attention in the text retrieval community, and more recent attention from the bioinformatics community [3, 13, 17, 29, 32, 33]. 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. As a more concrete example, consider the Jaccard coefficient [13], a distance measure beloved by streaming researchers. The Jaccard coefficient is only well defined for discrete data (such as web clicks or individual keystrokes) as thus cannot be used with real-valued time series.

There is a simple explanation for the data mining community's lack of interest in symbolic 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. This simple fact is at the core of almost all algorithms in time series data mining and indexing [14]. However, in spite of the fact that there are dozens of techniques for producing different variants of the symbolic representation [2, 11, 20], 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 [19].

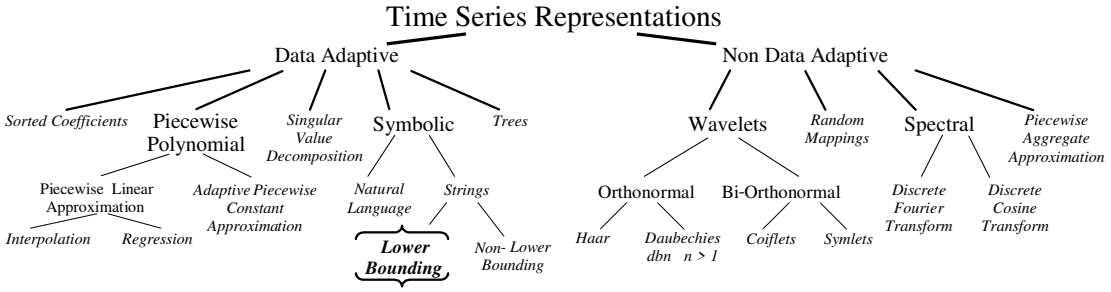

Figure 1: A hierarchy of all the various time series representations in the literature. The leaf nodes refer to the actual representation, and the internal nodes refer to the classification of the approach. The contribution of this paper is to introduce a new representation, the lower bounding symbolic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

None of the symbolic representations that we are aware of allow dimensionality reduction [2, 11, 20]. 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.

In [4], Babcock et. al ask if "*there is a need for database researchers to develop fundamental and general-purpose models… for data streams.*" The opinion of the authors is affirmative. In this work we take a step towards this goal by introducing a representation of time series that is suitable for streaming algorithms. It is dimensionality reducing, lower bounding and can be obtained in a streaming fashion.

As we shall demonstrate, the lower bounding 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 [21], classification [16], indexing [1, 14, 22, 35], and anomaly detection [9, 24, 31].

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. 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 [30, 23] for a more in depth review.

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* [1, 7, 14,22, 35].
- **Clustering:** Find natural groupings of the time series in database *DB* under some similarity/dissimilarity measure $D(Q,C)$ [21,25].
- **Classification:** Given an unlabeled time series Q, assign it to one of two or more predefined classes [16].
- **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 [26].
- **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 [9, 24, 31].

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 [14]. The basic approach is outlined in Table 1.

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.

Table 1: A generic time series data mining approach

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) [14], the Discrete Wavelet Transform (DWT) [7], Piecewise Linear, and Piecewise Constant models (PAA) [22], (APCA) [16, 22], and Singular Value Decomposition (SVD) [22]. Figure 2 illustrates the most commonly used representations.

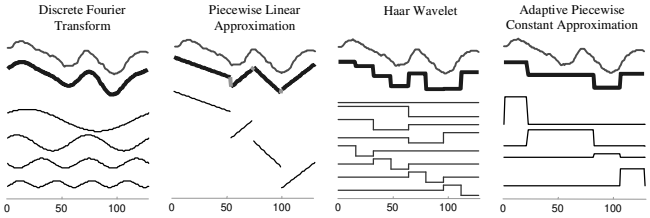

Figure 2: The most common representations for time series data mining. Each can be visualized as an attempt to approximate the signal with a linear combination of basis functions

Recent work suggests that there is little to choose between the above in terms of indexing power [23]; 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 [7].

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 [27]. Such limitations have lead researchers to consider using a symbolic representation of time series.

While there are literally hundreds of papers on discretizing (symbolizing, tokenizing, quantizing) time series [2, 20] (see [11] 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.

C	A time series $C = c_1,\ldots,c_n$
$\overline{C}$	A Piecewise Aggregate Approximation of a time series $\overline{C} = \overline{c}_1,\ldots,\overline{c}_w$
$\hat{C}$	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**}, $a = 3$)

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

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 [22, 35], 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 allows 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 [35].

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:

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

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.

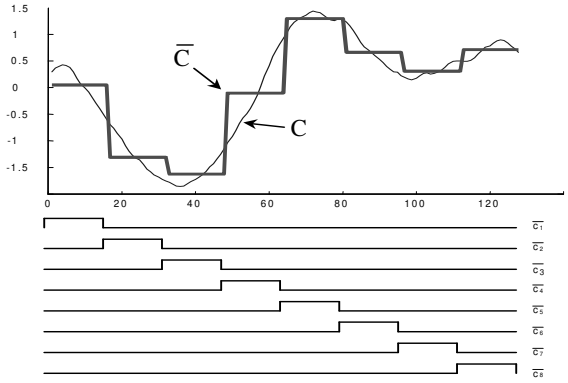

Figure 3: The PAA representation can be visualized as an attempt to model a time series with a linear combination of box basis functions. In this case, a sequence of length 128 is reduced to 8 dimensions

The PAA dimensionality reduction is intuitive and simple, yet has been shown to rival more sophisticated dimensionality reduction techniques like Fourier transforms and wavelets [22, 23, 35].

We normalize each time series to have a 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 [23].

3.2 Discretization

Having transformed a time series database into 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 [3, 28]. This is easily achieved since normalized time series have a Gaussian distribution [27]. To illustrate this, we extracted subsequences of length 128 from 8 different time series and plotted a normal probability plot of the data as shown in Figure 4.

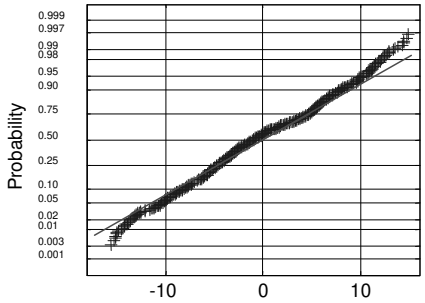

Figure 4: A normal probability plot of the cumulative distribution of values from subsequences of length 128 from 8 different datasets. The highly linear nature of the plot strongly suggests that the data came from a Gaussian distribution

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 [27].

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.

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

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

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.

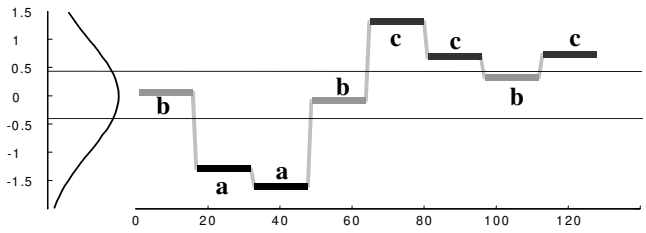

Figure 5: A time series is discretized by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into SAX symbols. In the example above, with n = 128, w = 8 and a = 3, the time series is mapped to the word **baabccbc**

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 i^{th} 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:

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

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

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 [23, 29]. Given two time series Q and C of the same length n, Eq. 3 defines their Euclidean distance, and Figure 6.A illustrates a visual intuition of the measure.

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

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:

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

This measure is illustrated in Figure 6.B. A proof that $DR(\overline{Q},\overline{C})$ lower bounds the true Euclidean distance appears in [22] (an alterative proof appears in [35]).

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:

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

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.

	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

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, $dist(\mathbf{a},\mathbf{b}) = 0$ and $dist(\mathbf{a},\mathbf{c}) = 0.67$.

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 in Figure 6.C.

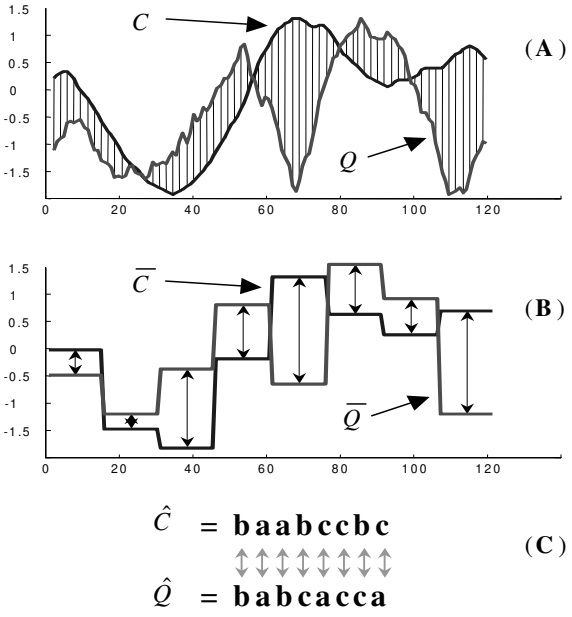

Figure 6: A visual intuition of the three representations discussed in this work, and the distance measures defined on them. **A**) The Euclidean distance between two time series can be visualized as the square root of the sum of the squared differences of each pair of corresponding points. **B**) The 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

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 infeasible 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.

$$Tightness\ of\ Lower\ Bound = \frac{MINDIST(\hat{Q},\hat{C})}{D(Q,C)} \quad (7)$$

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.

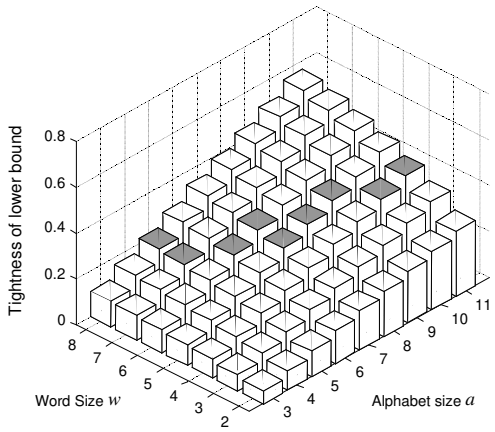

Figure 7: The empirically estimated tightness of lower bounds over the cross product of $a = [3\ldots11]$ and $w = [2\ldots8]$. The darker histogram bars illustrate combinations of parameters that require approximately equal space to store every possible word (approximately 4 megabytes)

The results suggest that using a low value for a results in weak bounds, but that there are diminishing returns for large values of a. The results also suggest that the parameters are not too critical; an alphabet size in the range of 5 to 8 seems to be a good choice.

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 [7, 14, 22, 35]. Figure 8 illustrates the idea.

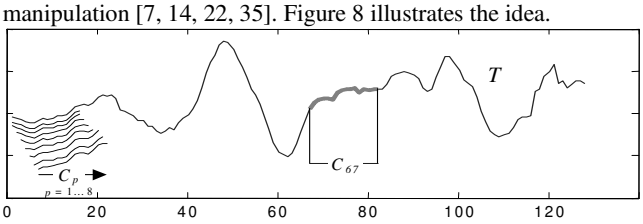

Figure 8: An illustration of the notation introduced in this section: A time series T of length 128, the *subsequence* C_{67} of length $n = 16$, and the first 8 subsequences extracted by a *sliding window*. Note that the sliding windows are overlapping

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 9.

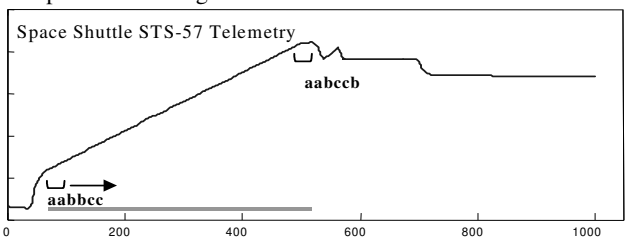

Figure 9: Sliding window extraction on Space Shuttle Telemetry data, with $n = 32$. At time point 61, the extracted word is **aabbcc**, and the next 401 subsequences also map to this word. Only a pointer to the first occurrence must be recorded, thus producing a large reduction in numerosity

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 5.48. This situation occurs quite frequently. For example, the last 200 time units of the data in Figure 9 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 $a = 5$).

We end this section with a visual comparison between SAX and the four most used representations in the literature (Figure 10).

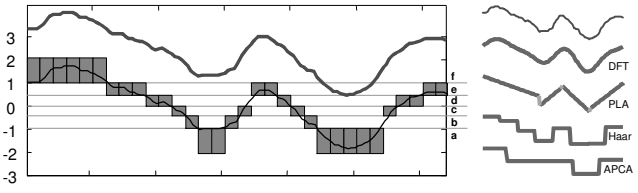

Figure 10: A visual comparison of SAX and the four most common time series data mining representations. A raw time series of length 128 is transformed into the word **ffffffeeeddcbaabceedcbaaaaacddee**. This is a fair comparison since the number of bits in each representation is the same

4. EXPERIMENTAL VALIDATION OF SAX

We performed various data mining tasks using SAX. For clustering and classification, we compared 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 paragraph we summarize the strawmen representations that we compare SAX to. We choose these two approaches since they are typical representatives of symbolic approaches in the literature.

André-Jönsson, and Badal [2] proposed the SDA algorithm that computes the changes between values from one instance to the next, and divide the range into user-predefined regions. 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 [20]. 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 [12,15, 21].

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 [23]. 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 11 we clustered nine time series from the Control Chart dataset, three each from the *decreasing trend*, *upward shift* and *normal* classes.

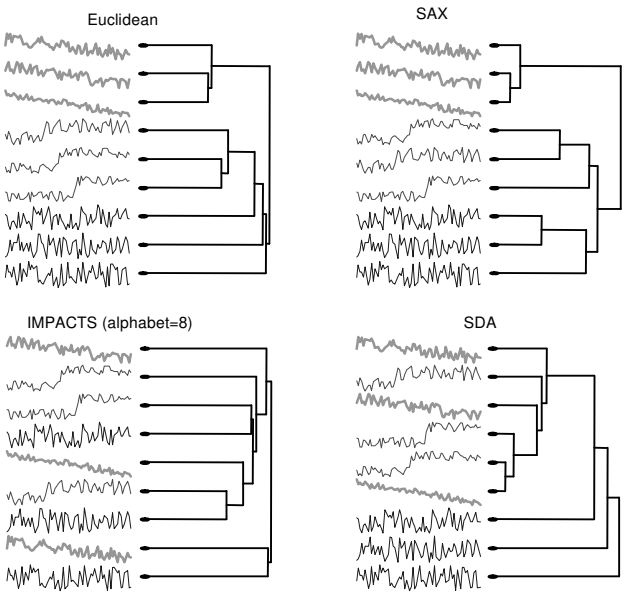

Figure 11: A comparison of the four distance measures' ability to cluster members of the Control Chart dataset. Complete linkage was used as the agglomeration technique

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. More generally, we observed that SAX closely mimics Euclidean distance on various datasets.

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 [15], so for completeness we will consider it here. We performed k-means on both the original raw data, and our symbolic representation. Figure 12 shows a typical run of *k*-means on a space telemetry dataset. Both algorithms converge after 11 iterations on average.

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 [12], this is what clustering with SAX implicitly does.

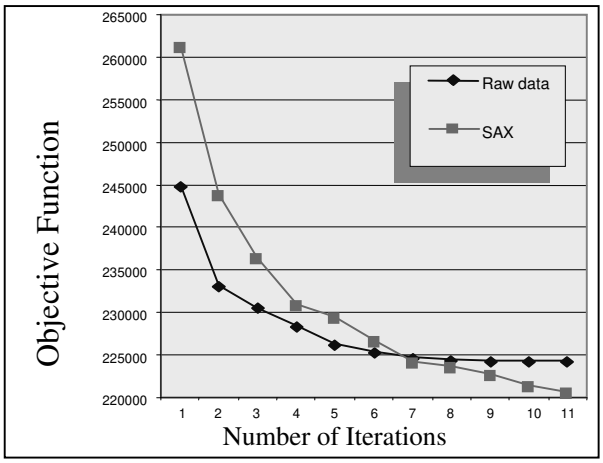

Figure 12: A comparison of the *k*-means clustering algorithm using SAX and the raw data. The dataset was Space Shuttle telemetry, 1,000 subsequences of length 512. Surprisingly, working with the symbolic approximation produces better results than working with the original data

4.2 Classification

Classification of time series has attracted much interest from the data mining community. The high dimensionality, high feature correlation, and typically high levels of noise found in time series provide an interesting research problem [23]. Although special-purpose algorithms have been proposed [25], 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 used leaving-one-out cross validation. We compare SAX with Euclidean distance, IMPACTS, SDA, and LP_∞. 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 [23].

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 13, we show the results 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 other approaches proposed in the literature.

4.2.2 Decision Tree Classification

Due to its poor scalability, Nearest Neighbor 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.

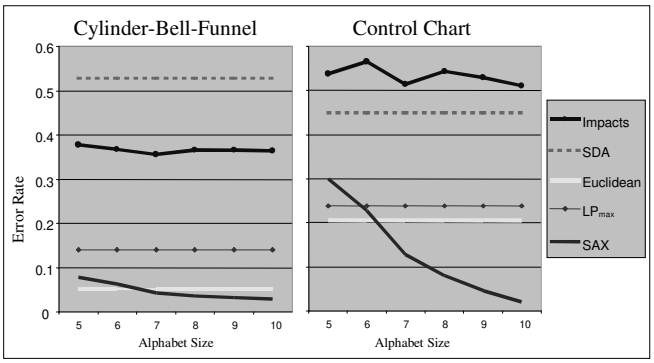

Figure 13: A comparison of five distance measures utility for nearest neighbor classification. We tested different alphabet sizes for SAX and IMPACTS. SDA's alphabet size is fixed at 5.

In an attempt to overcome this problem, Geurts [16] suggests representing the time series as a Regression Tree (RT) (this representation is essentially the same as APCA [22], 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 5.

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

Table 5: 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

Note that while our results are competitive with the RT approach, The RT representation is undoubtedly superior in terms of interpretability [16]. 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 [30]. Indeed, it is in this context that most of the representations enumerated in Figure 1 were introduced [7, 14, 22, 35]. Dozens of papers have introduced techniques to do indexing with a symbolic approach [2, 20], 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 [14].

To perform query by content, we built an index using SAX, and compared it to an index built using the Haar wavelet approach [7]. 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 [19]), 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 14 we show 4 representative examples.

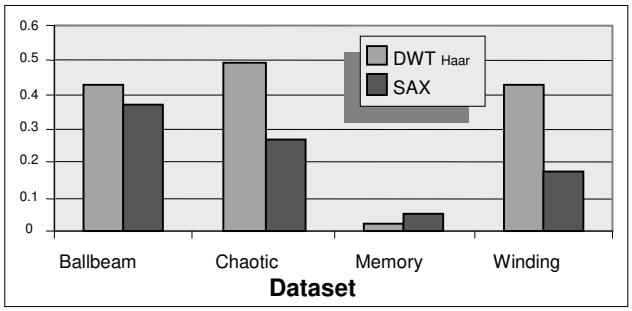

Figure 14: A comparison of indexing ability of wavelets versus SAX. The Y-axis is the percentage of the data that must be retrieved from disk to answer a 1-NN query of length 256, when the dimensionality reduction ratio is 32 to 1 for both approaches

Once again we find our representation competitive with existing 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 [9]. In order to achieve this, in [24] we combined a statistically sound scheme with an efficient combinatorial approach. The statistical 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 [20]; 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. [31] and the Immunology (IMM) inspired work of Dasgupta and Forrest [9]. 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 15 contains an experiment comparing all 5 techniques.

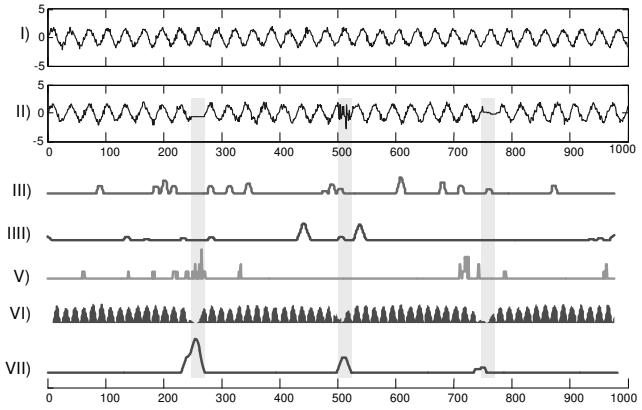

Figure 15: A comparison of five anomaly detection algorithms on the same task. **I)** The training data, a slightly noisy sine wave of length 1,000. **II)** The time series to be examined for anomalies is a noisy sine wave that was created with the same parameters as the training sequence, then an assortment of anomalies were introduced at time periods 250, 500 and 750. **III)** and **IIII)** The Markov Model technique using the IMPACTS and SDA representation 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.

4.4.2 Motif discovery

It is well understood in bioinformatics that overrepresented DNA sequences often have biological significance [3, 13, 29]. A substantial body of literature has been devoted to techniques to discover such patterns [17, 32, 33]. In a previous work, we defined the related concept of "time series motif" [26]. 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 [26], 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 [33]. 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[1]. 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. Figure 16 shows an example of a motif discovered in an industrial dataset [5] using this technique.

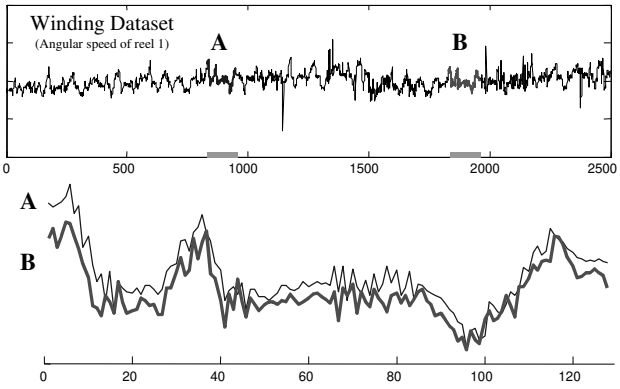

Figure 16: Above, a motif discovered in a complex dataset by the modified PROJECTION algorithm. Below, the motif is best visualized by aligning the two subsequences and "zooming in". The similarity of the two subsequences is striking, and hints at unexpected regularity

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.

5. CONCLUSIONS AND FUTURE DIRECTIONS

In this work we introduced the first dimensionality reduction, lower bounding, streaming 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 and motif discovery.

A host of future directions suggest themselves. In addition to use with streaming algorithms, there is an enormous wealth of useful definitions, algorithms and data structures in the bioinformatics literature that can be exploited by our representation [3, 13, 17, 28, 29, 32, 33]. It may be possible to create a lower bounding approximation of Dynamic Time Warping [6], by slightly modifying the classic string edit distance. Finally, there may be utility in extending our work to multidimensional time series [34].

6. REFERENCES

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

[2] André-Jönsson, H. & Badal. D. (1997). 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. pp 211-220.

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

[4] Babcock, B, Babu, S., Datar, M., Motwani, R. & Widom, J. (2002). Models and Issues in Data Stream Systems. *Invited Paper in proceedings of the 2002 ACM Symp. On Principles of Database Systems*. June 3-5, Madison, WI.

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

[5] Bastogne, T., Noura, H., Richard A. & Hittinger, J. .M. (2002). Application of Subspace Methods to the Identification of a Winding Process. In *proceedings of the 4th European Control Conference*, Vol. 5, Brussels.

[6] Berndt, D. & Clifford, J. (1994) Using Dynamic Time Warping to Find Patterns in Time Series. In *proceedings of the Workshop on Knowledge Discovery in Databases*, at the 12th Int'l Conference on Artificial Intelligence. July 31-Aug 4, Seattle, WA. pp 229-248.

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

[8] Cortes, C., Fisher, K., Pregibon, D., Rogers, A. & Smith, F. (2000). Hancock: a Language for Extracting Signatures from Data Streams. In *proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. Aug 20-23, Boston, MA. pp 9-17.

[9] Dasgupta, D. & Forrest, S. (1996) Novelty Detection in Time Series Data using Ideas from Immunology. In *proceedings of The International Conference on Intelligent Systems*. June 19-21.

[10] Datar, M. & Muthukrishnan, S. (2002). Estimating Rarity and Similarity over Data Stream Windows. In *proceedings of the 10th European Symposium on Algorithms*. Sep 17-21, Rome, Italy.

[11] Daw, C. S., Finney, C. E. A. & Tracy, E. R. (2001). Symbolic Analysis of Experimental Data. *Review of Scientific Instruments*. (2002-07-22).

[12] Ding, C., He, X., Zha, & Simon., H. (2002). Adaptive Dimension Reduction for Clustering High Dimensional Data. In *proceedings of the 2nd IEEE International Conference on Data Mining*. Dec 9-12. Maebashi, Japan. pp 147-154.

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

[14] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast Subsequence Matching in Time-Series Databases. *In proceedings of the ACM SIGMOD Int'l Conference on Management of Data.* May 24-27, Minneapolis, MN. pp 419-429.

[15] Fayyad, U., Reina, C. &. Bradley, P. (1998). Initialization of Iterative Refinement Clustering Algorithms. In *proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 194-198.

[16] Geurts, P. (2001). Pattern Extraction for Time Series Classification. In *proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. Sep 3-7, Freiburg, Germany. pp. 115-127.

[17] Gionis, A. & Mannila, H. (2003). Finding Recurrent Sources in Sequences. In *proceedings of the 7th International Conference on Research in Computational Molecular Biology*. Apr 10-13, Berlin, Germany. To Appear.

[18] Guha, S., Mishra, N., Motwani, R. & O'Callaghan, L. (2000). Clustering Data Streams. In *proceedings of the 41st Symposium on Foundations of Computer Science*. Nov 12-14, Redondo Beach, CA. pp 359-366.

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

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

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

[22] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In *proceedings of ACM SIGMOD Conference on Management of Data*. Santa Barbara, CA, May 21-24. pp 151-162.

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

[24] Keogh, E., Lonardi, S. & Chiu, W. (2002). Finding Surprising Patterns in a Time Series Database in Linear Time and Space. In the *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 550-556.

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

[26] Lin, J., Keogh, E., Lonardi, S. & Patel, P. (2002). Finding Motifs in Time Series. In proceedings of the *2nd Workshop on Temporal Data Mining*, at the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, July 23-26. pp. 53-68.

[27] Larsen, R. J. & Marx, M. L. (1986). An Introduction to Mathematical Statistics and Its Applications. Prentice Hall, Englewood, Cliffs, N.J. 2nd Edition.

[28] Lonardi, S. (2001). Global Detectors of Unusual Words: Design, Implementation, and Applications to Pattern Discovery in Biosequences. PhD thesis, Department of Computer Sciences, Purdue University, August, 2001.

[29] Reinert, G., Schbath, S. & Waterman, M. S. (2000). Probabilistic and Statistical Properties of Words: An Overview. *Journal of Computational. Bio*logy. Vol. 7, pp 1-46.

[30] Roddick, J. F., Hornsby, K. & Spiliopoulou, M. (2001). An Updated Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research. In *Post-Workshop Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining*. Berlin, Springer. Lecture Notes in Artificial Intelligence. Roddick, J. F. and Hornsby, K., Eds. 147-163.

[31] Shahabi, C., Tian, X. & Zhao, W. (2000). 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*. pp 55-68.

[32] Staden, R. (1989). Methods for Discovering Novel Motifs in Nucleic Acid Sequences. *Computer Applications in Biosciences.* Vol. 5(5). pp 293-298.

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

[34] Vlachos, M., Kollios, G. & Gunopulos, G. (2002). Discovering Similar Multidimensional Trajectories. In *proceedings of the 18th International Conference on Data Engineering*. Feb 26-Mar 1, San Jose, CA.

[35] Yi, B, K., & Faloutsos, C. (2000). Fast Time Sequence Indexing for Arbitrary L*p* Norms. In *proceedings of the 26st Int'l Conference on Very Large Databases*. Sep 10-14, Cairo, Egypt. pp 385-394.