

# Assumption-Free Anomaly Detection in Time Series

Li Wei Nitin Kumar Venkata Lolla Eamonn Keogh Stefano Lonardi Chotirat Ann Ratanamahatana  
University of California - Riverside  
Department of Computer Science & Engineering  
Riverside, CA 92521, USA  
{wli, nkumar, vlolla, eamonn, stelo, ratana}@cs.ucr.edu

## Abstract

Recent advancements in sensor technology have made it possible to collect enormous amounts of data in real time. However, because of the sheer volume of data most of it will never be inspected by an algorithm, much less a human being. One way to mitigate this problem is to perform some type of anomaly (novelty / interestingness/ surprisingness) detection and flag unusual patterns for further inspection by humans or more CPU intensive algorithms. Most current solutions are “custom made” for particular domains, such as ECG monitoring, valve pressure monitoring, etc. This customization requires extensive effort by domain expert. Furthermore, hand-crafted systems tend to be very brittle to concept drift. In this demonstration, we will show an online anomaly detection system that does not need to be customized for individual domains, yet performs with exceptionally high precision/recall. The system is based on the recently introduced idea of time series bitmaps. To demonstrate the universality of our system, we will allow testing on independently annotated datasets from domains as diverse as ECGs, Space Shuttle telemetry monitoring, video surveillance, and respiratory data. In addition, we invite attendees to test our system with any dataset available on the web.

## 1. Introduction

Recent advancements in sensor technology have made it possible to collect enormous amounts of data in real time. However, because of the sheer volume of data most of it is never inspected by an algorithm, much less a human being. One way to mitigate this problem is to perform some type of anomaly (novelty / interestingness/ surprisingness) detection and to flag unusual patterns for future inspection by humans or more CPU intensive algorithms. Most current solutions are “custom made” for particular domains, such as ECG monitoring, valve pressure monitoring, etc. This customization requires extensive effort by domain experts. Furthermore hand-crafted systems tend to be very brittle to concept drift.

In this demonstration, we will show an online anomaly detection system that does not need to be customized for individual domains, yet performs with exceptionally high precision/recall. The system is based on the recently introduced idea of time series bitmaps [11]. It allows

users to efficiently navigate through a time series of arbitrary length and identify portions that require further investigation. Figure 1 illustrates the graphical interface of our system<sup>1</sup>.

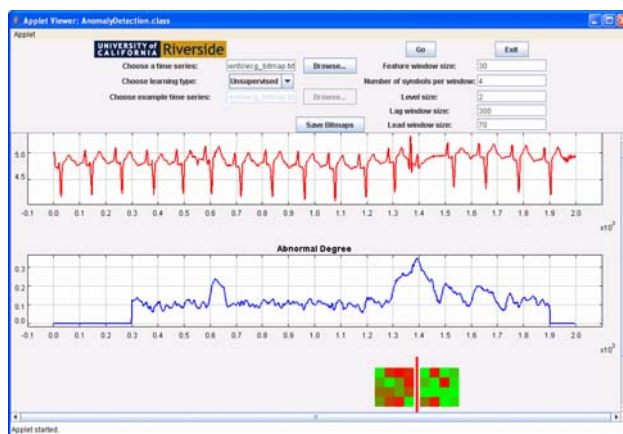


Figure 1. A snapshot of the anomaly detection tool.

To demonstrate the universality of our system, we will allow testing on independently annotated datasets from domains as diverse as ECGs, Space Shuttle telemetry monitoring, video surveillance, and respiratory data. In addition, we invite attendees to test our system with any dataset available on the web.

## 2. Background and Related Work

In this section, we give brief reviews of chaos games and symbolic representations of time series, which are at the heart of our anomaly detection technique.

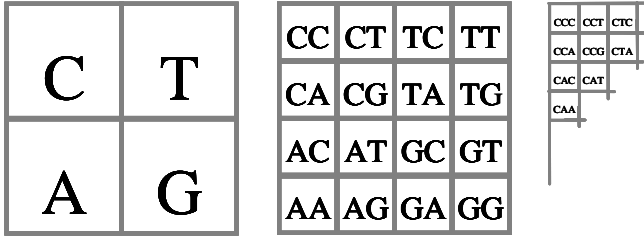
### 2.1 Chaos Game Representations

Our visualization technique is partly inspired by an algorithm to draw fractals called the *Chaos game* [1]. The method can produce a representation of DNA sequences, in which both local and global patterns are displayed.

The basic idea is to map frequency counts of DNA substrings of length  $L$  into a  $2L$  by  $2L$  matrix as shown in Figure 2, then color-code these frequency counts. From our point of view, the crucial observation is that the CGR

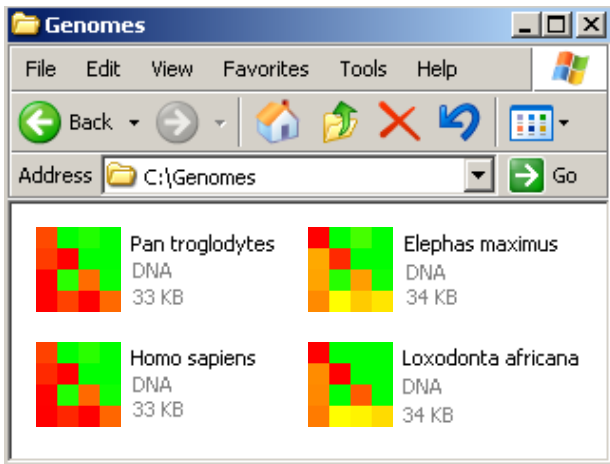
<sup>1</sup> We encourage the interested reader to visit [5] to view full color examples of all figures in this work.

representation of a sequence allows the investigation of the patterns in sequences, giving the human eye a possibility to recognize hidden structures.



**Figure 2. The quad-tree representation of a sequence over the alphabet {A,C,G,T} at different levels of resolution.**

We can get a hint of the potential utility of the approach if, for example, we take the first 5,000 symbols of the mitochondrial DNA sequences of four familiar species and use them to create their own file icons. Figure 3 below illustrates this. Note that *Pan troglodytes* is the familiar Chimpanzee, and *Loxodonta africana* and *Elephas maximus* are the African and Indian Elephants, respectively. Even if we did not know these particular animals, we would have no problem recognizing that there are two pairs of highly related species being considered.



**Figure 3. The bitmap representation of the gene sequences of four animals.**

With respect to the non-genetic sequences, Joel Jeffrey noted, “The CGR algorithm produces a CGR for any sequence of letters” [4]. However, it is only defined for discrete sequences, and most time series are real valued.

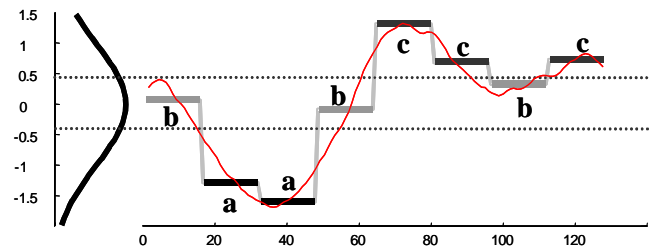
The results in Figure 3 encouraged us to try a similar technique on real valued time series data and investigate the utility of such a representation on the data mining task of anomaly detection. Since CGR involves treating a data input as an abstract string of symbols, a discretization method is necessary to transform continuous time series

data into discrete domain. For this purpose, we used the Symbolic Aggregate approxXimation (SAX) [8], which we review below.

## 2.2 Symbolic Time Series Representations

While there are at least 200 techniques in the literature for converting real valued time series into discrete symbols, the SAX technique of Lin *et. al.* [8] is unique and ideally suited for data mining. SAX is the only symbolic representation that allows the lower bounding of the distances in the original space.

The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. The mean value of each section is then calculated. By substituting each section with its mean, a reduced dimensionality piecewise constant approximation of the data is obtained. This representation is then discretized in such a manner as to produce a word with approximately equi-probable symbols. Figure 4 shows a short time series being converted into the SAX word **baabccbc**.



**Figure 4. A real valued time series can be converted to the SAX word baabccbc.**

It has been pointed out that when processing very long time series, it is not necessarily a good idea to convert the entire time series into a *single* SAX word [11]. Therefore, for long time series, we slide a shorter window, which is called feature window, across it and obtain a set of shorter SAX words.

Note that the user must choose both the length of the sliding feature window  $N$ , and the number  $n$  of equal sized sections in which to divide  $N$  (as we will see, there is no choice to be made for alphabet size). A good choice for  $N$  should reflect the natural scale at which the events occur in the time series. For example, for ECGs, this is about the length of one or two heartbeats. A good value for  $n$  depends on the complexity of the signal. Intuitively, one would like to achieve a good compromise between fidelity of approximation and dimensionality reduction. As we shall see, the proposed technique is not too sensitive to parameter choices.

## 3. Time Series Anomaly Detection

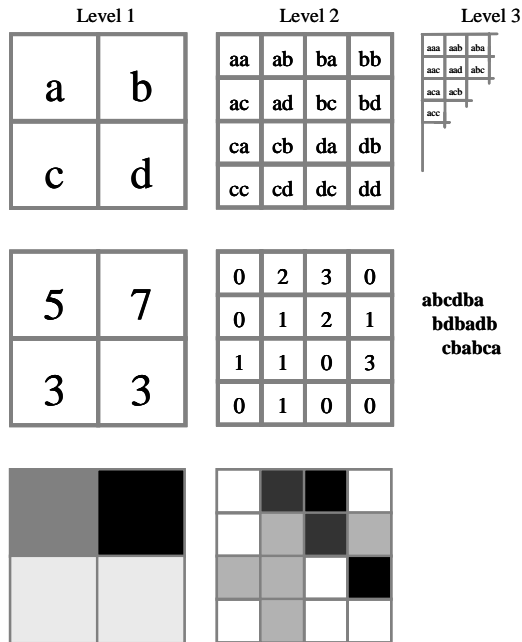
### 3.1 Time Series Bitmaps

At this point, we have seen that the Chaos game bitmaps can be used to visualize discrete sequences and that the SAX representation is a discrete time series representation that has demonstrated great utility for data mining. It is natural to consider combining these ideas.

The *Chaos game* bitmaps are defined for sequences with an alphabet size of four. SAX can produce strings on any alphabet sizes. As it turns out, many authors have reported a cardinality of four as an excellent choice for diverse datasets on assorted problems [2][3][6][7][8][9].

We need to define an initial ordering for the four SAX symbols **a**, **b**, **c**, and **d**. We use simple alphabetical ordering as shown in Figure 5.

After converting the original raw time series into the SAX representation, we can count the frequencies of SAX “subwords” of length  $L$ , where  $L$  is the desired level of recursion. Level 1 frequencies are simply the raw counts of the four symbols. For level 2, we count pairs of subwords of size 2 (**aa**, **ab**, **ac**, etc.). Note that we only count subwords taken from individual SAX words. For example, in the SAX representation in Figure 5 *middle right*, the last symbol of the first line is **a**, and the first symbol of the second word is **b**. However, we do not count this as an occurrence of **ab**.



**Figure 5. The generation of time series bitmaps.**

Once the raw counts of all subwords of the desired length have been obtained and recorded in the corresponding pixel of the grid, we normalize the frequencies by dividing it by the largest value. The pixel

values  $P$  thus range from 0 to 1. The final step is to map these values to colors. In the example above, we mapped to grayscale, with 0 = *white*, 1 = *black*. However, it is generally recognized that grayscale is not *perceptually uniform* [10]. A color space is said to be perceptually uniform if small changes to a pixel value are approximately equally perceptible across the range of that value. For all images in this paper, we encode the pixels values to be  $[P, 1-P, 0]$  in the RGB color space.

For bitmaps with same size, we define the distance between them as the summation of the square of the distance between each pair of pixels. More formally, for two  $n \times n$  bitmaps  $BA$  and  $BB$ , the distance between them

$$\text{is defined as } \text{dist}(BA, BB) = \sum_{i=1}^n \sum_{j=1}^n (BA_{ij} - BB_{ij})^2 .$$

### 3.2 Anomaly Detection

We create two concatenated windows and slide them together across the sequence. The latter one is called *lead* window, showing how far to look ahead for anomalous patterns. A reasonable value would be two or three times the length of the feature window. The former one is called *lag* window, whose size represents how much memory of the past to remember. Usually, it should be at least as long as the lead window. We convert each window into the SAX representation, count the frequencies of SAX “subwords” at the desired level, and get the corresponding bitmaps. The distance between the two bitmaps is measured and reported as an anomaly score at each time instance, and the bitmaps are drawn to visualize the similarities and differences between the two windows.

There are two ways to use the tool, unsupervised (one time series) and supervised (two time series). For unsupervised use, the user must specify the size of the lag window. For supervised use, the user must specify a time series file that he/she believes contains normal behavior for the system. For example, this could be 10 minutes of ECGs that are known to be normal, or a trace from a successful space mission. In this case, the entire training time series can be imagined as the lag window.

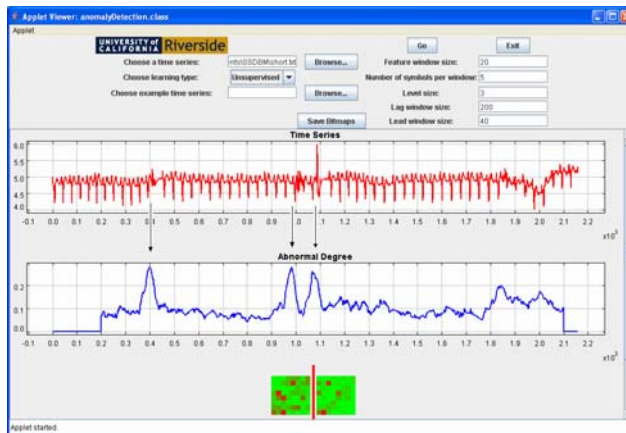
At each “step” of the sliding window we can incrementally ingress a new data point, and egress an old data point in constant time (updating only two pixels of each bitmap). Hence, the time complexity is linear in the length of the time series.

## 4. Experimental Evaluation

To demonstrate the universality of our system, we tested on independently annotated datasets from domains as diverse as ECGs, Space Shuttle telemetry monitoring, video surveillance, and respiratory data. Here we show only a subset of the experimental results due to space limitations. Our approach is also effective on time series

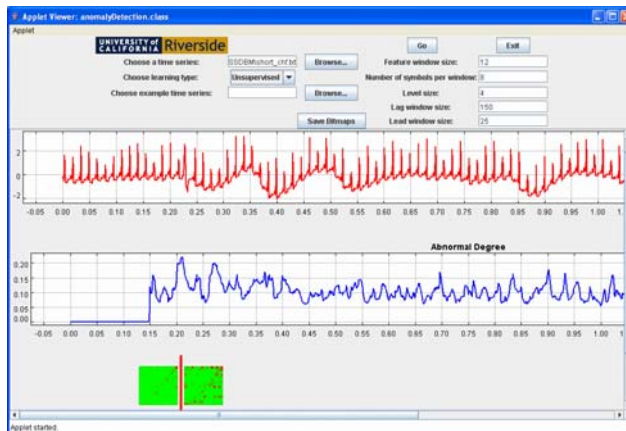
clustering and classification [11], but we focus on its utility for anomaly detection here. We urge the interested reader to consult [5] for large-scale color reproductions and additional details.

Figure 6 illustrates a subsection of an ECG data. A cardiologist annotated two premature ventricular contractions at approximately the 0.4 and 1.1 mark, respectively, and a supraventricular escape beat at about the 1.0 mark. Our approach easily detects all the three anomalies.



**Figure 6. Top)** A subsection of an ECG dataset. **Middle)** The abnormal score shows three strong peaks for the anomalous heartbeats. **Bottom)** The bitmaps before and after the third peak.

Figure 7 shows a very complex and noisy ECG. But according to a cardiologist, there is only one abnormal heartbeat at approximately the 0.23 mark. Our tool easily finds it.



**Figure 7. Top)** A subsection of an ECG dataset. **Middle)** The abnormal score shows a strong peak for the anomalous heartbeat. **Bottom)** The bitmaps before and after the strong peak.

## 5. Demonstration Plan

Our demonstration will consist of the following three parts.

- First, we will present some real-world applications in which our technique can be applied. These examples will provide the audience with insights into the task of time series anomaly detection.
- Second, by using real-world datasets from diverse domains, we will show the experimental evaluation of our system.
- Finally, we will invite audience to play the tool interactively themselves. The audience will be encouraged to test their own datasets.

Reproducible Results Statement: In the interests of competitive scientific inquiry, all datasets used in this work are available at the following URL [5]. This research was partly funded by the National Science Foundation under grant IIS-0237918.

## References

- [1] Barnsley, M.F., & Rising, H. (1993). *Fractals Everywhere*, second edition, Academic Press.
- [2] Celly, B. & Zordan, V. B. (2004). Animated People Textures. In proceedings of the 17th International Conference on Computer Animation and Social Agents. Geneva, Switzerland.
- [3] Chiu, B., Keogh, E., & Lonardi, S. (2003). Probabilistic Discovery of Time Series Motifs. In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- [4] Jeffrey, H.J. (1992). Chaos Game Visualization of Sequences. *Comput. & Graphics* 16, pp. 25-33.
- [5] Keogh, E. <http://www.cs.ucr.edu/~wli/SSDBM05/>
- [6] Keogh, E., Lonardi, S., & Ratanamahatana, C. (2004). Towards Parameter-Free Data Mining. In proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- [7] Lin, J., Keogh, E., Lonardi, S., Lankford, J.P. & Nystrom, D.M. (2004). Visually Mining and Monitoring Massive Time Series. In proceedings of the 10th ACM SIGKDD.
- [8] Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003) A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.
- [9] Tanaka, Y. & Uehara, K. (2004). Motif Discovery Algorithm from Motion Data. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAD). Kanazawa, Japan.
- [10] Wysocki, G. (1982). *Color science: Concepts and methods, quantitative data and formulae*, 2nd edition. New York, Wiley, 1982.
- [11] Kumar, N., Lolla N., Keogh, E., Lonardi, S., Ratanamahatana, C. & Wei, L. (2005). Time-series Bitmaps: A Practical Visualization Tool for Working with Large Time Series Databases. SIAM 2005 Data Mining Conference.