Track Title:	Special Track on Data Mining
Paper Title:	A Practical Tool for Visualizing and Data Mining Medical Time Series

Authors:

Li Wei (wli@cs.ucr.edu)
Nitin Kumar (nkumar@cs.ucr.edu)
Venkata Nishanth Lolla (vlolla@cs.ucr.edu)
Eamonn Keogh (eamonn@cs.ucr.edu)
Stefano Lonardi (stelo@cs.ucr.edu)
Chotirat Ann Ratanamahatana (ratana@cs.ucr.edu)

University of California - Riverside
Department of Computer Science & Engineering
Riverside, CA 92521, USA

&

Helga Van Herle (hvanherle@mednet.ucla.edu)

University of California, Los Angeles
David Geffen School of Medicine

Technical Areas: Data Mining, Time Series, Classification, Clustering, Anomaly Detection

Contact Info:

Dr. Eamonn Keogh
Department of Computer Science and Engineering
Surge Building
University of California Riverside
Riverside, CA 92521-0144
E-mail: eamonn@cs.ucr.edu
Phone: 1-951-827-2032
Fax: 1-951- 827-4643

Preference: Oral presentation

A Practical Tool for Visualizing and Data Mining Medical Time Series

Abstract

The increasing interest in time series data mining in the last decade has had surprisingly little impact on real world medical applications. Real world practitioners who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. In this work, we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our system has the unique advantage that it can be embedded directly into any standard graphical user interfaces, such as Microsoft Windows, thus making deployment easier. Our approach extracts features from a time series of arbitrary length and uses information about the relative frequency of these features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection. We demonstrate the utility of our approach with a set of comprehensive experiments on real datasets from a variety of medical domains, including ECGs and EEGs.

Keywords: Time Series, Chaos Game, Visualization.

1 Introduction

The increasing interest in time series data mining in the last decade has resulted the introduction of a variety of similarity measures/ representations/ definitions/ indexing techniques and algorithms (see, e.g., [1][2][3][4] [9][12][13][15][22]). Surprisingly, this massive research effort has had little impact on real world medical applications. Examples of implemented systems are rare exceptions [16]. Cardiologists and other medical practitioners who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. While it is difficult to firmly establish the reasons for the discrepancy between tool availability and practical adoption, the following reasons suggested themselves after an informal survey.

- Time series data mining tools often come with a bewildering number of parameters. It is not obvious to the practitioner how these should be set [14].
- Research tools often require (relatively) specialized hardware and/or software, rather than the ubiquitous desktop PC/Windows environment that prevails in most clinics.
- Many tools have a steep learning curve, requiring already busy doctors to spend many unproductive hours learning the system before any possibility of useful work.

In this work, we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our approach extracts features from a time series of arbitrary length, and uses information about the relative frequency of these features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of these bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection.

While our system can be used as an interactive tool, it also has the unique advantage that it can be embedded directly into any standard graphical user interfaces, such as Windows, Mac OS, etc. Since users navigate through files by looking at their icons, we decided to employ the bitmap representation as the icon corresponding to each time series. Simply by glancing at the icons contained in a folder of time series files, a user can quickly identify files that require further investigation. In Figure 1, we have illustrated a simple example[1].

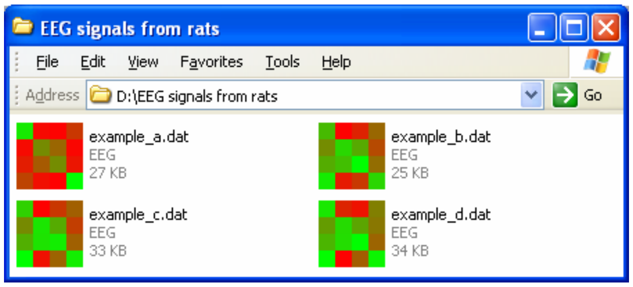


Figure 1. Four time series files represented as time series bitmaps. While they are all examples of EEGs, *example_a.dat* is from a normal trace, whereas the others contain examples of spike-wave discharges. The fact that there is some difference between one dataset and all the rest is immediately apparent from a casual inspection of the bitmap representation.

The utility of the idea shown in Figure 1 can be further enhanced by arranging the icons within the folder by *pattern similarity*, rather than the typical choices of arranging them by *size*, *name*, *date*, etc.. This can be achieved by using multidimensional scaling or a self-organizing map algorithm to arrange the icons.

[1] We encourage the interested reader to visit [10] to view full color examples of all figures in this work. We are grateful to Dr. Keogh for hosting the resource.

2 Background and Related Work

In this section, we give brief reviews of chaos games and symbolic representations of time series, which together are at the heart of our visualization/data mining technique.

2.1 Chaos Game Representations

Our visualization technique is partly inspired by an algorithm to draw fractals called the *Chaos game* [3]. 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 representation of a sequence allows the investigation of the patterns in sequences, giving the human eye a possibility to recognize hidden structures.

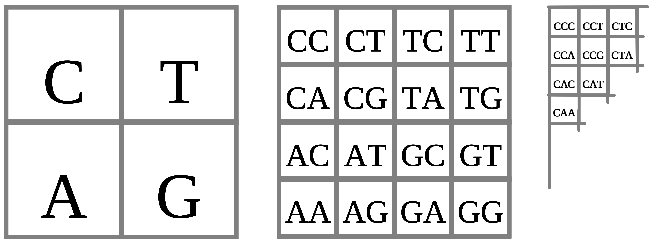


Figure 2. The quad-tree representation of a sequence over the alphabet {A,C,G,T} at different levels of resolution.

A biologist can recognize that a particular substring, say in a bacterial genome, is rarely used. This would suggest the possibility that the bacteria have evolved to avoid a particular restriction enzyme site, which means that it might not be easily attacked by a specific bacterio-phage.

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. 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.

With respect to the non-genetic sequences, Joel Jeffrey noted, *"The CGR algorithm produces a CGR for any sequence of letters"*[9]. However, it is only defined for discrete sequences, and most time series are real valued.

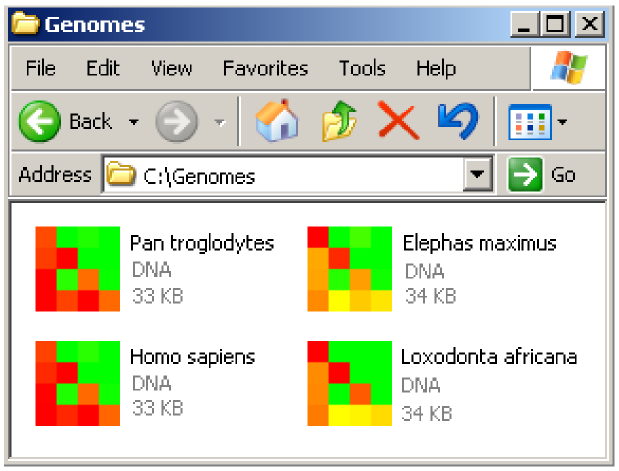


Figure 3. The gene sequences of mitochondrial DNA of four animals, used to create their own file icons using a chaos game representation. Note that *Pan troglodytes* is the familiar Chimpanzee, and *Loxodonta africana* and *Elephas maximus* are the African and Indian Elephants, respectively. The file icons show that humans and chimpanzees have similar genomes, as do the African and Indian elephants.

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 classic data mining tasks of clustering, classification, and visualization. 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 approXimation (SAX) [17], 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 [7], the SAX technique of Lin *et. al.* [17] 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 ability to efficiently lower bound distances is at the heart of hundreds of indexing algorithms and data mining techniques [2][6][11][13][17][20]. While we do not directly exploit the lower bounding property in this work, we note that if a representation is tightly lower bounding the original data, it must be representing it with great fidelity. It is this *implicit* property that we are exploiting. We can be sure that the SAX representation is accurately summarizing the time series, and as we will show, a minor modification of the chaos game can accurately summarize the SAX sequences.

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**.

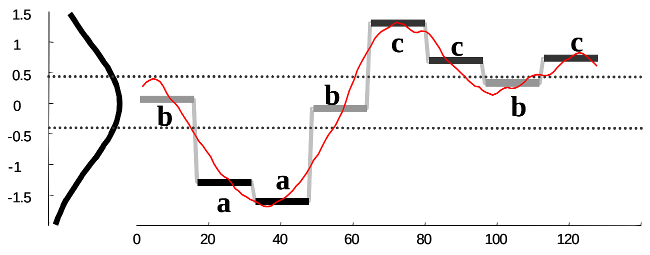


Figure 4. A real valued time series can be converted to the SAX word **baabccbc**. Note that all three possible symbols are approximately equally frequent.

Note that the user must choose both the length of the local sliding window *N*, and the number *n* of equal sized sections in which to divide it (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 of the complexity of the signal. Intuitively, one would like to achieve a good compromise between fidelity of approximation and dimensionality reduction. Two groups of researchers have independently suggested using Minimum Description Length MDL to set these parameters [14][19]. As we shall see, the proposed technique is not too sensitive to parameter choices.

The SAX representation has been successfully used by various groups of researchers for indexing, classification, clustering [17], motif discovery [5][6][19], rule discovery, visualization [16], and anomaly detection [14].

3 Time Series Bitmaps

At this point, the connection between the two “ingredients” for the time series bitmaps should be evident. We have seen in Section 2.1 that the *Chaos game* [3] bitmaps can be used to visualize discrete sequences, and we have seen in Section 2.2 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. It is fortuitous that DNA strings have this cardinality. SAX can produce strings on any alphabet sizes. As it turns out, a cardinality of four (or three) has been reported by many authors as an excellent choice for diverse datasets on assorted problems [5][6][14][16][17][19].

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**.

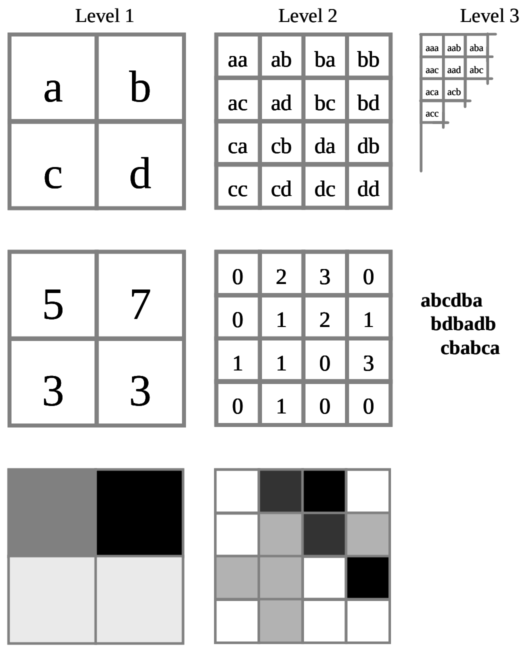


Figure 5. *Top*) The four possible SAX symbols are mapped to four quadrants of a square, and pairs, triplets, etc are recursively mapped to finer grids. *Middle*) We can extract counts of symbols from a SAX representation and record them in the grids. *Bottom*) The recorded values can be linearly mapped to colors, thus creating a square bitmap.

Once the raw counts of all subwords of the desired length have been obtained and recorded in the corresponding pixel of the grid, one more step is required. Since the time series in a data collection may be of various lengths, 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* [21]. 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.

Note that unlike the arbitrarily long, and arbitrarily shaped time series from which they where derived, for a fixed L, the bitmaps have a constant space and structure.

It is important to note that we do *not* suggest any utility in viewing a single time series bitmap. The representation is abstract, and we do not expect a user to be able to imagine the structure of time series given the bitmap. The utility of the bitmaps comes from the ability to efficiently compare and contrast them.

4 Time Series Thumbnails

A unique advantage of the time series bitmap representation is the fact that we can transparently integrate it into the user graphical interface of most standard operating systems.

Since most operating systems use the ubiquitous square icon to represent a file, we can arrange for the icons for time series files to appear as their bitmap representations. Simply by glancing at the contents of a folder of time series files, a user may spot files that require further investigation, or note natural clusters in the data.

The largest possible icon size varies by operating system. All modern versions of Microsoft Windows support 32 by 32 pixels, which is large enough to support a bitmap of level 5. As we will see, level 2 or 3 seems adequate for most tasks/datasets.

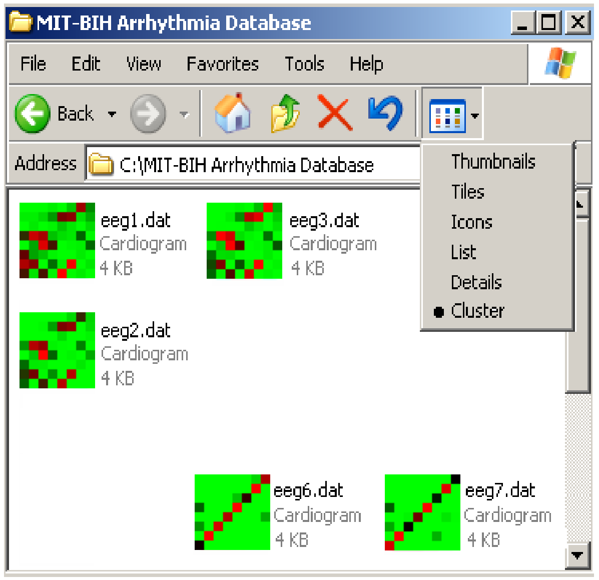
To augment the utility of the time series bitmaps, we can arrange for their placement on screen to reflect their structure. Normally, file icons are arranged by one of a handful of common criteria, such as *name*, *date*, *size*, etc.

We have created a simple modification of the standard Microsoft Windows (98 or later) file browser by introducing the concept of *Cluster View*. If *Cluster View* is chosen by the user, the time series thumbnails arrange themselves by similarity. This is achieved by performing Multi-Dimensional Scaling (MDS) of the bitmaps, and projecting them into a 2 dimensional space. For aesthetic reasons, we "snap" the icons to the closest grid point.

Figure 6 displays an example of *Cluster View* in Microsoft Windows XP Operating System.

In this example, the *Cluster View* is obtained for five MIT-BIH Arrhythmia Database files. It is evident in the figure that *eeg1.dat*, *eeg2.dat,* and *eeg3.dat* belong to one cluster whereas *eeg6.dat* and *eeg7.dat* belong to another. In this case, the grouping correctly reflects the fact that latter two files come from a different patient to first three.

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


MIT-BIH Arrhythmia Database

Address C:\MIT-BIH Arrhythmia Database

eeg1.dat Cardiogram 4 KB

eeg3.dat Cardiogram 4 KB

eeg2.dat Cardiogram 4 KB

eeg6.dat Cardiogram 4 KB

eeg7.dat Cardiogram 4 KB

Figure 6. A snapshot of a folder containing cardiograms when its files are arranged by "Cluster" option. Five cardiograms have been grouped into two different clusters based on their similarity.

5 Experimental Evaluation

In this section, we test our proposed approach with a comprehensive set of experiments. We will show some experiments that objectively measure the utility of our approach on clustering, classification, and anomaly detection. We note once again that the quality of illustrations here suffers from monochromic printing and small-scale reproduction. We urge the interested reader to consult [10] for large-scale color reproductions and additional details.

5.1 Clustering

We considered a four-class ECG clustering problem, where each class corresponds to a different patient. Figure 7 shows the clustering obtained with level 3 bitmaps, using parameters N = 150, n = 5. The results are correct, in that each time series from a given patient is assigned to its own sub-tree. For this problem, we found that we could vary the N and n parameters by a factor of 4 ($N > n$) and still obtain the correct clustering.

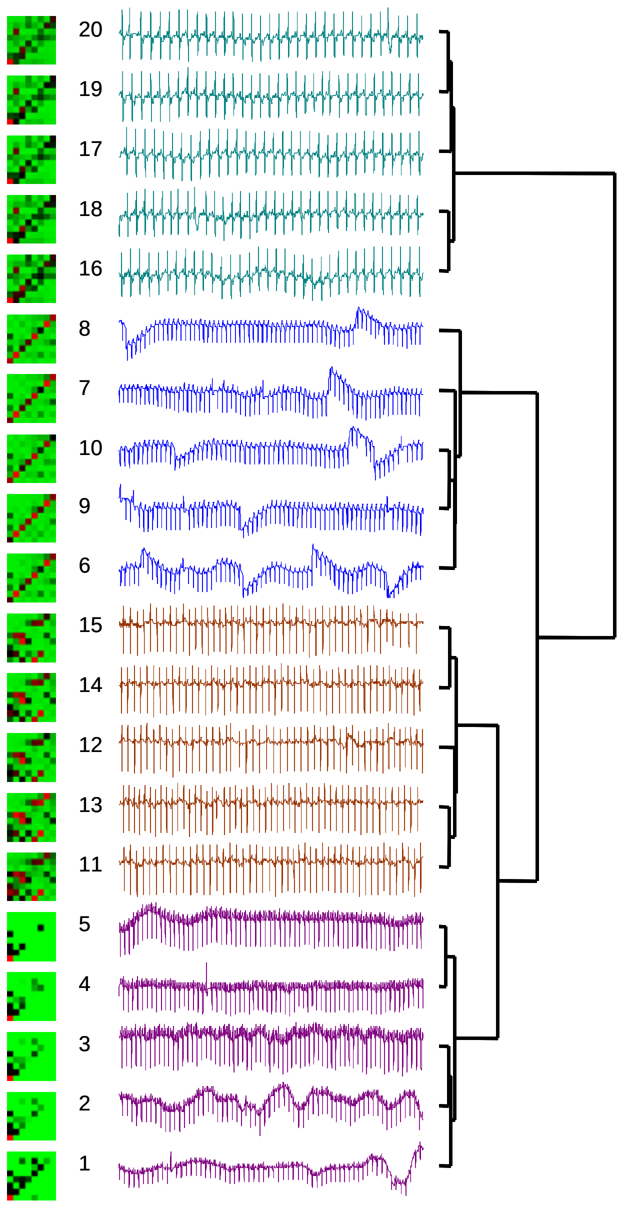


Figure 7. The clustering obtained by the time series thumbnail approach on a homogeneous data collection.

5.2 Classification

For classification, we considered an ECG classification problem. Our ECG dataset is a four-class problem derived from BIDMC Congestive Heart Failure Database of four patients. Each instance consists of 3,200 contiguous data points (about 20 heartbeats) randomly extracted from a long (several hours) ECG signal. Twenty instances are extracted from each class (patient). We compared to the ubiquitous Euclidean distance [12][13][15] and DTW [11][20].

For both datasets, we measure the error rates, using the one-nearest-neighbor with leaving-one-out evaluation method. The results are summarized in Table 1.

Table 1: Classification error rates.

	Euclidean	**DTW**	**Bitmaps**
ECG	42.25 %	16.25 %	7.50 %

We also considered a *Normal* vs *Arrhythmia* problem that appeared in [8]. Using Markov models, the authors reported an error rate of 2%. With our technique, under virtually any parameter settings, we achieve 0% error. We can achieve perfect classification using one nearest neighbor as above, or we can use MDS to project the data into 2 dimensional space and achieve perfect classification using a simple linear classifier, a decision tree or SVD. Figure 8 shows the data projected into 2D space, and the linear classifier learned.

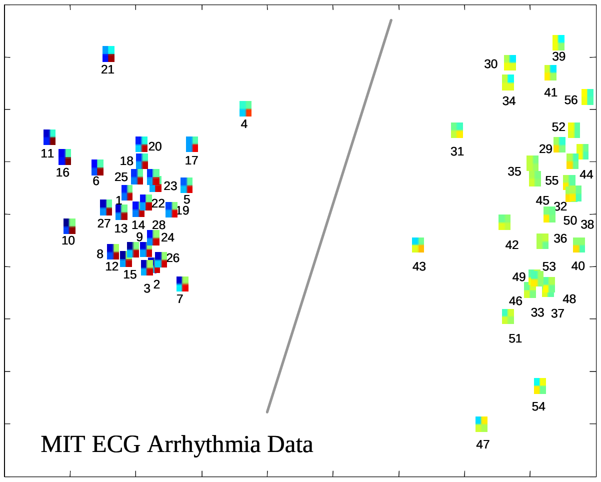
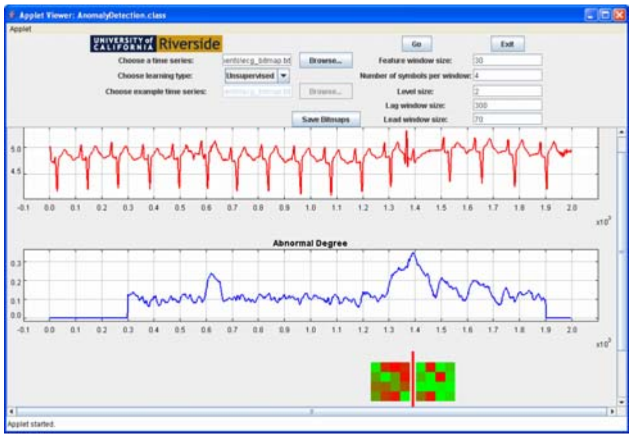


Figure 8. The MIT ECG Arrhythmia dataset projected into 2D space using only the information from level-2 time series bitmaps. The two classes are easily separated by a simple linear classifier (gray line).

5.3 Anomaly detection

The time series bitmap distance measure allows the creation of a simple anomaly detection algorithm.

We can create two concatenated windows and slide them together across the sequence. At each time instance, we build a time series bitmap for the two windows and measure the distance between them. We report this distance as an anomaly score. Figure 9 illustrates the idea on some annotated ECG data. This approach easily detects the single anomaly shown, and the rest of the annotated anomalies in this dataset (not shown).



Choose a time series: ents\ecg_bitmap.txt
Choose learning type: Unsupervised
Feature window size: 30
Number of symbols per window: 4
Level size: 2
Lag window size: 300
Lead window size: 70

Figure 9. Using time series bitmaps as an anomaly detector. *Top*) A subsection of an ECG dataset. A cardiologist annotated a premature ventricular contraction at approximately the 1.4 mark. *Middle*) The score of our approach shows a strong peak for the duration of the anomalous heartbeat. *Bottom*) The bitmaps before and after the peak are very different.

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

We have built an online version of this tool, which the readers may investigate with their own datasets (or several built-in examples). We encourage the interested reader to experiment with the tool [10].

In the simple example above, both sliding windows are of the same length. More generally, one may wish for the trailing window to be larger, so that it retains more of a "memory" of the previous data. We leave such considerations for future work.

Reproducible Results Statement: In the interests of competitive scientific inquiry, all datasets used in this work are available at the following URL [10]. This research was partly funded by the National Science Foundation under grant IIS-0237918.

References

[1] Aach, J., & Church, G. (2001). *Aligning gene expression time series with time warping algorithms*. Bioinformatics, Volume 17.

[2] Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). *Fast similarity search in the presence of noise, scaling, and translation in times-series databases*. In Proceedings of 21st International Conference on Very Large Databases.

[3] Barnsley, M.F., & Rising, H. (1993). *Fractals Everywhere*, second edition, Academic Press.

[4] Berndt, D., & Clifford, J. (1994). *Using dynamic time warping to find patterns in time series*, AAAI Workshop on Knowledge Discovery in Databases, pp. 229-248.

[5] Celly, B. & Zordan, V. B. (2004). Animated People Textures. In proceedings of the 17th International Conference on Computer Animation and Social Agents. Geneva, Switzerland.

[6] 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.

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

[8] Ge, X., & Smyth, P. (2000). *Deformable Markov model templates for time-series pattern matching*. In proceedings of the sixth ACM SIGKDD, pp. 81-90.

[9] Jeffrey, H.J. (1992). *Chaos Game Visualization of Sequences*. Comput. & Graphics 16, pp. 25-33.

[10] Keogh, E. http://www.cs.ucr.edu/~eamonn/CBMS05anonymous/

[11] Keogh, E. (2002). *Exact indexing of dynamic time warping*. In Proceedings of the twenty-eighth International Conference on Very Large Data Bases, pp. 406-417.

[12] Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra (2001). *Locally adaptive dimensionality reduction for indexing large time series databases*. In Proceedings of ACM SIGMOD Conference on Management of Data.

[13] Keogh, E. & Kasetty, S. (2002). *On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration*. In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[14] 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.

[15] Korn, F., Jagadish, H., & Faloutsos, C. (1997). *Efficiently supporting ad hoc queries in large datasets of time sequences*. In Proceedings of SIGMOD, pp. 289-300.

[16] 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.

[17] 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.

[18] Shneiderman, B. (2002). *Inventing discovery tools: combining information visualization with data mining*. Information Visualization 1(1): 5-12.

[19] 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 (JSAI). Kanazawa, Japan.

[20] Ratanamahatana, C.A., & Keogh, E. (2004). *Everything you know about Dynamic Time Warping is Wrong*. 3rd Workshop on Mining Temporal and Sequential Data, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[21] Wyszecki, G. (1982). *Color science: Concepts and methods, quantitative data and formulae*, 2nd edition. New York, Wiley, 1982.

[22] Tsuruoka, Y. & Tamura, Y. (2001). *Time Series Analysis of Bio-Medical Signals.* 14th IEEE Symposium on Computer-Based Medical Systems (CBMS'01), Maryland, USA.