A Practical Tool for Visualizing and Data Mining Medical Time Series

Li Wei Nitin Kumar Venkata Lolla
Eamonn Keogh Stefano Lonardi
Chotirat Ann Ratanamahatana

University of California, Riverside
Department of Computer Science & Engineering
{wli, nkumar, vlolla, eamonn, stelo, ratana}@cs.ucr.edu

Helga Van Herle

University of California, Los Angeles
David Geffen School of Medicine
hvanherle@mednet.ucla.edu

Abstract

The increasing interest in time series data mining has had surprisingly little impact on real world medical applications. 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 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 bitmaps, a user can quickly discover clusters, anomalies, and other regularities within the data collection. We demonstrate the utility of our approach with a set of comprehensive experiments on real datasets from a variety of medical domains.

1. Introduction

The increasing interest in time series data mining has resulted the introduction of a variety of similarity measures/ representations/ definitions/ indexing techniques and algorithms (see, e.g., [1][2][5][8][9][10]). Surprisingly, the massive research effort has had little impact on real world applications. 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. Possible reasons for this discrepancy are the bewildering number of parameters, the specialized hardware and/or software, and the long time learning required by the time series data mining tools.

In this work, we attempt to address this problem by introducing a 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 the relative frequency of these features to color a bitmap. By visualizing the similarities and differences among the bitmaps, a user can quickly discover clusters, anomalies, and other regularities within the data collection.

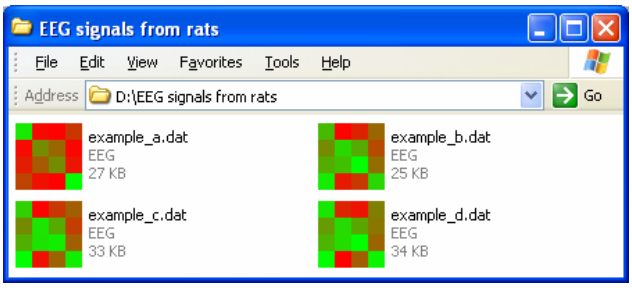

Figure 1. Four time series bitmaps.

Our system can both be used as an interactive tool and be embedded directly into any standard operation system. We employ the bitmap representation as the icon for time series file. Simply by glancing at the icons of time series files, a user can quickly identify files that require further investigation. Figure 1 illustrates a simple example for four EEG time series. Time series *example_a.dat* is from a normal trace, whereas the others contain

examples of spike-wave discharges. The difference between one dataset and all the rest is immediately apparent from a casual inspection of the bitmap representation.

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* [1]. The method produces a representation of DNA sequences, in which both local and global patterns are displayed.

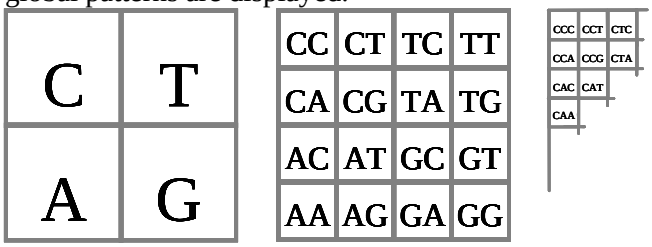

Figure 2. The quad-tree representation of a sequence over the alphabet {A,C,G,T} at different levels of resolution.

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 the frequency counts. The CGR representation of a sequence allows the investigation of the patterns in sequences, making it possible for human eyes to recognize hidden structures.

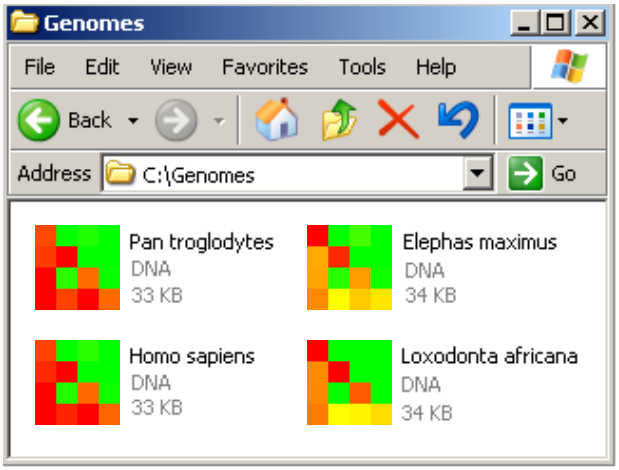

Figure 3. The bitmap representation of the gene sequences of four animals.

In Figure 3, we take the first 5,000 symbols of the mitochondrial DNA sequences of four species and use them to create file icons. 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 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 time series data. Since CGR is only defined for discrete sequences and most time series are real valued, a discretization method is necessary to transform continuous time series data into discrete domain. For this purpose, we used the Symbolic Aggregate approXimation (SAX) [11], as reviewed 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 [3], the SAX technique of Lin *et. al.* [11] 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. If a representation is tightly lower bounding the original data, it must be representing it with great fidelity.

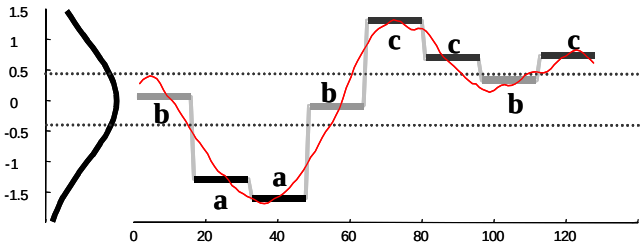

Figure 4. A real valued time series can be converted to the SAX word baabccbc.

The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. By substituting each section with its mean, a reduced dimensionality piecewise constant approximation is obtained. This representation is then discretized to produce a word with approximately equi-probable symbols. The whole conversion process is shown in Figure 4.

Users need to choose the length of the local sliding window *N*, and the number *n* of equal sized sections in which to divide *N*. A good choice for *N* should reflect the natural scale at which the events occur in the time series. A good value for *n* depends on the complexity of the signal.

3. Time Series Bitmaps

At this point, we could combine the idea of *Chaos game* [1] and the SAX representation to get the bitmap representation for time series. We use SAX on alphabet size four because the *Chaos game* bitmaps are defined on sequence with alphabet size of four. We use simple alphabetical ordering as shown in Figure 5 as an initial ordering for the four SAX symbols **a**, **b**, **c**, and **d**.

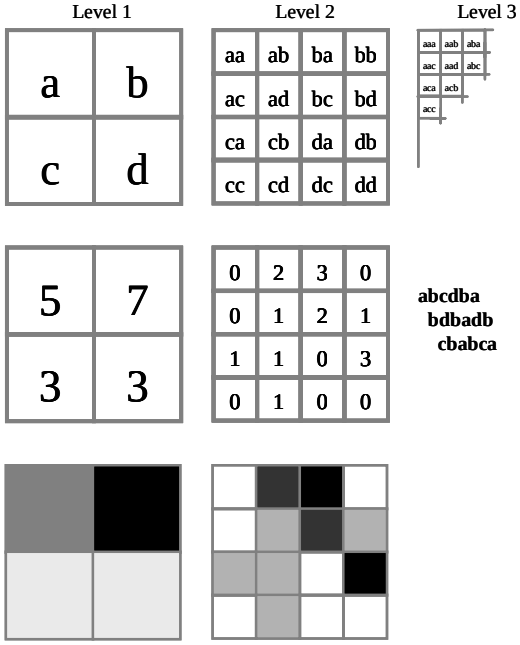

Figure 5. The generation of time series bitmaps.

After converting the original raw time series into the SAX representation, we 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**.

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 them 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* and 1 = *black*. However, it is generally recognized that grayscale is not *perceptually uniform* [13]. A color space is said to be perceptually uniform if small changes to a pixel 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.

It is important to note that we do *not* suggest any utility in viewing a single time series 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 that we can transparently integrate it into most standard operating systems. For time series files, we generate the bitmap representations and use them as the file icons. Simply by glancing the icons of time series files, a user may spot files that require further investigation or note natural clusters in the data.

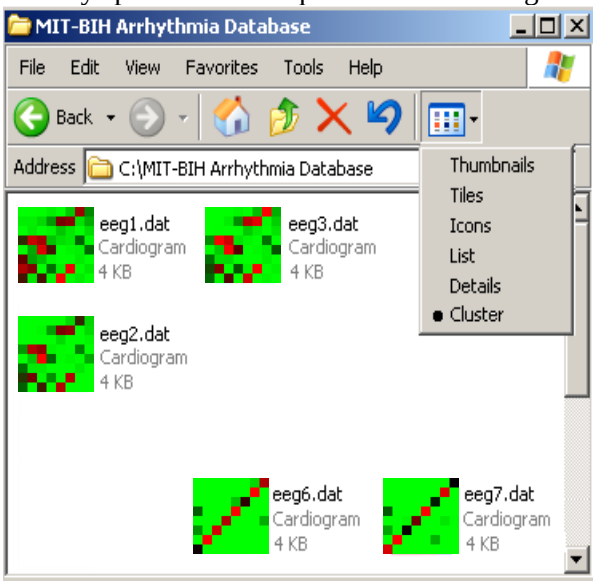

Figure 6. A snapshot of a folder arranged by "Cluster" option.

Normally, file icons are arranged by one of a handful of common criteria, such as *name*, *date*, *size*, etc. To augment the utility of the time series bitmaps, 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. Figure 6 displays an example of *Cluster View* in Microsoft Windows XP Operating System obtained for five MIT-BIH Arrhythmia Database files. It is evident 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 $dist(BA, BB) = \sum_{i=1}^{n}\sum_{j=1}^{n}(BA_{ij} - BB_{ij})^2$.

5. Experimental Evaluation

In this section, we will show some experiments that objectively measure the utility of our approach on classification, clustering, and anomaly detection. Since the quality of illustrations here suffers from monochromic printing and small-scale reproduction, we urge the interested reader to consult [6] for large-scale color reproductions and additional details.

5.1 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 [8][9][10] and DTW [6][12].

Table 1: Classification error rates.

	Euclidean	DTW	Bitmaps
ECG	42.25 %	16.25 %	7.50 %

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

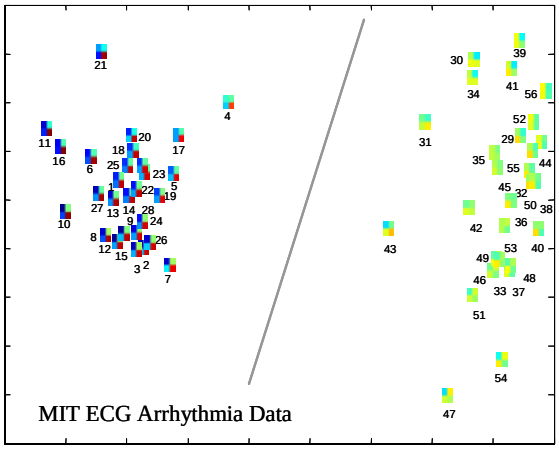

Figure 7. Classification by projection.

We also considered a *Normal* vs *Arrhythmia* problem that appeared in [4]. 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 7 shows the data projected into 2D space, and the linear classifier learned (the gray line).

5.2 Clustering

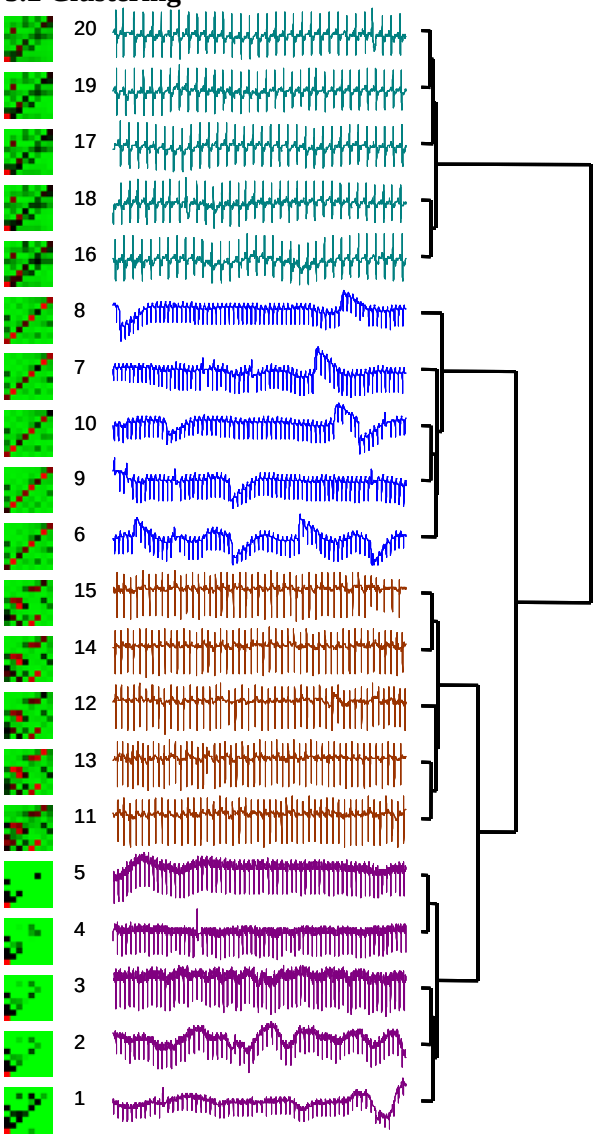

Figure 8. Clustering results.

We considered a four-class ECG clustering problem, where each class corresponds to a different patient. Figure 8 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.

5.3 Anomaly detection

The time series bitmap distance measure allows the creation of a simple anomaly detection algorithm. We create two concatenated windows and slide them together across the sequence. At each time instance, time series bitmaps are built for the two windows and the distance between them is reported as an anomaly score.

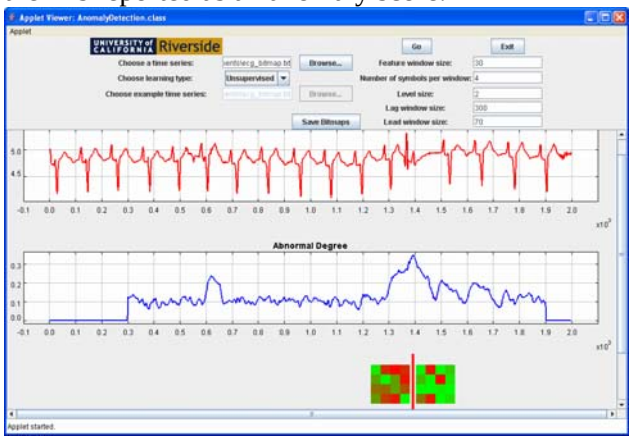

Figure 9. Time series anomaly detector.

Figure 9 illustrates the idea on some annotated ECG data. The top part is a subsection of an ECG dataset. A cardiologist annotated a premature ventricular contraction at approximately the 1.4 mark. The score in the middle shows a strong peak for the duration of the anomalous heartbeat. The bottom shows that the bitmaps before and after the peak are very different. Our approach easily detects the single anomaly shown, and the rest of the annotated anomalies in this dataset (not shown).

We have built an online version of this tool [6], which the readers may investigate with their own datasets (or several built-in examples).

Reproducible Results Statement: In the interests of competitive scientific inquiry, all datasets used in this work are available at the following URL [6]. 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] 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.

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

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

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

[6] Keogh, E. http://www.cs.ucr.edu/~wli/CBMS05/

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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