A Novel Bit Level Time Series Representation with Implication of Similarity Search and Clustering

Chotirat Ratanamahatana[1], Eamonn Keogh[1], Anthony J. Bagnall[2], and Stefano Lonardi[1]

[1] Dept. of Computer Science & Engineering, Univ. of California, Riverside, CA 92521 USA
{ratana, eamonn, stelo}@cs.ucr.edu
[2] School of Computing Sciences, University of East Anglia, Norwich, UK
ajb@cmp.uea.ac.uk

Abstract. Because time series are a ubiquitous and increasingly prevalent type of data, there has been much research effort devoted to time series data mining recently. As with all data mining problems, the key to effective and scalable algorithms is choosing the right representation of the data. Many high level representations of time series have been proposed for data mining. In this work, we introduce a new technique based on a bit level approximation of the data. The representation has several important advantages over existing techniques. One unique advantage is that it allows raw data to be directly compared to the reduced representation, while still guaranteeing lower bounds to Euclidean distance. This fact can be exploited to produce faster exact algorithms for similarly search. In addition, we demonstrate that our new representation allows time series clustering to scale to much larger datasets.

1 Introduction

Time series are a ubiquitous and increasingly prevalent type of data. Because of this fact, there has been much research effort devoted to time series data mining in the last decade [1],[2],[3],[4]. As with all data mining problems, the key to effective and scalable algorithms is choosing a suitable representation of the data. Many high level representations of time series have been proposed for data mining. In this work, we introduce a novel technique based on a bit level approximation of the data. As we will show, our clipped representation has several important advantages over existing techniques. The proposed approach is not only a new representation; it is a new type of representation. For data adaptive, non-data adaptive, and model-based approaches, the user has a choice (implicit or explicit) of the compression ratio. This allows the user to fine tune the parameters to achieve the ideal compression/ fidelity tradeoff for their particular application.

In contrast, with the clipped representation, the data *itself* dictates the compression ratio; the user has no choice to make. This may be seen as somewhat of a disadvantage (although removing parameters from a data mining task is often a good thing

T.B. Ho, D. Cheung, and H. Liu (Eds.): PAKDD 2005, LNAI 3518, pp. 771 – 777, 2005.

[5]). However, this lack of flexibility is counterbalanced by another unique property of the clipped representation. For all other dimensionality reduction approaches, we must transform the query into the same representation as the dimensionality reduced database, i.e. having a loss of fidelity for the candidate matches stored in the index and a loss of fidelity for the query. This in turn produces weak lower bounds, and thus weak pruning power. In contrast, the clipped representation is unique in that the original raw query can be compared directly to the clipped candidate sequences, thus producing tighter lower bounds, greater pruning power and faster query by content.

2 The Clipped Representation

Our proposed representation works by replacing each real valued data point with a single bit. gives the visual intuition.

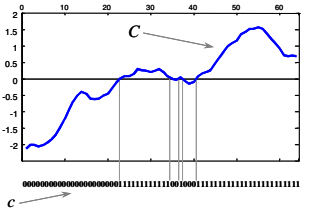

Fig. 1. A time series, *C*, of length 64, is converted to the clipped representation, *c*, by observing each element of *C*; if its value is strictly above zero, the corresponding bit is set to 1, and to 0 otherwise

More formally, we can define *c*, the clipped representation of *C* as:

$$c(i) = \begin{cases} 1 & if \quad C(i) > \mu \\ 0 & otherwise \end{cases} \tag{1}$$

where μ is the mean of *C*. Since the importance of normalizing the data before attempting any clustering, classification or indexing [3] is well-established, we can simply assume $\mu = 0$, without loss of generality for the rest of this work. Note that this representation has been considered before in the statistical community [6], but its utility for data mining, namely, the ability to lower bound distance functions, is first documented here.

2.1 Lower Bounding Euclidean Distance

Suppose we have 2 time series, a query $Q = Q_1,Q_2,\ldots,Q_i,\ldots,Q_n$, and a candidate match $C = C_1,C_2,\ldots,C_j,\ldots,C_n$. The Euclidean Distance can simply be used to compare the two time series. However, if we have a clipped time series *c*, and a raw time series *Q*, we can also lower bound the squared Euclidean distance between *C* and *Q*, using equation 2) below. Due to space limitations, the proof of this *LB_clipped* is omitted and can be found in [7]. However, gives its visual intuition.

$$LB\_clipped(Q,c) \equiv \sum_{i=1}^{n} \begin{cases} Q_i^2 & if\ (Q_i > 0\ \ and\ \ c_i = 0)\ or (Q_i \le 0\ \ and\ \ c_i = 1) \\ 0 & Otherwise \end{cases} \quad (2)$$

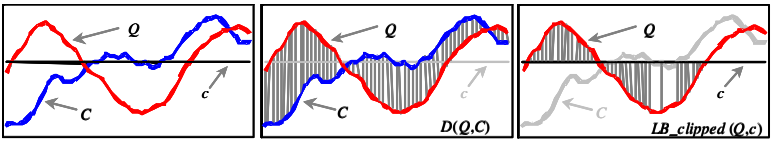

Fig. 2. The distance returned by both LB_clipped(*Q, c*) and *D(Q,C)* is the sum of squared lengths of the gray hatch lines. Because every hatch line for *LB_clipped(Q,c)* is matched with corresponding line in *D(Q,c)* which is at least as long, we must have *LB_clipped(Q,c)* ≤ *D(Q,c)*

2.2 Run Length Encoding

Consider the clipped sequence *c*, which we have been using as a running example. Its value is **00000000000000000000001111111111110010001111111111111111111111111.** Note that we could write this as 22#**0**, 11#**1**, 2#**0**, 1#**1**, 3#**0**, 24#**1**, which we can interpret as 22 zeros followed by 12 ones, etc. The shorter format allows us to fit more data in main memory. In fact, we can be even terser; because we always toggle from zero to one or vice versa, so we only need to record the parity of the first bit, giving us 22#**0**, **11**,**2**,**1**,**3**,**24**. This classic lossless compression technique is known as Run Length Encoding (RLE). To make the representation even shorter, we can represent the parity bits of 0 and 1 with two special characters, e.g. "@" and "!," respectively; our run length encoding now can be represented as @**22**,**11**,**2**,**1**,**3**,**24**. We can use this to further reduce the clipped representation of the data. Note that while the example above illustrates the idea with ASCII characters, we actually do RLE at the bit level.

2.3 Numerosity Reduction

Even though the run length-encoding scheme itself gives an impressive compression ratio, we can improve it by numerosity reduction on sliding windows. This step is motivated by observing that while applying a sliding window on the streaming data, time series in consecutive sliding windows are very often identical in the clipped representation, except for the first and the last values that are omitted and added, respectively. If the time series in each sliding window has this property, we can exploit this fact and just record the maximum amount of time this property has consecutively been observed, along with a special character, $, that represents this reduction. Consider the run length encoding from our example in the previous section and let the encoding of the next five sliding windows be:

@**22**,**11**,**2**,**1**,**3**,**24**@*21*,**11**,**2**,**1**,**3**,*25*@*20*,**11**,**2**,**1**,**3**,*26*@*19*,**11**,**2**,**1**,**3**,*27*@**18**,**22**,**2**,**1**,**3**,**27**,**1**@*17*,**22**,**2**,**1**,**3**,**2**
7,*2*.

We can readily see that the first four windows are very similar and can be reduced to one since the only values differ from each other are the first and the last (italicized

for clarity). However, the 5th window cannot be combined with the previous one since the last bit has changed from 1 to 0, but it can be combined with its next window. As a result, the final encoding with numerosity reduction becomes @22,11,2,1,3,24$3@18,22,2,1,3,27,1$1.

As before, although we demonstrate the idea with ASCII text, we actually encode everything at the bit level. With the Power Demand dataset of size 10,000 data points, numerosity reduction together with Huffman coding yields a huge compression ratio of 1057:1. Note that while the factor of 32 to 1 achieved by clipping is lossy, the remaining factor of approximately 33 to 1 is lossless with respect to the clipped data.

3 Empirical Evaluations

In this section, we will provide an extensive empirical comparison among the raw and various representations of compressed data in two major data mining tasks, time series indexing and clustering. Twelve datasets were used in our indexing experiments, and two were used for clustering experiments (only subsets of results are shown here due to space limitations). We also tested on a wide range of both real and synthetic datasets. The datasets range from 66 Kilobytes to 2 Gigabytes in size (see [7] for complete details).

3.1 Experimental Methodology

For indexing, we will demonstrate the superiority of our clipped representation in terms of number of disk accesses. We compare our proposed method with the classic Piecewise Aggregate Approximation (PAA) and Discrete Fourier Transform (DFT), all preserving similar compression ratio. We then demonstrate that clipped series can produce clusters similar to those obtained with the raw data when clustering a very large real world database introduced in section 3.3. We show that clipping performs favorably when compared to clustering with unclipped data since clustering can be done faster and with much less memory requirement.

For similarity search, we performed all experiments over a range of query lengths. Since we want to include PAA in our experiments, the query length is somewhat limited. We therefore consider query lengths of 256 and 512 data points. We tested our approach on a variety of twelve datasets with various properties within the data, obtained from the UCR Time Series Data Mining Archive [8]. The sizes of the datasets range from 6,875 data points to 198,400 data points. Leaving-one-out cross validation is used; on each run, we randomly pick a query from a database, create a run-length encoding with numerosity reduction for the rest of the data, and determine the resultant compression ratio. We then create PAA and DFT on the same data and with the same compression ratio (or with smaller compression ratio, in favor of PAA and DFT) then measure the number of random disk accesses for the nearest neighbor queries of all methods. To determine the number of dimensionality reduction (m) in PAA and DFT in these cases, we assume that each value in PAA and DFT can be represented by only *two* bytes (instead of 4 or 8 bytes) to demonstrate that our results are still competitive among all the approaches. In addition, to avoid any possibility of implementation bias, the number of I/O disk accesses of each method is measured

instead of recording the actual running time. This is done by first computing the lower bound distances using LB_clipped and Euclidean distance, between a query and all the sequences in the dataset. Then to retrieve the nearest neighbor, each sequence is visited in the order according to the lower bound values. We count the number of times the real disk accesses must be made. These numbers also indicates the tightness of the lower bounds for each representation. The results are averaged over 100 separate runs for each dataset. For simplicity, we only report results for one-nearest neighbor queries.

3.2 Indexing Results

As noted above, the amount of compression is dictated by the data itself. For the twelve datasets considered the compression ratios range between 60.2:1 to 1,089.5:1. We compare different representations in terms of I/O random disk accesses during the process of the 1-nearest neighbor retrieval of a query time series. In particular, in each run, we reduce the dimensionality of the data from n to m using Clipped, PAA, and DFT representations, and build their indices on the reduced spaces based on their lower bounds between each subsection (sliding window) of the time series and the query. To allow a visual comparison, we normalize each experiment on each dataset by the worst performing algorithm; the raw numbers are available in [7]. Fig. 3 shows the number of disk accesses with lower bounding the Euclidean distance, using the three dimensionality-reduction techniques over the range of query lengths of 256 and 512 data points. In general, the results show that the clipped representation greatly outperforms or at least is comparable to the other approaches, expressing the superiority in its tightness of the lower bounds. Again, we would like to emphasize that our results here are obtained by conservatively assuming only *two*-byte requirement to represent each number in PAA and DFT. If we assume 4 or 8 bytes or without the parameter m adjusted, the results will be much improved.

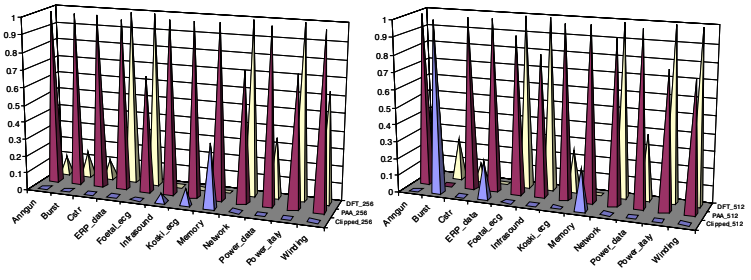

Fig. 3. Number of disk accesses with lower bounding of Euclidean distance, normalized by the worst performing approach, using the 3 representations for query lengths of 256 and 512 points

3.3 General Compression-Based Clustering

We examine a class of problems where a DFT approach should produce good results, and show that clipping is better than the most commonly used DFT approach described in [2].

To demonstrate how clipping can help with a real world large dataset, we cluster optical recording data from a bee's olfactory system [9]. The data consists of 980 images, each image containing of 688x520 measurements. If we consider each position in the image as a time series, the data consists of 357,760 time series of length 980. Preliminary analysis has shown that clustering the series based on similarity in time produces results that have a sensible physiological interpretation [9]. We cluster with *k*-means (with *k* set to 16) restarted 50 times from random initial centroids, and take as the best clustering the one with the lowest within-cluster variation.

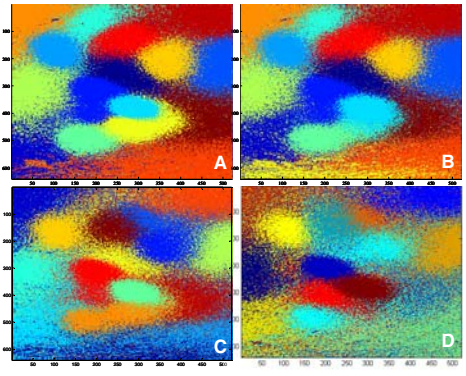

Fig. 4. A) 16 clusters produced using all 2GB of raw data. B) Clusters formed using the clipped data with 32:1 compression ratio. The spatial cluster co-occurrences between this plot and A) shows its effectiveness in the clipped data reduction technique. C) Clusters formed using PAA with 59 coefficients, giving 20:1 compression ratio. D) Clusters formed using first 17 DFT coefficients, giving 29.7:1 compression ratio

4 Conclusions

In this paper, we have shown that a simple dimensionality reduction technique, i.e. the clipped representation, can outperform more sophisticated techniques by a few orders of magnitude. We have shown that our proposed clipped representation can improve the compression ratio by a wide margin, while being able to maintain or increase the tightness of its lower bound, which allows even faster nearest neighbor queries, especially in ones that require Dynamic Time Warping distance measure. Other than producing faster exact algorithms for similarity search, we have also demonstrated that our clipped representation approach can support clustering and scale to much larger datasets.

Acknowledgements. This research was partly funded by the NSF under grant IIS-0237918.

References

1. Aach, J. & Church, G. Aligning gene expression time series with time warping algorithms. Bioinformatics(17) (2001) 495-508.
2. Berndt, D., Clifford, J. Using dynamic time warping to find patterns in time series. AAAI-94 Workshop on Knowledge Discovery in Databases. (1994) 229-248.
3. Keogh E., Kasetty, S. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In the 8th ACM SIGKDD (2002) 102-111.
4. Yi B K., Faloutsos, C. Fast time sequence indexing for arbitrary Lp norms. VLDB (2000) 385-94.
5. Keogh, E., Lonardi, S., Ratanamahatana, CA. Towards Parameter-Free Data Mining. In proceedings of SIGKDD (2004).
6. Kedem B., Slud, E. On Goodness of Fit of Time Series Models: An Application of Higher Order Crossings, Biometrika, Vol 68, (1981) 551-556
7. Ratanamahatana, C.A., Keogh, E., Bagnall, A.J., Lonardi, S. A Novel Bit Level Time Series Representation with Implication of Similarity Search and Clustering. (2004) [http://www.cs.ucr.edu/downloads/techrpt/TR_clippedpaper.pdf].
8. Keogh E., Folias, T. The UCR time Series Data Mining archive. (2002) [http://www.cs.ucr.edu/~eamonn/TSDMA].
9. Galan, R.F., Sachse, S., Galizia, C.G., Herz, A.V.M. "Odor-driven attractor dynamics in the antennal lobe allow for simple and rapid olfactory pattern classification." Neural Computation (2004)
10. Bagnall, A. J., Janacek, G. Clustering time series from ARMA models with clipped data, SIGKDD (2004).