Discovering the Intrinsic Cardinality and Dimensionality of Time Series using MDL

Bing Hu Thanawin Rakthanmanon Yuan Hao Scott Evans[1] Stefano Lonardi Eamonn Keogh
Department of Computer Science & Engineering
University of California, Riverside
Riverside, CA 92521, USA
[1]GE Global Research
{bhu002, rakthant, yhao002}@ucr.edu, [1]evans@ge.com, {stelo, eamonn}@cs.ucr.edu

Abstract—Most algorithms for mining or indexing time series data do not operate directly on the original data, but instead they consider alternative representations that include transforms, quantization, approximation, and multi-resolution abstractions. Choosing the best representation and abstraction level for a given task/dataset is arguably the most critical step in time series data mining. In this paper, we investigate techniques to discover the natural intrinsic representation model, dimensionality and alphabet cardinality of a time series. The ability to discover these intrinsic features has implications beyond selecting the best parameters for particular algorithms, as characterizing data in such a manner is useful in its own right and an important sub-routine in algorithms for classification, clustering and outlier discovery. We will frame the discovery of these intrinsic features in the Minimal Description Length (MDL) framework. Extensive empirical tests show that our method is simpler, more general and significantly more accurate than previous methods, and has the important advantage of being essentially parameter-free.

Keywords: Time Series, MDL, Dimensionality Reduction

I. INTRODUCTION

Most algorithms for indexing or mining time series data operate on higher-level representations of the data, which include transforms, quantization, approximations and multi-resolution approaches. For instance, Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Adaptive Piecewise Constant Approximation (APCA) and Piecewise Linear Approximation (PLA) are models that all have their advocates for various data mining tasks, and each has been used extensively [5]. However the question of choosing the *best* abstraction level and/or representation of the data for a given task/dataset still remains open. In this work, we investigate this problem by discovering the natural intrinsic model, dimensionality and (alphabet) cardinality of a time series. We will frame the discovery of these intrinsic features in the Minimal Description Length (MDL) framework [7][11][18][21]. MDL is the cornerstone of many bioinformatics algorithms [6][20], but it is arguably underutilized in time series data mining [8][17].

The ability to discover the intrinsic dimensionality and cardinality of time series has implications beyond setting the best parameters for data mining algorithms, as characterizing data in such a manner is useful in its own right to understand/describe the data and an important sub-routine in algorithms for classification, clustering and outlier discovery [19][26]. To illustrate this, consider the three unrelated datasets in Figure 1.

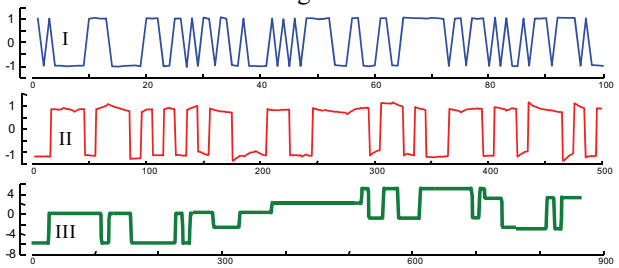

Figure 1. Three unrelated industrial time series with low intrinsic cardinality. I) Evaporator (channel one). II) Winding (channel five). III) Dryer (channel one).

The number of unique values in each time series is, from top to bottom, 14, 500 and 62. However, we might reasonably claim, that the *intrinsic* alphabet cardinality is instead 2, 2, and 12 respectively. As it happens, an understanding of the processes that produced these data would perhaps support this claim [10]. In these datasets, and indeed in many real-world datasets, there is a significant difference between the *actual* and *intrinsic* cardinality. Similar remarks apply to dimensionality.

Before we define more precisely what we mean by *actual* versus *intrinsic* cardinality, we should elaborate on the motivations behind our considerations. Our objective is generally not simply to save memory[1]: if we are wastefully using eight bytes per time point instead of using the mere three bytes made necessary by the intrinsic cardinality, the memory space saved is significant, but memory is getting cheaper every day, and is rarely a bottleneck in data mining tasks. There are instead many other reasons why we may wish to find the true intrinsic model, cardinality and dimensionality of the data, for example:

- There is an increasing interest in using specialized hardware for data mining [22]. However, the complexity of implementing data mining algorithms in hardware typically grows super linearly with the cardinality. For example, FPGAs usually cannot handle cardinalities greater than 256 [22].

[1] Although Section I.*A* shows an example where this *is* useful.

1550-4786/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDM.2011.54

- Some data mining algorithms benefit from having the data represented in the lowest *meaningful* cardinality. As a trivial example, in the stream: ..0, 0, 1, 0, 0, 1, 0, 0, 1, we can easily find the rule that a '1' follows two appearances of '0'. However, notice that this rule is *not* apparent in this string: ..0, 0, 1.0001, 0.0001, 0, 1, 0.000001, 0, 1 even though it is essentially the same.
- Most time series indexing algorithms critically depend on the ability to reduce the dimensionality [5] or the cardinality [13] of the time series (or both [1] [2]), and searching over the compacted representation in main memory. However, setting the best level of representation remains a black art.
- In resource-limited devices, it may be helpful to remove the spurious precision induced by a cardinality/dimensionally that is too high. We elaborate on this issue below.
- Knowing the intrinsic model, cardinality and dimensionality of a dataset allows us to create very simple outlier detection models. We simply look for data where the parameters discovered in new data differ from our expectations learned on training data. This is a simple idea, but it can be very effective.

To enhance our appreciation of the potential utility of knowing the intrinsic cardinality and dimensionality of the data, we briefly consider an application in classification.

A. A CONCRETE EXAMPLE

For concreteness we present a simple scenario that shows the utility of understanding the intrinsic cardinality/dimensionality of data. Suppose we wish to build a time series classifier into a device with a limited memory footprint such as a cell phone or pacemaker [23]. Let us suppose we have only 20kB available for the classifier, and that (as is the case with the benchmark dataset, *TwoPat* [10]) each time series exemplar has a dimensionality of 128 and takes 4 bytes per value.

One could choose decision trees or Bayesian classifiers because they are space efficient, however it is well known that nearest neighbor classifiers are *very* difficult to beat for time series problems [5]. If we had simply stored forty random samples in the memory for our nearest neighbor classifier, the average error rate over fifty runs would be a respectable 58.7% for a four-class problem. However, we could also down-sample the *dimensionality* by a factor of two, either by skipping every second point, or by averaging pairs of points (as in SAX [13]), and place eighty reduced quality samples in memory. Or perhaps we could instead reduce the *alphabet cardinality*, by reducing the precision of the original four bytes to just one byte, thus allowing 160 reduced-fidelity objects to be placed in memory. Many other combinations of dimensionality and cardinality reduction could be tested, which would trade reduced fidelity to the original data for more exemplars stored in memory. In this case, a dimensionality of 32 and a cardinality of 6 allow us to place 852 objects in memory and achieve an error rate of about 90.75%, a remarkable accuracy improvement given the limited resources. As we shall see, we found this combination of parameters using our MDL technique.

In general, testing all the combinations of parameters is computationally infeasible. Furthermore, while in this case we have class labels to guide us through the search of parameter space, this would not be the case for other unsupervised data mining algorithms, such as clustering, motif discovery [14], outlier discovery [3][24][26], and etc.

As we shall show, our MDL framework allows us to automatically discover the parameters that reflect the intrinsic model/cardinality/dimensionally of the data without requiring external information or expensive cross validation search.

II. DEFINITIONS AND NOTATION

We begin with the definition of a time series:

Definition 1: A *time series* T is an ordered list of numbers. $T=t_1,t_2,...,t_m$. Each value t_i is a finite precision number and m is the length of time series T.

Before continuing we must justify the decision of (slightly) quantizing the time series. MDL is only defined for discrete values[2], but most time series are real-valued. The obvious solution is to reduce the original number of possible values to a manageable amount. However the reader may object that such a drastic reduction in precision must surely lose some significant information. However this is *not* the case. To illustrate this point, we performed a simple experiment. From each of the twenty diverse datasets in the UCR archive [10] we randomly extracted one hundred pairs of time series. For each pair of time series we measured their Euclidean distance in the original high dimensional space, and then in the quantized 256-cardinality space, and used these pairs of distances to plot a point in a scatter plot. Figure 2 shows the results.

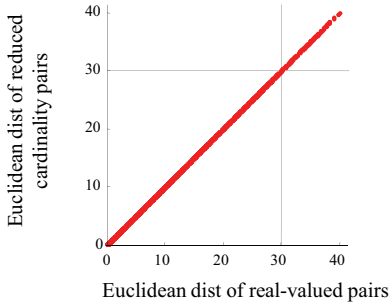

Figure 2. Each point on this plot corresponds to a pair of time series: the x-axis corresponds to their Euclidean distance, while the y-axis corresponds to the Euclidean distance between the 8-bit quantized representation of the same pair.

The figure illustrates that all the points fall close to the diagonal, thus the quantization makes no *perceptible* difference. Beyond this subjective visual test, we also reproduced the heavily cited UCR time series classification benchmark experiments [10], replacing the original data

[2] The closely related technique of MML (Minimum Message Length [25]) *does* allow for continuous real-valued data. However, here we stick with the more familiar MDL formulation.

with the 256-cardinality version. In no case did it make more than one tenth of one percent difference to classification accuracy (full details are at [28]). Given this, we simply reduce all the time series data to its 256 cardinality version in this work, by using *discretization*:

Definition 2: *Discretization* is a function used to normalize a real-valued time series T into b-bit discrete values in the range $[-2^{b-1}, 2^{b-1}-1]$. It is defined as following:

$$Discretization_b(T) = round\left(\frac{T - min}{max - min}\right) * (2^b - 1) - 2^{b-1}$$

where *min* and *max* are the minimum and maximum value in T, respectively[3].

For any time series T, we are interested in determining how many bits it takes to represent it. We can thus define the description length of a time series.

Definition 3: A *description length DL* of a time series T is the total number of bits required to represent it. When Huffman coding is used to compress the time series T, the description length of time series T is defined by:

$$DL\ (T) = |\ HuffmanCoding(T)\ |$$

In the current literature, the number of bits required to store the time series depends on the idiosyncrasies of the data format or hardware device, not on any intrinsic properties of the data or domain. However we are really interested in knowing the minimum number of bits to exactly represent the data, the *intrinsic* amount of information in the time series. Unfortunately, in the general case this is not calculable, as it is the Kolmogorov complexity of the time series [12]. However, we can approximate the Kolmogorov complexity by compressing the data, using say Huffman coding [7][24][27]. The (lossless) compressed file size is clearly an upper bound to the *DL* of the time series [4].

One of the key steps in finding the intrinsic cardinality and/or dimensionality is converting a given time series to other representation or model, e.g., by using DFT or DWT. We call that representation, a *hypothesis*:

Definition 4: A *hypothesis* H is a representation of a discrete time series T after applying a transformation M.

In general, there are many possible transforms M. Examples include the Discrete Wavelet Transform (DWT), the Discrete Fourier transform (DFT), the Adaptive Piecewise Linear Approximation (APCA), the Piecewise Linear Approximation (PLA), etc.[5]. Figure 6 shows three illustrative examples, DFT, APCA, and PLA. In this paper, we demonstrate our ideas using these three the most commonly used representations (two are relegated to the expanded version in [28]). Note however that our ideas apply to all time series models (see [5] for a survey of time series representations).

Note that we use the term *model* interchangeably with the term *hypothesis* in this work.

[3] This slightly awkward formula is necessary because we use the symmetric range [-128,127]. If we use range [1, 256] instead we get a more elegant: $Discretization(T) = round\left(\frac{T-min}{max-min}\right) * \left(2^8 - 1\right) + 1$.

Definition 5: A *reduced description length* of a time series T given hypothesis H is the number of bits used for encoding time series T, exploiting information in the hypothesis H, i.e., $DL\ (T \mid H)$, and the number of bits used for encoding H, i.e., $DL\ (H)$. Thus, the reduced description length is defined as:

$$DL\ (T, H) = DL\ (H) + DL\ (T \mid H)$$

The first term, $DL\ (H)$, called the *model cost*, is the number of bits required to store the hypothesis H. In brief, the model cost for say the piecewise linear approximation would include the bits needed to encode the mean, slope and length of each linear segment.

The second term, $DL\ (T \mid H)$, called the *correction cost* (in some works it is called the *description cost* or *error term*) is the number of bits required to rebuild the entire time series T from the *given* hypothesis H.

There are many possible ways to encode T using H. However, if we just simply store the differences (i.e. the difference vector) between T and H, we can easily re-generate a whole time series T from the information we have. Thus, we simply use $DL\ (T \mid H) = DL\ (T\text{-}H)$.

We will demonstrate how to calculate the reduced description length more in detail in the next section.

III. MDL MODELING OF TIME SERIES

A. AN INTUITIVE EXAMPLE OF OUR BASIC IDEA

For concreteness, we will consider a simple worked example comparing two possible dimensionalities of data. Note that here we are assuming a cardinality of 16, and a model of APCA. However, more generally we do not need to make such assumptions. Let us consider a sample time series T of length 24:

$T =$ **1 1 1 2 3 4 5 6 7 8 9 10 11 11 12 12 12 12 11 11 10 10 9 7**

In Figure 3 we show a plot of this data.

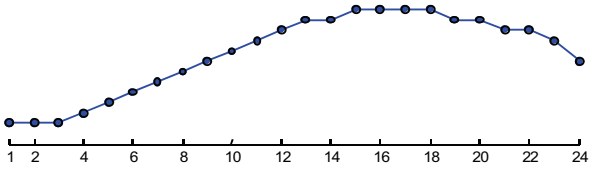

Figure 3. A sample time series T.

We can attempt to model this data with a single constant line, a special case of APCA. We begin by finding the mean of *all* the data, which (rounding in our integer space) is eight. We can create a hypothesis H_1 to model this data, which as shown in Figure 4. It is simply a constant line with a mean of eight. There are 16 possible values this model *could* have had. Thus $DL\ (H_1) = 4$ bits.

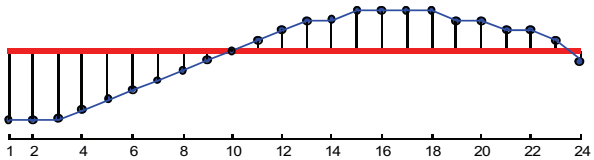

Figure 4. Time series T (blue/fine), approximated by a one-dimensional APCA approximation H_1 (red/bold). The error for this model is represented by the vertical lines.

This model H_1 has an error modeling T, and we must account for it. The errors e_1, represented by the length of the vertical lines in Figure 4 are:

e_1 = **7 7 7 6 5 4 3 2 1 0 -1 -2 -3 -3 -4 -4 -4 -4 -3 -3 -2 -2 -1 1**

As noted in Definition 5, the cost to represent these errors is the correction cost, the number of bits encoding e_1 using Huffman coding, which is 82 bits. Thus the overall cost to represent T with a one-dimensional model or its reduced description length is:

$$DL(T, H_1) = DL(T|H_1) + DL(H_1)$$

$$DL(T, H_1) = 82 + 4 = 86 \text{ bits}$$

We can now test to see if hypothesis H_2, which models the data with *two* constant lines could reduce the description length. Figure 5 shows the two segment approximation lines created by APCA.

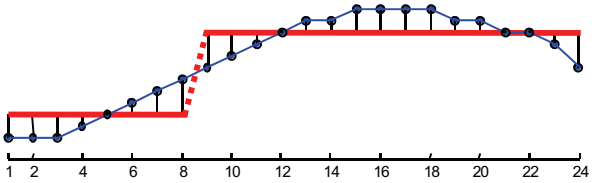

Figure 5. Time series T (blue/fine), approximated by a two-dimensional APCA approximation H_2 (red/bold). Vertical lines represent the error.

As we expect, the error e_2 shown as the vertical lines in Figure 5 is smaller than the error e_1. In particular, the error e_2 is:

e_2 = **2 2 2 1 0 -1 -2 -3 3 2 1 0 -1 -1 -2 -2 -2 -2 -1 -1 0 0 1 3**

The number of bits encoding e_2 using Huffman coding or the correction cost to generate the time series T given the hypothesis H_2, $DL(T|H_2)$, is 65 bits. Although the correction cost is smaller than one-dimensional APCA, the model cost is larger. In order to store two constant lines, two constant numbers corresponding to the height of each line and a pointer indicating the end position of the first line are required. Thus, the reduced description length of model H_2 is:

$$DL(T, H_2) = DL(T|H_2) + DL(H_2)$$

$$DL(T, H_2) = 65 + 2 * log_2(16) + \lceil log_2(24) \rceil = 78 \text{ bits}$$

Because we have $DL(T, H_2) < DL(T, H_1)$, we prefer H_2 as a proper number of segments for our data.

Clearly we are not done yet, we should also test H_3, H_4, H_5, etc., corresponding to 3, 4, 5, etc. piecewise constant segments. Moreover, we can also test alterative models corresponding to different levels of DFT or PLA representation. In addition, we can also test different cardinalities, because it is possible that the 16-value cardinality was unnecessary for this domain. For example, suppose we had been given T_2 instead:

T_2 = **0 0 0 0 4 4 4 4 4 0 0 0 0 8 8 8 8 8 8 12 12 12 12 12**

Here, if we tested multiple hypotheses as to the cardinality of this data, we would hope to find that the hypothesis H_4^c that attempts to encode the data with a cardinality of just 4 would result in the smallest model.

We have shown a detailed example for APCA; however, essentially all the time series representations can be encoded in a similar way. As shown with three representative examples in Figure 6, essentially all the time series models consist of a set of basic functions (i.e., coefficients) that are linearly combined to produce an approximation of the data.

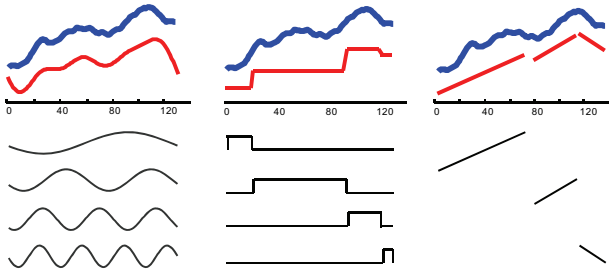

Figure 6. A time series T shown in bold/blue and three different models of it shown in fine/red: from left to right: DFT, APCA, and PLA.

As we apply our ideas to each representation, we must be careful to correctly "charge" each model for its approximation level. For example, each APCA segment requires two numbers, to encode its mean value and its length. However, PLA segments require three numbers, mean value, segment length and slope. Each DFT coefficient requires two numbers to encode the amplitude and phase of each sine wave, however, because of the complex conjugate property, we get a "free" coefficient for each one we record [2][5]. In previous comparisons of the indexing performance of various time series representations, many authors [9] have given an unfair advantage to one representation by the counting cost to represent an approximation incorrectly. The ideas in this work do explicitly assume a fair comparison. Fortunately, the community seems more aware of this problem in recent years [2] [16].

In the next section we give both the generic version of the MDL model discovery for time series algorithm, and a concrete instantiations for APCA. Other two instantiations for DFT and PLA are explained in [28].

B. GENERIC MDL FOR TIME SERIES ALGORITHM

In the last section, we use a toy example for demonstrating how to compute the reduced description length of a time series with competing hypothesis. In this section, we will show a detailed generic version of our algorithm, and then explain our algorithm in detail for the three most commonly used time series representations.

Our algorithm can not only discover the intrinsic cardinality and dimensionality of an input time series, but can also be used to find the right model or data representation for the given time series. TABLE I shows a high-level view of our algorithm for discovering the best model, cardinality, and dimensionality which will minimize the total number of bits required to store the input time series.

Because MDL is a heart of our algorithm, the first step in our algorithm is to quantize a real-value time series into

a discrete-value (but still fine-grained) time series, T (line 1). Next, we consider each model, cardinality, and dimensionality one by one (line 3-5). Then a hypothesis H is created based on the selected model and parameters (line 6). For example, a hypothesis H shown in Figure 5 is created when the model M=APCA, cardinality c=16, and dimensionality d=2; note that, in that case, the length of input time series was m=24.

The reduced description length defined in Definition 5 is then calculated (line 7), and our algorithm returns the model and parameters which minimized the reduced description length for encoding T (line 8-13).

TABLE I. GENERIC MDL FOR TIME SERIES ALGORITHM

Input: *TS* : time series
Output: *best_model* : best model
best_card : best cardinality
best_dim : best dimensionality

```
1   T = Discretization(TS)
2   bsf = ∞
3   for all M in {APCA,PLA,DFT}
4     for all cardinality c
5       for all dimensionality d
6         H = ModelRespresentation(T,M,c,d)
7         toal_cost = DL(H)+ DL(T|H)
8         if (bsf > toal_cost)
9           bsf = toal_cost
10          best_model = M
11          best_card  = c
12          best_dim   = d
13        end if
14      end for
15    end for
16  end for
```

For concreteness we will now consider specific versions of our generic algorithm. See [28] for more details.

C. ADAPTIVE PIECEWISE CONSTANT APPROXIMATION

As we have seen in a previous section, an APCA model is simple; it contains only constant lines. The pseudo code for APCA shown in TABLE II is very similar to the generic algorithm. First of all, we do quantization on the input time series (line 1). Then, we evaluate all cardinalities from 2 to 256 and dimensionalities from 2 to the maximum possible number, which is a half of the length of input time series TS (line 3-4).

TABLE II. OUR ALGORITHM SPECIFIC TO APCA

Algorithm: IntrinsicDiscovery for APCA

```
1   T = Discretization(TS)
2   bsf = ∞
3   for c = 2 to 256
4     for d = 2 to m/2
5       H = APCA(T,c,d)
6       model_cost =d*log2(c)+(d-1)*log2(m)
7       total_cost = model_cost + DL(T-H)
8       if (bsf > total_cost)
9         bsf = total_cost
10        best_card = c
11        best_dim  = d
12      end if
13    end for
14  end for
```

Note that if the dimensionality was more than $m/2$, some segments will contain only one point. Then, a hypothesis H is created using the values of cardinality c and dimensionality d, as shown in Figure 5 when c=16 and d=2. The model contains d constant segments so the model cost is the number of bits required for storing d constant numbers, and d-1 pointers to indicate the offset of the end of each segment (line 6). The difference between T and H is also required to rebuild T. The correction cost (Definition 5) is computed; then the reduced description length is the combination of the model cost and the correction cost (line 7). Finally, the hypothesis which minimized this value is returned as an output of the algorithm (line 8-13).

The algorithms for Piecewise Linear Approximation and DFT are similar and are relegated to the extended version of this paper [28].

IV. EXPERIMENTAL EVALUATION

To ensure that our experiments are *easily* reproducible, we have built a website which contains all data and code, together with the raw spreadsheets for the results [28]. In addition this website contains additional experiments that are omitted here for brevity.

A. AN EXAMPLE APPLICATION IN PHYSIOLOGY

The *Muscle* dataset studied by Mörchen and Ultsch [15] describes the muscle activation of a professional inline speed skater. The authors calculated the muscle activation from the original EMG (Electromyography) measurements by taking the logarithm of the energy derived from a wavelet analysis. Figure 7.*top* shows an excerpt. At first glance it seems to have two states, which correspond to our (perhaps) naive intuitions about skating and muscle physiology.

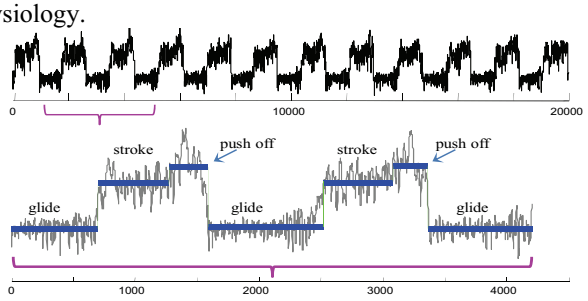

Figure 7. *top*) An excerpt from the Muscle dataset. *bottom*) A zoomed-in section of the Muscle dataset which had its model, dimensionality and cardinality set by MDL.

We can test this binary assumption by using MDL to find the model, dimensionality and cardinality. The results for model and dimensionality are objectively correct, as we might have expected given the results in the previous section, but the results for cardinality, shown in Figure 8.*left* are worth examining.

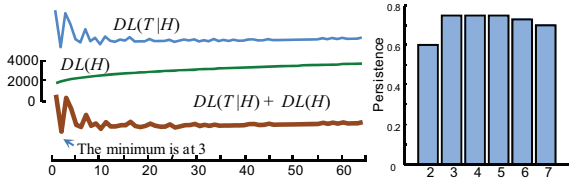

Figure 8. *left*) The description length of the muscle activation time series is minimized with a cardinality of three, which is the correct answer. *right*) The PERSIST algorithm, [15] predicts a value of four.

Our MDL method suggests a cardinality of three. Glancing back at Figure 7.*bottom* shows why. At the end of the *stroke* there is an additional level corresponding to an additional *push-off* by the athlete. This feature was noted by physiologists that worked with Mörchen and Ultsch [15]. However, their algorithm weakly predicts a value of four[4]. Here once again we find the MDL can beat rival approaches, even though the rival approach attempted the most favorable parameter tuning.

V. CONCLUSIONS

We have shown that a simple methodology based on MDL can robustly specify the intrinsic model, cardinality and dimensionality of time series data from a wide variety of domains. Our method has significant advantages over revival methods in that it is more general and is essentially parameter-free. We have further shown applications of our ideas to resource-limited classification and anomaly detection. We have given away all our (admittedly very simple) code and datasets so that others can confirm and build on our results [28].

ACKNOWLEDGEMENTS AND NOTES

This project was supported by the Department of the United States Air Force, Air Force Research Laboratory under Contract FA8750-10-C-0160, and by NSF grants 0803410/ 0808770. The first two authors contributed equally and did the bulk of the work, and should be consider *joint* first authors.

REFERENCES

[1] I. Assent, R. Krieger, F. Afschari, and T. Seidl. The TS-Tree: Efficient Time Series Search and Retrieval. *EDBT*, 2008.

[2] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh. *i*SAX 2.0: Indexing and Mining One Billion Time Series, *International Conference on Data Mining.* 2010.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey, *ACM* Comput. Surv. 41, 3, 2009.

[4] S. De Rooij and P. Vitányi. Approximating Rate-Distortion Graphs of Individual Data: Experiments in Lossy Compression and Denoising. *IEEE Trans' on Computers.*

[5] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *VLDB,* pp. 1542-1552, 2008.

[6] S.C. Evans et al. MicroRNA target detection and analysis for genes related to breast cancer using MDL compress. *EURASIP J. Bioinform. Syst. Biol.,* pp. 1-16, 2007.

[7] P.D. Grünwald, I.J. Myung, and M.A. Pitt, Advances in Minimum Description Length: Theory and Applications, *MIT Press,* 2005.

[8] I. Jonyer, L. B. Holder, and D. J. Cook, Attribute-Value Selection Based on Minimum Description Length. *International Conference on Artificial Intelligence*, 2004.

[9] E. Keogh and M. J. Pazzani, A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases. *PAKDD,* pp.122-133, 2000.

[10] E. Keogh, Q. Zhu, B. Hu, Y. Hao , X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification /Clustering Homepage: *www.cs.ucr.edu/~eamonn/time_series_data*/, 2011.

[11] P. Kontkanen and P. Myllym. MDL histogram density estimation. *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*, 2007.

[12] M. Li and P. Vitanyi. An Introduction to Kolmogorov Complexity and Its Applications. *2nd Ed, Springer*, 1997.

[13] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Journal of DMKD* 15, 2, pp. 107-144, 2007.

[14] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. *In Proc. of 2nd Workshop on Temporal Data Mining,* 2002.

[15] F. Mörchen and A. Ultsch. Optimizing time series discretization for knowledge discovery. *KDD,* 2005.

[16] T. Palpanas, M. Vlachos, E. Keogh, and D. Gunopulos. Streaming Time Series Summarization Using User-Defined Amnesic Functions. *IEEE Trans. Knowl. Data Eng.* 20, 7, pp. 992-1006, 2008.

[17] S. Papadimitriou, A. Gionis, P. Tsaparas, A. Väisänen , H. Mannila and C. Faloutsos . Parameter-free spatial data mining using MDL. *ICDM*, 2005

[18] E.P.D. Pednault. Some Experiments in Applying Inductive Inference Principles to Surface Reconstruction. *IJCAI*, pp. 1603-1609, 1989.

[19] P. Protopapas, J. M. Giammarco, L. Faccioli, M. F. Struble, R. Dave, and C. Alcock. Finding outlier light-curves in catalogs of periodic variable stars. *Monthly Notices of the Royal Astronomical Society,* 369, pp. 677–696, 2006.

[20] J. Rissanen. Stochastic Complexity in Statistical Inquiry. World Scientific, Singapore, 1989.

[21] J. Rissanen, T. Speed and B. Yu. Density estimation by stochastic complexity. *IEEE Trans. On Information Theory,* 38, 315-323, 1992.

[22] D. Sart, A. Mueen, W. Najjar, V. Niennattrakul, and E. Keogh. Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs. *IEEE International Conference on Data Mining,* pp. 1001- 1006, 2010.

[23] A. Vahdatpour and M. Sarrafzadeh. Unsupervised Discovery of Abnormal Activity Occurrences in Multi-dimensional Time Series, with Applications in Wearable Systems. *SIAM International Conference on Data Mining*, 2010.

[24] N. Vereshchagin and P. Vitanyi. Rate distortion and denoising of individual data using Kolmogorov complexity. IEEE Trans. *Information Theory* 56, 7, pp. 3438–3454, 2010.

[25] C.S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal* 11, 2, pp.185-194, 1968.

[26] D. Yankov, E. Keogh, and U. Rebbapragada. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. *Knowl. Inf. Syst.* 17, 2, pp. 241-262, 2008.

[27] H.J. Zwally and P. Gloersen. Passive microwave images of the polar regions and research applications. *Polar Records* 18, pp. 431-450, 1977.

[28] Project URL: *www.cs.ucr.edu/~bhu002/MDL/MDL.html* This URL contains all data and code used in this paper. In addition it contains many additional experiments omitted for brevity.

[4] The values for $k = 3$, 4 or 5 do not differ by more than 1%.