Diagnosing similarity of oscillation trends in time series

Leonardo E. Mariote, Claudia Bauzer Medeiros, Ricardo da S. Torres
leonardo.mariote@students.ic.unicamp.br, {cmbm,rtorres}@ic.unicamp.br
Institute of Computing, University of Campinas - CP6176
Campinas, SP, CEP 13084-851, Brazil

Abstract

Sensor networks have increased the amount and variety of temporal data available, requiring the definition of new techniques for data mining. Related research typically addresses the problems of indexing, clustering, classification, summarization, and anomaly detection. They present many ways for describing and comparing time series, but they focus on their values. This paper concentrates on a new aspect - that of describing oscillation patterns. It presents a technique for time series similarity search, based on multiple temporal scales, defining a descriptor that uses the angular coefficients from a linear segmentation of the curve that represents the evolution of the analyzed series. Preliminary experiments with real datasets showed that our approach correctly characterizes the oscillation of time series.

1 Introduction

An increasing number of applications periodically collect information from some environment, creating and analyzing temporal data, e.g., rainfall or temperature. Sensor networks [12, 2, 10] are prime data providers, e.g. in biodiversity systems or urban traffic analysis. The use of this huge amount of data requires the definition of techniques for mining these data. Similarity search is a basic operation, used in the majority of the mining operations, being thus an important problem to be investigated.

Several research initiatives have addressed this problem, using different methods for compacting, indexing, and comparing time series. Some of these methods are specific for comparing entire time series, while others can work with subsequences [3, 13, 5]. The problem of searching for subsequences is more difficult, since the method should be able to align a subsequence query with the right instant in time when some time series has the desired behavior.

In the rest of this paper we will use the following terminology. Consider a database containing several time series.

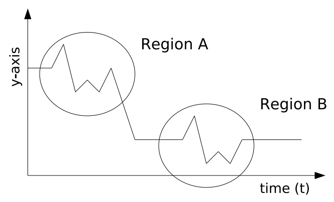

Figure 1: The problem of displacement in the values axis.

The goal of similarity search is to find, in this database, all series which are similar to another series provided as input. We call this input series a *query series*.

Similarity mechanisms are based on the notions of *feature descriptor* and *distance function*. A feature descriptor is typically a set of values, organized according to some structure (e.g., vector), that summarizes the evolution of values in a time series. Two series are considered similar if the distance between their descriptors is below some threshold. Challenges in this area include designing functions that will extract the appropriate descriptors, and determining adequate distance functions. The pair $<$ feature descriptor, distance function $>$ is called a *data descriptor*. Threshold values, and adequate distance functions must be validated by experts for each application domain.

This paper addresses the problem of similarity search from a different perspective, motivated by user needs. It presents a new data descriptor, based on describing patterns of data evolution and their oscillation trends, instead of concentrating on actual data values. Our descriptor, TIDES (TIme series oscillation DEScriptor), allows summarization and comparison of time series oscillations for different temporal scales. Figure 1 shows an example of the kind of concern met by our work. The curve in regions A and B evolves in the same way, but these two sub-series are displaced relative to the y-axis. If they were submitted to a conventional descriptor, they would probably be considered non similar.

The main contribution of this paper is the proposal of a new kind of time series similarity evaluation, based on the

TIDES multiscale descriptor.

2 Related work

Temporal data mining has been subject to extensive research. Several projects address the problems of compacting, comparing, and indexing time series. A family of feature extraction functions is based on analyzing spectral information present in the curve. Examples include the use of Fourier Transform [3], *Singular Value Decomposition (SVD)*, and *Wavelet Transformations*. Although these solutions are able to represent behavior with a small number of coefficients, they require the curve to be locally stationary in time. Furthermore, there can be problems when the query involves some kind of scale or displacement processing.

Other kinds of feature extraction functions have been studied, like *Piecewise Aggregate Approximation (PAA)* [9] and *Adaptive Piecewise Constant Approximation (APCA)* [4]. The technique proposed in [9] defines a simple way to extract features from a time series. At regular times it extracts data samples (e.g., the average of the period), and represents an interval with this value. According to the interval used, it can produce a satisfactory representation of the curve. The ACPA approach works in a similar way, but it adjusts the length of the segments to the variation of data: the length of segments, and the number of segments to represent a period can vary. The difficulty of APCA lies in indexing, since the number of elements that represents one series is variable. To address this problem, [4] proposes a solution based on two distance measures.

There are other techniques to characterize the evolution of a series. Some are based on linear segmentation [7, 5]. In [7], for example, the curve that represents the evolution of a time series is decomposed into a sequence of linear segments. After this decomposition, many measures can be obtained and manipulated. According to [7], techniques that use geometrical characteristics for the curve can produce good results.

Another approach that uses linear segmentation is proposed in [5]. In this method, a user can assign a weight to each segment. These weights are stored with the endpoints of the segments, and are used in the distance function. This enables the user to choose which segments are more important than others in a similarity query. Although this technique gives more control to the user, it does not address the scale and displacement problems.

The second step in similarity search is the definition of an appropriate distance function – such as Euclidean (L2) and Manhattan (L1). The distance used in a technique is tied to the feature vector defined. However, for some feature vectors, these distances are not appropriate [6, 11]. Other distance functions address the problems of scale and displacement in the time axis, like the *Time Warping Distance* [6, 11]. This kind of approach was considered unfit for indexing in the literature [1]. Dynamic Time Warping (DTW) [6] can solve this problem.

The simple DTW approach, however, does not address the problem of displacements on the y-axis. Most of the descriptors proposed in the literature compare pairwise values, and do not consider its oscillation behavior (that characterizes their evolution). Two series of values S_1 and S_2 are similar but relatively displaced in the value axis if $S_1 =< s_{1,1}, s_{1,2}, ..., s_{1,n} >$ and $S_2 =< s_{2,1}, s_{2,2}, ..., s_{2,n} >$, and $s_{1,i} - s_{2,i} \approx \delta, \forall i \in \{1..n\}$ (see Figure 1).

As will be seen, our approach to overcome this problem is to use the angular coefficient of the linear segments extracted for the series analyzed. We can describe how the time series (and not their values) evolve along time. This should be matched with the DTW techniques, thus simultaneously providing solutions for displacements in the time axis.

Another kind of similarity approach is the symbolic representation of the time series [9, 6]. In this approach, the time series is somehow converted into a sequence of symbols. The conversion is associated with the technique used for characterizing the corresponding curve. This representation allows the use of text-match algorithms to compare series, which are seen as strings of symbols. To calculate the distance between two series represented in a symbolic way, it is necessary to define a distance metric between the symbols used. Other technique address specific situations – e.g. [13] deals with data that represent periodic phenomenon.

Although these works address important points in similarity search, as far as we know none of them focus on the problem of displacements in the y-axis. Moreover, oscillation patterns, our main concern, are not considered either.

3 The TIDES Descriptor

3.1 Overview

From a high level point of view, the TIDES descriptor is derived as follows. While some researchers use spectral information for the data [3], our work describes the data with a sequence of linear segments, using a symbolic representation for each segment. The information used for representing each segment is its angular coefficient.

Our descriptor is detailed in two basic steps. In the first step, we introduce how a single scale of a series is described. In the second step, we explain how to obtain the multiscale description of the series, and how to merge the information present in all these scales. Our solution combines approaches from APCA and symbolic description.

3.2 Feature vector extraction

3.2.1 Transformation into segments

Our feature descriptor represents the data as a set of linear segments, using a linear segmentation algorithm, the bottom-up approach [8]. Each of the segments has an angular coefficient with respect to the vertical axis. This coefficient represents the oscillation trend of a series in a time interval. In our experiments (Section 4), we used the following ways to frame the segmentation:

- Produce the best representation with k segments;
- Produce the best representation such that the maximum error for any segment does not exceed some threshold e. This threshold is an input parameter for the method, and varies according to the application.

Once the data has been segmented, the angular coefficient of each segment is obtained. This coefficient is processed and stored, as described in the following.

3.2.2 Feature Vector Representation

Our feature vector is based on the notion of angular coefficient to characterize a series' oscillation patterns. An angular coefficient represents the slope of a segment wrt the vertical axis. Consider a series S that has been transformed into n segments $< g_1, g_2, ..., g_n >$. The feature vector is defined as

$$V =<< a_1, l_1 >, < a_2, l_2 >, ..., < a_n, l_n >> \quad (1)$$

where a_i is the angular coefficient of segment g_i and l_i is its length. Figure 2 shows a segmented representation of one time series, and the angular coefficients obtained.

The use of angle values, however, may introduce several problems in similarity computation, and in vector storing (e.g., due to precision issues or overflows). For this reason, we adopted a widespread solution - symbolic representation [9]. In this approach, angular coefficients in one series are submitted to a classification function F_{class}, and are assigned to a class (which is represented by a symbol). These classes are created by clustering functions. Thus,

$$V =<< y_1, l_1 >, ..., < y_n, l_n >> \quad (2)$$

where y_i is the symbol that represents the class assigned to segment g_i.

An angular coefficient can vary between 0^o and 180^o (or 0 and π radians), in relation to the vertical axis. The range of possible values (0 to 180) is partitioned in n_g sets, where n_g is defined by domain experts.

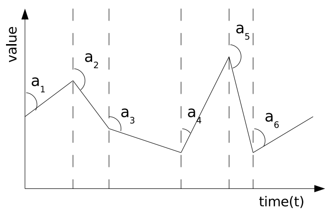

Figure 2: Angular coefficients from one time series.

3.3 Distance function

Once the feature vector is extracted, the next step is to define the distance function.

Symbolic representation supports several kinds of distance functions, e.g., based on dynamic programming [6]. However, in our implementation we used a vertent of the Manhattan Distance Function (L1). To do so, two feature vectors, V_1 and V_2 with different number of segments are normalized into new feature vectors, NV_1 and NV_2, with the same size. Normalization is based on creation of additional (virtual) points to align segments.

The distance between NV_1 and NV_2 is given by:

$$D(NV_1, NV_2) = \sum_{i=1}^{i=n} d(< y_i, l_i >_1, < y_i, l_i >_2) \quad (3)$$

where $d(< y_i, l_i >_1, < y_i, l_i >_2)$ is the distance between the ith (normalized) segment of vectors NV_1 and NV_2. Distance d is computed as:

$$d(< y_i, l_i >_1, < y_i, l_i >_2) = (y_{i,1} - y_{i,2}) * l_i \quad (4)$$

where the distance between any two consecutive symbols is 1 – e.g., $d(n^{th} symbol, (n+4)^{th} symbol) = 4$.

3.4 Multiscale Description

So far, our descriptor provides the oscillation behavior of a series that has been approximated by linear segments. This approximation, however, depends on the number of points chosen in the segmentation process. This may not be sufficient to describe actual oscillation trends. To solve this, we extend our solution to a multiscale approach instead of representing a series by one V, we describe it using multiple V. Each vector corresponds to a segmentation obtained with a different number of points.

The multiscale feature vector of TIDES is thus:
$V =< V_1, V_2, ...V_k, ...V_n >$,
where k is the number of segments used to approximate the series in a given time granularity. In particular, V_1 uses the endpoints of the series, and corresponds to a single segment. The value of n is defined by domain experts.

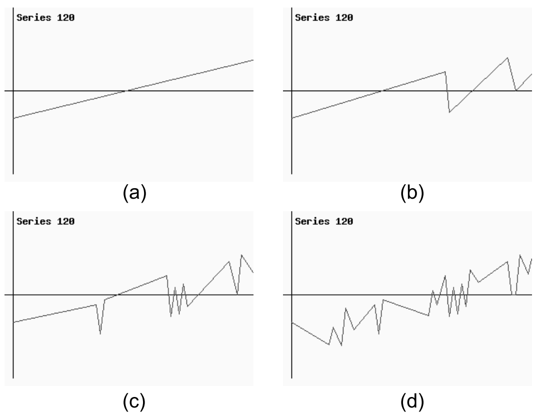

Figure 3: Four different representations for the series 120, using one (a), five (b), thirteen (c) and thirty (d) segments.

Figure 3 illustrates the idea, with the segmentation of a series extracted from the Synthetic Control dataset [1], identified by number 120. It shows how to approximate the series with one (a), five (b), thirteen (c), and thirty (d) segments. Each scale corresponds to a distinct level of detail – e.g., with one segment it is possible to see the rough oscillation tendency (ascending, descending, stationary).

Intuitively, the multiscale approach allows users to analyze a series under distinct granularities. For instance, suppose two series are similar at levels 1 and 2, and dissimilar afterwards. This means that only for very coarse time granularities they show the same tendency. In another example, two series may be similar in a fine granularity say, n segments but dissimilar at levels 1 and 2. In this case, typically one curve is ascending (in terms of the y axis) and the other descending, but oscillations follow the same behavior.

Thus, the descriptor is composed of an array of series of symbols. The comparison between two series is done in two basic steps. First of all, the sequences of symbols relative to each scale are compared, using the distance function defined. As an example, suppose two series, $ms_1 = (s_{1,1}, s_{1,2}, ..., s_{1,m})$ and $ms_2 = (s_{2,1}, s_{2,2}, ..., s_{2,m})$, where ms represents a multiscale series, and $s_{1,i}$ represents a sequence of l_i segments that encodes the sequence s_1. To compare these series, we should compare $s_{1,1}$ with $s_{2,1}$, $s_{1,2}$ with $s_{2,2}$, and so on.

The distance function for the multiscale case is based on computing Equation 3 repeatedly for each scale considered. Hence, the distance between two normalized multiscale feature vectors $VA =< VA_1...VA_n >$ and $VB =< VB_1...VB_n >$ is based on the multiscale distance vector

$$DM =<< D(VA_1, VB_1) > ... < D(VA_n, VB_n) >> \quad (5)$$

[1]www.cs.ucr.edu/ eamonn/timeseriesdata/

Distance vector DM can subsequently be used to compute several kinds of distance functions – e.g., sum or Euclidean distance. The distance vector itself is a good indicator of the scale(s) for which two series can be considered most similar, as illustrated in the next section.

4 Experimental analysis

4.1 Datasets used

Two kinds of time series were used. The first, the Synthetic Control dataset series, were used to validate two important TIDES features: y-axis invariance and multiscale analysis. This data set has 600 series (grouping its training and test sets), with 60 points each.

Our second test set used real data – monthly average maximum and minimum temperature readings since 1961 – from five Brazilian cities in São Paulo state (Campinas, Jaboticabal, Sao Carlos, Sorocaba and Taubate). We used this data to construct a set of 1336 series of 48 points each, and classified them according to the month where the series started – e.g., all series beginning in january of some year belong to class 1. This second test set was used to compare the precision of TIDES against Linear Scan. We selected this kind of phenomenon because temperature oscillation is similar in a given geographical area, in the same season, even though values themselves may vary.

4.2 Invariance to noise in the y-axis

Figure 1 illustrates the problem of noise in the y-axis, where similarity between the two sub-series is not captured by other methods. This section comments on experiments conducted to evaluate TIDES when comparing series that present this sort of relative behavior. In this case, we used a simpler (single scale) version of the descriptor.

We introduced some noise in the dataset creating, for all the time series, a new one with a value displacement in every point. As an example of that, suppose $S = (s_1, s_2, ..., s_n)$ a time series. We created a series $S' = (s_1 + \delta, s_2 + \delta, ..., s_n + \delta)$. In the expanded database, S' should be returned as one of the most similar series when the query series is S.

As expected, the artificial series S' were always returned as the most similar series (after the S series itself). We repeated this experiment for all the original 600 series. It shows TIDES immunity to displacement in the y-axis.

4.3 Introducing multiscale description

One of the advantages of TIDES is that it allows similarity evaluation at different granularities. This is adequate for

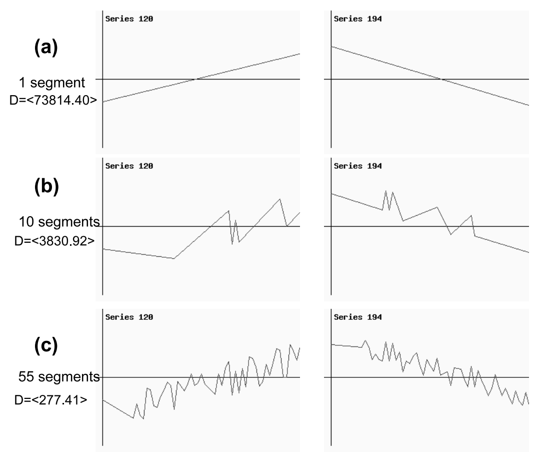

Figure 4: Multiscale comparison between series 120 and 194, using three different granularities.

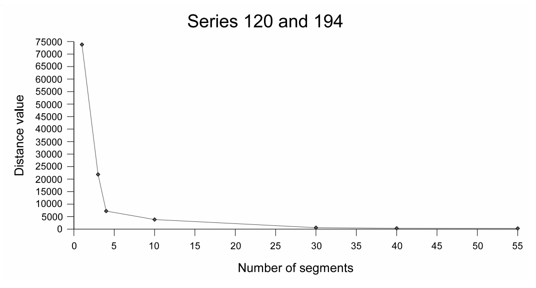

Figure 5: Multiscale Distance: series 120 and 194.

cases in which users want to examine a given phenomenon from distinct scale perspectives – e.g., in economy.

Figures 4 and 5 show the result of a multiscale comparison using TIDES, for two series in the Synthetic Control dataset – 120 and 194. In more detail, Figure 4 shows the curves that describe these series for 1 segment (NV_1), 10 segments (NV_{10}), and 55 segments (NV_{55}), respectively Figures 4(a), 4(b) and 4(c). The figure also shows that, for the 1-segment representation, the distance computed was 73814,4, whereas for a 55-segment it went down to 277,4.

Suppose the similarity threshold defined by application domain experts is 300 – i.e., D(NV_{120}, NV_{194}) $<=$ 300. Then, the results obtained by TIDES for these three scales is that they are 1-dissimilar, 10-dissimilar, but 55-similar.

Figure 5 shows a curve that plots the distinct distance values between the same two series (120 and 194), for several scales. Here, at lower granularities they have very different oscillation tendencies, but as we “zoom into” more detailed scales they present similar oscillation behavior.

Figure 6 helps understanding our descriptor. It shows the sequence of angular coefficients of these two series, with

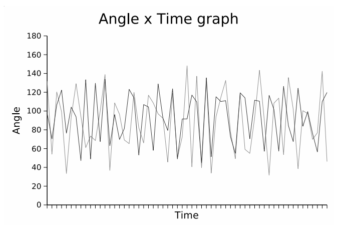

Figure 6: Angle variation: series 120 and 194.

time, for the smallest granularity (NV_{55}). One can see that their oscillation patterns are very close, if one considers a larger number of segments.

4.4 Comparison with Linear Scan

We finish our evaluation with a comparison of single granularity TIDES and Linear Scan (a point-to-point comparison) descriptors, which illustrates their differences, using our temperature time series dataset. All the series had 47 segments ($n = 47$), and we used $n_g = 100$. Two series were considered to be similar if they started in the same (or neighboring) months.

In this experiment, we conducted two kinds of tests. The first computed the k nearest neighbors of a query series Q, for the two descriptors – for $k = \{30, 50, 70, 100\}$. Precision was computed based on the number of similar series found in each k-NN query. This first test used all 1336 series in the set as query series.

The second test was also based on k-NN computation, for the same values of k, but in this case the result eliminated series originating from the same city as Q. For instance, if Q concerned the city of Campinas, then the result only contained series from all other 4 cities. The goal of this second test was to increase the influence of oscillation in the comparison (and not just the values).

To clarify, Sao Carlos is always warmer than Campinas, because of their geographical characteristics. In a given summer, each city shows similar patterns of temperature oscillation, but values are distinct. Linear scan will first select all series from Campinas, at summer time, considering them to be closer to each other than any series from Sao Carlos. TIDES, on the other hand, will prioritize similar oscillation trends between Sao Carlos and Campinas in the same season, and consider them to be more similar than two series from Campinas in different years.

Columns TEST1 and TEST2 of table 1 respectively present the percentage a of correct answers for the first and second tests, where $a = \frac{similarseries}{k} * 100$.

The table shows that TIDES is more appropriate than Linear Scan to describe oscillations. In the TIDES column, the % value indicates how much TIDES is better wrt Lin-

Table 1: Comparison between TIDES and Linear Scan.

k	TEST1		TEST2	
	Lin.Scan	TIDES	Lin.Scan	TIDES
30	90	81(-9%)	78	77(-1%)
50	85	78(-7%)	73	74(1%)
70	78	76(-2%)	69	72(3%)
100	70	73(3%)	62	69(7%)

ear Scan. In TEST2,TIDES rapidly presented better results than Linear Scan, for a relatively small number of series returned (see precision for k = 50). The larger the size of the result, the better TIDES performs (precision for k = 100).

In the first type of test, however, Linear Scan showed good results in the cases where tests were run with small k. It is only when k increased that the oscillation factor started to show its influence, in which case TIDES presented a better performance (precision when k=100).

5 Conclusions and future work

The number of applications that require management of time series is growing every day. This has prompted many kinds of research on time series management and analysis, which frequently rely on similarity search functionalities. This kind of search is generally centered on comparing evolution patterns of data values.

Our work contributes to this effort, from another perspective, considering needs of another nature. It proposes a multiscale descriptor – TIDES – whose purpose is to characterize series' oscillation behavior. We point out two advantages of using TIDES for this kind of characteristics. First, it is immune to series displacement along the y-axis. Second, since it is multiscale, it can be customized to distinct application domains, where oscillation similarity is a matter of time granularity. Experiments conducted show that TIDES serves its purpose, where another frequently used descriptor, based on linear scan of the series, does not.

Future and ongoing work involve several activities. One issue is to develop an evaluation methodology, involving user interaction, with help of experts in interface design, to tune the scale threshold. Another is to perform more experiments using large series datasets with real data.

Acknowledgments This work was partially funded by CPqD Foundation, CAPES, FAPESP, CNPq grants and CNPq projects WebMAPS and RPG.

References

[1] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *VLDB 95*, pages 490–501, 1995.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, March 2002.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[4] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD '01*, pages 151–162, 2001.

[5] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *KDD'98*, pages 239–241, 1998.

[6] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, 2005.

[7] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *3rd KDDM Conference*, pages 24–30, 1997.

[8] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *IEEE ICDM '01*, pages 289–296, 2001.

[9] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *ACM DMKD '03*, pages 2–11, 2003.

[10] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM WSNA '02*, pages 88–97, 2002.

[11] S. Park, D. Lee, and W. W. Chu. Fast retrieval of similar subsequences in long sequence databases. In *KDEX '99*, page 60, 1999.

[12] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proceedings of the First European Workshop on Sensor Networks (EWSN)*, Jan. 2004.

[13] H. Wu, B. Salzberg, G. C. Sharp, S. B. Jiang, H. Shirato, and D. Kaeli. Subsequence matching on structured time series data. In *SIGMOD '05*, pages 682–693, 2005.