Data Min Knowl Disc
DOI 10.1007/s10618-012-0250-5

Experimental comparison of representation methods and distance measures for time series data

Xiaoyue Wang · Abdullah Mueen · Hui Ding · Goce Trajcevski · Peter Scheuermann · Eamonn Keogh

Received: 26 April 2010 / Accepted: 12 January 2012

Abstract The previous decade has brought a remarkable increase of the interest in applications that deal with querying and mining of time series data. Many of the research efforts in this context have focused on introducing new representation methods for dimensionality reduction or novel similarity measures for the underlying data. In the vast majority of cases, each individual work introducing a particular method has made specific claims and, aside from the occasional theoretical justifications, provided quantitative experimental observations. However, for the most part, the comparative aspects of these experiments were too narrowly focused on demonstrating the benefits of the proposed methods over some of the previously introduced ones. In order to provide a comprehensive validation, we conducted an extensive

Responsible editor: Geoffrey I. Webb.

Research supported by NSF awards 0803410 and 0808770, NSF-CNS grant 0910952.

X. Wang (✉) · A. Mueen · E. Keogh
University of California Riverside, Riverside, CA, USA
e-mail: xwang@cs.ucr.edu

A. Mueen
e-mail: mueen@cs.ucr.edu

E. Keogh
e-mail: eamonn@cs.ucr.edu

H. Ding · G. Trajcevski · P. Scheuermann
Northwestern University, Evanston, IL, USA
e-mail: hdi117@eecs.northwestern.edu

G. Trajcevski
e-mail: goce@eecs.northwestern.edu

P. Scheuermann
e-mail: peters@eecs.northwestern.edu

Published online: 10 February 2012

experimental study re-implementing eight different time series representations and nine similarity measures and their variants, and testing their effectiveness on 38 time series data sets from a wide variety of application domains. In this article, we give an overview of these different techniques and present our comparative experimental findings regarding their effectiveness. In addition to providing a unified validation of some of the existing achievements, our experiments also indicate that, in some cases, certain claims in the literature may be unduly optimistic.

Keywords Time series · Representation · Distance measure · Experimental comparison

1 Introduction

Time series data are being generated at an unprecedented scale and rate from almost every application domain, e.g., daily fluctuations of stock market, traces of dynamic processes and scientific experiments, medical and biological experimental observations, various readings obtained from sensor networks, position updates of moving objects in location-based services, etc. As a consequence, in the last decade there has been a dramatically increasing amount of interest in querying and mining such data which, in turn, resulted in a large amount of work introducing new methodologies for indexing, classification, clustering and approximation of time series (Faloutsos et al. 1994; Han and Kamber 2005; Keogh 2006).

Two main goals of managing time series data are the *effectiveness* and the *efficiency*, and the two key aspects towards achieving them are: (1) *representation methods*, and (2) *similarity measures*. Time series are essentially *high dimensional* data (Han and Kamber 2005) and working directly with such data in its raw format is very expensive in terms of both processing and storage cost. It is thus highly desirable to develop representation techniques that can reduce the dimensionality of time series, while still preserving the fundamental characteristics of a particular data set. In addition, unlike canonical data types, e.g., nominal/categorical or ordinal variables (Olofsson 2005), where the distance definition between two values is usually fairly straightforward, the *distance* between time series needs to be carefully defined in order to properly capture the semantics and reflect the underlying (dis)similarity of such data. This is particularly desirable for similarity-based retrieval, classification, clustering and other querying and mining tasks over time series data (Han and Kamber 2005).

Many techniques have been proposed for representing time series with reduced dimensionality, for example: *Discrete Fourier Transformation* (DFT) (Faloutsos et al. 1994), *Single Value Decomposition* (SVD) (Faloutsos et al. 1994), *Discrete Cosine Transformation* (DCT) (Korn et al. 1997), *Discrete Wavelet Transformation* (DWT) (Chan and Fu 1999), *Piecewise Aggregate Approximation* (PAA) (Keogh et al. 2001b), *Adaptive Piecewise Constant Approximation* (APCA) (Keogh et al. 2001a), *Chebyshev polynomials* (CHEB) (Cai and Ng 2004), *Symbolic Aggregate approXimation* (SAX) (Lin et al. 2007) and *Indexable Piecewise Linear Approximation* (IPLA) (Chen et al. 2007a). In conjunction with these techniques, there are over a dozen distance measures used for evaluating similarity of time series presented in

the literature, e.g., *Euclidean distance* (ED) (Faloutsos et al. 1994), *Dynamic Time Warping* (DTW) (Berndt and Clifford 1994; Keogh and Ratanamahatana 2005), distance based on *Longest Common Subsequence* (LCSS) (Vlachos et al. 2002), *Edit Distance with Real Penalty* (ERP) (Chen and Ng 2004), *Edit Distance on Real sequence* (EDR) (Chen et al. 2005a), DISSIM (Frentzos et al. 2007), *Sequence Weighted Alignment model* (Swale) (Morse and Patel 2007), *Spatial Assembling Distance* (SpADe) (Chen et al. 2007b) and similarity search based on *Threshold Queries* (TQuEST) (Aßfalg et al. 2006). Quite a few of these works, as well as some of their extensions, have been widely cited in the literature and applied to facilitate query processing and data mining of time series data.

Given the multitude of competitive techniques, we believe that there is a strong need for a comprehensive comparison which, in addition to providing a foundation for benchmarks, may also reveal certain omissions in the comparative observations reported in the individual works. In the common case, every newly-introduced representation method or distance measure has claimed a particular superiority over some of the existing results. However, it has been demonstrated that some empirical evaluations may have been inadequate (Keogh and Kasetty 2003) and, worse yet, some of the results may be contradictory. For example, one paper shows the result that "*wavelets outperform the DFT*" (Popivanov and Miller 2002), another suggests that "*DFT filtering performance is superior to DWT*" (Kawagoe and Ueda 2002) and yet another shows the result: "*DFT-based and DWT-based techniques yield comparable results*" (Wu et al. 2000). Clearly, not all of these experimental results can generalize simultaneously. An important consequence of this observation is that there is a risk that such (or similar) results may not only cause a confusion to newcomers and practitioners in the field, but also cause a waste of time and research efforts due to assumptions based on results that do not generalize.

Motivated by these observations, we have conducted the most extensive set of time series experiments to-date, re-evaluating the state-of-the-art representation methods and similarity measures for time series that appeared in high quality conferences and journals. Specifically, as the main contributions of this work, we have:

- Re-implemented 8 different representation methods for time series, and compared their *pruning power* over various time series data sets.
- Re-implemented 9 different similarity measures and their variants, and compared their effectiveness using 38 real world data sets from highly diverse application domains.
- Provided certain analysis and conclusions based on the experimental observations.

We note that all of our source code implementations and the data sets are publicly available on our website (http://www.ece.northwestern.edu/~hdi117/tsim.htm).

The rest of this paper is organized as follows. Section 2 reviews the concept of time series, and gives an overview of the definitions of different representation techniques and similarity measures investigated in this work. Sections 3 and 4 present the main contribution of this work—the results of the extensive experimental evaluations of different representation methods and similarity measures, respectively. In Sect. 5, we summarize some of the myths and possible misunderstandings about DTW. Section 6 concludes the paper and discusses possible future extensions of the work.

2 Preliminaries

Typically, most of the existing works on time series assume that *time* is discrete. For simplicity and without any loss of generality, we make the same assumption here. Formally, a *time series* data is defined as a sequence of pairs $T = [(p_1, t_1), (p_2, t_2), \ldots, (p_i, t_i), \ldots, (p_n, t_n)](t_1 < t_2 < \cdots < t_i < \cdots < t_n)$, where each p_i is a data point in a d-dimensional data space, and each t_i is the time stamp at which the corresponding p_i occurs.[1] If the sampling rates of two time series are the same, one can omit the time stamps and consider them as sequences of d-dimensional data points. Such a sequence is called the *raw representation* of the time series. In reality however, sampling rates of time series may be different. Furthermore, some data points of time series may be dampened by noise or even completely missing, which poses additional challenges to the processing of such data. For a given time series, its number of data points n is called its *length*. The portion of a time series between two points p_i and p_j (inclusive) is called a *segment* and is denoted as s_{ij}. In particular, the segment $s_{i(i+1)}$ between two consecutive points is called a *line segment*.

In the following subsections, we briefly review the representation methods and similarity measures studied in this work. We note that this is not intended to be a complete survey of the available techniques and is only intended to provide the necessary background for following and understanding our experimental evaluations.

2.1 Representation methods for time series

There is a plethora of time series representation methods, each of them proposed for the purpose of supporting similarity search and data mining tasks.

A classification of the major techniques, organized in a hierarchical manner, is shown in Fig. 1.

As illustrated, there are two basic categories:

- *Data Adaptive* representations: in this category, a common representation will be chosen for all items in the database that minimizes the global reconstruction error.
- *Non-Data Adaptive* representations: in contrast, these methods consider local properties of the data, and construct an approximate representation accordingly.

For example, Adaptive Piecewise Constant Approximation (APCA, an adaptive technique) transforms each time series by a set of constant value segments of varying lengths such that their individual reconstruction errors are minimal. On the other hand, Piecewise Aggregate Approximation (PAA, a non-adaptive technique), approximates a time series by dividing it into equal-length segments and recording the mean value of the datapoints that fall within the segment. This representation does not adapt to each individual data item thus is less efficient than the adaptive representation.

The representations annotated with an asterisk ($*$) in Fig. 1 have the very desirable property of allowing *lower bounding*. This property, essentially, allows one to define a

[1] We do not differentiate between the *time of occurrence* and the *time of detection* of a particular event in this work (Bennet and Galton 2004) or, to phrase it in a different context—we do not distinguish the *valid time* from the *transaction time* (Tansel et al. 1993).

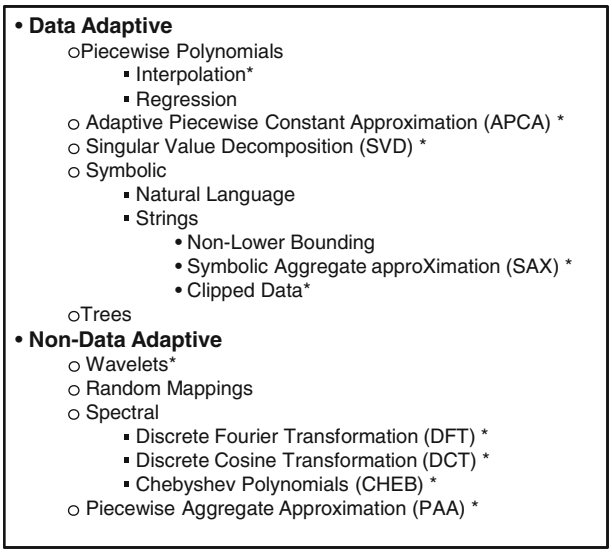

Fig. 1 A hierarchy of representation methods

distance measure that can be applied to the reduced-size (i.e., compressed) representations of the corresponding time series, that is guaranteed to be less than or equal to the true distance which is measured on the raw data. The main benefit of the lower bounding property is that it allows using the respective reduced-size representations to index the data, with a guarantee of *no false negatives* (Faloutsos et al. 1994). The list of representations considered in this study includes (in approximate order of introduction) DFT, DCT, DWT, PAA, APCA, SAX, CHEB and IPLA. The only lower bounding omissions from our experiments below are the eigenvalue analysis techniques such as SVD and PCA (Korn et al. 1997). While such techniques give optimal linear dimensionality reduction, we believe they are untenable for large data sets. For example, while (Steinbach et al. 2003) notes that they can transform 70000 time series in under 10 min, the assumption is that the data is memory resident. However, transforming out-of-core (disk resident) data sets using these methods becomes unfeasible. Note that the available literature seems to agree with us on this point. For (at least) DFT, DWT and PAA, there are more than a dozen projects that use these representations to index over 100,000 objects for query-by-humming (Zhu and Shasha 2003; Karydis et al. 2005), Mo-Cap indexing (Cardle 2004), etc. At the time of writing this article, however, we are unaware of any projects of a similar scale that use SVD.

2.2 Similarity measures for time series

We now give an overview of the 9 similarity measures evaluated in this work which, for convenience, are summarized in Fig. 2.

Given two time series T_1 and T_2, a similarity function $Dist$ calculates the distance between the two time series, denoted by $Dist(T_1, T_2)$. In the following we will refer

- **Lock-step Measure**
  - L_p-norms
    - L_1-norm (Manhattan Distance)
    - L_2-norm (Euclidean Distance)
    - L_{inf}-norm
  - DISSIM
- **Elastic Measure**
  - Dynamic Time Warping (DTW)
  - Edit distance based measure
    - Longest Common SubSequence (LCSS)
    - Edit Sequence on Real Sequence (EDR)
    - Swale
    - Edit Distance with Real Penalty (ERP)
- **Threshold-based Measure**
  - Threshold query based similarity search (TQuEST)
- **Pattern-based Measure**
  - Spatial Assembling Distance (SpADe)

Fig. 2 A summary of similarity measures

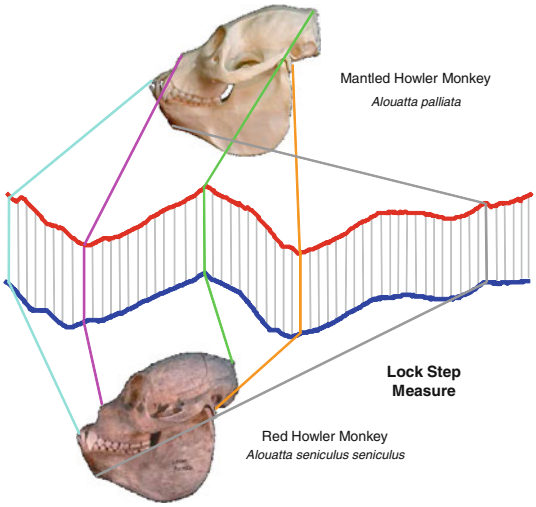

Fig. 3 An illustration of a *Lock Step* measure. Note the "one-to-one" mapping of data points. The distance measure is proportional to the length of the *gray lines*

to distance measures that compare the ith point of one time series to the ith point of another as *lock-step measures* (e.g., Euclidean distance and the other Lp norms), and distance measures that allow comparison of one-to-many points (e.g., DTW) and one-to-many/one-to-none points (e.g., LCSS) as *elastic measures*. Figures 3 through 6 provide illustrations of the corresponding intuitions behind the major classes of distance measures. Note that in every case, the two time series are shown shifted apart in the y-axis for visual clarity, however they would typically be normalized and therefore overlapping (Keogh and Kasetty 2003). Figure 3 shows the intuition behind Lock Step measures, a class which includes the ubiquitous Euclidean distance.

The most straightforward similarity measure for time series is the *Euclidean Distance* (Faloutsos et al. 1994), along with its variants based on the common L_p-norms (Yi and Faloutsos 2000). In particular, in this work we used L_1

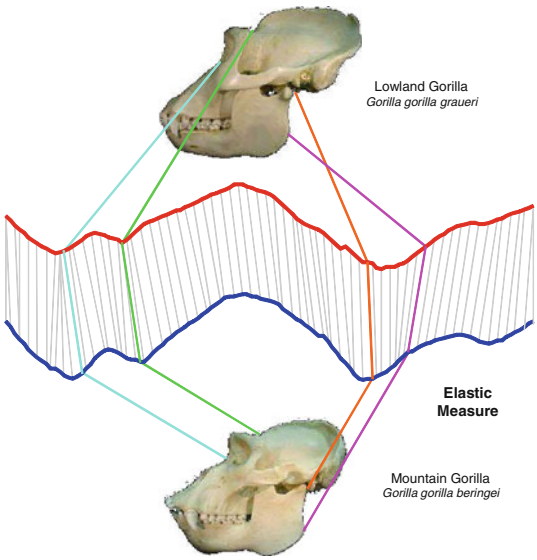

Fig. 4 An illustration of *Elastic* measure. Note that unlike Lock Step measures, here we allow the possibility of "one-to-many" mapping of the data points, but each data point must be matched. The distance measure is proportional to the length of the *gray lines*

(Manhattan), L_2 (Euclidean) and L_∞ (Maximum) norms (cf. Yi and Faloutsos 2000). In the sequel, the terms Euclidean distance and L_2 norm will be used interchangeably. In addition to being relatively straightforward for intuitive understanding, the Euclidean distance and its variants have several other advantages. An important one is that the complexity of evaluating these measures is linear to the length of the time serieses, and they are easy to implement and indexable with any access method and, in addition, they are *parameter-free*. Furthermore, as we will demonstrate, the Euclidean distance is surprisingly competitive with the other, more complex approaches, especially if the size of the training set/database is relatively large. However, since the mapping between the points of two time series is *fixed*, these distance measures are very sensitive to noise and misalignments in time, and are unable to handle *local time shifting*, i.e., similar segments that are out of phase.

The DISSIM distance (Frentzos et al. 2007) aims at computing the similarity of time series with different sampling rates. However, the original similarity function is numerically too difficult to compute, and the authors proposed an approximated distance with a formula for computing the error bound.

Inspired by the need to handle time warping in similarity computation, Berndt and Clifford (Berndt and Clifford 1994) introduced DTW, a classical speech recognition tool, to the data mining community, in order to allow a time series to be "stretched" or "compressed" to provide a better match with another time series. Figure 4 illustrates the intuition behind DTW and other elastic measures.

Several lower bounding measures have been introduced to speed up similarity search using DTW (Yi et al. 1998; Kim et al. 2001; Keogh 2002; Keogh and Ratanamahatana 2005), and it has been shown that the amortized cost for computing DTW on large data sets is linear (Keogh 2002; Keogh and Ratanamahatana 2005). The original

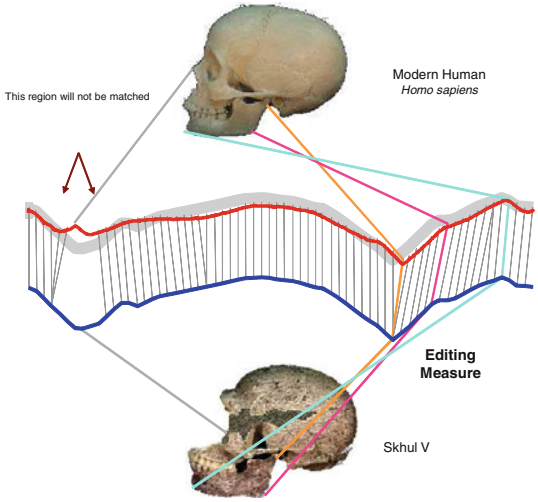

Fig. 5 An illustration of an *Editing* measure. Note that, similarly to the elastic measures, we allow the possibility of "one-to-many" mapping of the data points. However in addition, we also allow the possibility of *not* matching some (one or more) points. The distance measure is proportional to the length of the *gray lines*

DTW distance is also parameter free, however, as has been reported in Keogh and Ratanamahatana (2005), Vlachos et al. (2006) enforcing a *temporal constraint* δ on the warping window size of DTW not only improves its computation efficiency, but also improves its accuracy for measuring time series similarity, as extended warping may introduce pathological matchings between two time series and distort the true similarity. The constraint warping is also utilized for developing the lower-bounding distance (Keogh and Ratanamahatana 2005) as well as for indexing time series based on DTW (Vlachos et al. 2006).

Another group of similarity measures for time series has been developed based on the concept of the *edit distance* for strings. The main intuition behind the Editing measures is visualized in Fig. 5.

The best known example from this category is the LCSS distance, which is based on the *longest common subsequence* model (André-Jönsson and Badal 1997; Vlachos et al. 2002). To adapt the concepts used in matching characters and strings in the settings of time series, a *threshold parameter* ε was introduced, the semantics of which is that two points from two time series are considered to match if their distance is less than ε. The work reported in Vlachos et al. (2002) also took into consideration an additional constraint—the matching of points along the temporal dimension, using a so called *warping threshold* δ. A lower-bounding measure and indexing technique for LCSS were introduced in Vlachos et al. (2006).

EDR (Chen et al. 2005a) is another similarity measure based on the edit distance. Similar to LCSS, EDR also uses a threshold parameter ε, except its role is to quantify the distance between a pair of points to 0 or 1. Unlike LCSS, EDR assigns penalties to the gaps between two matched segments according to the lengths of the gaps.

The ERP distance (Chen and Ng 2004) attempts to combine the merits of both DTW and EDR, by introducing the concept of a *constant reference point* for computing the distance between gaps of two time series. Essentially, if the distance between two

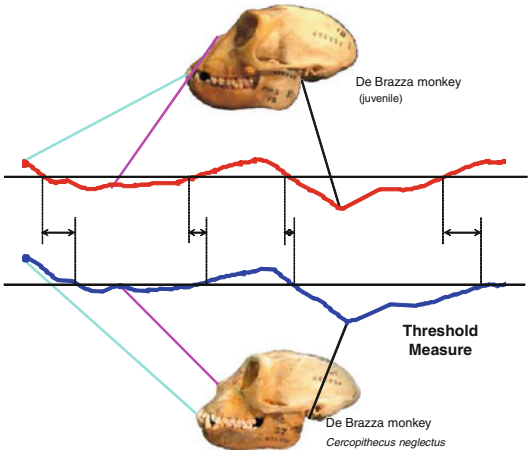

Fig. 6 An illustration of a *Threshold* measure. The distance measure is proportional to the length of the *double-headed arrows*

points is too large, ERP simply uses the distance value between one of those point and the reference point.

Recently, a new approach for computing the edit distance based similarity measures was proposed in Morse and Patel (2007). Whereas traditional tabular dynamic programming was used for computing DTW, LCSS, EDR and ERP, a matching threshold is used to divide the data space into grid cells and, subsequently, matching points are found by hashing. An important feature of the similarity model Swale(cf. Morse and Patel 2007) is that it rewards matching points and penalizes gaps. In addition to the matching threshold ε, Swale requires the tuning of two parameters: the *matching reward* weight r and the *gap penalty* weight p.

The TQuEST distance (Aßfalg et al. 2006) introduced a rather novel approach to computing the similarity measure between time series. The main idea behind TQuEST is that, given a threshold parameter τ, a time series is transformed into a sequence of so-called *threshold-crossing* time intervals, where the points within each time interval have a value greater than τ. Each time interval is then treated as a point in a two dimensional space, where the starting time and ending time constitute the two dimensions. The similarity between two time series is then defined as the Minkowski sum of the two sequences of time interval points (Flato 2000). Figure 6 visually illustrates the intuition behind the threshold measures.

The last approach considered in this work is SpADe (Chen et al. 2007b), which is a pattern-based similarity measure for time series. The key idea behind the presented algorithm is to find out matching segments within the entire time series, called *patterns*, by allowing shifting and scaling in both the temporal and amplitude dimensions. The problem of computing similarity value between time series is then transformed to the one of finding the most similar set of matching patterns. SpADe requires tuning a number of parameters, such as the temporal scale factor, amplitude scale factor, pattern length, sliding step size, etc.

3 Comparison of time series representations

We compare all the major time series representations that have been proposed in the literature, including SAX, DFT, DWT, DCT, PAA, CHEB, APCA and IPLA. We note that all the representation methods studied in this paper allow lower bounding, and any of them can be used to index the Euclidean Distance, the Dynamic Time Warping, and at least some of the other elastic measures. While various subsets of these representations have been compared before, to the best of our knowledge, this is the first attempt to compare all of them together. One obvious question that needs to be considered is what metric should should be used for comparison? We postulate that the wall clock time is a poor choice, because it may be open to an implementation bias (Keogh and Kasetty 2003). Instead, we believe that using the *tightness of lower bounds* (TLB) is a very meaningful measure (Keogh et al. 2001b), and this also appears to be the current consensus in the literature (Cai and Ng 2004; Chen and Ng 2004; Chen et al. 2007a; Keogh 2002, 2006; Keogh et al. 2001a; Keogh and Ratanamahatana 2005; Ratanamahatana and Keogh 2005; Vlachos et al. 2006). Formally, given two time series, T and S, the corresponding TLB is defined as

$$\text{TLB} = \textit{Lower Bound Dist}\,(T, S)/\textit{True Euclidean Dist}(T, S)$$

The advantage of using TLB is twofold:

1. It is a completely *implementation-free* measure, independent of hardware and software choices, and is therefore completely reproducible.
2. It allows a *very accurate prediction* of the indexing performance.

If the value of TLB is zero, then any indexing technique is condemned to retrieving every single time series from the disk. On the other hand, if the value of TLB is one then, after some trivial processing in main memory, we could simply retrieve a single object from the disk and guarantee that we have obtained the true nearest neighbor. Note that, in general, the speedup obtained is non-linear in TLB, that is to say, if one representation has a lower bound that is twice as large as another, we can usually expect a much greater than twofold decrease in the number of disk accesses.

As part of this work, we randomly sampled T and S (with replacement) $1,000$ times for each combination of parameters. We varied the time series length among the values of $\{480, 960, 1,440, 1,920\}$, as well as the number of coefficients per time series available to the dimensionality reduction approach among the values of $\{4, 6, 8, 10\}$ (each coefficient takes 4 bytes). For SAX, we hard coded the cardinality to 256. Figure 7 shows the result of one such experiment with an ECG data set.

At a first glance, the results of this experiment may appear surprising, as they show that there is very little difference between representations, in spite of the apparent results to the contrary in the literature. However, we believe that some of these results may be due to some errors or bias in the experiments. For example, a recent study showed that DFT is much worse than all the other approaches (Chen et al. 2007a), however it appears that the complex conjugate property of DFT was not exploited. As another example, it was suggested that "*it only takes* 4 *to* 6 *Chebyshev coefficients to deliver the same pruning power produced by* 20 *APCA coefficients*"

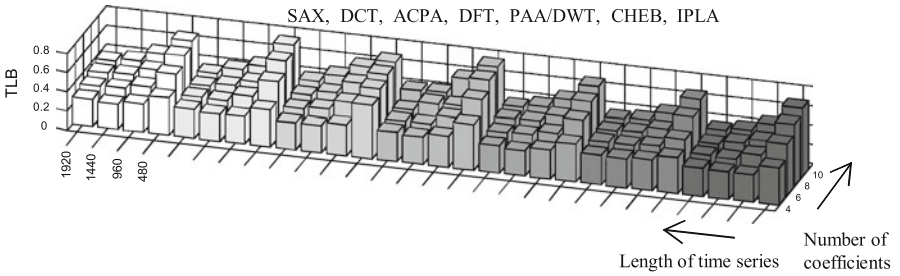

Fig. 7 The tightness of lower bounds (TLB) for various time series representations on an ECG data set

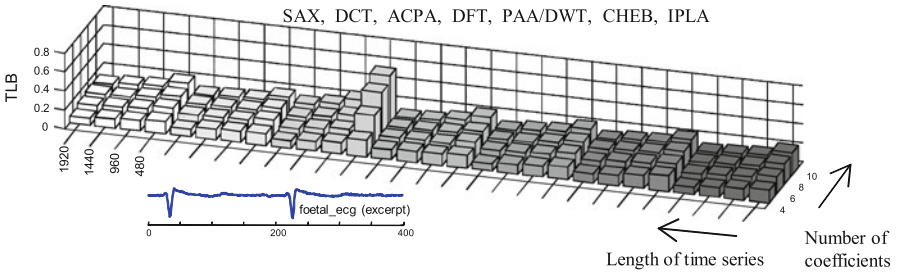

Fig. 8 The tightness of lower bounds (TLB) for various time series representations on a relatively bursty data set (see *inset*)

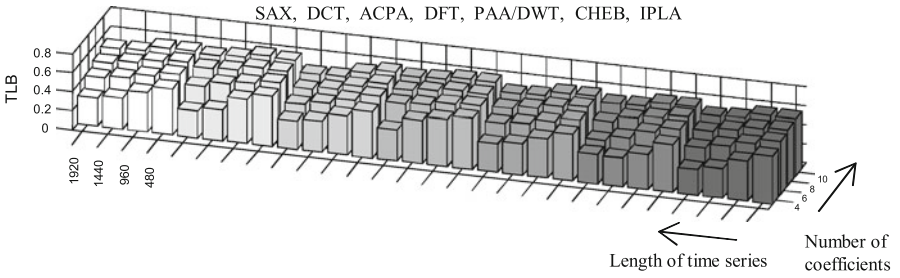

Fig. 9 The tightness of lower bounds (TLB) for various time series representations on a periodic data set of tide levels

(Cai and Ng 2004), however this claim has since been withdrawn by the authors, who explained it was due to a coding error (Ng 2006). Of course there are some variabilities and differences depending on the data sets. For example, on a highly periodic data set the spectral methods are better, whereas on bursty data sets APCA can be significantly better, as shown in Fig. 8.

In contrast, in Fig. 9 we can see that highly periodic data can slightly favor the spectral representations (DCT, DFT, CHEB) over the polynomial representations (SAX, APCA, DWT/PAA, IPLA).

However it is worth noting that the differences presented in these figures are the most extreme cases found in a search spanning over 80 diverse data sets from the publicly available UCR Time Series Data Mining Archive (Keogh et al. 2006). This,

in turn, makes it very likely that, in general, there is very little to choose between representations in terms of pruning power.

4 Comparison of time series similarity measures

In this section we present our experimental evaluation on the accuracy of different similarity measures.

4.1 The effect of data set size on accuracy and speed

We first discuss an extremely important finding which, in some circumstances makes some of the previous findings on efficiency, and the subsequent findings on accuracy, moot. This finding has been noted before (Ratanamahatana and Keogh 2005), but does not seem to be appreciated by the database community.

For an elastic distance measure, both the *accuracy* of classification (or precision/recall of similarity search), and the *amortized speed*, depend critically on the size of the data set. Specifically, on one hand, as data sets get larger, the amortized speed of elastic measures approaches that of lock-step measures, on the other hand, the accuracy/precision of lock-step measures approaches that of the elastic measures. This observation has significant implications for much of the research in the literature. Many papers present results along the lines of "*I have shown on these* 80 *time series that my elastic approach is faster than DTW and more accurate that Euclidean distance, so if you want to index a million time series, use my method*". However, our observation suggests that even if the method is faster than DTW, the speed difference will decrease for larger data sets. Furthermore, for large data sets, the differences in accuracy/precision will also diminish or disappear. To demonstrate our claim we conducted experiments on two highly warped data sets that are often used to highlight the superiority of elastic measures, Two-Patterns and CBF. Because these are synthetic data sets, one has the luxury of creating as many instances as needed, using the data generation algorithms proposed in the original papers (Geurts 2001, 2002). However, it is critical to note that the same effect can be seen on all the data sets considered in this work. For each problem we created 10, 000 test time series, and increasingly large training data sets of size 50, 100, 200, . . ., 6, 400. We measured the classification accuracy of 1NN for the various data sets (explained in more detail in Sect. 4.2.1), using both Euclidean distance and DTW with 10% warping window, and the results are shown in Fig. 10.

Note that for small data sets, DTW is significantly more accurate than Euclidean distance in both cases. However, for CBF, by the time we have a mere 400 time series in our training set, there is no statistically significant difference. For Two-Patterns it takes longer for Euclidean Distance to converge to DTW's accuracy, nevertheless, by the time we have seen a few thousand objects there is no statistically significant difference.

This experiment can also be used to demonstrate our claim that the amortized speed of a (lower-boundable) elastic method approaches that of Euclidean distance. Recall that Euclidean distance has a time complexity of $O(n)$ and that a single DTW

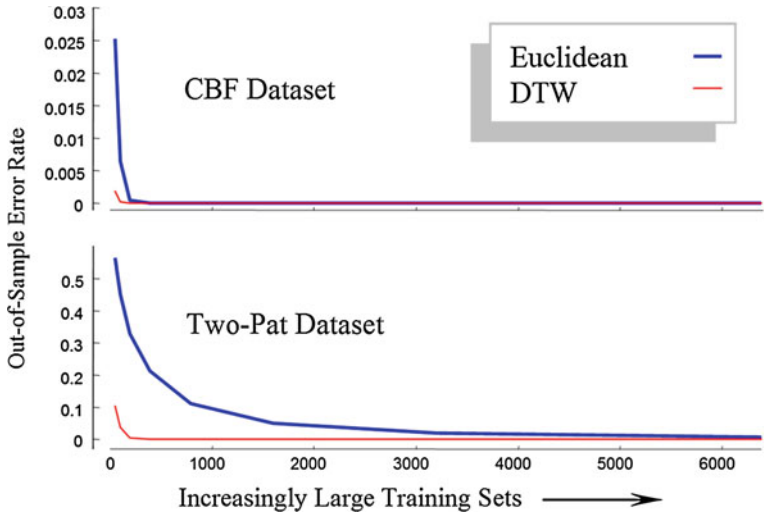

Fig. 10 The error rate for 1-Nearest Neighbor Classification for increasingly large instantiations of two classic time series benchmarks

calculation has a time complexity of $O(nw)$, where w is the warping window size. However for similarity search or 1NN classification, the amortized complexity of DTW is $O((P \cdot n) + (1 - P) \cdot nw)$, where P is the fraction of DTW calculations pruned by a linear time lower bound such as LB_Keogh (Keogh 2002). A similar result can be achieved for LCSS as well, and possibly for the other measures. In the Two-Pattern experiments above, when classifying with only 50 objects, $P = 0.1$, so we are forced to do many Full DTW calculations. However, by the time we have 6, 400 objects, we empirically find out that $P = 0.9696$, so about 97% of the objects are disposed of in the same time as it takes to do a Euclidean distance calculation. To ground this into concrete numbers, it takes less that one second to find the nearest neighbor to a query in the database of 6, 400 Two-Patterns time series, on our off-the-shelf desktop computer, even if we use the pessimistically wide warping window. We note that this time is for just sequential search with a lower bound—no attempt was made to index the data.

To summarize, many of the results reporting on the advantages of a particular distance measure being the fastest or most accurate one may have been biased by the lack of tests on very (or even slightly) large(r) data sets.

4.2 Accuracy of similarity measures

In this section, we evaluate the accuracy of the similarity measures presented in Sect. 2. We first explain the methodology of our evaluation, as well as the parameters that need to be tuned for each similarity measure. We then present the results of our experiments and discuss several interesting findings.

4.2.1 Accuracy evaluation framework

Accuracy evaluation answers one of the most important questions about a similarity measure: why is this a good measure for describing the (dis)similarity between time series? We found that accuracy evaluation is often insufficient in existing literature: it has been either based on subjective evaluation, e.g., (Chen et al. 2005a; Aßfalg et al. 2006), or using clustering with small data sets which are not statistically significant, e.g., (Vlachos et al. 2006; Morse and Patel 2007). In this work, we use an objective evaluation method recently proposed (Keogh and Kasetty 2003). The idea is to use a one nearest neighbor (1NN) classifier (Tan et al. 2005; Han and Kamber 2005) on labelled data to evaluate the efficacy of the distance measure used. Specifically, each time series has a correct class label, and the classifier tries to predict the label as that of its nearest neighbor in the training set. There are several advantages with this approach. First, it is well known that the underlying distance metric is critical to the performance of 1NN classifier (Tan et al. 2005), hence, the accuracy of the 1NN classifier directly reflects the effectiveness of the similarity measure. Second, the 1NN classifier is straightforward to implement and is parameter free, which makes it easy for anyone to reproduce our results. Third, it has been proved that the error ratio of 1NN classifier is at most twice the Bayes error ratio (Duda and Hart 1973). Finally, we note that while there have been attempts to classify time series with decision trees, neural networks, Bayesian networks, supporting vector machines, etc., the best published results (by a large margin) come from simple nearest neighbor methods (Xi et al. 2006).

Algorithm 1 Time series classification with 1NN classifier

Input: Labelled time series data set $\mathbb{T}$, similarity measure operator $SimDist$, number of crosses k
Output: Average 1NN classification error ratio and standard deviation
1: Randomly divide $\mathbb{T}$ into k stratified subsets $\mathbb{T}_1, \ldots, \mathbb{T}_k$
2: Initialize an array $ratios[k]$
3: **for** Each subset $\mathbb{T}_i$ of $\mathbb{T}$ **do**
4: **if** $SimDist$ requires parameter tuning **then**
5: Randomly split $\mathbb{T}_i$ into two equal size stratified subsets $\mathbb{T}_{i1}$ and $\mathbb{T}_{i2}$
6: Use $\mathbb{T}_{i1}$ for parameter tuning, by performing a leave-one-out classification with 1NN classifier
7: Set the parameters to values that yields the minimum error ratio from the leave-one-out tuning process
8: Use $\mathbb{T}_i$ as the training set, $\mathbb{T} - \mathbb{T}_i$ as the testing set
9: $ratio[i] \leftarrow$ the classification error ratio with 1NN classifier
10: **return** Average and standard deviation of $ratios[k]$

To evaluate the effectiveness of each similarity measure, we use a cross-validation algorithm as described in Algorithm 1, based on the approach suggested in Salzberg (1997). We first use a stratified random split to divide the input data set into k subsets for the subsequent classification (line 1) in order to minimize the impact of skewed class distribution. The number of cross validations k is dependent on the data sets and we explain shortly how we choose the proper value for k. We then carry out the cross validation, using one subset at a time for the training set of the 1NN classifier, and the rest

Table 1 Parameter tuning for similarity measures

Parameter	Min value	Max value	Step size
DTW.δ	1	$25\% \cdot n$	1
LCSS.δ	1	$25\% \cdot n$	1
LCSS.ε	$0.02 \cdot Stdv$	$Stdv$	$0.02 \cdot Stdv$
EDR.ε	$0.02 \cdot Stdv$	$Stdv$	$0.02 \cdot Stdv$
Swale.ε	$0.02 \cdot Stdv$	$Stdv$	$0.02 \cdot Stdv$
Swale.$reward$	50	50	–
Swale.$penalty$	0	$reward$	1
TQuEST.τ	$Avg - Stdv$	$Avg + Stdv$	$0.02 \cdot Stdv$
SpADe.$plength$	8	64	8
SpADe.$ascale$	0	4	1
SpADe.$tscale$	0	4	1
SpADe.$slidestep$	$plength/32$	$plength/8$	$plength/32$

$k-1$ subsets as the testing set (lines 3–9). If the similarity measure $SimDist$ requires parameter tuning, we divide the training set into two equal size stratified subsets, and use one of the subset for parameter tuning (lines 4–7). We perform an exhaustive search for all the possible (combinations of) value(s) of the similarity parameter, and conduct a leave-one-out classification test with a 1NN classifier. We record the error ratios of the leave-one-out test, and use the parameter values that yield the minimum error ratio. Finally, we report the average error ratio of the 1NN classification over the k cross validations, as well as the standard deviation (line 10).

Algorithm 1 requires that we provide an input k for the number of cross validations. In our experiments, we need to take into consideration the impact of training data set size discussed in Sect. 4.1. Therefore, our selection of k for each data set attempts to strike a balance between the following factors:

1. The training set size should be selected to enable discriminativity, i.e., one can tell the performance difference between different distance measures.
2. The number of items in the training set should be large enough to represent each class. This is especially important when the distance measure needs parameter tuning.
3. The number of cross validations should be between 5 and 20 in order to minimize bias and variation, as recommended in Kohavi (1995).

The actual number of splits is empirically selected such that the training error for 1NN Euclidean distance (which we use as a comparison reference) is not perfect, but significantly better than the default rate.

Several of the similarity measures that we investigated require the setting of one or more parameters. The proper values for these parameters are key to the effectiveness of the measure. However, most of the time only empirical values are provided for each parameter in isolation. In our experiments, we perform an exhaustive search for all the possible values of the parameters, as described in Table 1.

For DTW and LCSS measures, a common optional parameter is the window size δ that constrains the temporal warping, as suggested in Vlachos et al. (2006). In our experiments we consider both the version of distance measures without warping and with warping. For the latter case, we search for the best warping window size up to 25% of the length of the time series n. An additional parameter for LCSS, which is also used in EDR and Swale, is the matching threshold ε. We search for the optimal threshold starting from $0.02 \cdot Stdv$ up to $Stdv$, where $Stdv$ is the standard deviation of the data set. Swale has two other parameters, the matching reward weight and the gap penalty weight. We fix the matching reward weight to 50 and search for the optimal penalty weight from 0 to 50, as suggested by the authors. Although the warping window size can also be constrained for EDR, ERP and Swale, we only consider full matching for these distance measures in our current experiments—and the rationale for this choice was the fairness. Namely, while each of the three approaches may be amenable to less-than-full matching, this was never proposed, nor considered, as a feature in the original works. For TQuEST, we search for the optimal querying threshold from $Avg - Stdv$ to $Avg + Stdv$, where Avg is the average of the time series data set. For SpADe, we tune four parameters based on the original implementation and use the parameter tuning strategy, i.e. search range, step size, as suggested by the authors. In Table 1, $plength$ is the length of the patterns, $ascale$ and $tscale$ are the maximum amplitude and temporal scale differences allowed respectively, and $slidestep$ is the minimum temporal difference between two patterns.

4.3 Analysis of classification accuracy

In order to provide a comprehensive evaluation, we have performed the experiments on 38 diverse time series data sets, from the UCR Time Series repository (Keogh et al. 2006), which make up somewhere between 90 and 100% of all publicly available, labeled time series data sets in the world. We refer the interested reader to "Appendix" for more details on the datasets. For several years everyone in the data mining/database community has been invited to contribute data sets to this archive, and 100% of the donated data sets have been archived. This ensures that the collection represents the interest of the data mining/database community, and not just one group. All the data sets have been normalized to have a maximum scale of 1.0 and all the time series are z-normalized. The entire simulation was conducted on a computing cluster at Northwestern University, with 20 multi-core workstations running for over a month. The results are presented in Table 2, such as the standard deviation of the cross validations, are hosted on our web site (http://www.ece.northwestern.edu/~hdi117/tsim.htm).

To provide a more intuitive illustration of the performance of the similarity measures compared in Table 2, we now use scatter plots to conduct pair-wise comparisons. In a scatter plot, the error ratios of the two similarity measures under comparison are used as the x and y coordinates of a dot, where each dot represents a particular data set. Where a scatter plot has the label "A versus B", a dot above the line indicates that A is more accurate than B (since these are error ratios). The further a dot is from the

Table 2 Error ratio of different similarity measures on 1NN classifier

Data Set	Crosses#	ED	L_1-norm	L_∞-norm	DISSIM	TQuEST	DTW	DTW (c)[a]	EDR	ERP	LCSS	LCSS(c)	Swale	Spade
50words	5	0.407	0.379	0.555	0.378	0.526	0.375	0.291	0.271	0.341	0.298	0.279	0.281	0.341
Adiac	5	0.464	0.495	0.428	0.497	0.718	0.465	0.446	0.457	0.436	0.434	0.418	0.408	0.438
Beef	2	0.4	0.55	0.583	0.55	0.683	0.433	0.583	0.4	0.567	0.402	0.517	0.384	0.5
Car	2	0.275	0.3	0.3	0.217	0.267	0.333	0.258	0.371	0.167	0.208	0.35	0.233	0.25
CBF	16	0.087	0.041	0.534	0.049	0.171	0.003	0.006	0.013	0	0.017	0.015	0.013	0.044
ChlorineConcentration	9	0.349	0.374	0.325	0.368	0.44	0.38	0.348	0.388	0.376	0.374	0.368	0.374	0.439
Cinc_ECG_torso	30	0.051	0.044	0.18	0.046	0.084	0.165	0.006	0.011	0.145	0.057	0.023	0.057	0.148
Coffee	2	0.193	0.246	0.087	0.196	0.427	0.191	0.252	0.16	0.213	0.213	0.237	0.27	0.185
DiatomSizeReduction	10	0.022	0.033	0.019	0.026	0.161	0.015	0.026	0.019	0.026	0.045	0.084	0.028	0.016
ECG200	5	0.162	0.182	0.175	0.16	0.266	0.221	0.153	0.211	0.213	0.171	0.126	0.17	0.256
ECGFiveDays	26	0.118	0.107	0.235	0.103	0.181	0.154	0.122	0.111	0.127	0.232	0.187	0.29	0.265
FaceFour	5	0.149	0.144	0.421	0.172	0.144	0.064	0.164	0.045	0.042	0.144	0.046	0.134	0.25
FacesUCR	11	0.225	0.192	0.401	0.205	0.289	0.06	0.079	0.05	0.028	0.046	0.046	0.03	0.315
Fish	5	0.319	0.293	0.314	0.311	0.496	0.329	0.261	0.107	0.216	0.067	0.16	0.171	0.15
Gun_Point	5	0.146	0.092	0.186	0.084	0.175	0.14	0.055	0.079	0.161	0.098	0.065	0.066	0.007
Haptics	5	0.619	0.634	0.632	0.64	0.669	0.622	0.593	0.466	0.601	0.631	0.58	0.581	0.736
InlineSkate	6	0.665	0.646	0.715	0.65	0.757	0.557	0.603	0.531	0.483	0.517	0.525	0.533	0.643
ItalyPowerDemand	8	0.04	0.047	0.044	0.043	0.089	0.067	0.055	0.075	0.05	0.1	0.076	0.082	0.233
Lighting2	5	0.341	0.251	0.389	0.261	0.444	0.204	0.32	0.088	0.19	0.199	0.108	0.16	0.272
Lighting7	2	0.377	0.286	0.566	0.3	0.503	0.252	0.202	0.093	0.287	0.282	0.116	0.279	0.557
MALLAT	20	0.032	0.041	0.079	0.042	0.094	0.038	0.04	0.08	0.033	0.088	0.091	0.09	0.167
MedicalImages	5	0.319	0.322	0.36	0.329	0.451	0.286	0.281	0.36	0.309	0.349	0.357	0.348	0.434

Table 2 continued

Data Set	Crosses#	ED	L_1-norm	L_∞-norm	DISSIM	TQuEST	DTW	DTW (c)[a]	EDR	ERP	LCSS	LCSS(c)	Swale	Spade
Motes	24	0.11	0.082	0.24	0.08	0.211	0.09	0.118	0.095	0.106	0.064	0.077	0.073	0.103
OliveOil	2	0.15	0.236	0.167	0.216	0.298	0.1	0.118	0.062	0.132	0.135	0.055	0.097	0.207
OSULeaf	5	0.448	0.488	0.52	0.474	0.571	0.401	0.424	0.115	0.365	0.359	0.281	0.403	0.212
Plane	6	0.051	0.037	0.033	0.042	0.038	0.001	0.032	0.001	0.01	0.016	0.062	0.023	0.006
SonyAIBORobotSurface	16	0.081	0.076	0.106	0.088	0.135	0.077	0.074	0.084	0.07	0.228	0.155	0.205	0.195
SonyAIBORobotSurfaceII	12	0.094	0.084	0.135	0.071	0.186	0.08	0.083	0.092	0.062	0.238	0.089	0.281	0.322
StarLightCurves	9	0.142	0.143	0.151	0.142	0.13	0.089	0.086	0.107	0.125	0.118	0.124	0.12	0.142
SwedishLeaf	5	0.295	0.286	0.357	0.299	0.347	0.256	0.221	0.145	0.164	0.147	0.148	0.14	0.254
Symbols	30	0.088	0.098	0.152	0.093	0.078	0.049	0.096	0.02	0.059	0.053	0.055	0.058	0.018
Synthetic_control	5	0.142	0.146	0.227	0.158	0.64	0.019	0.014	0.118	0.035	0.06	0.075	0.06	0.15
Trace	5	0.368	0.279	0.445	0.286	0.158	0.016	0.075	0.15	0.084	0.118	0.142	0.108	0
TwoLeadECG	25	0.129	0.154	0.151	0.163	0.266	0.033	0.07	0.065	0.071	0.146	0.154	0.149	0.017
Two-Patterns	5	0.095	0.039	0.797	0.036	0.747	0	0	0.001	0.01	0	0	0	0.052
Wafer	7	0.005	0.004	0.021	0.005	0.014	0.015	0.005	0.002	0.006	0.004	0.004	0.004	0.018
WordsSynonyms	5	0.393	0.374	0.53	0.375	0.529	0.371	0.315	0.295	0.346	0.294	0.28	0.274	0.322
Yoga	11	0.16	0.161	0.181	0.167	0.216	0.151	0.151	0.112	0.133	0.109	0.134	0.43	0.13

[a] DTW (c) denotes DTW with constrained warping window, same for LCSS

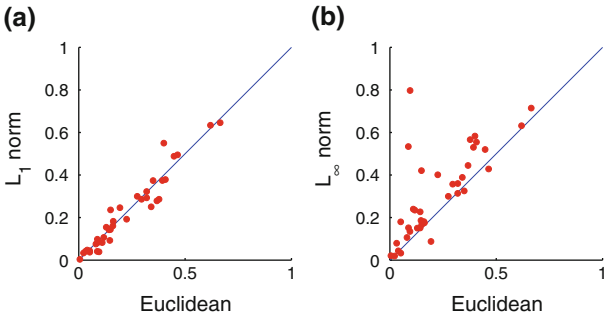

Fig. 11 Comparison of various L_p-norms, above the *line* Euclidean outperforms L_1norm/L_∞norm. **a** Euclidean versus L_1 norm. **b** Euclidean versus L_∞ norm

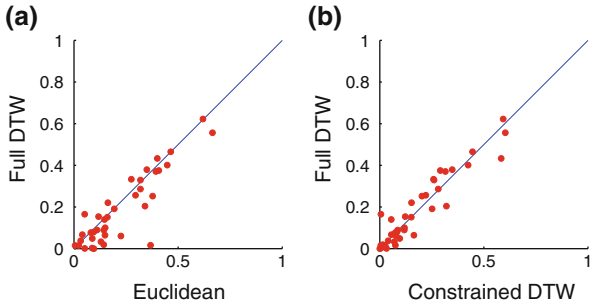

Fig. 12 Comparison of DTW, above the *line* Euclidean/constrained DTW outperforms Full DTW. **a** Euclidean versus Full DTW. **b** Constrained DTW versus Full DTW

line, the greater the margin of accuracy improvement. The more dots on one side of the line indicates that the worse one similarity measure is compared to the other.

First, we compare the different variances of L_p-norms.

Figure 11 shows that the Euclidean distance and the Manhattan distance have a very close performance, while both largely outperform the L_∞-norm. This is expected, as a consequence of its definition: the L_∞-norm uses the maximum distance between two sets of time series points, and is more sensitive to noise (Han and Kamber 2005).

Next we illustrate the performance of DTW against Euclidean. Figure 12a shows that Full DTW is clearly superior over Euclidean on the data sets we tested. Figure 12b shows that the effectiveness of constrained DTW is the same (or even slightly better) than that of Full DTW. This means that we could generally use the constrained DTW instead of DTW to reduce the time for computing the distance and to utilize proposed lower bounding techniques (Keogh and Ratanamahatana 2005).

Unless otherwise stated, in the following we compare the rest of the similarity measures against Euclidean distance and Full DTW, since Euclidean distance is the fastest and most straightforward measure, and DTW is the oldest elastic measure.

The performance of DISSIM against that of Euclidean and DTW is shown in Fig. 13. It can be observed that the accuracy of DISSIM is slightly better than Euclidean distance; however, it is apparently inferior to DTW.

The performance of TQuEST against that of Euclidean and DTW is shown in Fig. 14. On most of the data sets, TQuEST is worse than Euclidean and DTW

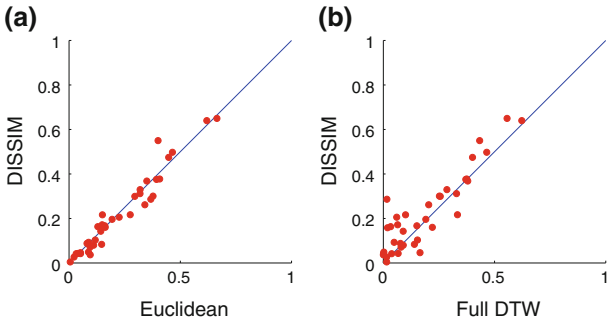

Fig. 13 Comparison of DISSIM, above the *line* Euclidean/Full DTW outperforms DISSIM. **a** Euclidean versus DISSIM. **b** Full DTW versus DISSIM

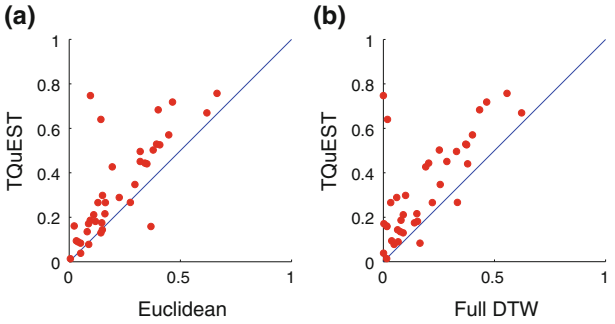

Fig. 14 Comparison of TQuEST, above the *line* Euclidean/Full DTW outperforms TQuEST. **a** Euclidean versus TQuEST. **b** Full DTW versus TQuEST

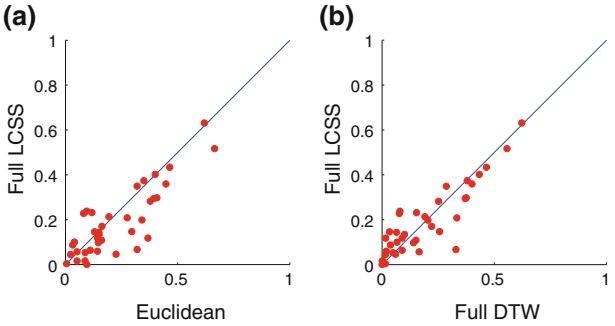

Fig. 15 Comparison of LCSS, above the *line* Euclidean/Full DTW outperforms Full LCSS. **a** Euclidean versus Full LCSS. **b** Full DTW versus Full LCSS

distances. While the outcome of this experiment cannot account for the usefulness of TQuEST, it indicates that there is a need to investigate the characteristics of the data set for which TQuEST is a favorable measure.

The respective performances of LCSS, EDR and ERP against the Euclidean and DTW measures are illustrated in Figs. 15, 16 and 17, where the left portions of each figure represent the comparison against the Euclidian distance, and the right portions

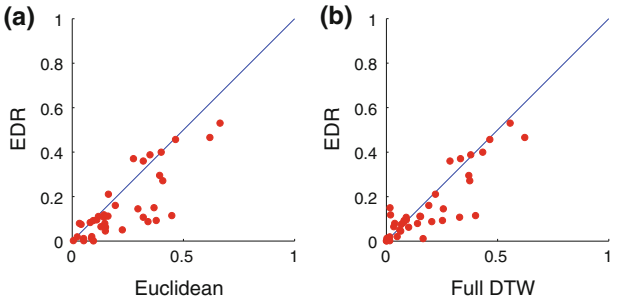

Fig. 16 Comparison of EDR, above the *line* Euclidean/Full DTW outperforms EDR. **a** Euclidean versus EDR. **b** Full DTW versus EDR

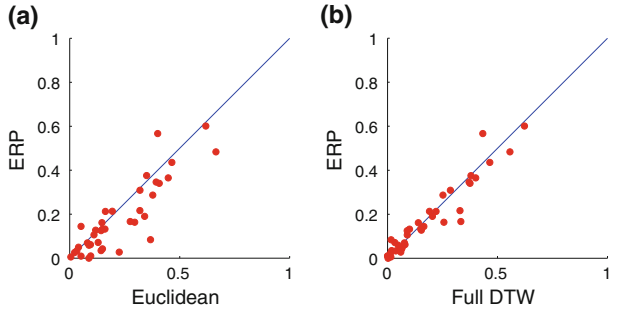

Fig. 17 Comparison of ERP, above the *line* Euclidean/Full DTW outperforms ERP. **a** Euclidean versus ERP. **b** Full DTW versus ERP

represent the comparison against DTW. An obvious conclusion is that all three distances outperform the Euclidean distance by a large percentage. However, while it is commonly believed that these edit distance based similarity measures are superior to DTW (Chen and Ng 2004; Chen et al. 2005b, 2007b), our experiments suggest that this need not the case in general. As shown, only EDR is *potentially* slightly better than Full DTW, whereas the performance of LCSS and ERP are very close to DTW. Even for EDR, a more formal analysis using a two-tailed, paired t-test is required to reach any statistically significant conclusion (Salzberg 1997). We also studied the performance of constrained LCSS, as shown in Fig. 18. It can be observed that the constrained version of LCSS is even slightly better than the unconstrained one, while it also reduces the computation cost and gives rise to an efficient lower-bounding measure (Vlachos et al. 2006).

Next, we compare the performance of Swale against that of Euclidean distance and LCSS, as Swale aims at improving the effectiveness of LCSS and EDR. The results are shown in Fig. 19, and suggest that Swale is clearly superior to Euclidean distance, and yields an almost identical accuracy as LCSS.

Finally, we compare the performance of SpADe against that of Euclidean distance and DTW. The results are shown in Fig. 20. In general, the accuracy of SpADe is close to that of Euclidean but is inferior to DTW distance, although on some data sets SpADe outperforms the other two. We believe that one of the biggest challenges for

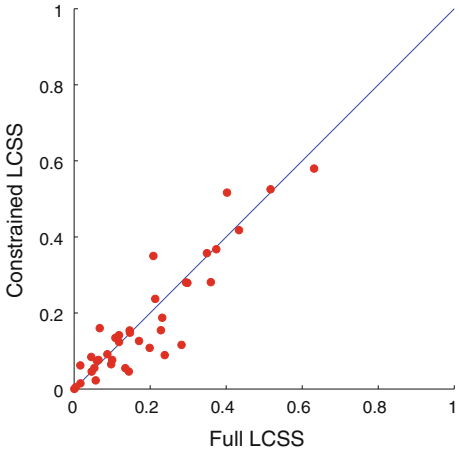

Fig. 18 Comparison of constrained LCSS, above the *line* Full LCSS outperforms constrained LCSS.

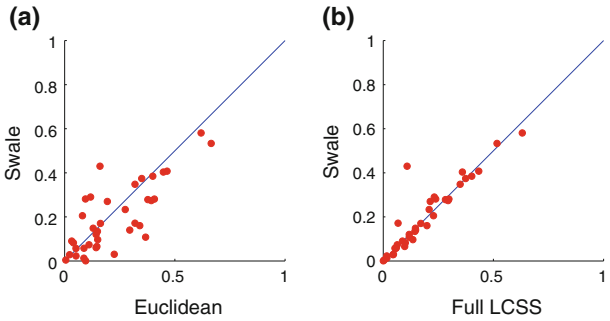

Fig. 19 Comparison of Swale, above the *line* Euclidean/Full LCSS outperforms Swale. **a** Euclidean versus Swale. **b** Full LCSS versus Swale

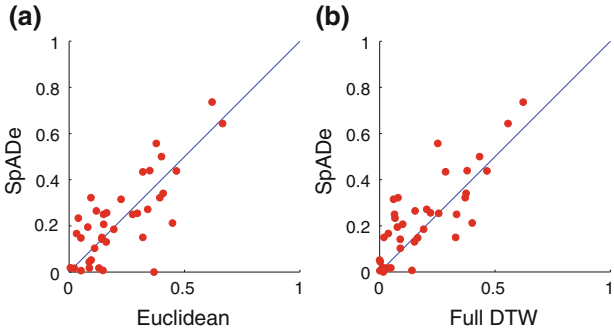

Fig. 20 Comparison of SpADe, above the *line* Euclidean/Full DTW outperforms SpADe. **a** Euclidean versus SpADe. **b** Full DTW versus SpADe

SpADe is that it has a large number of parameters that need to be tuned. Given the small tuning data sets, it is very difficult to pick the right values. However, we note again that the outcome of this experiment cannot account for the utility of SpADe. For example, one major contribution of SpADe is to detect interesting patterns online for stream data.

In summary, we found through experiments that there is no clear evidence that one similarity measure exists that is superior to others in the literature in terms of accuracy. While some similarity measures are more effective on certain data sets, they are usually inferior on some other data sets. This does not mean that the time series community should settle for the existing similarity measures—quite the contrary. However, we believe that more caution needs to be exercised in order to avoid the possibility of making certain mistakes and drawing invalid conclusions, some of which we address in detail in the subsequent Sect. 5.

5 Exploring misunderstandings surrounding dynamic time warping

DTW is one of the earliest similarity measures for time series proposed in the literature. Having shown that, on average, the constrained DTW is no worse than the more recently introduced similarity measures in terms of accuracy across a wide range of problems, we now address some persistent myths about it, including some that have limited its adoption.

5.1 DTW is too slow to be of practical use

In the literature, it is often claimed that DTW is too slow to be of practical use. Consider the following quotes:

> "...too slow for practical applications, especially when the underlying similarity measure is based on DTW "(Alon et al. 2005). "The expensive DTW method prohibits high performance and real-time applications" (Lin 2006). "However, the computational load of DTW is so expensive that it is intractable for many real-world problems" (Jia et al. 2004). "DTW (is) very expensive, and are not applicable for multi-media data" (Aßfalg et al. 2008). "computing cost of DTW algorithm is high" (Zhang and Kinsner 2009),"DTW-based techniques suffer for performance inefficiencies" (Papadopoulos 2008).

The literature is replete with similar claims, however, in every case where a particular claim is backed up with an experimental verification, we believe that subtle errors may account for the respective results. We reimplemented the experiments in dozens of papers that implicitly or explicitly suggest that DTW is too slow to be practical (Park and Kim 2006; Jia et al. 2004; Kumar et al. 2007; Aßfalg et al. 2008) and found that doing nothing more than a simple sequential scan with a lower bound search (as in Algorithm 2), we could do DTW search between two to five orders of magnitude faster than the claimed wall clock times. Of course, to be fair to the original authors we note that at least some fraction of our results are surely due to improvements in machine speed, and our efficient implementation. Our point is simply that if a newcomer to

the field reads such a paper and is tempted to shy away from DTW after reading of experiments that took minutes or hours, she should note that the experiments in question can be done in less than one second on a modern machine using twenty lines of matlab.

The idea of "*DTW is too slow*" seems to come from reading old literature, perhaps combined with implementation bias (Keogh and Kasetty 2003), and it perpetuates itself from paper to paper. However, the fact that we could easily do the many hundreds of millions of DTW calculations required for this paper should help to dispel this myth. Less than one percent of papers in the time series data mining/database literature consider a data set that is larger than 10,000 objects, yet we can easily search 10,000 time series of, say, length 256, using DTW—in well under one second.

5.2 There is room to speed up similarly search under DTW

Another common and persistent misunderstanding about DTW is that it can be further sped up by improving current lower bounds. Needless to say, it is generally the case that tighter lower bounds are better. However, there are diminishing returns for tightness of lower bounds, and as we will show, we have long ago reached a point where it is no longer worthwhile to attempt to improve lower bounds for DTW.

To eliminate the confounding factors of indexing structures, buffering policies etc, we consider the simplest lower bounding search algorithm, which assumes that all the data is in main memory, illustrated by Algorithm 2 below. The algorithm assumes that C_i is the ith time series in database C, which contains N time series, and Q is a query issued to it.

Algorithm 2 Lower_Bounding_Sequential_Scan(Q,C)

```
1: best_so_far = Inf
2: for all sequences in database C do
3:    LB_dist = lower_bound_distance(C_i, Q) //Cheap lower bound
4:    if LB_dist < best_so_far then
5:       true_dist = DTW(C_i, Q) // Expensive DTW
6:       if true_dist < best_so_far then
7:          best_so_far = true_dist
8:          index_of_best_match = i
```

It is easy to see that the time taken for this search algorithm depends only on the data itself, and the tightness of the lower bound. If the lower bound is trivially loose, say we hard-code it to zero, then the test in line 4 will always be true, and we must do the expensive DTW calculation in line 5 for every object in the database. In contrast, if the lower bound is relativity tight, then a large fraction of tests in line 4 will fail, and we can skip that fraction of DTW calculations.

There are a handful of ways to slightly speed up this algorithm. For example, both DTW and most of the lower bounds can be implemented as early abandoning (Keogh et al. 2009), and we could first sort the time series objects by their lower bound distance before entering the loop in line 2 (this increases the speed at which

the *best_so_far* decreases, making the test in line 4 fail more often). However, these produce only modest speed-ups, since most of the strength of this simple algorithm comes from having a tight lower bound.

As with different representations (cf. Sect. 3), the tightness of a lower bound can be measured by a simple ratio T:

$$T = lower\_bound\_distance(C_i, Q)/\mathrm{DTW}(C_i, Q)$$

It is clear that $0 \leq T \leq 1$. Note that T must be measured by a large random sampling. How tight are current lower bounds? Let us start by considering the envelope-based lower bound (LB_Keogh) introduced in 2002. Its value depends somewhat on δ, the temporal constraint (cf. Sect. 2.2) and on the data itself. In general, smooth data sets tend to allow tighter lower bounds than noisy ones. However, values of 0.6 are typical.

We are now in a position to consider the question, can DTW search be further sped up by improving the current lower bound, as frequently claimed? Rather than implementing these various bounds and risking criticism of a poor implementation, we perform the following idealized experiment. We imagine that we have a sequence of progressively better lower bounds. In particular, each time we calculate LB_Keogh we also calculate "magic" lower bounds which are tighter. To calculate these tighter lower bounds we must "cheat". We also calculate the true DTW distance and the difference d.

$$d = \mathrm{DTW}(C_i, Q) - \mathrm{LB\_Keogh}(C_i, Q)$$

We can then add in a fraction of the difference to LB_Keogh to see what effect a tighter lower bound will have. Concretely, we create two idealized lower bounds:

$$\mathrm{LB\_Magic\_50} = \mathrm{LB\_Keogh}(C_i, Q) + (0.50 \times d)$$
$$\mathrm{LB\_Magic\_100} = \mathrm{LB\_Keogh}(C_i, Q) + (1 \times d)$$

Note the following: although the magic lower bounds have been given extra information, they have not been penalized for it in terms of time complexity. They are assumed to take *exactly* as long to compute as LB_Keogh. Furthermore, note what an extraordinary advantage has been given to these lower bounds—LB_Magic_100 is a logically *perfect* lower bound; it cannot be improved upon. Without doing any experiments it is possible to predict the performance of LB_Magic_100. It will only have to do the expensive calculation in line 5 of Algorithm 2 for $O(log_2(N))$ times.

We can test the utility of these lower bounds by searching increasingly large data sets. We measure the wall clock time to find the nearest neighbor to a randomly chosen query (which did not come from the data set), averaging over 30 queries. We used a data set of star light curves (Keogh et al. 2009). Figure 21[2] shows the results.

[2] In this experiment only the relative times matter. However, the reader may wonder why the absolute times are large. The original time series, which are based on a few dozen (unevenly spaced in time) samples, are greatly over sampled to a length 1,024 by the astronomers as a side effect of their interpolation/smoothing

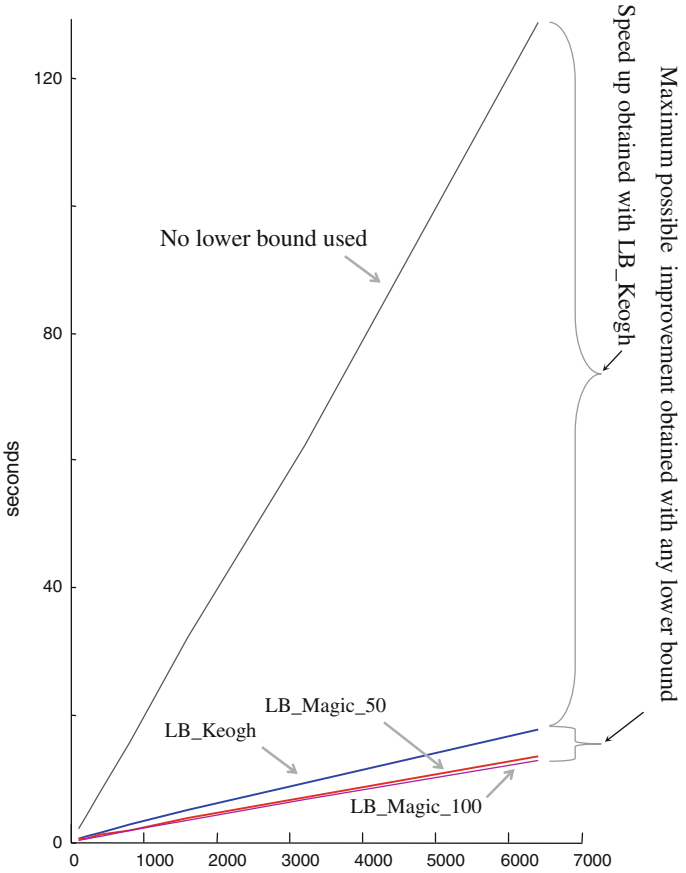

Fig. 21 The wall clock time to answer a one-nearest neighbor query in increasingly large instances of a *star* light curve dataset, for four rival methods[2]

As we can see, the first (non-trival) lower bound introduced for DTW in 2002 really does produce a significant speedup. However, even if we use the idealized optimal lower bound, the *most* of an improvement we could ever hope to obtain is a search that is 1.37 times faster. These results are hard to reconcile with some claims in the literature. As indicated in Fig. 21, speedups beyond certain factor and based solely on improvements of the lower bounds, may not be attainable in practice. It might be argued that this result is an anomaly of some kind; however, essentially the same results are seen on other data sets (Ratanamahatana and Keogh 2005). We see the same basic pattern regardless of the data set, the value of the temporal constraint, the length of the time series, the size of the data set, etc. To our knowledge, there is only

Footnote 2 continued
algorithm, and we used a pessimistic temporal constraint of $\delta = 10\%$. If we re-sample them to a more reasonable length of 256, and use the (empirically) best temporal constraint of $\delta = 4\%$, we can find the nearest neighbor in a data set of size 6,400 in well under second. This would also *improve* the accuracy (cf.Table 2).

one paper that offers a plausible speedup based on a tighter lower bound—Lemire (2009) suggests a mean speedup of about 1.4 based on a tighter bound. These results are reproducible, and testing on more general data sets we obtained similar results (speedups of between 1.0 and 1.3).

We note that other independent research with scrupulously fair and reproducible findings have confirmed these claims. For example, (Assent et al. 2009) discovered that while the boundary-based lower bounds introduced in Zhou and Wong (2007) offer slightly tighter bounds, no speed-up could be obtained due to the overhead in the slightly more expensive lower bound calculations. Likewise, while the FTW lower bounds introduced in Sakurai et al. (2005) may offer tighter bounds, similarity search under FTW is significantly slower (about an order of magnitude) than using just LB_Keogh. Quite simply, the large overhead in creating the lower bound here does not manage to break even. In fact, for reasonable values of the threshold δ, the FTW lower bound can take longer to calculate than the original DTW distance!

In summary, while it may be possible to speed up similarity search for DTW (see Assent et al. (2009) for example), it is *not* possible to do so significantly by tightening the lower bounds.

5.3 There is room to produce a more accurate distance measure than DTW

Our comparative experiments have shown that while elastic measures are, in general, better than lock-step measures, there is little difference between the various elastic measures. This result explicitly contradicts the results of many papers introducing distance measures that could potentially yield better performance than DTW, the original and simplest elastic measure. How are we to reconcile these two conflicting claims? We believe the following demonstration will shed some light on the issue. We classified 20 of the data sets hosted at the UCR archive, using the suggested twofold splits that were established several years ago. We used a distance measure called ANA (explained below) which has a single parameter that we adjusted to get the best performance. Figure 22 compares the results of our algorithm with Euclidean distance.

As we can see, the ANA algorithm is consistently better than Euclidean distance, often significantly so. Furthermore, ANA is as fast as Euclidean distance, is indexable and only has a single parameter. Given this, can a paper on ANA be published in a good conference or journal? It is time to explain how ANA works. We downloaded the mitochondrial DNA of an animal from Genbank (www.ncbi.nlm.nih.gov/genbank/). We converted the DNA to a string of integers, with A (Adenine) = 0, C (Cytosine) = 1, G (Guanine) = 2 and T (Thymine) = 3. So the DNA string $GATCA\ldots$ becomes 2, 0, 3, 1, 0, ...

Given that we have a string of 16,564 integers, we can use the first n integers as weights when calculating the weights of the Euclidean distance between our time series of length n. So ANA is nothing more than the weighed Euclidean distance, weighed by the DNA string. More concretely, if we have a string S: $S = 3, 0, 1, 2, 0, 2, 3, 0, 1, \ldots$ and some time series, say of length 4, then the weight vector W with p = 1 is 3, 0, 1, 2, and the ANA distance is simply:

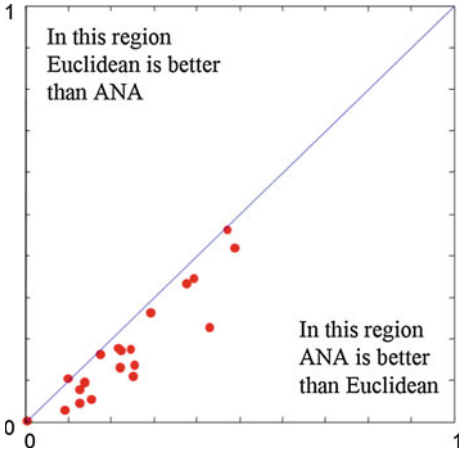

Fig. 22 The classification error ratios of ANA compared to Euclidean distance

$$\mathrm{ANA}(A, B, W) = \sqrt{\sum_{i=1}^{4}(A_i - B_i)^2 \times W_i}.$$

After we test the algorithm, if we are not satisfied with the result, we simply shift the first location in the string, so that we are using locations 2 to $n + 1$ of the weight string. We continue shifting until the string is exhausted and report the best result in Fig. 22. At this point the reader will hopefully say "but that is not fair, you cannot change the parameter after seeing the results, and report the best results". However, we believe this effect may explain many of the apparent improvements over DTW, and in some cases the authors have explicitly acknowledged this (Chen et al. 2005b). Researchers do seem to be adjusting the parameters after seeing the results on the test set, although of course they are in no way deliberately attempting to gain an advantage. In summary, based on the experiments conducted in this paper and all the reproducible fair experiments in the literature, there is no large body of evidence that any distance measure that is systematically better than DTW in general. Furthermore, there is at best very scant evidence that there is any distance measure that is systematically better than DTW in particular domains (say, just ECG data, or just noisy data).

Could these results actually explain the apparently optimistic results in the literature? While we can never know for sure, we can present two pieces of supporting evidence.

A known example of peeking at the test data: In a recent paper on time series classification, a new distance measure was introduced, called DPAA (Karamitopoulos and Evangelidis 2009). The new measure was tested on the twenty datasets in the UCR archive and it was reported that: "*there are (nine) datasets where DPAA outperforms DTW*". The authors were presented with an earler draft of this chapter, and were asked whether it would be possible that some fraction of their reported results could be attributed to simply adjusting the parameters after seeing the results of the test set. The authors of Karamitopoulos and Evangelidis (2009), who were extraordinarily gracious and cooperative (for which we are truly indebted) did acknowledge

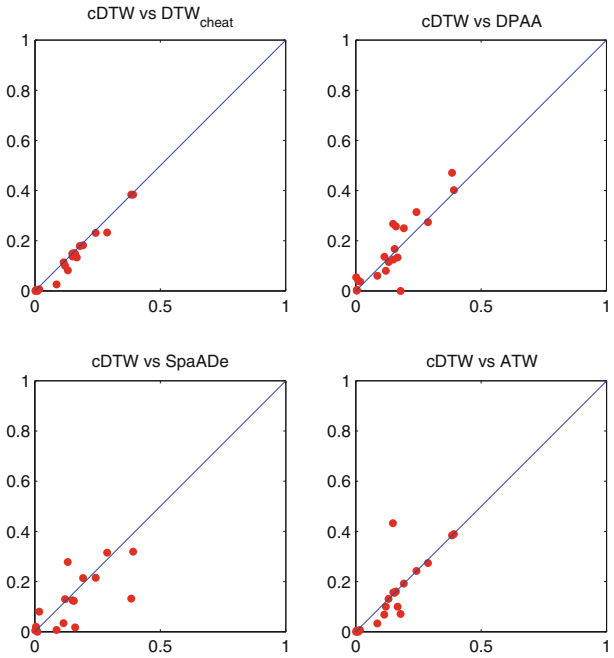

Fig. 23 Comparison of constrained DTW with four other approaches, above the *line* constrained DTW is better, below the *line* the rival approach wins

that they had adjusted some parameters after looking at the test set, but they did so being confident that this was irrelevant, and their results would hold (L. Karamitopoulos and G. Evangelidis, personal communication, Oct 2009). They agreed to rerun all the experiments in a strictly blind fashion—they would learn the parameters required by their method only from the training set, and use those parameters for classifying the test data. A week later, after carefully conducting the experiments, they wrote to us, ruefully noting "*according to the new results, DTW (always) outperforms DPAA*" acknowledging that the apparent utility of their method may have resulted due to some omissions in the assumptions which could affect the fairness of the experiments.

A speculative example of peeking at the test data: The above example is a rare case where we can be sure that feedback from the test data occurred. We can use this example and an original experiment to ask what feedback from the test data might look like in the more general case. Consider the bottom-right portion of Fig. 23. It shows a distance measure compared to constrained DTW on 20 datasets from the UCR archive (essentially a subset of the datasets considered here, but each with one fixed training/test split). If we dismiss the one poor showing (perhaps we could reasonably explain it away as having been an unreliable result on the smallest dataset), then we might take this figure as evidence of the superiority of ATW. After all, it is generally not worse than constrained DTW, and on 3 or 4 datasets it appears to be better. While the visual evidence for ATW is only suggestive, the three companion figures offer *stronger* evidence of superiority. Let us consider them one by one.

- **DTW$_{\text{cheat}}$**: Here we compared constrained DTW with itself; however, we "cheated" by adjusting the single parameter requirement *after* seeing the testing results. Even though there is only one [relatively insensitive, see (Ratanamahatana and Keogh 2005)] parameter for us to play with here, the results are apparently better, and without the knowledge of our cheating, a reader might assume that DTW$_{\text{cheat}}$ algorithm is a useful contribution.
- **DPAA:** Here we compared constrained DTW with the published results for DPAA discussed in the previous section (Karamitopoulos and Evangelidis 2009). Although the results appear to be more of a mixed bag, on a few datasets DPAA does appear to be better. However, recall that when the authors reran the experiments, only looking at the training data, the results were consistently worse than constrained DTW.
- **SpADe:** Here we compared constrained DTW with the results published in Chen et al. (2007b).[3] In this case the results do appear to offer more hope for optimism. Several of the results really do seem significantly better than constrained DTW. However, for this case we have some additional data. The authors tested this approach on a rigorously fair blind test, the SIGKDD challenge held in 2007. Here the authors were only given the labeled training data with the *unlabeled* testing data, and had to submit the class predictions for the unlabeled testing data to an independent judge. In this fair test they lost to Euclidean distance on 9 out of the 20 problems, and lost to constrained DTW on 17 out of the 20 problems (the three wins were by the very small margins of 2.0, 0.4 and 0.1%). Perhaps we could attempt to explain this discrepancy away by assuming that the datasets in the SIGKDD challenge data are very different to the datasets in the UCR archive, and the results in the figure below really *do* represent real cases where the method is better. However, this is not the case. Unknown to all participants in the contest, most of the datasets used in the contest are the same ones that had been publicly available at the UCR archive for many years, just with minor changes in train/test splits to make them less recognizable. For example, in Chen et al. (2007b) two of the best datasets for their method are Adiac and FACE, where they report significant improvements over Euclidean distance and constrained DTW. However, when faced with a minor rearrangement of these two datasets in the blind contest, the quality of results plunged. The SpADe method got 0.098 error for FACE, whereas Euclidean distance got about half that error (0.0447) and cDTW did even better (0.043). Likewise, for Adiac, a dataset known to be highly "warped", they got 0.3039, which is the same as Euclidean distance, but orders of magnitude worse than constrained DTW, which got just 0.0654 error.

We can now revisit the question asked above; do we think that, at least for some problems, ATW represents an improvement over constrained DTW? We hope that simply the three case studies we have just shown will give the reader a pause.

The point of this section is simply to suggest to the community that experimental studies that do not very explicitly state how the parameters were set are likely to be of

[3] Only 17 of the 20 datasets were available at the time this was published.

a very limited value. Under such circumstances the reader simply may not be able to decide whether the proposed method is making a contribution.

6 Conclusion and future work

In this paper, we conducted an extensive experimental consolidation on the state-of-the-art representation methods and similarity measures for time series data. We re-implemented and evaluated 8 different dimension-reduction representation methods, as well as 9 different similarity measures and their variants. Our experiments were carried on 38 diverse time series data sets from various application domains. Based on the experimental results we obtained, we make the following conclusions:

1. The tightness of lower bounding, thus the pruning power, thus the indexing effectiveness of the different representation methods for time series data have, for the most part, very little difference on various data sets.
2. For time series classification, as the size of the training set increases, the accuracy of elastic measures converge with that of Euclidean distance. However, on small data sets, elastic measures, e.g., DTW, LCSS, EDR and ERP etc. can be significantly more accurate than Euclidean distance and other lock-step measures, e.g., L_∞-norm, DISSIM.
3. Constraining the warping window size for elastic measures, such as DTW and LCSS, can reduce the computation cost and enable effective lower-bounding, while yielding the same or even better accuracy.
4. The accuracy of edit distance based similarity measures, such as LCSS, EDR and ERP, are *very* close to that of DTW, a 40-year-old, much simpler technique.
5. The accuracy of several novel types of similarity measures, such as TQuEST and SpADe, are in general inferior to elastic measures.
6. If a similarity measure is not accurate enough for the task, getting more training data *really* helps. This is shown in Fig. 10 where the error rate of both DTW and Euclidean distance is reduced by more than an order of magnitude when we go from a training set of size 50 to size 2,000.
7. If getting more data is not possible, then trying the other measures *might* help; however, extreme care must be taken to avoid overfitting. If we test enough measures on a single train/test split, there is an excellent possibility of finding a measure that improves the accuracy by chance, but will not generalize.

An interesting problem which we hope will be addressed in the near future is the one of obtaining some "meta-characteristics" of time series data. Namely, as we mentioned in Sects. 3 and 4.3, our experiments did not provide any conclusive evidence supporting a "universally better" representation method and indicated that on some datasets, certain similarity measures are better than the rest, while being worse on other datasets. At this time, it remains an open question whether some attributes related to the (semantics of the) datasets can be identified, which could be used as indicators for a preference of a given similarity measure.

Acknowledgments We would like to sincerely thank all the donors of the data sets, without which this work would not have been possible. We would also like to thank Lei Chen, Jignesh Patel, Beng Chin Ooi,

Yannis Theodoridis and their respective team members for generously providing the source code to their similarity measures as references and for providing timely help on this project. We would like to thank Michalis Vlachos, Themis Palpanas, Chotirat (Ann) Ratanamahatana and Dragomir Yankov for useful comments and discussions. Needless to say, any errors herein are the responsibility of the authors only. We would also like to thank Gokhan Memik and Robert Dick at Northwestern University for providing the computing resources for our experiments. We have made the best effort to faithfully re-implement all the algorithms, and evaluate them in a fair manner. The purpose of this project is to provide a consolidation of existing works on querying and mining time series, and to serve as a starting point for providing references and benchmarks for future research on time series. We welcome all kinds of comments on our source code and the data sets of this project (http://www.ece.northwestern.edu/~hdi117/tsim.htm), and suggestions on other experimental evaluations.

Appendix: The datasets

Table 3 list some summary statistics about the 38 datasets used in this work. More details, including pointers to the first paper that used each dataset (where known) are at www.cs.ucr.edu/~eamonn/time_series_data/ (Keogh et al. 2006).

The *type* field requires a brief explanation. Some datasets are **synthetic**, created by some researcher(s) to test some property of a time series classification algorithm. We note that none of the current authors have created any of the synthetic datasets used here. Some datasets are **real**, which simply means they were recorded as natural time series from some physical process, from a beating heart to a robot walking. Finally, some datasets are **shape**, this are one dimensional time series that were extracted by processing some two dimensional shapes, such as fish profiles or the silhouettes of aircraft.

Table 3 The time series classification datasets used

Data set	Number of classes	Number of objects	Time series length	Type
50 words	50	905	270	Real
Adiac	37	781	176	Shape
Beef	5	60	470	Real
Car	4	120	577	Shape
CBF	3	930	128	Synthetic
ChlorineConcentration	3	4, 307	166	Real
Cinc_ECG_torso	4	1, 420	1, 639	Real
Coffee	2	56	286	Real
DiatomSizeReduction	4	322	345	Real
ECG200	2	200	96	Real
ECGFiveDays	2	884	136	Real
FaceFour	4	112	350	Shape
FacesUCR	14	2, 250	131	Shape
Fish	7	350	463	Shape

Table 3 continued

Data set	Number of classes	Number of objects	Time series length	Type
Gun_Point	2	200	150	Real
Haptics	5	463	1, 092	Real
InlineSkate	7	650	1, 882	Real
ItalyPowerDemand	2	1, 096	24	Real
Lighting2	2	121	637	Real
Lighting7	7	143	319	Real
MALLAT	8	2, 400	1, 024	Synthetic
MedicalImages	10	1, 141	99	Real
Motes	2	1, 272	84	Real
OliveOil	4	60	570	Real
OSULeaf	6	442	427	Shape
Plane	7	210	144	Shape
SonyAIBORobotSurface	2	980	65	Real
SonyAIBORobotSurfaceII	2	621	70	Real
StarLightCurves	3	9, 236	1, 024	Real
SwedishLeaf	15	1, 125	128	Shape
Symbols	6	1, 020	398	Real
Synthetic_control	6	600	60	Synthetic
Trace	4	200	275	Synthetic
TwoLeadECG	2	1, 162	82	Real
Two-Patterns	4	5, 000	128	Synthetic
Wafer	2	7, 174	152	Real
WordsSynonyms	25	905	270	Real
Yoga	2	3, 300	426	Shape

References

Aßfalg J, Kriegel H-P, Kröger P, Kunath P, Pryakhin A, Renz M (2006) Similarity search on time series based on threshold queries. In: EDBT

Aßfalg J, Kriegel H-P, Kroger P, Kunath P, Pryakhin A, Renz M (2008) Similarity search in multimedia time series data using amplitude-level features. In: MMM'08, pp 123–133

Additional experiment results for representation and similarity measures of time series. http://www.ece.northwestern.edu/~hdi117/tsim.htm

Alon J, Athitsos V, Sclaroff S (2005) Online and offline character recognition using alignment to prototypes. In: ICDAR'05, pp 839–845

André-Jönsson H, Badal DZ (1997) Using signature files for querying time-series data. In: PKDD

Assent I, Wichterich M, Krieger R, Kremer H, Seidl T (2009) Anticipatory dtw for efficient similarity search in time series databases. PVLDB 2(1):826–837

Bennet B, Galton A (2004) A unifying semantics for time and events. Artif Intell 153(1–2):13–48

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop, pp 359–370

Cai Y, Ng RT (2004) Indexing spatio-temporal trajectories with chebyshev polynomials. In: SIGMOD conference

Cardle M (2004) Automated motion EDITInG. In: Technical report, Computer Laboratory, University of Cambridge, Cambridge
Chan K-p, Fu AW-C (1999) Efficient time series matching by wavelets. In: ICDE
Chen L, Ng RT (2004) On the marriage of lp-norms and edit distance. In: VLDB
Chen L, Özsu MT, Oria V (2005a) Robust and fast similarity search for moving object trajectories. In: SIGMOD conference
Chen L, Özsu MT, Oria V (2005b) Using multi-scale histograms to answer pattern existence and shape match queries. In: SSDBM
Chen Q, Chen L, Lian X, Liu Y, Yu JX (2007a) Indexable PLA for efficient similarity search. In: VLDB
Chen Y, Nascimento MA, Ooi BC, Tung AKH (2007b) SpADe: On shape-based pattern detection in streaming time series. In: ICDE
Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York
Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. In: SIGMOD conference
Flato E (2000) Robust and efficient computation of planar minkowski sums. Master's thesis, School of Exact Sciences, Tel-Aviv University
Frentzos E, Gratsias K, Theodoridis Y (2007) Index-based most similar trajectory search. In: ICDE
Geurts P (2001) Pattern extraction for time series classification. In: PKDD
Geurts P (2002) Contributions to decision tree induction: bias/variance tradeoff and time series classification. PhD thesis, University of Liège, Belgium
Jia S, Qian Y, Dai G (2004) An advanced segmental semi-markov model based online series pattern detection. In: ICPR (3)'04, pp 634–637
Jiawei H, Kamber M (2005) Data mining: concepts and techniques. Morgan Kaufmann Publishers, California
Karamitopoulos L, Evangelidis G (2009) A dispersion-based paa representation for time series. In: CSIE (4), pp 490–494
Karydis I, Nanopoulos A, Papadopoulos AN, Manolopoulos Y (2005) Evaluation of similarity searching methods for music data in P2P networks. IJBIDM 1(2)
Kawagoe K, Ueda T (2002) a similarity search method of time series data with combination of Fourier and wavelet transforms. In: TIME
Keogh EJ (2002) Exact indexing of dynamic time warping. In: VLDB
Keogh EJ (2006) A decade of progress in indexing and mining large time series databases. In: VLDB
Keogh EJ, Kasetty S (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Min Knowl Discov 7(4):349–371
Keogh EJ, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7(3): 358–386
Keogh EJ, Chakrabarti K, Mehrotra S, Pazzani MJ (2001a) Locally adaptive dimensionality reduction for indexing large time series databases. In: SIGMOD conference, pp 151–162
Keogh EJ, Chakrabarti K, Pazzani MJ, Mehrotra S (2001b) Dimensionality reduction for fast similarity search in large time series databases. Knowl Inf Syst 3(3):263–286
Keogh E, Xi X, Wei L, Ratanamahatana C (2006) The UCR time series dataset. http://www.cs.ucr.edu/~eamonn/time_series_data/
Keogh EJ, Wei L, Xi X, Vlachos M, Lee S-H, Protopapas P (2009) Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. VLDB J 18(3):611–630
Kim S-W, Park S, Chu WW (2001) An index-based approach for similarity search supporting time warping in large sequence databases. In: ICDE
Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI
Korn F, Jagadish HV, Faloutsos C (1997) Efficiently supporting ad hoc queries in large datasets of time sequences. In: SIGMOD conference
Kumar A, Jawahar CV, Manmatha R (2007) Efficient search in document image collections. In: ACCV (1)'07, pp 586–595
Lemire D (2009) Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern recognition, pp 2169–2180
Lin Y (2006) Efficient human motion retrieval in large databases. In: GRAPHITE, pp 31–37

Lin J, Keogh EJ, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144
Morse MD, Patel JM (2007) An efficient and accurate method for evaluating time series similarity. In: SIGMOD conference
Ng RT (2006) Note of caution. http://www.cs.ubc.ca/~rng/psdepository/chebyReport2.pdf
Olofsson P (2005) Probability, statistics and stochastic processes. Wiley-Interscience, Hoboken
Papadopoulos AN (2008) Trajectory retrieval with latent semantic analysis. In: SAC'08, pp 1089–1094
Park S, Kim S-W (2006) Prefix-querying with an l1 distance metric for time-series subsequence matching under time warping
Popivanov I, Miller RJ (2002) Similarity search over time-series data using wavelets. In: ICDE
Ratanamahatana CA, Keogh EJ (2005) Three myths about dynamic time warping data mining. In: SDM
Sakurai Y, Yoshikawa M, Faloutsos C (2005) Ftw: fast similarity search under the time warping distance. In: PODS'05, pp 326–337
Salzberg S (1997) On comparing classifiers: pitfalls to avoid and a recommended approach. Data Min Knowl Discov 1(3):317–328
Steinbach M, Tan P-N, Kumar V, Klooster SA, Potter C (2003) Discovery of climate indices using clustering. In: KDD
Tan P-N, Steinbach M, Kumar V (2005) Introduction to data mining. Addison-Wesley, Reading
Tansel A, Clifford J, Jajodia S, Segev A, Snodgrass R (1993) Temporal databases: theory and implementation. Benjamin/Cummings Publishing Co., Menlo Park
Vlachos M, Gunopulos D, Kollios G (2002) Discovering similar multidimensional trajectories. In: ICDE, pp 673–684
Vlachos M, Hadjieleftheriou M, Gunopulos D, Keogh EJ (2006) Indexing multidimensional time-series. VLDB J 15(1):1–20
Workshop and challenge on time series classification at SIGKDD (2007). http://www.cs.ucr.edu/~eamonn/SIGKDD2007TimeSeries.html
Wu Y-L, Agrawal D, Abbadi AE (2000) A comparison of DFT and DWT based similarity search in time-series databases. In: CIKM
Xi X, Keogh EJ, Shelton CR, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: ICML
Yi B-K, Faloutsos C (2000) Fast time sequence indexing for arbitrary Lp norms. In: VLDB
Yi B-K, Jagadish HV, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. In: ICDE. IEEE Computer Society
Zhang G HB, Kinsner W (2009) Electrocardiogram data mining based on frame classification by dynamic time warping matching. Comput Methods Biomech Biomed Eng
Zhou M, Wong MH (2007) Boundary-based lower-bound functions for dynamic time warping and their indexing. In: ICDE'07, pp 1307–1311
Zhu Y, Shasha D (2003) Warping indexes with envelope transforms for query by humming. In: SIGMOD conference