Data Min Knowl Disc
DOI 10.1007/s10618-013-0312-3

CID: an efficient complexity-invariant distance for time series

Gustavo E. A. P. A. Batista · Eamonn J. Keogh · Oben Moses Tataw · Vinícius M. A. de Souza

Received: 9 June 2011 / Accepted: 23 March 2013

Abstract The ubiquity of time series data across almost all human endeavors has produced a great interest in time series data mining in the last decade. While dozens of classification algorithms have been applied to time series, recent empirical evidence strongly suggests that simple nearest neighbor classification is exceptionally difficult to beat. The choice of distance measure used by the nearest neighbor algorithm is important, and depends on the invariances required by the domain. For example, motion capture data typically requires invariance to warping, and cardiology data requires invariance to the baseline (the mean value). Similarly, recent work suggests that for time series clustering, the choice of clustering algorithm is much less important than the choice of distance measure used.In this work we make a somewhat surprising claim. There is an invariance that the community seems to have missed, complexity invariance. Intuitively, the problem is that in many domains the different classes may have different complexities, and pairs of complex objects, even those which subjectively may seem very similar to the human eye, tend to be further apart under current distance measures than pairs of simple objects. This fact introduces errors

Responsible editor: Kristian Kersting.

G. E. A. P. A. Batista (✉) · E. J. Keogh · O. M. Tataw
University of California, Riverside, 900 University Ave, Riverside, CA 92521, USA
e-mail: gbatista@cs.ucr.edu

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

O. M. Tataw
e-mail: tatawm@cs.ucr.edu

V. M. A. de Souza
Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo,
Caixa Postal 668, São Carlos, SP 13560-970, Brazil
e-mail: vsouza@icmc.usp.br

Published online: 12 April 2013

in nearest neighbor classification, where some complex objects may be incorrectly assigned to a simpler class. Similarly, for clustering this effect can introduce errors by "suggesting" to the clustering algorithm that subjectively similar, but complex objects belong in a sparser and larger diameter cluster than is truly warranted.We introduce the first complexity-invariant distance measure for time series, and show that it generally produces significant improvements in classification and clustering accuracy. We further show that this improvement does not compromise efficiency, since we can lower bound the measure and use a modification of triangular inequality, thus making use of most existing indexing and data mining algorithms. We evaluate our ideas with the largest and most comprehensive set of time series mining experiments ever attempted in a single work, and show that complexity-invariant distance measures can produce improvements in classification and clustering in the vast majority of cases.

Keywords Time series · Classification · Clustering · Similarity measures · Complexity

1 Introduction

Time series data occur in almost all domains, and this fact has created a great interest in time series data mining. There is a plethora of classification algorithms that can be applied to time series; however, all of the current empirical evidence suggests that simple nearest neighbor classification is very difficult to beat (Ding et al. 2008). Likewise, recent work suggests that for time series clustering, the choice of clustering algorithm is much less important than the choice of distance measure used (Keogh et al. 2006). This only leaves open the question of *which* distance measure to use. The choice of distance measure depends on the domain. In particular, it depends on the invariances required by the domain. For example, motion capture data typically requires invariance to "warping" (local non-linear accelerations) (Keogh et al. 2009), and gene expression data typically requires invariance to uniform scaling (linear "stretching") (Keogh 2003). With this in mind, the last decade has seen the introduction of hundreds of techniques designed to efficiently measure the similarity between time series with invariance to (various combinations of) the distortions of warping (Keogh et al. 2009), uniform scaling (Keogh 2003), offset (Faloutsos et al. 1994), amplitude scaling, phase (Keogh et al. 2009), occlusions, uncertainty and wandering baseline.

Surprisingly, there is an invariance that the community has missed, *complexity invariance*. We defer a detailed discussion of complexity until later in this work, but for the moment the reader's intuitive idea of "*having more peaks, valleys and features*" will suffice.

The problem lies in the fact that in many domains the different classes may have different complexities, and pairs of complex objects, even those which subjectively may seem very similar, tend to be further apart under current distance measures than pairs of simple objects. This fact introduces errors in nearest neighbor classification, because complex objects are assigned to a simpler class. In this work we introduce the first complexity-invariant distance measure for time series, and show that it generally produces significant improvements in classification and clustering accuracy.

Our complexity-invariant distance measure is simple, parameter-free, and increases the time complexity only by a barely perceptible amount.

We further show that this improvement does not compromise the efficiency of algorithms that make frequent calls to a distance measure (classification (Ding et al. 2008), clustering (Elkan 2003), motif discovery (Mueen et al. 2009) and outlier detection (Protopapas et al. 2006; Yankov et al. 2008)), since we can lower bound the measure and use a minor modification of triangular inequality, and thus avail of all existing indexing and data mining techniques.

It is critical to note that the problem we have observed is not solved or mitigated by generalizing from one nearest neighbor to k-nearest neighbors, or by smoothing the data, or using dynamic time warping (DTW), etc. We will present extensive empirical evidence that our improvements in classification and clustering are valid, significant and result from the reason we claim.

The rest of this paper is organized as follows. In Sect. 2 we review the current set of known invariances, and more general related work. In Sect. 3 we present a distance measure that is invariant to the complexity of time series. In Sect. 4 we evaluate our ideas with a large and comprehensive set of time series experiments and show that complexity-invariant distance measures can produce improvements in classification and clustering in the vast majority of cases. In Sect. 5 we compare the complexity estimate measure used in our approach with other measures previously proposed, and show that our complexity measure outperforms the other measures for most data sets. In Sect. 6 we discuss some important properties of our proposed distance that allow it to be indexed by virtually all existing indexing schemes based on lower bounding and triangular inequality. In Sect. 7 we discuss how CID can be useful for other data mining tasks that involve similarity calculations and show empirical evidence of CID utility to discords discovery. Finally, in Sect. 8 we offer conclusions and suggest areas for future work.

2 Background and related work

We begin by introducing Euclidean distance, and use this as a starting point to consider other distance measures in the next section.

Suppose we have two time series, Q and C, of length n.

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n$$
$$C = c_1, c_2, \ldots, c_i, \ldots, c_n$$

If we wish to compare two time series, we can use the ubiquitous Euclidean distance:

$$ED(Q, C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

This distance measure is visualized in Fig. 1. While Euclidean distance is a simple measure, it is competitive for many problems (Ding et al. 2008). Nevertheless, in many domains the data is distorted in some way, and either the distortion must be removed

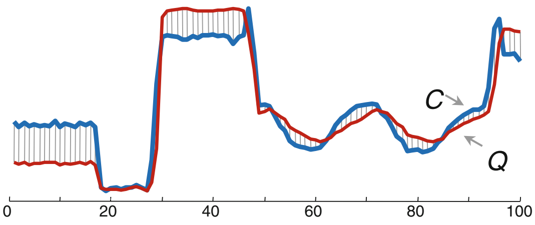


Fig. 1 The Euclidean distance between two time series can be visualized as the square root of the sum of the squared length of the *vertical hatch lines*

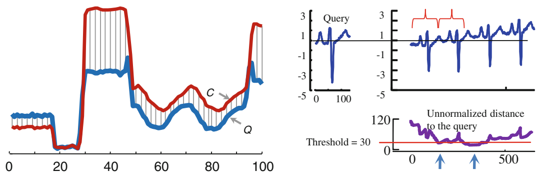


Fig. 2 *Left* If compared before amplitude scaling, these two time series appear very different. *Right* When matching a heartbeat query against a stream, the first two match well, but subsequently the drifting offset means the remaining heartbeats are not discovered

before using Euclidean distance, or a more robust measure must be used.[1] We will now review all known distortions and the techniques for achieving invariance to them.

2.1 A review of all known invariances

For any given domain the set of required invariances is domain dependent, and include:

2.1.1 Amplitude and offset invariance

If we try to compare two time series measured on different scales, say Celsius and Fahrenheit, they will not match well, even if they have similar shapes. To measures the true underlying similarity we must first make their amplitudes the same. As shown in Fig. 2 *left*, the example of Euclidean distance shown in Fig. 1 already had this normalization, and the original data has a significant difference in amplitude.

Similarly, even if two time series have identical amplitudes, they may have different offsets (different mean values). As shown in Fig. 2 *right*, even a small change in offset rapidly dominates the Euclidean distance, leading to false negatives; i.e., missed heartbeats by a heartbeat detector.

A classic example of where these invariances are critical is gait recognition from video, where zoom and pan/tilt correspond to amplitude and offset, respectively. Both these invariances can be trivially achieved by z-normalizing the data (Faloutsos et al. 1994).

[1] In practice these two options can be logically equivalent. For example, DTW can be seen as a more robust distance measure, or it can be seen as using the Euclidean distance after a dynamic programming algorithm has removed the warping.

2.1.2 Local scaling ("warping") invariance

This invariance is necessary in almost all biological signals, including gait, motion capture, handwriting and ECGs. Fig. 3 shows an example of two insect behaviors which match only when one is locally warped to align with the other (Andino et al. 2000).

Given the ubiquity of the domains in which this invariance is required, there are hundreds of papers on the topic. However, recent empirical evidence strongly suggests that a 40-year-old technique, DTW, works exceptionally well (Ding et al. 2008).

2.1.3 Uniform scaling invariance

In contrast to the *localized* scaling that DTW deals with, in many data sets we must also deal with *global* scaling. For example, Fig. 4 shows two yeast cell-cycle gene expression time series which are known to be related (Li et al. 2002). If we try to match the shorter one against the prefix of the longer one, it matches poorly. However, if we globally stretch it by a factor of 1.41, it becomes a much better match.

The main difficulty in creating uniform scaling invariance is that we typically do not know the scaling factor ahead of time, and are thus condemned to testing all possibilities within a given range (Keogh 2003).

2.1.4 Phase invariance

This form of invariance is important when matching periodic time series such as star-light curves (Keogh et al. 2009), gait, heartbeats, etc. It is also important when matching

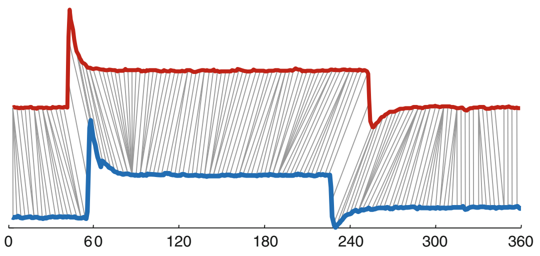


Fig. 3 Two time series of insect behavior matched with invariance to warping. The alignment was calculated by the DTW algorithm

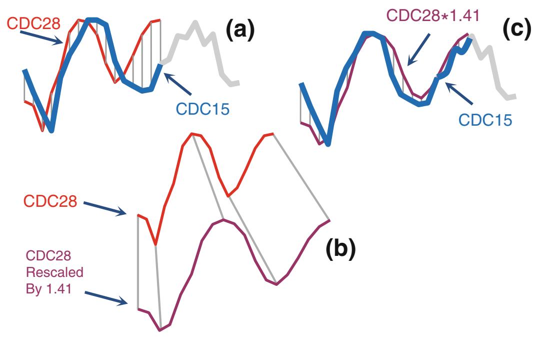


Fig. 4 **a** A full gene expression, CDC28, matches poorly to the prefix of a related gene, CDC15. **b** If we rescale it by a factor of 1.41, it matches CDC15 much more closely (**c**)

two-dimensional shapes which have been converted to one-dimensional "time" series, a representational trick which has gained popularity in recent years (cf. Fig. 6) (Keogh et al. 2009).

Several authors have suggested achieving this invariance by finding a cardinal alignment to which all time series are aligned. However, recent evidence suggests that this may be very brittle (Žunic et al. 2006), and the only currently known way to guarantee phase invariance is to test all possible alignments, as show in Fig. 5.

2.1.5 Occlusion invariance

This form of invariance occurs in domains where a small subsequence of a time series may be missing. As a simple example, imagine that we want to build a robust index for sign language motion capture data, such that the query "*one small step for* **a** *man*" will find "*one small step for man*," in spite of the fact that one sequence is missing the "*a*." Fig. 6 shows a more visually intuitive example.

Time series measures that handle this invariance are very similar to DTW, but have the extra ability to ignore sections (possibility with some penalty) that are too difficult to match (Vlachos et al. 2003). Such measures are often called Longest-Common-Subsequence after the analogue problem in string matching (Vlachos et al. 2003).

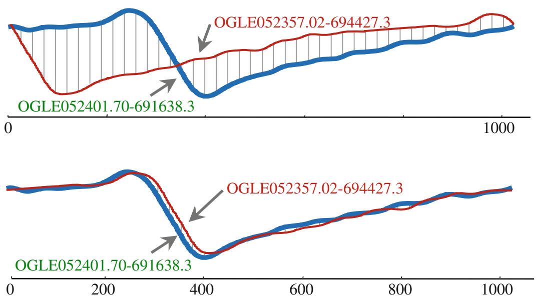


Fig. 5 *Top* Two star light curves are obviously out of phase. *Bottom* By holding one time series fixed, and testing all circular shifts of the other, we can achieve phase invariance

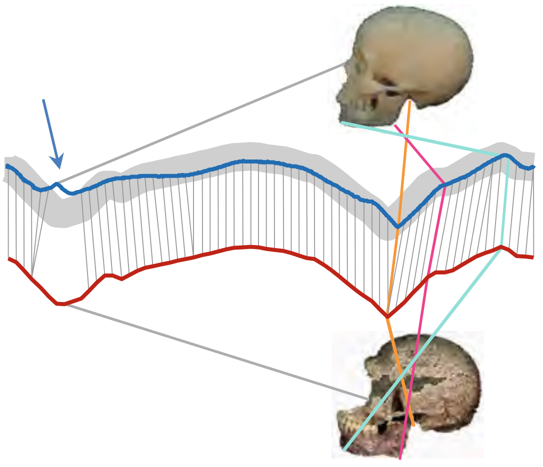


Fig. 6 Occlusion invariance can be achieved by selectively declining to match subsections of a time series. In this case it is robust to the missing nose region of the ancient skull (*bottom*)

2.1.6 Multiple invariances

We conclude with a simple experiment to demonstrate what the reader will have surely anticipated: that some problems require *multiple* invariances.

We created a 3-class star light curve data set with a test set of 128 examples (two of which are shown in Fig. 5), and a training set of 1,024 objects.[2]

While the original light curves have periods that range from hours to weeks, they are normalized to have a standard length of 1,024 to give them *uniform scaling* invariance. Furthermore, since it is the *relative*, not *absolute*, changes in apparent brightness that matter, the data is z-normalized to adjust the *amplitude* and *offset*. The data is then universally phased using a state-of-the-art domain aware algorithm to achieve phase invariance (Protopapas et al. 2006).

We measured the accuracy of this data set using 1-nearest neighbor classifier with Euclidean distance, achieving a respectable 80.47 %. We then checked to see if using DTW to achieve *local scaling* invariance would help, and indeed our accuracy jumped to 86.72 %.

However, rather than stopping here, we decided to test the universal phasing assumption. Suppose we ignored it and tested DTW for all possible alignments/shifts. After testing the *phase*-invariant version of DTW (Keogh et al. 2009), we found that the accuracy increased to an impressive 91.4 %. Clearly, the current universal phasing algorithm does not produce perfect alignments.

Finally, we note the obvious, that in some cases invariances can *decrease* accuracy. An obvious example is in the classification of shapes converted to time series (as in Fig. 6). If we wanted to discriminate between the shapes ‘`p`’ and ‘`d`’, this would be trivial, but adding phase (rotation) invariance would make it impossible.

The complexity invariance that we are proposing in this work is not a special case of any of the above or any combination of the above, but rather a new invariance whose need has escaped the attention of the community.

3 Complexity-invariant distance for time series

We are finally in a position to introduce the core contribution of this work. As we have seen, the research community has proposed a diverse set of invariances for time series in the last two decades. However, there is one invariance that has been missed, *complexity invariance* (for the moment, the reader’s natural intuition of complex as having many peaks and values will suffice.) In many (perhaps *most*) domains, different classes may have widely varying complexities. We can most readily see this on time series derived from shapes (as in Fig. 6). For example, as we show in Fig. 7, leaves range in complexity from a simple pointed ellipse (ovate), shown bottom right, to a jagged-edged familiar maple leaf (palmate), shown bottom left.

The reason why this matters is that pairs of complex objects, even those which subjectively may seem very similar to the human eye, tend to be further apart under

[2] This experiment, like all others in this work, is 100 % reproducible; see Sect. 4 for our experimental philosophy.

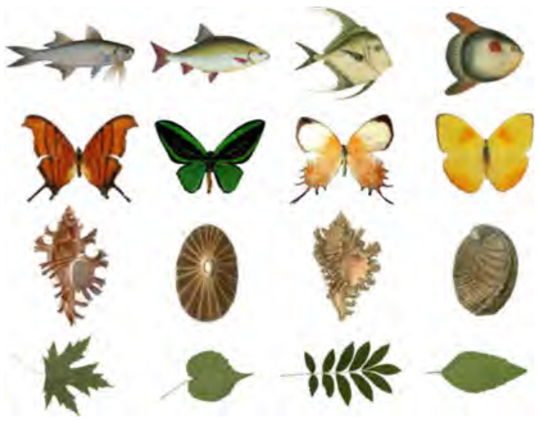


Fig. 7 Examples of objects from a single domain, which have different shape complexities

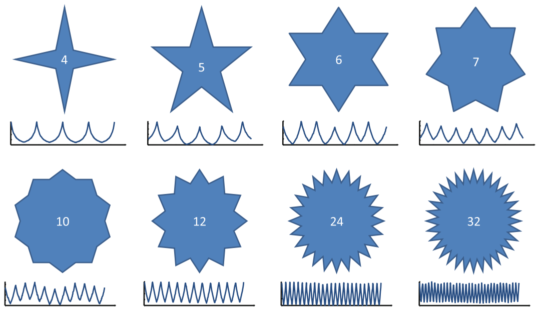


Fig. 8 A set of geometric figures with increasing shape complexity and the respective "time" series extracted by calculating the distance between the central point and its contour

current distance measures than pairs of simple objects. This fact introduces errors in nearest neighbor classification, because complex objects are incorrectly assigned to a simpler class.

We first illustrate the necessity for a complexity-invariant distance for time series using a synthetic data set of figures with different shape complexities. Note that our use of synthetic data here is for clarity; as we hinted at in Fig. 7 and will show later, the complexity problem occurs in many real data sets.

Figure 8 presents some geometric figures with increasing shape complexity. Each figure is labeled with its number of edges for reference. As in Fig. 6, the two-dimensional shapes are converted to a single-dimensional "time" series by calculating the distance between the central point and the figure contour. Figure 8 also presents the z-normalized time series created for each figure.

We calculated distances between every pair of time series in Fig. 8 using Euclidean distance and present the results in Table 1.

We can use the numbers in Table 1 (and the dendrogram created from them, as shown in Fig. 10 *left*) to explore the most similar shapes according Euclidean distance. Considering the figures with simple shapes, the results are intuitive: The 4-edge star is the most similar object to the 5-edge star and vice-versa; the 7-edge figure is the most similar to the 6-edge star, etc. However, when we consider figures with complex shapes, the results are not so clear. The simplest figure, the 4-edge star, is the most

Table 1 Euclidean distance matrix for the geometric figures data set

	4	5	6	7	10	12	24	32
4		1.000	1.122	1.231	1.181	1.048	1.155	1.170
5			1.318	1.068	1.103	1.153	1.165	1.180
6				1.088	1.097	1.103	1.186	1.200
7					1.217	1.199	1.198	1.191
10						1.263	1.195	1.214
12							1.135	1.199
24								1.191
32								

similar to the 32-edge and the 12-edge figures. For the 24-edge figure, even though the 12-edge is the nearest one, the second and third most similar objects are the 4 and 5-edges stars, respectively.

This example illustrates a phenomenon that can be summarized in a simple statement: the distance between pairs of complex time series is frequently greater than the distance between pairs of simple time series. In fact, complex time series are commonly considered more similar to simple time series than to other complex time series they look like.

3.1 CID

Complexity invariance uses information about complexity differences between two time series as a correction factor for existing distance measures. In this section we restrict our discussion to Euclidean distance, and we consider the use of complexity invariance in other distance measures in Sect. 4.4. The Euclidean distance, $ED(Q, C)$, between two time series Q and C, can be made complexity-invariant by introducing a correction factor:

$$CID(Q, C) = ED(Q, C) \times CF(Q, C),$$

where CF is a complexity correction factor defined as:

$$CF(Q, C) = \frac{\max\left(CE(Q), CE(C)\right)}{\min\left(CE(Q), CE(C)\right)}$$

and $CE(T)$ is a complexity estimate of a time series T.

Before discussing how CE can be implemented, it is worth noticing that CF accounts for differences in the complexities of the time series being compared. CF forces time series with very different complexities to be further apart. In the case that all time series have the same complexity, CID simply degenerates to Euclidean distance.

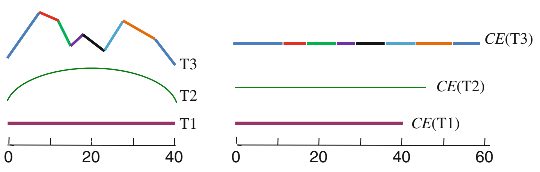


Fig. 9 *Left* Three time series can have their complexity measured by stretching them and measuring the length of the resulting lines (*right*)

Table 2 Matlab code for complexity invariance distance

```
function d = CID(Q, C)
CE_Q = sqrt(sum(diff(Q).^2));
CE_C = sqrt(sum(diff(C).^2));
d = sqrt(sum((Q - C).^2)) * (max(CE_Q,CE_C)/min(CE_Q,CE_C));
```

This formulation leaves only the question of how to measure the complexity. The complexity of a time series can be estimated by a multitude of different approaches, such as Kolmogorov complexity (Keogh et al. 2007; Li and Vitnyi 2008), many variants of entropy (Andino et al. 2000; Aziz and Arif 2006), fractal dimension (Schroeder 2009), the number of zero crossings, etc.

We can consider the desirable properties any such complexity measure should have:

1. It should have low time and space complexity;
2. It should have few parameters, ideally none;
3. It should be intuitive and interpretable.

Given these above desideratum, we now present *one* possible complexity measure. It has $O(1)$ space and $O(n)$ time complexity, is completely parameter-free, and has a natural interpretation. It also was empirically the best on average of the dozen or so possible measures we tried. Nevertheless, we emphasize that we are not claiming this is the optimal measure (if such a thing is even well defined). Recall that the contribution of this work is to point out the complexity invariance problem, and to show an existence proof of a method to mitigate it. We return to this discussion on Sect. 5 when we compare our complexity estimate measure to other measures proposed in literature.

Our approach to estimate the complexity of a time series is very simple. It is based on the physical intuition that if we could "stretch" a time series until it becomes a straight line, a complex time series would result in a longer line than a simple time series. Figure 9 illustrates the intuition of this idea with some examples.

Our complexity estimate can be computed as follows:

$$CE(Q) = \sqrt{\sum_{i=1}^{n-1} (q_i - q_{i+1})^2} \tag{1}$$

The CID approach for complexity estimation can be easily implemented in any programming language. In order to reinforce this statement, Table 2 lists not the pseudo-code algorithm, but the entire Matlab code to compute CID.

Table 3 CID distance matrix for the geometric figures data set

	4	5	6	7	10	12	24	32
4		1.000	1.061	1.229	1.707	1.839	3.843	5.042
5			1.307	1.138	1.700	2.159	4.135	5.422
6				1.096	1.599	1.953	3.982	5.214
7					1.651	1.977	3.744	4.819
10						1.439	2.580	3.396
12							2.018	2.761
24								1.446
32								

Notice that the realization of the complexity estimate in Eq. 1 takes in account differences of consecutive measurements and ignores differences of time when those measurements occurred. Therefore, $CE(T) = 0$ when T is a constant time series. We made this simplification since we apply CID to compare time series with the same sampling rate and number of observations. Thus, all time differences sum up to a constant value. However, extending Eq. 1 to take in account when the observations occurred is a trivial task. Also notice that in order to provide consistent complexity estimates across an entire data set, Eq. 1 requires that the time series under comparison should have normalized amplitudes and offsets.

The code in Table 2 is computationally efficient since computing the complexity estimates is $O(n)$. Therefore, the time complexity of the entire code is $O(n)$. However, we can make this code a little more efficient if the complexity estimates are pre-computed and stored in a table. For example, in an exhaustive k-nearest neighbor search, each query must be compared to the entire database. Complexity estimates for the query and all time series in the database can be pre-computed and stored with negligible overhead. In this case, lines 2 and 3 in Table 2 can be replaced by a simple table lookup.

We can get our first hint of the utility of CID by considering the geometric figures data set of the previous section. Table 3 presents the pairwise CID distances. Note that CID distances tend to be small between time series with similar complexities and increase as we compare time series with different complexities.

We also clustered this data set using hierarchical clustering with average linkage. Figure 10 presents the results for Euclidean distance (*left*) and CID (*right*). Here the results for Euclidean distance are counterintuitive, with complex figures linked directly to clusters formed by simpler figures.

3.2 CID natural problems

The skeptical reader may wonder if the need for a complexity-invariant distance is restricted to our contrived synthetic example. Before applying CID to a large set of data sets, we present in this section two examples of CID applied to natural data.

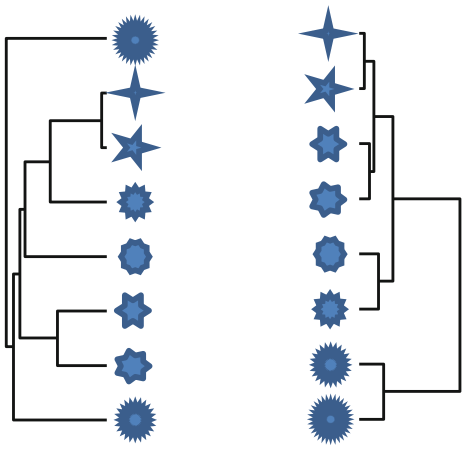


Fig. 10 Hierarchical clustering of geometric figures data set using Euclidean distance (*left*) and CID (*right*)

The first example consists of images of leaves that have been converted to time series (cf. Fig. 6). The classification of leaves is an important problem in agriculture and in understanding biodiversity, and has attracted significant recent interest (Hearn 2009).

Pairs of leaf images with different shape complexity were randomly selected and clustered with Euclidean distance and CID using average linkage. Figure 11 illustrates the results that were obtained.

Even though all the pairs of leaves are quite similar to the human eye, Euclidean distance can only correctly identify two clusters (marked by heavy lines in Fig. 11 *left*). Not surprisingly, these two clusters are formed by the two pairs of leaves with the simplest shapes. Complex leaves do not form their own clusters and are attached directly to clusters formed by simpler leaves. In contrast, CID was able to identify clusters of pairs of leaves with complex shapes, as shown in Fig. 11 *right*.

The second example consists of electrocardiograph (ECG) data, one of the most important applications of time series in the medical domain. ECG is the standard test to

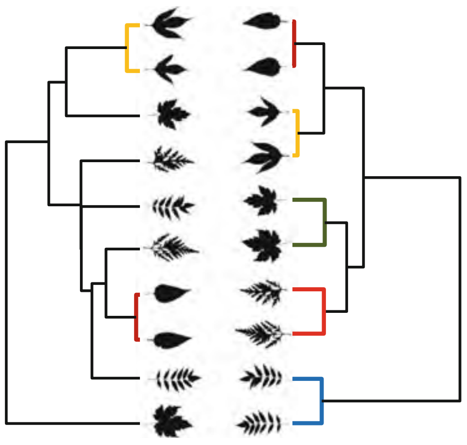


Fig. 11 Hierarchical clustering of leaf images with Euclidean distance (*left*) and CID (*right*)

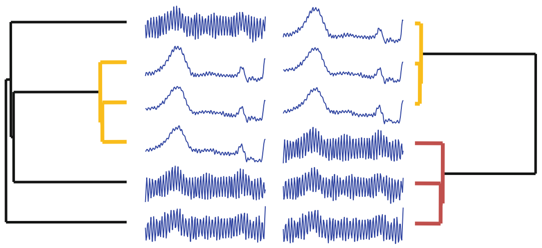


Fig. 12 Hierarchical clustering of ECG heartbeats with Euclidean distance (*left*) and CID (*right*)

diagnose and assess the extension of a number of heart diseases, including myocardial infarction.

We selected two ECG time series with different apparent complexities from PhysioNet (Goldberger et al. 2000), and split the original time series in individual heartbeats. Figure 12 shows that Euclidean distance correctly clustered the simple heartbeats together, but failed to group the complex heartbeats. In contrast, CID correctly identified two distinguished groups, clustering together simple and complex time series in different groups.

We return to the time series extracted from leaf images to get a better intuition of why CID works. The central idea behind CID is the correction factor (CF) defined by Eq. 3.1. Notice that $CF \geq 1$, since it is the ratio of the larger complexity by the smaller complexity. This property makes CID always larger or equal to Euclidean distance, and has two major consequences. The first one will be better explored in Sect. 6 and allows CID to be indexed by virtually all existing techniques proposed for Euclidean distance based on lower-bounding or triangular inequality. The second consequence is that CID makes objects with different complexities further apart, but assigns the same value as Euclidean distance for objects with equal complexities. We can *think* of CID as a measure that uses Euclidean distance for time series with the same complexities, but increases the distance for objects with different complexities.

We can better illustrate this idea with an example. Figure 13 shows the correction factors calculated for the leafs clustered in Fig. 11. As one may expect, same-class leaves have similar complexities, resulting in correction factors close to one. Take as example the maple-shaped leaf *Acer saccharinum*. The correction factor for the these two leaves is only $CF = 1.01$, meaning that the Euclidean distance will be increased by approximately 1 %. In contrast, the distance among the *Acer saccharinum* and other species can be raised up to 325 %. This property of CID avoids the Euclidean distance problem of frequently consider complex objects similar to simple ones.

The reader may be wondering why Euclidean distance frequently considers complex time series to be more similar to simple time series than to other complex (but apparently similar) time series. Figure 14 presents a comparison of three leaves.

Figure 14 presents a comparison of a leaf with a simple shape, *Boehmeria cylindrical*, and a leaf of complex shape, *Ambrosia artemisiifolia* (*top*); and a comparison of two exemplars of *Ambrosia artemisiifolia* (*bottom*). According to Euclidean distance, the distance between *Boehmeria cylindrical* and *Ambrosia artemisiifolia* (the distance between the top two time series) is smaller than the distance between the two exemplars *Ambrosia artemisiifolia* (the two bottom time series).

Fig. 13 Correction factors (CF) calculated for the same leaves clustered in Fig. 11. Same class, and therefore, similar complexity leaves tend to have smaller CFs than different class and different complexity leaves

	1.00	1.23	1.27	1.37	2.53	2.56	3.04	3.43	9.44	10.77
	1.23	1.00	1.04	1.11	2.06	2.08	2.47	2.80	7.69	8.78
	1.27	1.04	1.00	1.08	1.99	2.01	2.39	2.70	7.43	8.48
	1.37	1.11	1.08	1.00	1.86	1.87	2.22	2.51	6.91	7.89
	2.53	2.06	1.99	1.86	1.00	1.01	1.20	1.35	3.73	4.25
	2.56	2.08	2.01	1.87	1.01	1.00	1.19	1.34	3.70	4.22
	3.04	2.47	2.39	2.22	1.20	1.19	1.00	1.13	3.11	3.55
	3.43	2.80	2.70	2.51	1.35	1.34	1.13	1.00	2.75	3.14
	9.44	7.69	7.43	6.91	3.73	3.70	3.11	2.75	1.00	1.14
	10.77	8.78	8.48	7.89	4.25	4.22	3.55	3.14	1.14	1.00

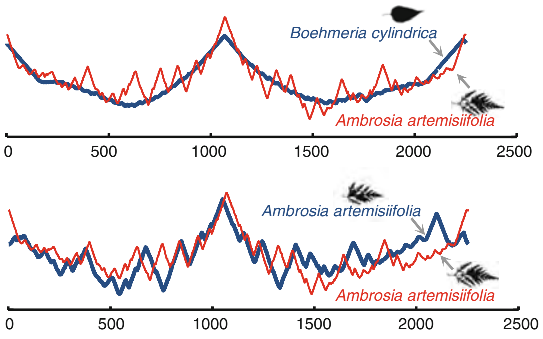


Fig. 14 *Top* A comparison of a simple-shaped leaf (*Boehmeria cylindrical*) and a complex-shaped leaf (*Ambrosia artemisiifolia*) that results in smaller Euclidean distance than when two complex-shaped leaves (*Ambrosia artemisiifolia*) are compared to each other (*bottom*)

The intuition behind this unintuitive result is that simple time series frequently present an "average" pattern that helps to minimize the Euclidean distance. In contrast, complex time series have a large number of peaks, in different quantities, amplitudes and durations. It is very unlikely that all these features match, even to another similar time series, thus the many minor local differences rapidly accumulate to produce a large global Euclidean distance.

3.3 A discussion of our complexity estimate

We are finally in a position to review our ideas on complexity-invariant distance and to prepare the reader for the upcoming sections.

Our main argument is that complexity matters for relevant tasks in time series mining. This follows from the fact that most time series data mining tasks, including classification, clustering, repeated pattern (motif) discovery and anomaly detection use similarity calculations as a subroutine. Thus, any contributions to improve similarity calculations have the potential to have a significant impact. In the remainder of this paper we show strong and statistically significant empirical evidence for classification and clustering in Sects. 4.1 and 4.3, respectively, and anomaly detection in Sect. 7.

Our observations about the need for complexity-invariant distance measures allow for two lines of research, on the complexity estimate and the distance function itself. We evaluated the complexity invariance with Euclidean distance and DTW. We show in Sect. 4.4 that CIDDTW, CID with DTW as base distance, can statistically outperform

DTW for time series classification. Moreover, we prove in Sect. 6 that the complexity-invariant distance can inherit the lower-bounding and triangular inequality properties of its base distances. In practice, it means that the complexity-invariant distance avail of all existing indexing techniques available for its base distance.

The second possible extension is the complexity estimate. In this work we proposed a very simple, yet effective, complexity estimate. However, we note this is not the central contribution of this paper. We naturally need some complexity estimate to show that the complexity invariance matters, and the complexity estimate of Eq. 1 provided the best empirical results. In particular, we demonstrate that our simple complexity estimate outperforms nine other well-known estimates for time series classification in Sect. 5. Nevertheless, the discovery of an even better measure that can outperform our results would only strengthen our original observations.

We believe the results of Sect. 5 provide strong arguments in favor of our complexity estimate. Although simple, our measure provides excellent estimate for time series complexity and has just two simple requirements. The times series under comparison should have the same length (also required by Euclidean distance) and should be z-normalized. Z-normalization is a necessary procedure to meaningfully compare times series in most domains, as discussed in (Rakthanmanon et al. 2012).

We believe our complexity estimate works so well because it is a good estimate of the intrinsic dimensionality of the time series. The intrinsic dimensionality of a time series is frequently significantly less than recorded dimensionality (Hu et al. 2011). This is the reason that time series are typically amiable to efficient indexing. It is generally possible to index time series of length say 1,024, in spite of the fact that classic curse-of-dimensionality results suggest that you cannot index data with dimensionality much greater than 10 or 20.

We performed a simple experiment to show that there is no strong connection between our complexity estimate and the recorded dimensionality of a time series. We created a random walk with 1,024 observations and down sampled it by a factor of 4 and by a factor 16. We calculated the complexity estimate according Eq. 1 for the original time series and the two down sampled versions. The complexity estimated varied only 0.96 % (factor of 4) and 3.92 % (factor of 16).

This experiment suggests the complexity estimate is a property of the shape itself, not of the recorded dimensionality that happened to be used to record the time series. This is a desirable property, allowing us to meaningfully compare data that was sampled at different sampling rates.

We believe the complexity estimate is related to the intrinsic dimensionality of a time series. In general, the higher the intrinsic dimensionality the higher the complexity estimates. Beyond this it is much more difficult to say anything concrete. The intrinsic dimensionality of a time series depends on the underlying model used. A pure sine wave has a low intrinsic dimensionality in DFT space, but a high intrinsic dimensionality if modelled in the Haar wavelet space. Note however that in a dataset where everything has identical intrinsic dimensionality, it is still possible that two classes can have very different complexities, and thus CID can still help.

4 Empirical evaluation

We begin by stating our experimental philosophy. We have designed all experiments such that they are not only reproducible, but easily reproducible. For example, if a reader wishes to reproduce any figure in this paper, they can simply run the script we provide to do so. The scripts are designed to be easily expandable in order to include new data sets or distance measures, for example.

To this end, we have built a webpage which contains all data sets and code used in this work, together with spreadsheets which contain the raw numbers displayed in all of the figures. In addition, this webpage contains many additional experiments which we could not fit into this work; however, we note that this paper is completely self-contained. We have made the script available in Matlab, since this is essentially free for anyone in academia, and a completely free clone version (Octave) is available.

4.1 Basic accuracy

We performed our evaluation using a large set of time series classification data sets. In total, the evaluation includes 43 data sets from different domains, including medicine, entomology, engineering, astronomy, signal processing, and many others. Twenty of these data sets have been available for 5 years at (Keogh et al. 2006), and used in more than one hundred papers.

We tested the accuracy of both Euclidean distance and CID using the 5-year-old predefined splits for 20 of the data sets that came from (Keogh et al. 2006), and defined similar splits for the remaining 23 data sets. We emphasize that in the latter case, we split the data just once, before testing any accuracy results. A detailed description of the data sets is available at the paper website (Batista 2011).

Recall that neither distance measure has any parameters to be learned in the training stage. Our experimental results are visually summarized in Fig. 15; the raw numbers are available at (Batista 2011).

We believe that the results speak for themselves. However, notice that we are publishing results for all classification benchmark data sets we have at hand, and not for a subset of any kind. Nevertheless, CID improved classification accuracy in 32 of the 43 data sets, had the same performance as Euclidean distance in 3 data sets and decreased the accuracy in only 8.

We analyzed CID and Euclidean distance results using the Wilcoxon signed-ranks test. This is the statistical test recommended by Demšar (2006) to compare two algorithm over multiple data sets. The test indicates that the null-hypothesis, i.e. both distances have the same performance, should be rejected with 95 % confidence level. Therefore, CID outperforms Euclidean distance with statistical significance.

We conclude this section by observing that CID improves accuracy because of the reasons we claim; i.e., CID avoids considering overly simple objects as similar to complex objects. In order to verify this hypothesis we performed a simple experiment: we collected data on all time series (from all data sets) that were misclassified by Euclidean distance and correctly classified by CID, and compared the complexity of the nearest neighbors provided by each distance measure. In 87.54 % of the cases, CID

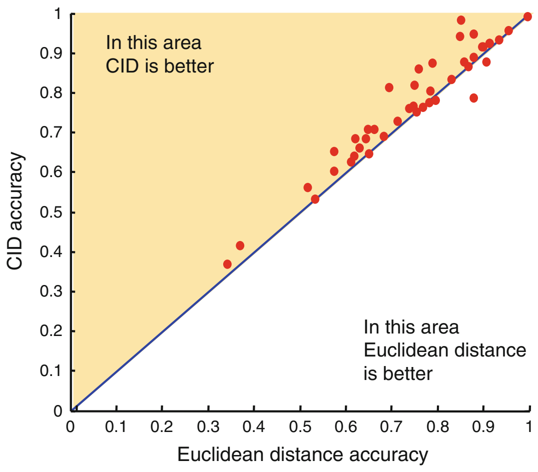


Fig. 15 CID compared with Euclidean distance using 1-nearest neighbor classifier accuracy. Each point represents one data set, with its *x-axis* value being its Euclidean distance accuracy, and its *y-axis* value being its CID accuracy

assigned a more complex nearest neighbor than Euclidean distance did. It is reasonable to assume that, if CID could improve accuracy for any other (unknown) reason, we would expect that only approximately 50 % of the CID nearest neighbors would be more complex than the Euclidean distance nearest neighbors.

4.2 The Texas sharpshooter fallacy

As impressive as these results are, we must be careful to avoid a simple logic error that seems pervasive in time series classification papers. Many papers in the last three or four years test their algorithm/distance measure on all 20 data sets in the UCR archive, and note that their method wins on some, ties on many, and loses on some. They then claim "*our method has better accuracy in some domains; therefore we could use it in those domains, and thus it has value.*" However, it is not useful to have an algorithm that can be accurate on *some* problems unless you can tell in advance on *which* problems it will be more accurate (this is a subtle version of the Texas sharpshooter fallacy.)

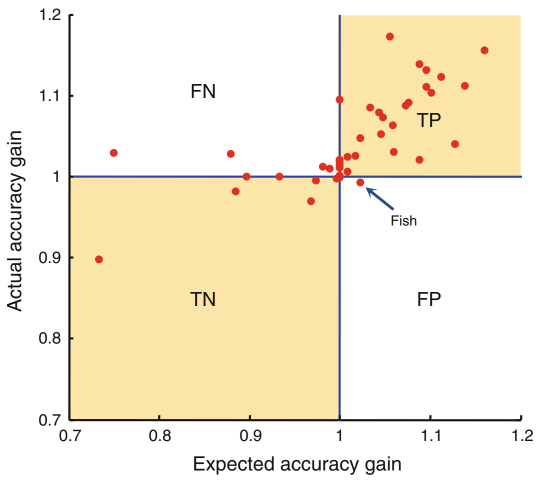
In order for the algorithm to be useful, we must show that we can predict ahead of time when our method will have superior accuracy. The obvious way (but not the only possible way) to do this is to test the accuracy of both Euclidean distance and CID looking *only* at the training data, and using these results to choose which algorithm to use to classify an object from the testing set.

We measured the accuracy of Euclidean distance and CID on training data using leaving-one-out cross-validation and calculated the expected accuracy gain as:

$$gain = \frac{accuracy\ CID}{accuracy\ Euclidean}.$$

Obviously, gain values greater than one indicate that we expect CID will outperform Euclidean distance on a given data set; and gain values lower than one indicate the opposite. We also measured the *actual* accuracy gain (or loss) using testing data. The

Fig. 16 Expected accuracy gain calculated on training data versus actual accuracy gain on testing data



results are shown in Fig. 16. Note that the figure is essentially a real-valued version of a contingency table, and we have labeled the four regions with the four familiar labels. Let us consider the four cases we observe in the order of the evidence of utility for CID:

TP In this region we claimed ahead of time that CID would improve accuracy, and we were correct. Gratifyingly, the vast majority of data points are in this region;

TN In this region we *correctly* claimed ahead of time that CID would decrease accuracy;

FN In this region we claimed ahead of time that CID would decrease accuracy, but the accuracy actually increased. This represents a lost opportunity to improve, but note that we are no worse off than if we had not tried CID;

FP This region is the only truly bad case for our method. Data points falling in this region represent cases where we thought we could improve accuracy, but did not. The good news is that only one case falls into this category, the Fish data set. For this data set we predicted an improvement of only 2 % and obtained an accuracy reduction of 0.7 %. In general, all cases near point (1, 1) are not of special interest, since they represent marginal increases/decreases in accuracy.

4.3 Clustering with CID

As one may have anticipated by Figs. 10, 11 and 12, CID is also useful for clustering. We performed an evaluation using representatives of hierarchical, partitional and spectral clustering, the three most important clustering paradigms. For hierarchical clustering, we use the agglomerative hierarchical clustering with average linkage, and divide the resulting dendrogram in the smallest height that results the desired number of clusters. For partitional clustering, we use the classic Partitioning Around Medoids (PAM) algorithm (Kaufman and Rousseeuw 2005). Finally, for spectral clustering, we use the popular *normalized spectral clustering* (Ng et al. 2001).

We compared CID and Euclidean distance in the same 43 data sets. We used the ubiquitous Rand index (Rand 1971) as the main measure to assess our results. In brief, the Rand index is based upon counting pairs of points (time series objects, in our case) on which two clusterings agree or disagree. In our experiments, one clustering is given by the clustering algorithm and the other is the naturally occurring classes in each data set. Given a set of cluster $V = \{V_1, \ldots V_k\}$ outputted the clustering algorithm and the set of classes $C = \{C_1, \ldots C_k\}$, there are four categories in which a pair of points might fall:

- a: number of pairs in the same cluster in V and in the same class in C;
- b: number of pairs in different clusters in V and in different classes in C;
- c: number of pairs in the same cluster in V but in different classes in C;
- d: number of pairs in different clusters in V but in the same class in C.

In simple terms, a and b are the number of agreements between the two clustering, and c and d are the number of disagreements. The Rand index is defined as:

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}.$$

We start presenting our results with hierarchical clustering in Fig. 17. CID significantly improved the Rand index for most of the data sets. In numbers, CID resulted in larger Rand indexes for 28 data sets, in equal for 5 data sets and lower values for 10 data sets. However, Fig. 17 clearly shows that for the 10 data sets that CID lost to Euclidean distance, in only 4 of them the differences are significant.

We also compared the performance of CID and Euclidean distance for hierarchical clustering using the Wilcoxon signed-ranks test. The test indicates that the null-hypothesis, i.e. both distances have the same performance, should be rejected with 95 % confidence level. Therefore, CID outperforms Euclidean distance with statistical significance.

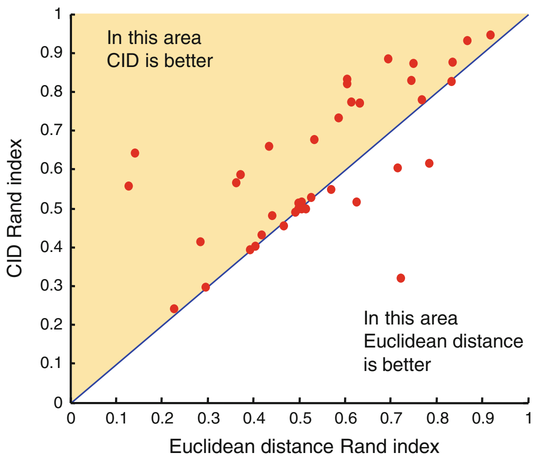


Fig. 17 CID compared with Euclidean distance using hierarchical clustering algorithm. Each point represents one data set, with its *x-axis* value being the Euclidean distance Rand index, and its *y-axis* value being the CID Rand index

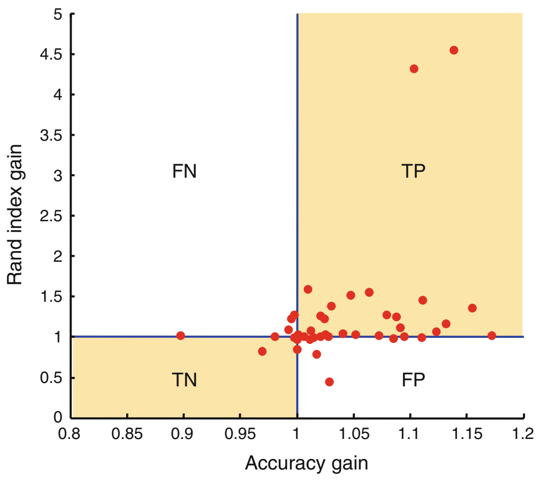


Fig. 18 Accuracy gain calculated on training data versus Rand index gain for hierarchical clustering

Once again, we face the problem of the Texas sharpshooter fallacy, that is, the utility of these results is conditioned to the the ability of predicting in which data sets we can improve clustering ahead of time. Fortunately, for most data sets that CID improves classification accuracy, it also improves clustering Rand index. Obviously, such analysis would require class labeled data. Although such labels may not be available for every training data, for most domains it is possible to obtain partially labeled data, and use such data to estimate accuracy gain.

Figure 18 shows the relationship between accuracy gain (calculated in the training set) and Rand index gain for the hierarchical clustering. The plot was divided into four regions as in Fig. 16, and most of the points fall in the TP area. The FP area represents the region in which we incorrectly predicted a Rand index improvement based on an accuracy gain, and there are only a few points is this area, most of them representing small marginal Rand index losses.

Figure 19 presents the results for partitional (PAM) algorithm. In order to reduce possible performance differences due chance, we ran paired experiments. It means that all experiments for CID and Euclidean distance were run with the exact same settings. For instance, the same initial medoids (randomly chosen) were assigned to both distances in each run. We also repeated the experiments 10 times to reduce variance.

In summary, CID presented mean Rand index values greater than Euclidean distance for 31 data sets, same mean Rand index for 1 data set and lower mean Rand index values for 11 data sets. However, for only one of the 11 data sets the difference is significant. We also used the Wilcoxon signed-ranks test to compared the performance of CID and Euclidean for partitional clustering. The test indicates that CID outperforms Euclidean distance with statistical significance at 95 % confidence level.

Figure 20 shows the relationship between accuracy gain and Rand index gain. Again, most of the points fall in the TP area, and only a few points are in the FP area, all of them representing small marginal Rand index loses.

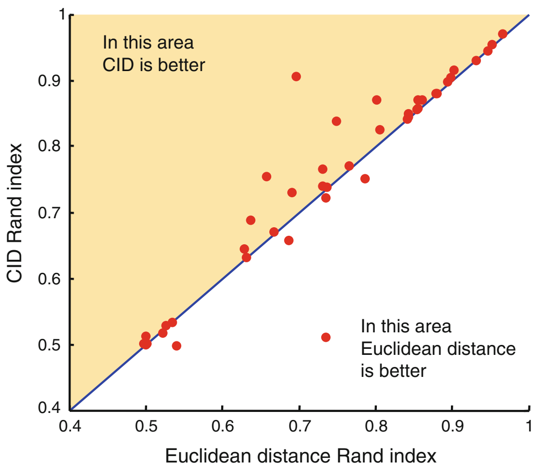


Fig. 19 CID compared with Euclidean distance using *k*-medoids (PAM) clustering algorithm. Each point represents one data set, with its *x-axis* value being the Euclidean distance Rand index, and its *y-axis* value being the CID Rand index

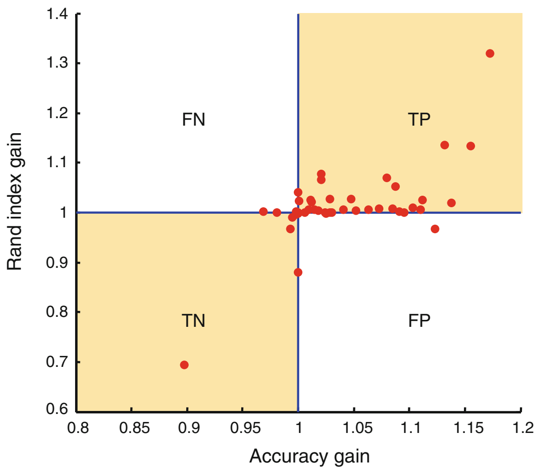


Fig. 20 Accuracy gain calculated on training data versus Rand index gain for the *k*-medoids (PAM) clustering algorithm

We finally present the results for spectral clustering in Fig. 21. CID improved Rand index compared to Euclidean distance for 23 data sets, presented the same performance for 2 data sets and presented inferior Rand index for 18 data sets. Although the win/draw/loss numbers suggest that CID beats Euclidean distances for a small margin, an analysis of Fig. 21 shows that, in general, the wins have larger margins than the losses.

Once again, we analyzed CID and Euclidean distance results using the Wilcoxon signed-ranks test. The test indicates that the null-hypothesis should *not* be rejected with the results obtained for spectral clustering.

Figure 22 shows the relationship between accuracy gain and Rand index gain. Again, most of the points fall in the TP area, and only a few points are in the FP area, most of them representing small marginal Rand index losses.

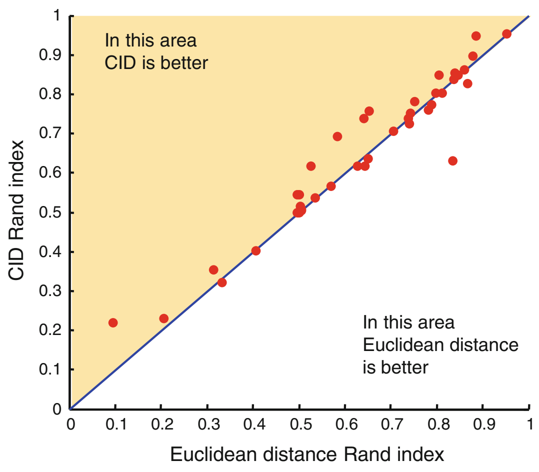


Fig. 21 CID compared with Euclidean distance using spectral clustering algorithm. Each point represents one data set, with its *x-axis* value being the Euclidean distance Rand index, and its *y-axis* value being the CID Rand index

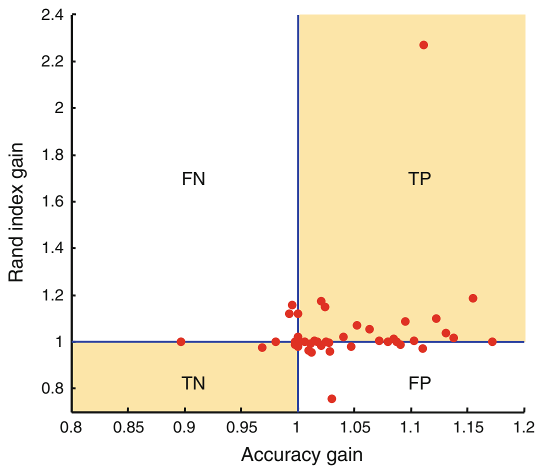


Fig. 22 Accuracy gain calculated on training data versus Rand index gain for the spectral clustering algorithm

4.4 Is there some other way?

The reader may wonder if the results of CID are simply a reinterpretation of some other invariance. In particular, it might be imagined that DTW could eliminate some of the effects of differing complexities. In brief, the answer is no. Complexity invariance is *not* subsumed by any other known invariance. To see this we applied the correction factor of Sect. 3 to DTW, and named this new measure of CIDDTW. We then compared the classification accuracy of both DTW and CIDDTW using a one-nearest neighbor classifier. Figure 23 presents the results we obtained.

For DTW, we searched for the best Sakoe-Chiba band size using the one-nearest neighbor classifier on training data only, and used lower-bounding to speed up calculations (Keogh 2002). For CIDDTW we did *not* search for the best band size, and simply used the window sizes obtained for DTW. Therefore, the results in Fig. 24

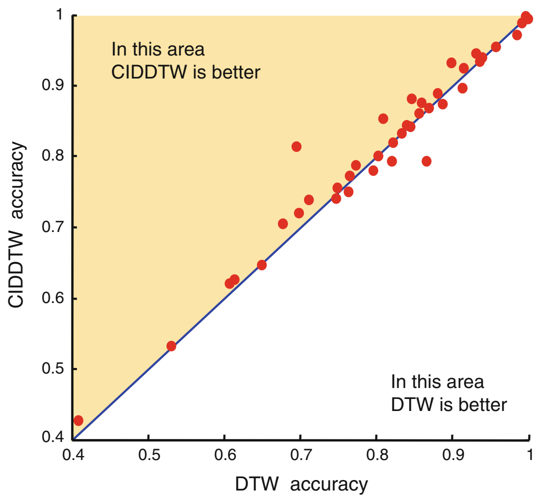


Fig. 23 CIDDTW compared with DTW distance using one nearest neighbor classifier accuracy

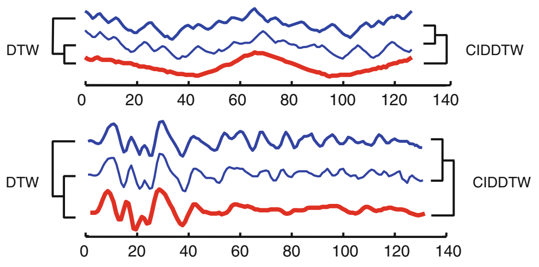


Fig. 24 Some examples misclassified by DTW with a one-nearest neighbor classifier, and correctly classified by CIDDTW. Data sets are SwedishLeaf (*top*) and FaceAll (*bottom*)

might be pessimistic for CIDDTW. Nevertheless, CIDDTW outperformed DTW in 24 data sets, drew in 7 and lost in 12.

We compared DTW and CIDDTW results using the Wilcoxon signed-ranks test. The test indicates that the null-hypothesis should be rejected with 95 % confidence level. Therefore, CIDDTW outperforms DTW distance with statistical significance.

DTW is one of the most popular distance measures for time series due to its ability to align, in a non-linear manner, time series that are locally out of phase (cf. Fig. 3) (Ding et al. 2008; Keogh 2002). This feature can be used to *partially* deal with the complexity problem. Intuitively, DTW can be used to align peaks and valleys of complex time series, reducing the distance between them. However, complex time series frequently have *different amounts* of peaks and valleys and aligning some of them do not fully solve the problem. In Fig. 24 we show that there are instances where DTW can be "fooled" by simple time series in several classification domains.

We would like to make our point clear here. If the DTW distance completely subsumed the complexity invariance then CIDDTW would not be able to outperform the DTW distance in some data sets, since the complexity invariance would be fixing something that was already incorporated by the DTW distance.

Another possible question is if CID *alone* can beat DTW for classification accuracy. We performed such comparison and show the results in Fig. 25.

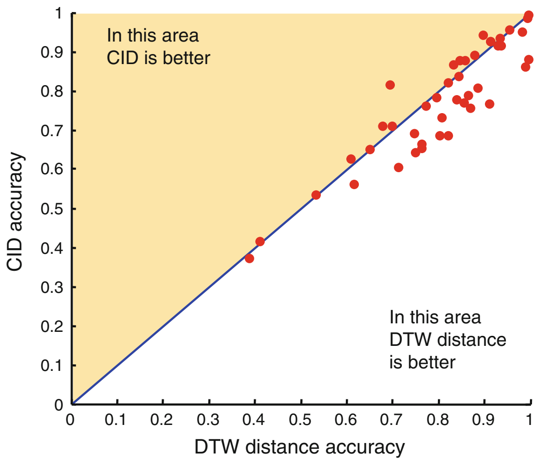


Fig. 25 CID compared with DTW distance using one nearest neighbor classifier accuracy

It is clear from Fig. 25 that DTW beats CID for most of the data sets. This is an expected result, since it is well-known that several time series domains can benefit from some degree of nonlinear time stretching, a problem that is not directly tackled by CID. At a first glance, DTW seems to be the distance of choice for most data sets. However, besides classification accuracy, time complexity is also a variable that must be taken in account.

We must notice that CID is *essentially free* compared to Euclidean distance. By essentially free we mean that the computational cost and the implementation cost, as illustrated by the three-line code in Table 2, are negligible. For instance, in order to run the classification experiments with all 43 data sets,[3] Euclidean distance required 193.12 s while CID required just 193.95 s, including the calculation of the complexities for each time series. In contrast, DTW distance required 91,108.35 s (i.e. more than 25 h) to finish the same experiment.

In addition to time complexity for in-memory search, we should also consider the access time when data is disk-resident, a central concern for several Data Mining applications. As we will demonstrate in Sect. 6, CID can utilize a myriad of indexing techniques proposed for Euclidean distance that rely on triangular inequality or lower-bounding properties. In contrast, it is well-known that DTW does not obey the triangular inequality and has resisted most attempts of indexing. Successful attempts to speed up DTW calculations are mostly based on lower-bounding (Keogh 2002).

5 Other measures for complexity

There are literally dozens of complexity measures that are applicable to time series data. Most of these measures are variations of concepts from information theory (espe-

[3] The running times were obtained in a Intel Core i7, 3.4 Ghz, 8 Gb of RAM computer. These results are also reproducible, the paper website has ready-to-use scripts that report running times and spreadsheet results with detailed execution times for each data set.

cially entropy), Kolmogorov complexity and chaos (fractal) complexity. In this section, we compare the complexity estimate measure used in CID with nine other complexity measures for time series. We provide a short description of each measure:

Fractal dimension This complexity measure uses the concept of fractals and estimates the amount of self-similarity existing in each time series. We use the popular correlation dimension (Schroeder 2009) calculated over a set of time series subsequences extracted with a sliding window;
Absolute difference This complexity measure is similar to the one used in CID; however we compute the absolute differences between consecutive observations, instead of the squared difference. More formally, $CE(Q) = \sum |q_i - q_{i+1}|$;
Compression This complexity measure approximates the Kolmogorov complexity with the Lempel–Ziv compression length. Each time series is first converted to symbols using SAX (Lin et al. 2003) and later compressed using a file compression utility. The complexity estimate of the time series is simply the compressed file size in bytes;
Pseudo-pairs This complexity measure was formulated to directly quantify the idea that "pairs of complex objects tend to be further apart". A subset of random points is removed from each time series, and the time series is interpolated back to the initial number of observations. The original time and its slightly changed counterpart form a pseudo-pair, and the complexity estimate is the Euclidean distance between them. Due to the fact it is a stochastic complexity measure, in our experiments we repeated this process and used the averaged complex estimates over 20 runs;
Edges This complexity measure uses the number of edges, that can also be interpreted as the number of trend changes or the number of times the first derivative changes sign, as a complexity estimate;
Zero-crossings The number of zero-crossings is the number of times the signal crosses an imaginary zero line, that is, the signal changes sign. It is a simple and well-known complexity estimate. In the area of speech analysis it is frequently used to detect voiced segments apart from unvoiced sounds and noisy breaks (Rabiner and Schafer 1978);
Spectrum This complexity measure uses the idea that a complex signal should be composed by high amplitude peaks in a multitude of frequencies. This complexity measure tries to quantify this idea by computing the area under the frequency spectrum;
Fourier entropy Fourier or spectral entropy (Rezek 1998) is obtained by interpreting the frequency spectrum of a time series as a probability density function. This is obtained by normalizing the frequency spectrum resulting in an area under the spectrum equals to one. The complexity estimate is the entropy ($\sum_f(-p_f \log(p_f))$) of the probability density function;
Permutation entropy This complexity measure is calculated as the entropy of a set of patterns (Bandt and Pompe 2002). Those patterns are obtained by generating all permutations of natural numbers between 0 and $n - 1$, being n a parameter value usually chosen in the interval $[3 \ldots 7]$. For instance, for $n = 3$, the valid permutations are $\{[0, 1, 2], [1, 0, 2], \ldots, [2, 1, 0]\}$. The pattern $[0, 1, 2]$ should be interpreted as a sequence of three observations in a time series in which the second

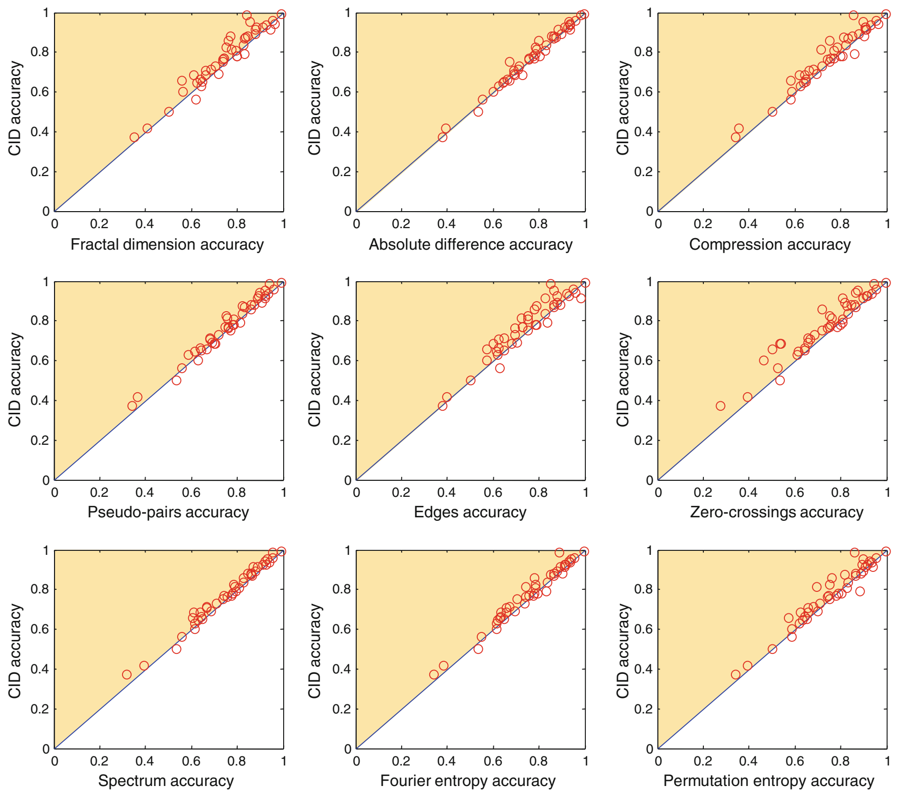


Fig. 26 A comparison of nine complexity measures. The complexity measure used in CID obtained the best average results

observation is greater than the first one, and the third observation is greater then the second one; the pattern [1, 0, 2] should be interpreted as a sequence of three values where the second observation is smaller than the first one and the third observation is greater than the first one; and so on. The probability of each pattern is obtained by running a sliding window of size n across the time series and counting the occurrences of each pattern. The complexity estimate is the entropy of the set of patterns.

Figure 26 presents a classification accuracy comparison among the CID complexity estimate and nine other complexity measures using 1-nearest neighbor classifier. CID obtained the best average results over all other complexity measures in the evaluation. In addition, remember that the complexity estimate used in CID can be computed in linear time, has no parameters and is easily interpretable.

6 Useful CID properties

As we are producing a new distance measure, the vast majority of our empirical evaluation has focused on questions of classification and clustering *accuracy* (cf. Sects. 4.1, 4.2 and 4.3). However, given that we have demonstrated that CID is an accurate measure, the next natural question to ask is about its efficiency. As the CID measure is only $O(n)$, it is clearly efficient for simple main memory similarity search problems. In fact, the time difference between Euclidean distance and CID is only perceptible with very careful experiments.

However, the issue is less clear for disk-resident problems, or for higher-level problems that require (at least in principle) a number of comparisons that are quadratic in the number of objects. Examples of such problems include motif discovery (Mueen et al. 2009), some types of clustering and outlier detection (Protopapas et al. 2006; Yankov et al. 2008).

Essentially all algorithms in the literature that mitigate the potential intractability of disk-resident or higher-level data mining use one (or both) of two well-known ideas, *lower bounding* and the *triangular inequality*. Many of the novel distance measures proposed for time series do not (or initially did not) allow leveraging off lower bounding and the triangular inequality (Ding et al. 2008). For example, the rotation invariance that greatly improved the classification accuracy of star light curves in Sect. 2.1.6 does not allow triangular inequality acceleration (Keogh et al. 2009), and DTW initially only allowed very weak lower bounds until the invention of envelope-based lower bounds (Keogh 2002). In this section we consider each speedup technique with reference to CID.

6.1 Lower bounding of CID

Since the introduction of the GEMINI framework (Faloutsos et al. 1994) the idea of lower bounding has been a cornerstone technique for speeding up the indexing and mining of time series (and many other kinds of data). The basic idea is to produce a low-dimensionality representation of the data and produce a distance measure defined by the reduced data such that:

$$D_{reduced\ data}\,(A, B) \leq D_{original\ data}\,(A, B)$$

Once this has been done, it is trivial to index the data with any off-the-shelf multi-dimensional index structure. While the details differ slightly for each index structure, the basic idea is always the same. The query is projected into the same reduced dimensionality space, and using $D_{reduced\ data}$ we search the approximate data (held in the main memory). We then proceed to load the most promising object from the disk and measure its distance to the query in the original dimensional space, updating the *best-so-far* variable. We continue to retrieve the most promising candidates, updating the *best-so-far* if appropriate, until the next most promising candidate has a value (measured in the $D_{reduced\ data}$ space) that is greater than the *best-so-far*. At this point we can admissibly abandon the search.

Can we use lower bounding with CID? The answer is affirmative and general. We can use *any* of the dozens of lower bounding representations for time series (Ding et al. 2008) with CID by only changing a single line of code. The intuition behind this result is to note that CID can only be greater than or equal to ED. For example, the first paper in the GEMINI framework used the Fourier Transform (DFT) for lower

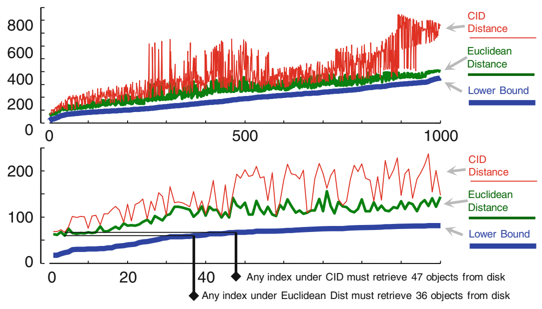


Fig. 27 *Top* A visual intuition of using *lower* bounding to accelerate disk-based search. *Bottom* A zoom-in of the above. For ED or CID we must keep retrieving objects from the disk until the *smallest* true distance calculated is less than the next candidate's *lower-bound* distance. For ED, this happens after we have seen 36 objects, and for CID after 47 objects

bounding (Faloutsos et al. 1994):

$$D_{DFT}(A, B) \leq D_{original\ data}(A, B)$$

But CID is always greater than or equal to ED, hence:

$$D_{DFT}(A, B) \leq D_{original\ data}\,(A, B) \leq D_{CID}(A, B)$$

So we automatically inherit the lower bounding:

$$D_{DFT}(A, B) \leq D_{CID}(A, B)$$

We must regard this news with some caution. It is always possible to create a lower bound to *any* distance measure by hard coding $D_{reduced\ data}$ to zero. In that case, the index structure degenerates to a brute force search. What we need is for the lower bound to be a *tight* estimate of the distance. We can see how we fare here with a simple experiment. We created a database of 1,000 random walk time series of length 128. We approximated the data with an eight-dimensional PAA approximation.[4] We sorted the main-memory approximations of the candidates by listing the *most promising first*, producing the heavy (blue) line in Fig. 27.

The first item in this list has a smallest value of 16.8. This is not a particularly tight lower bound, as the true ED, plotted above it in medium (green) is 63.1, and the CID plotted above them, both in light (red), is 68.6.

Note that we can tell from this plot the best possible pruning for any indexing structure given our assumed parameters. We can prune off at most 964 objects under ED, and at most 953 objects under CID. For all intents and purposes, these numbers are the same: CID allows 98.9 % of the pruning that ED allows. Furthermore, for larger data sets (recall this was a mere 1,000 objects) this percentage grows ever closer to 100 %.

In summary, CID can use the hundreds of indexing and mining algorithms that exploit ED lower bounding (Ding et al. 2008; Faloutsos et al. 1994) with the most trivial coding effort and a negligible loss of efficiency.

[4] For data lengths/reduced dimensionality that are powers of two, PAA is exactly equivalent to Haar wavelets (Ding et al. 2008).

6.2 CID obeys the ρ-relaxed triangular inequality

There are literally dozens of indexing and data mining algorithms for metric spaces that rely on triangular inequality to prune distance calculations (Chávez et al. 2001; Elkan 2003; Moore 2000). Although the details of how triangular inequality is employed vary for different algorithms, we consider a concrete technique to illustrate the general idea.

6.2.1 A brief review of Orchard's algorithm

One of the simplest indexing techniques known is Orchard's algorithm (Li and Vitnyi 2008; Orchard 1991). The general idea behind it is to store a table with the distances between every pair of objects in a database.[5] In order to decide which database instance is the nearest neighbor to a user-given query Q, Orchard's algorithm first calculates the distance $D(Q, C_i)$ between Q and a randomly chosen database instance C_i.

Using $D(Q, C_i)$ and the triangular inequality property, we can safely prune some distance calculations between Q and other training instances. Suppose that we want to know if we can prune the distance calculation to a given instance C_j. Using triangular inequality:

$$D(C_i, C_j) \leq D(C_i, Q) + D(Q, C_j)$$

Reordering the last equation to isolate the distance between the query object and the pruning candidate C_j, $D(Q, C_j)$:

$$D(Q, C_j) \geq D(C_i, C_j) - D(C_i, Q)$$

Note that when we have:

$$D(C_i, C_j) \geq 2D(C_i, Q) \tag{2}$$

we can safely conclude that:

$$D(Q, C_j) \geq D(C_i, Q).$$

Therefore, when the distance between the database instances C_i and C_j is at least twice the distance between C_i and Q, we can safely prune C_j, since it cannot be closer to the query object Q than C_i. Notice that no distance calculations were necessary at classification time, just constant time table lookups. This is because $D(C_i, C_j)$ is a known quantity since Orchard's algorithm stores a table with distances between all pairs of training instances. Figure 28 illustrates this idea.

When a distance calculation cannot be pruned, such as in the case of C_k in Fig. 28, the distance $D(C_k, Q)$ is calculated and if $D(C_k, Q) < D(C_i, Q)$, C_k becomes the new nearest neighbor currently known (i.e., the *best-so-far*), consequently reducing the pruning radius.

[5] Therefore, Orchard's algorithm requires $O(m^2)$ in space, where m is the number of database objects. However, Ye et al. (2009) shows how to significantly reduce the space requirement, while producing nearly identical speedup.

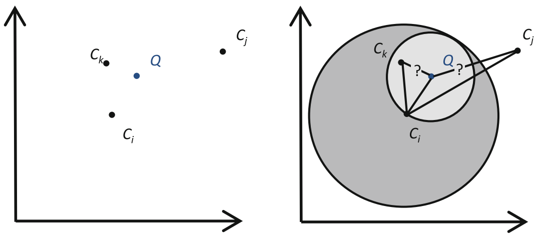


Fig. 28 *Left* Assume we know the pairwise distances between C_i, C_j and C_k. A newly arrived query Q must be answered. *Right* After calculating the distance $D(Q, C_i)$ we can conclude that items with a distance to C_i less than or equal to $2 \times D(Q, Ci)$ (i.e., the *dark gray area*) *might* be the nearest neighbor of Q, but everything else, including C_j in this example, can be excluded from consideration

Our interest in Orchard's algorithm is due to the fact that this algorithm relies *exclusively* on the triangular inequality to improve search performance. However, many other indexing schemes are strongly related (Hjaltason and Samet 2003).

6.2.2 The relaxed triangular inequality

CID does *not* obey the triangular inequality; however, it does obey a relaxed version of this property:

$$D_{CID}(A, B) \le \rho(D_{CID}(A, C) + D_{CID}(C, B))$$

with $\rho = CF(A, B)$. We postpone a formal proof of this property to Appendix A.[6]

The ρ-relaxed triangular inequality property implies that we can search in CID space using the same algorithms designed for metric spaces, after some scaling (Chávez et al. 2001). In practice, all metric indexing techniques can be adapted to CID by changing just a few lines of source code. For concreteness we illustrate this with Orchard's algorithm.

Suppose we have calculated the distance $D_{CID}(Q, C_i)$, and want to know if an instance C_j can be safely pruned.

We start with the ρ-relaxed triangular inequality:

$$D_{CID}(C_i, C_j) \le \rho(D_{CID}(C_i, Q) + D_{CID}(Q, C_j))$$

and reorder to isolate $D_{CID}(Q, C_j)$ on the left-hand side of the equation:

$$\rho D_{CID}(Q, C_j) \ge D_{CID}(C_i, C_j) - \rho D_{CID}(C_i, Q)$$

[6] Notice that $\rho = CF(A, B)$ is not a bounded value; however, for indexing purposes this fact has no major consequences. ρ-relaxed triangular inequality implies 2ρ-inframetric inequality. This means that the following property also holds: $D_{CID}(A, B) \le 2\rho \max(D_{CID}(A, C), D_{CID}(C, B))$.

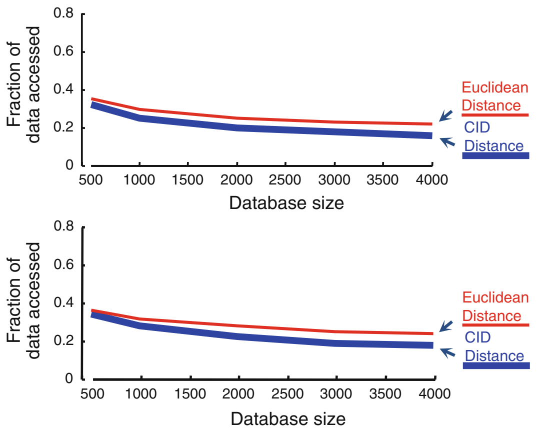


Fig. 29 A comparison of fraction of data accessed with Euclidean distance and CID using random walk time series with 32 (*top*) and 1,024 (*bottom*) observations indexed with Orchard's algorithm

This equation is equivalent to:

$$D_{CID}(Q, C_j) \geq \frac{1}{\rho} D_{CID}(C_i, C_j) - D_{CID}(C_i, Q)$$

Therefore, we require that:

$$D_{CID}(C_i, C_j) \geq 2\rho D_{CID}(C_i, Q) \tag{3}$$

with $\rho = CF(C_i, C_j)$ in order to safely prune the calculation of $D(Q, C_j)$.

Thus, to adapt Orchard's algorithm to CID we just need to make two simple modifications:

1. We must use Eq. 3 instead of Eq. 2 when pruning database instances;
2. We must store the complexity estimates, CE, of each database instance. The space overhead for storing the complexity estimates is $O(m)$, where m is the number of database objects, and thus is small relative to the typical overhead for most indexing data structures.

In order to illustrate that pruning with ρ-relaxed triangular inequality under CID can be effective, we ran a series of experiments using a database of random walk time series indexed with Orchard's algorithm. Our experiment consisted of running paired tests; in other words, the same database, queries and other parameters were used to measure performance for both CID and Euclidean distance. We used the mean percentage of the database accessed in each query as the main form to assess the performance of each technique.

Figure 29 presents the results for database sizes ranging from 500 up to 4,000 time series. For each database size we ran a series of 1,000 nearest neighbor queries. We ran experiments with time series lengths of 32, 64, 128, 256, 512 and 1,024 observations. In every experiment CID outperformed Euclidean distance in the number of distance calculations pruned. For brevity, we show in Fig. 29 the results for time series of 32 (top) and 1,024 (bottom) observations.

To be clear, we are not claiming that CID outperforms Euclidean distance for all indexing algorithms in general or for every data set when using Orchard's algorithm. For brevity, we reserve a more detailed performance analysis of indexing CID space for future work.

7 The utility of CID for other data mining problems: anomaly detection

We have shown that considering the relative complexity of time series generally improves the accuracy of classification and clustering, two of the cornerstone techniques of data mining. However, we believe that CID has the potential to improve the results for any application that uses similarity calculations as a subroutine. For time series, this is a very large set of tasks; similarity calculations are used for segmentation, summarization, forecasting, visualization, rule-discovery and anomaly detection.

For brevity, we consider just anomaly detection, which has been an area of active research for over two decades, attracting the attention of researchers in biology, physics, astronomy and statistics, in addition to the more recent work by the data mining community. The problem, and closely related tasks are variously referred to as the detection of "aberrant behavior", "novelties", "faults", "surprise", "deviants", "temporal change" and "outliers". The literature on this topic is vast, we refer the reader to (Chandola et al. 2009) for an excellent survey. The number of proposed techniques is likewise vast, but there is recent evidence that distance-based methods work very well. For example, Chandola et al., after the most extensive and rigorous empirical comparison we are aware of, noted "... on 19 different publicly available data sets, comparing 9 different techniques (distance-based methods are) the best overall technique among all techniques". Distance based methods come in several flavors, for concreteness we consider one popular definition, time series discords (Yankov et al. 2008).

Informally, given a time series C, with $|C| = n$, it has $n - m + 1$ subsequences of length m. The time series discord is the subsequence that is maximally furthest from its nearest neighbor. Thus discords capture the intuition of "I have never seen anything like this before". Naively, discovering the time series discord seems to require a $(n \times n - 1)/2$ distance calculations; however, Yankov et al. has shown that that can be discovered in $O(n)$ time.

Does CID have implications for discord discovery? While this is not the venue to attempt to forcefully answer this question, we will show an existence proof that it may. In Fig. 30 we show a snippet of ECG data from the MIT-BIH Atrial Fibrillation data set (Goldberger et al. 2000). In particular, a section taken from patient 08405. As the reader will appreciate, this section of ECG straddles a point where the data has become significantly noisier. This change is almost certainly an artifact of the recording apparatus, and not medically significant. Nevertheless a large fraction of the thousands of ECG recordings archived at www.physionet.org have similar non-stationary changes in noise level.

To create an unambiguous ground-truth anomaly, we reversed the direction of a randomly chosen heartbeat. The resulting anomaly has the same properties as the original data (mean, variance etc.); however it is glaringly obvious even to a casual

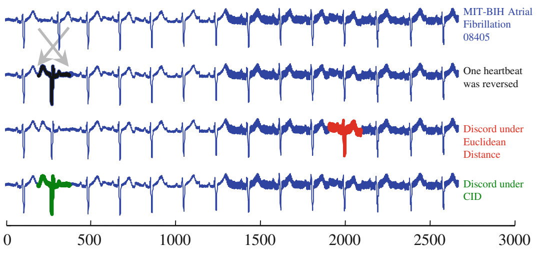


Fig. 30 From *top* to *bottom* A section of MIT-BIH Atrial Fibrillation, record 08045. A single heartbeat, show in *bold*/*black*, is reversed to create a trivial and obvious anomaly. Using the Euclidean distance, the time series discord, shown in *bold*/*red* is simply a normal heartbeat from the noisier half of the signal. In contrast, using the CID distance we find that the backwards heartbeat, shown in *bold*/*green*, is correctly identified as the discord (Color figure online)

inspection. We ran discord discovery on this data set, setting the length of the discord desired (the only parameter) to be about the length of a single heartbeat. Using the Euclidean distance, the algorithm fails to find the planted anomaly, and instead points to a noisy, but otherwise perfectly normal heartbeat. In retrospect, this is not that surprising. The curse of dimensionality tells us that in high-dimensional space all points tend to be far away from each other. For most interesting real-world time series problems this is not an issue, because the intrinsic dimensionally is much lower than the apparent dimensionally. However in this case, all the subsequences in the second half of this snippet are very far from every other subsequence. In fact, there is not much difference in the discord score between all of them, but as we are dealing with real numbers, one does have a largest value by a small margin, and it is marked in bold/red in Fig. 30.

We reran this experiment, making exactly one change, replacing the Euclidean distance with CID. As we might expect, the time taken for the algorithm did not differ in a perceptible way, but the CID-discord is the objectively correct reversed heartbeat as shown in Fig. 30 in bold/green.

We regard this experiment as visually intuitive existence proof, not a forceful demonstration. Nevertheless, in ongoing work we have seen subtle natural examples where CID produces more intuitive results.

8 Conclusion and future work

In this work we have surveyed all the existing invariances for time series similarity measures, and demonstrated the previously unknown need for complexity invariance. We have introduced CID, a simple and parameter-free method to mitigate the problem, and demonstrated its utility in improving classification and clustering accuracy on dozens of data sets.

We have further shown that the use of CID need not compromise efficiency, since it can be used with the vast majority of indexing and mining algorithms with only very minor changes.

What then are the shortcomings of CID? As shown in Figs. 15, 17 and 19, while it helps on most data sets, it does not help on every data set. Moreover, there is currently no way to predict which data sets it will work on, other than empirical testing. More generally, while we have developed a detailed intuition as to why CID works, we are lacking a detailed theoretical framework to fully explain and justify it. This is an avenue for future work. Future work also includes considering the utility of CID for the problem of motif discovery (Mueen et al. 2009) and outlier detection (Protopapas et al. 2006; Yankov et al. 2008).

Acknowledgements Thanks to Abdullah Mueen and Pavlos Protopapas for their help with the star light curve experiments, to Bing Hu and Yuan Hao for their help preparing some of the datasets, and to Thanawin Rakthanmanon, Ronaldo C. Prati, Edson T. Matsubara, André G. Maletzke and Claudia R. Milaré for their suggestions on a draft version of this paper. This work is an expanded version of a paper that appeared in SDM 2011 (Batista et al. 2011), in which we show CID is useful for classification. In this expanded version we also show that the complexity invariance is useful for other data mining tasks such as clustering and anomaly detection. The first author is in a leave from Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Brazil. This work was funded by NSF awards 0803410 and 0808770, FAPESP awards 2009/06349-0 and 2012/07295-3 and a gift from Microsoft.

Appendix

A ρ-relaxed triangular inequality proof

In this section we prove that CID obeys the ρ-relaxed triangular inequality:

$$D_{CID}(A, B) \leq \rho(D_{CID}(A, C) + D_{CID}(C, B))$$

We start our proof by stating the triangular inequality of Euclidean distance:

$$ED(A, B) \leq ED(A, C) + ED(C, A)$$

Remember that the complexity correction factor CF is a quantity greater than or equal to one; therefore, we can multiply both sides of the inequality by $CF(A, B)$:

$$ED(A, B)CF(A, B) \leq CF(A, B)(ED(A, C) + ED(C, A))$$

The left-hand side of the inequality is our definition of CID; hence:

$$D_{CID}(A, B) \leq \rho(ED(A, C) + ED(C, A))$$

with $\rho = CF(A, B)$.

Finally, we can again use the fact that CF is greater than or equal to one to change Euclidean distances to CID on the right-hand side of the equation:

$$D_{CID}(A, B) \leq \rho(ED(A, C)CF(A, C) + ED(C, A)CF(C, A))$$

And, therefore:

$$D_{CID}(A, B) \leq \rho(D_{CID}(A, C) + D_{CID}(C, A))$$

References

Andino SG, de Peralta Menendez RG (2000) Measuring the complexity of time series: an application to neurophysiological signals. Hum Brain Mapp 11(1):46–57

Aziz W, Arif M (2006) Complexity analysis of stride interval time series by threshold dependent symbolic entropy. Eur J Appl Physiol 98:30–40. doi:10.1007/s00421-006-0226-5

Bandt C, Pompe B (2002) Permutation entropy: a natural complexity measure for time series. Phys Rev Lett 88(17). doi:10.1103/PhysRevLett.88.174102

Batista G (2011) Website for this paper. http://www.icmc.usp.br/~gbatista/cid (Online)

Batista G, Wang X, Keogh EJ (2011) A complexity-invariant distance measure for time series. In: Proceedings of the 2011 SIAM International Conference on Data Mining (SDM), pp 699–710. http://www.siam.omnibooksonline.com/2011datamining/data/papers/106.pdf

Chandola V, Cheboli D, Kumar V (2009) Detecting anomalies in a time series database. CS Technical Report 09–004, Computer Science Department, University of Minnesota

Chávez E, Navarro G, Baeza-Yates R, Marroquín JL (2001) Searching in metric spaces. ACM Comput Surv 33:273–321. doi10.1145/502807.502808

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Ding H, Trajcevski G, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. In: International Conference on Very Large Data Bases, pp 1542–1552

Elkan C (2003) Using the triangle inequality to accelerate k-means. In: International Conference on Machine Learning, pp 147–153

Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. SIGMOD Rec 23:419–429. doi:10.1145/191843.191925

Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE (2000) PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation 101(23):e215–e220

Hearn DJ (2009) Shape analysis for the automated identification of plants from images of leaves. Taxon 58:934–954(21). http://www.ingentaconnect.com/content/iapt/tax/2009/00000058/00000003/art00021

Hjaltason GR, Samet H (2003) Index-driven similarity search in metric spaces (survey article). ACM Trans Database Syst 28:517–580. doi:10.1145/958942.958948

Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E (2011) Discovering the intrinsic cardinality and dimensionality of time series using MDL. In: IEEE International Conference on Data Mining(ICDM), pp 1086–1091

Kaufman L, Rousseeuw PJ (2005) Finding groups in data: an introduction to cluster analysis. Wiley, New York

Keogh E (2002) Exact indexing of dynamic time warping. In: International Conference on Very Large Data Bases, pp 406–417. http://portal.acm.org/citation.cfm?id=1287369.1287405

Keogh E (2003) Efficiently finding arbitrarily scaled patterns in massive time series databases. In: Knowledge Discovery in Databases: PKDD 2003, vol 2838, pp 253–265. doi:10.1007/978-3-540-39804-2_24

Keogh EJ, Xi X, Wei L, Ratanamahatana C (2006) The UCR time series classification/clustering homepage. http://www.cs.ucr.edu/~eamonn/time_series_data/ (Online)

Keogh E, Lonardi S, Ratanamahatana CA, Wei L, Lee SH, Handley J (2007) Compression-based data mining of sequential data. Data Min Knowl Discov 14:99–129. http://portal.acm.org/citation.cfm?id=1231311.1231321

Keogh E, Wei L, Xi X, Vlachos M, Lee SH, Protopapas P (2009) Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. VLDB J 18:611–630. doi:10.1007/s00778-008-0111-4

Li M, Vitnyi PM (2008) An introduction to Kolmogorov complexity and its applications, 3rd edn. Springer Publishing Company, Incorporated, Heidelberg

Li K, Yan M, Yuan S (2002) A simple statistical model for depicting the cdc15-synchronized yeast cell-cycle regulated gene expression data. Stat Sin 12:141–158

Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: 8th ACM SIGMOD Workshop on Research Issues in, Data Mining and Knowledge Discovery, pp 2–11

Moore A (2000) The anchors hierarchy: using the triangle inequality to survive high-dimensional data. In: Conference on Uncertainty in Artificial Intelligence, pp 397–405

Mueen A, Keogh E, Bigdely-Shamlo N (2009) Finding time series motifs in disk-resident data. In: IEEE International Conference on Data Mining, pp 367–376. doi:10.1109/ICDM.2009.15

Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: analysis and an algorithm. In: Dietterich TG, Becker S, Ghahramani Z (eds) Advances in neural information processing systems, MIT Press, Cambridge, pp 849–856

Orchard M (1991) A fast nearest-neighbor search algorithm. In: Acoustics, Speech, and Signal Processing, 1991. ICASSP-91, 1991 International Conference on, vol 4, pp 2297–2300. doi:10.1109/ICASSP.1991.150755

Protopapas P, Giammarco JM, Faccioli L, Struble MF, Dave R, Alcock C (2006) Finding outlier light curves in catalogues of periodic variable stars. Mon Notices R Astron Soc 369(2):677–696

Rabiner L, Schafer R (1978) Digital Processing of Speech Signals. Prentice Hall, Englewood Cliffs

Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: ACM KDD, pp 262–270

Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846–850

Rezek I (1998) Stochastic complexity measures for physiological signal analysis. IEEE Trans Biomed Eng 44(9):1186–1191

Schroeder M (2009) Fractals Chaos, Power Laws: minutes from an infinite paradise. Dover Publications, New York

Vlachos M, Hadjieleftheriou M, Gunopulos D, Keogh EJ (2003) Indexing multi-dimensional time-series with support for multiple distance measures. In: ACM KDD, pp 216–225

Yankov D, Keogh E, Rebbapragada U (2008) Disk aware discord discovery: finding unusual time series in terabyte sized datasets. Knowl Info Syst 17:241–262. doi:10.1007/s10115-008-0131-9

Ye L, Wang X, Keogh EJ, Mafra-Neto A (2009) Autocannibalistic and anyspace indexing algorithms with application to sensor data mining. In: SIAM International Conference on Data Mining, pp 85–96. http://www.siam.org/proceedings/datamining/2009/dm09_009_yel.pdf

Žunic J, Rosin P, Kopanja L (2006) Shape orientability. In: Computer Vision ACCV 2006, pp 11–20. doi:10.1007/11612704_2