DOI 10.1007/s10115-004-0172-7
Springer-Verlag London Ltd. © 2004
Knowledge and Information Systems (2005) 8: 154–177

Knowledge and Information Systems

Clustering of time-series subsequences is meaningless: implications for previous and future research

Eamonn Keogh, Jessica Lin

Computer Science & Engineering Department, University of California–Riverside, Riverside, CA, USA

Abstract. Given the recent explosion of interest in streaming data and online algorithms, clustering of time-series subsequences, extracted via a sliding window, has received much attention. In this work, we make a surprising claim. Clustering of time-series subsequences is meaningless. More concretely, clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random. While this constraint can be intuitively demonstrated with a simple illustration and is simple to prove, it has never appeared in the literature. We can justify calling our claim surprising because it invalidates the contribution of dozens of previously published papers. We will justify our claim with a theorem, illustrative examples, and a comprehensive set of experiments on reimplementations of previous work. Although the primary contribution of our work is to draw attention to the fact that an apparent solution to an important problem is incorrect and should no longer be used, we also introduce a novel method that, based on the concept of time-series motifs, is able to meaningfully cluster subsequences on some time-series datasets.

Keywords: Clustering; Data mining; Rule discovery; Subsequence; Time series

1. Introduction

A large fraction of attention from the data-mining community has focuses on time-series data (Keogh and Kasetty 2002; Roddick and Spiliopoulou 2002). This is plausible and highly anticipated because time-series data is a by-product in virtually every human endeavor, including biology (Bar-Joseph et al. 2002), finance (Fu et al. 2001; Gavrilov et al. 2000; Mantegna 1999), geology (Harms et al. 2002b), space exploration (Honda et al. 2002; Yairi et al. 2001), robotics (Oates 1999) and human motion

Received 19 November 2003
Revised 8 January 2004
Accepted 16 February 2004
Published online 31 August 2004

analysis (Uehara and Shimada 2002). Of all the techniques applied to time series, clustering is perhaps the most frequently used (Halkidi et al. 2001), being useful in its own right as an exploratory technique and as a subroutine in more complex data-mining algorithms (Bar-Joseph et al. 2002; Bradley and Fayyad 1998). Given these two facts, it is hardly surprising that time-series clustering has attracted an extraordinary amount of attention (Bar-Joseph et al. 2002; Cotofrei 2002; Cotofrei and Stoffel 2002; Das et al. 1998; Fu et al. 2001; Gavrilov et al. 2000; Harms et al. 2002a, 2002b; Hetland and Satrom 2002; Honda et al. 2002; Jin et al. 2002a, 2002b; Keogh 2002a; Keogh et al. 2001; Li et al. 1998; Lin et al. 2002; Mantegna 1999; Mori and Uehara 2001; Oates 1999; Osaki et al. 2000; Radhakrishnan et al. 2000; Sarker et al. 2002; Steinback et al. 2002; Tino et al. 2000; Uehara and Shimada 2002; Yairi et al. 2001). The work in this area can be broadly classified into two categories:

- **Whole clustering:** The notion of clustering here is similar to that of conventional clustering of discrete objects. Given a set of individual time-series data, the objective is to group similar time series into the same cluster.
- **Subsequence clustering:** Given a single time series, sometimes in the form of streaming time series, individual time series (subsequences) are extracted with a sliding window. Clustering is then performed on the extracted time-series subsequences.

Subsequence clustering is commonly used as a subroutine in many other algorithms, including rule discovery (Das et al. 1998; Fu et al. 2001; Harms et al. 2002a, 2002b; Hetland and Satrom 2002; Jin et al. 2002a, 2002b; Mori and Uehara 2001; Osaki et al. 2000; Sarker et al. 2002; Uehara and Shimada 2002; Yairi et al. 2001), indexing (Li et al. 1998; Radhakrishnan et al. 2000), classification (Cotofrei 2002; Cotofrei and Stoffel 2002), prediction (Schittenkopf et al. 2000; Tino et al. 2000) and anomaly detection (Yairi et al. 2001). For clarity, we will refer to this type of clustering as STS (subsequence time series) clustering.

In this work, we make a surprising claim. Clustering of time-series subsequences is meaningless! In particular, clusters extracted from these time series are forced to obey certain constraints that are pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random.

Because we use the word *meaningless* many times in this paper, we will take the time to define this term. All useful algorithms (with the sole exception of random-number generators) produce output that depends on the input. For example, a decision-tree learner will yield very different outputs on, say, a credit-worthiness domain, a drug-classification domain and a music domain. We call an algorithm *meaningless* if the output is independent of the input. As we prove in this paper, the output of STS clustering does not depend on input and is therefore meaningless.

Our claim is surprising because it calls into question the contributions of dozens of papers. In fact, the existence of so much work based on STS clustering offers an obvious counterargument to our claim. It could be argued: *Because many papers have been published that use time-series subsequence clustering as a subroutine and these papers produced successful results, time-series subsequence clustering must be a meaningful operation.*

We strongly feel that this is not the case. We believe that, in all such cases, the results are consistent with what one would expect from random cluster centres. We recognize that this is a strong assertion, so we will demonstrate our claim by reimplementing the most successful (i.e. the most referenced) examples of such work

and showing with exhaustive experiments that these contributions inherit the property of meaningless results from the STS clustering subroutine.

The rest of this paper is organized as follows. In Sect. 2, we will review the necessary background material on time series and clustering, then briefly review the body of research that uses STS clustering. In Sect. 3, we will show that STS clustering is meaningless with a series of simple intuitive experiments; then in Sect. 4, we will explain *why* STS clustering cannot produce useful results. In Sect. 5, we show that the many algorithms that use STS clustering as a subroutine produce results indistinguishable from random clusters. Because the main contribution of this paper may be considered negative, Sect. 6 demonstrates a simple algorithm that can find clusters in at least some trivial datasets. This algorithm is not presented as the best way to find clusters in time-series subsequences; it is simply offered as an existence proof that such an algorithm exists and to pave the way for future research. In Sect. 7, we conclude and summarize some comments from researchers that have read an earlier version of this paper and verified the results.

2. Background material

In order to frame our contribution in the proper context, we begin with a review of the necessary background material.

2.1. Notation and definitions

We begin with a definition of our data type of interest, time series:

Definition 1. *Time series:* A time series $T = t_1, \ldots, t_m$ is an ordered set of m real-valued variables.

Data-mining researchers are typically not interested in any of the global properties of a time series; rather, researchers confine their interest to subsections of the time series, called subsequences.

Definition 2. *Subsequence:* Given a time series T of length m, a subsequence C_p of T is a sampling of length $w < m$ of contiguous positions from T, that is, $C_p = t_p, \ldots, t_{p+w-1}$ for $1 \leq p \leq m - w + 1$.

In this work, we are interested in the case where all the subsequences are extracted and then clustered. This is achieved by use of a sliding window.

Definition 3. *Sliding window:* Given a time series T of length m and a user-defined subsequence length of w, a matrix S of all possible subsequences can be built by sliding a window across T and placing subsequence C_p in the pth row of S. The size of matrix S is $(m - w + 1)$ by w.

Figure 1 summarizes all the above definitions and notations.

Note that, while S contains exactly the same information[1] as T, it requires significantly more storage space.

[1] S contains the same information as T except that the subsequences are usually normalized individually before inserting to S. Normalization is an important and indispensable step in the sense that it allows identification of similar patterns in time series with different amplitude, scaling, etc.

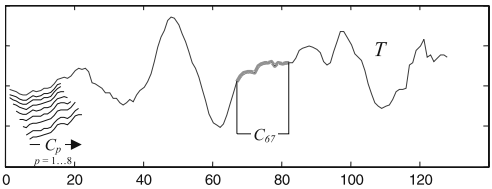


Fig. 1. An illustration of the notation introduced in this section: a *time series T* of length 128, a *subsequence* of length $w = 16$, beginning at datapoint 67, and the first eight subsequences extracted by a *sliding window*

2.2. Background on clustering

One of the most widely used clustering approaches is hierarchical clustering due to the great visualization power it offers (Keogh and Kasetty 2002; Mantegna 1999). Hierarchical clustering produces a nested hierarchy of similar groups of objects, according to a pairwise distance matrix of the objects. One of the advantages of this method is its generality because the user does not need to provide any parameters such as the number of clusters. However, its application is limited to only small datasets due to its quadratic computational complexity. Table 1 outlines the basic hierarchical clustering algorithm.

Table 1. An outline of hierarchical clustering

Algorithm Hierarchical clustering	
1.	Calculate the distance between all objects. Store the results in a distance matrix.
2.	Search through the distance matrix and find the two most similar clusters/objects.
3.	Join the two clusters/objects to produce a cluster that now has at least two objects.
4.	Update the matrix by calculating the distances between this new cluster and all other clusters.
5.	Repeat step 2 until all cases are in one cluster.

A faster method to perform clustering is *k*-means (Bradley and Fayyad 1998). The basic intuition behind *k*-means (and a more general class of clustering algorithms known as iterative refinement algorithms) is shown in Table 2.

Table 2. An outline of the *k*-means algorithm

Algorithm *k-means*	
1.	Decide on a value for k.
2.	Initialize the k cluster centres (randomly, if necessary).
3.	Decide the class memberships of the N objects by assigning them to the nearest cluster centre.
4.	Reestimate the k cluster centres by assuming the memberships found above are correct.
5.	If none of the N objects changed membership in the last iteration, exit. Otherwise, go to 3.

The *k*-means algorithm for N objects has a complexity of O($kNrD$), where k is the number of clusters specified by the user, r is the number of iterations until

convergence and D is the dimensionality of time series (in the case of STS clustering, D is the length of the sliding window, w). While the algorithm is perhaps the most commonly used clustering algorithm in the literature, it does have several shortcomings, including the fact that the number of clusters must be specified in advance (Bradley and Fayyad 1998; Halkidi et al. 2001).

It is well understood that some types of high-dimensional clustering may be meaningless. As noted by Agrawal et al. (1993) and Bradley and Fayyad (1998), in high dimensions, the very concept of nearest neighbor has little meaning because the ratio of the distance to the nearest neighbor over the distance to the average neighbor rapidly approaches 1 as the dimensionality increases. However, time series, while often having high dimensionality, typically have a low intrinsic dimensionality (Keogh et al. 2001) and can therefore be meaningful candidates for clustering.

2.3. Background on time-series data mining

The last decade has seen an extraordinary interest in mining time-series data, with at least one thousand papers on the subject (Keogh and Kasetty 2002). Tasks addressed by the researchers include segmentation, indexing, clustering, classification, anomaly detection, rule discovery and summarization.

Of the above, a significant fraction use subsequence time-series clustering as a subroutine. Below we enumerate some representative examples.

- There has been much work on finding association rules in time series (Das et al. 1998; Fu et al. 2001; Harms et al. 2002a, 2002b; Jin et al. 2002a, 2002b; Keogh and Kasetty 2002; Mori and Uehara 2001; Osaki et al. 2000; Uehara and Shimada 2002; Yairi et al. 2001). Virtually all work is based on the classic paper of Das et al. that uses STS clustering to convert real-valued time series into symbolic values, which can then be manipulated by classic rule-finding algorithms (Das et al. 1998).
- The problem of anomaly detection in time series has been generalized to include the detection of surprising or interesting patterns (which are not necessarily anomalies). There are many approaches to this problem, including several based on STS clustering (Yairi et al. 2001).
- Indexing of time series is an important problem that has attracted the attention of dozens of researchers. Several of the proposed techniques make use of STS clustering (Li et al. 1998; Radhakrishnan et al. 2000).
- Several techniques for classifying time series make use of STS clustering to preprocess the data before passing to a standard classification technique such as a decision tree (Cotofrei 2002; Cotofrei and Stoffel 2002).
- Clustering of streaming time series has also been proposed as a knowledge-discovery tool in its own right. Researchers have suggested various techniques to speed up the STS clustering (Fu et al. 2001).

The above is just a small fraction of the work in the area; more extensive surveys may be found in Keogh (2002a) and Roddick and Spiliopoulou (2002).

3. Demonstrations of the meaninglessness of STS clustering

In this section, we will demonstrate the meaninglessness of STS clustering. In order to demonstrate that this meaninglessness is a result of the way the data is obtained

by sliding windows and not some quirk of the clustering algorithm, we will also do whole clustering as a control (Gavrilov et al. 2000; Oates 1999). We will begin by using the well-known k-means algorithm because it accounts for the lion's share of all clustering in the time-series data-mining literature. In addition, the k-means algorithm uses Euclidean distance as its underlying metric, and again the Euclidean distance accounts for the vast majority of all published work in this area (Cotofrei 2002; Cotofrei and Stoffel 2002; Das et al. 1998; Fu et al. 2001; Harms et al. 2002a; Jin et al. 2002a; Keogh et al. 2001), and as empirically demonstrated in Keogh and Kasetty (2002), it performs better than the dozens of other recently suggested time-series distance measures.

3.1. k-means clustering

Because k-means is a heuristic, hill-climbing algorithm, the cluster centres found may not be optimal (Halkidi et al. 2001). That is, the algorithm is guaranteed to converge on a local, but not necessarily global, optimum. The choices of the initial centres affect the quality of results. One technique to mitigate this problem is to do multiple restarts and choose the best set of clusters (Bradley and Fayyad 1998). An obvious question to ask is how much variability in the shapes of cluster centres we get between multiple runs. We can measure this variability with the following equation:

- Let $A = (\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_k)$ be the cluster centres derived from one run of k-means.
- Let $B = (\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_k)$ be the cluster centres derived from a different run of k-means.
- Let $dist(\bar{a}_i, \bar{a}_j)$ be the distance between two cluster centres, measured with Euclidean distance.

Then the distance between two sets of clusters can be defined as

$$cluster\_distance(A, B) \equiv \sum_{i=1}^{k} \min\left[dist(\bar{a}_i, \bar{b}_j)\right], \quad 1 \le j \le k. \tag{1}$$

The simple intuition behind the equation is that each individual cluster centre in A should map onto its closest counterpart in B, and the sum of all such distances tells us how similar two sets of clusters are.

An important observation is that we can use this measure not only to compare two sets of clusters derived for the same dataset but also two sets of clusters that have been derived from *different* data sources. Given this fact, we propose a simple experiment.

We performed three random restarts of k-means on a stock-market dataset, and saved the three resulting sets of cluster centres into set $\hat{X}$. We also performed three random restarts on random-walk dataset, saving the three resulting sets of cluster centres into set $\hat{Y}$. Note that the choice of three was an arbitrary decision for ease of exposition; larger values do not change the substance of what follows.

We then measured the average cluster distance (as defined in Eq. (1)), between each set of cluster centres in $\hat{X}$, to each other set of cluster centres in $\hat{X}$. We call this number *within_set_$\hat{X}$_distance*.

$$within\_set\_\hat{X}\_distance = \frac{\sum_{i=1}^{3}\sum_{j=1}^{3} cluster\_distance(\hat{X}_i, \hat{X}_j)}{9} \tag{2}$$

We also measured the average cluster distance between each set of cluster centres in $\hat{X}$, to cluster centres in $\hat{Y}$; we call this number *between_set_$\hat{X}$_and_$\hat{Y}$_distance*.

$$between\_set\_\hat{X}\_and\_\hat{Y}\_distance = \frac{\sum_{i=1}^{3}\sum_{j=1}^{3} cluster\_distance(\hat{X}_i, \hat{Y}_j)}{9} \quad (3)$$

We can use these two numbers to create a fraction,

$$\text{clustering meaningfulness}(\hat{X}, \hat{Y}) \equiv \frac{within\_set\_\hat{X}\_distance}{between\_set\_\hat{X}\_and\_\hat{Y}\_distance}. \quad (4)$$

We can justify calling this number *clustering meaningfulness* because it clearly measures just that. If, for any dataset, the clustering algorithm finds similar clusters each time regardless of the different initial seeds, the numerator should be close to 0. In contrast, there is no reason why the clusters from two completely different, unrelated datasets should be similar. Therefore, we should expect the denominator to be relatively large. So, overall, we should expect that the value of *clustering meaningfulness*($\hat{X}$, $\hat{Y}$) be close to 0 when $\hat{X}$ and $\hat{Y}$ are sets of cluster centres derived from different datasets.

As a control, we performed the exact same experiment on the same data, but using subsequences that were randomly extracted, rather than extracted by a sliding window. We call this whole clustering.

Because it might be argued that any results obtained were the consequence of a particular combination of k and w, we tried the cross product of $k = \{3, 5, 7, 11\}$ and $w = \{8, 16, 32\}$. For every combination of parameters, we repeated the entire process 100 times, and averaged the results. Figure 2 shows the results.

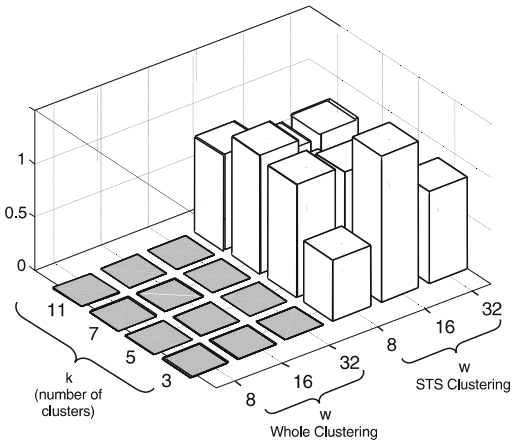


Fig. 2. A comparison of the clustering meaningfulness for whole clustering and STS clustering using k-means with a variety of parameters. The two datasets used were Standard and Poor's 500 Index closing values and random-walk data

The results are astonishing. The cluster centres found by STS clustering on any particular run of k-means on the stock-market dataset are not significantly more similar to each other than they are to cluster centres taken from random-walk data! In other words, if we were asked to perform clustering on a particular stock-market dataset, we could reuse an old clustering obtained from random-walk data, and no one could tell the difference!

We reemphasize here that the difference in the results for STS clustering and whole clustering in this experiment (and all experiments in this work) are due exclusively to the feature-extraction step. In particular, both are being tested on the same datasets with the same parameters of w and k using the same algorithm.

We also note that the exact definition of *clustering meaningfulness* is not important to our results. In our definition, each cluster centre in A maps onto its closest match in B. It is possible, therefore, that two or more cluster centres from A map to one centre in B, and some clusters in B have no match. However, we tried other variants of this definition, including pairwise matching, minimum matching and maximum matching, together with dozens of other measurements of clustering quality suggested in the literature (Halkidi et al. 2001); it simply makes no significant difference to the results.

3.2. Hierarchical clustering

The previous section suggests that k-means clustering of STS time series does not produce meaningful results, at least for stock-market data. Two obvious questions to ask are: Is this true for STS with other clustering algorithms? And is this true for other types of data? We will answer the former question here and the latter question in Sect. 3.3.

Hierarchical clustering, unlike k-means, is a deterministic algorithm. So we can't reuse the experimental methodology from the previous section exactly; however, we can do something very similar.

First we note that hierarchical clustering can be converted into a partitional clustering, by cutting the first k links (Mantegna 1999). Figure 3 illustrates the idea. The resultant time series in each of the k subtrees can then be merged into single cluster prototypes. When performing hierarchical clustering, one has to make a choice about how to define the distance between two clusters; this choice is called the linkage method (cf. step 3 of Table 1).

Three popular choices are complete linkage, average linkage and Ward's method (Halkidi et al. 2001). We can use all three methods for the stock-market dataset and place the resulting cluster centres into set X. We can do the same for random-walk data and place the resulting cluster centres into set Y. Having done this, we can extend the measure of clustering meaningfulness in Eq. (4) to hierarchical clustering and run a similar experiment as in the last section, but using hierarchical clustering. The results of this experiment are shown in Fig. 4.

Once again, the results are astonishing. While it is well understood that the choice of linkage method can have minor effects on the clustering found, the results above tell us that, when doing STS clustering, the choice of linkage method has as much effect as the choice of dataset! Another way of looking at the results is as follows: If we were asked to perform hierarchical clustering on a particular dataset, but we did not have to report which linkage method we used, we could reuse an old random-walk clustering and no one could tell the difference without rerunning the clustering for every possible linkage method.

3.3. Other datasets and algorithms

The results in the two previous sections are extraordinary, but are they the consequence of some properties of stock market data, or as we claim, a property of

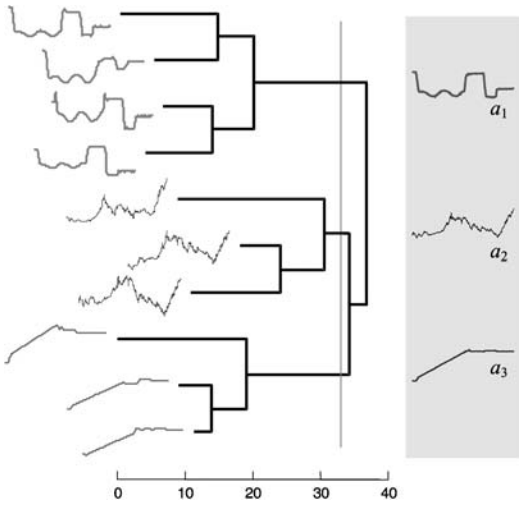


Fig. 3. A hierarchical clustering of ten time series. The clustering can be converted to a k partitional clustering by sliding a cutting line until it intersects k lines of the dendrograms, then averaging the time series in the k subtrees to form k cluster centres (gray panel)

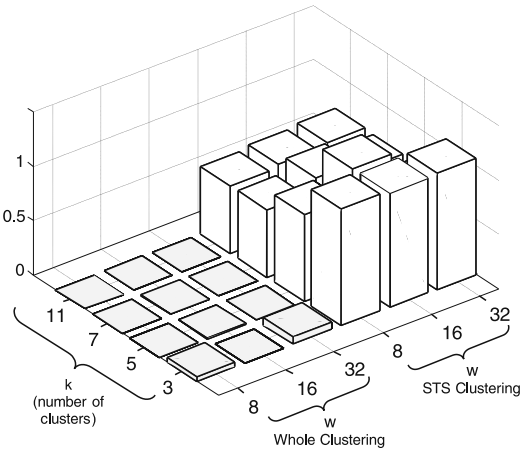


Fig. 4. A comparison of the clustering meaningfulness for whole clustering and STS clustering using hierarchical clustering with a variety of parameters. The two datasets used were Standard and Poor's 500 Index closing values and random-walk data

the sliding-window feature extraction? The latter is the case, which we can simply demonstrate. We visually inspected the UCR archive of time-series datasets for the two time-series datasets that appear the least alike (Keogh 2002b). The best two candidates we discovered are shown in Fig. 5.

We repeated the experiment of Sect. 3.2 using these two datasets in place of the stock market data and the random-walk data. The results are shown in Fig. 6.

In our view, this experiment sounds the death knell for clustering of STS time series. If we cannot easily differentiate between the clusters from these two vastly different time series, then how could we possibly find meaningful clusters in any data?

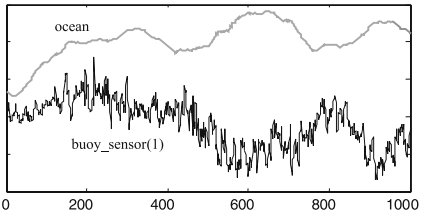


Fig. 5. Two subjectively very dissimilar time series from the UCR archive. Only the first 1,000 datapoints are shown. The two time series have very different properties of stationarity, noise, periodicity, symmetry, autocorrelation, etc.

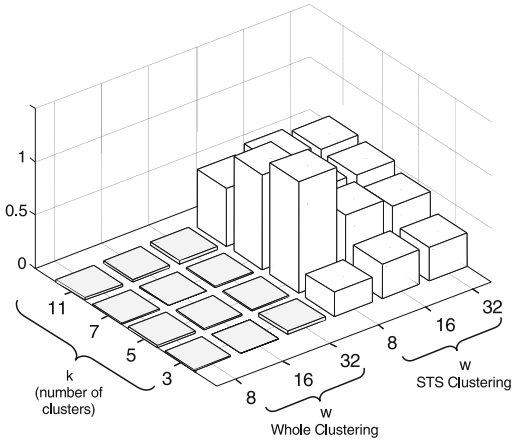


Fig. 6. A comparison of the clustering meaningfulness for whole clustering and STS clustering, using *k*-means with a variety of parameters. The two datasets used were buoy_sensor(1) and ocean

In fact, the experiments shown in this section are just a small subset of the experiments we performed. We tested other clustering algorithms, including Expectation Maximization (EM) and Self-Organizing Maps (SOMs) (van Laerhoven 2001). We tested on 42 different datasets (Keogh 2002a; Keogh and Kasetty 2002). We experimented with other measures of clustering quality (Halkidi et al. 2001). We tried other variants of *k*-means, including different seeding algorithms. Although Euclidean distance is the most commonly used distance measure for time-series data mining, we also tried other distance measures from the literature, including Manhattan, L_∞, Mahalanobis distance and dynamic time warping distance (Gavrilov et al. 2000; Keogh 2002a; Oates 1999). We tried various normalization techniques, including Z-normalization, 0-1 normalization, amplitude only normalization, offset only normalization, no normalization, etc. In every case, we are forced to the inevitable conclusion: whole clustering of time series is usually a meaningful thing to do, but sliding-window time-series clustering is *never* meaningful.

4. Why is STS clustering meaningless?

Before explaining why STS clustering is meaningless, it will be instructive to visualize the cluster centres produced by both whole clustering and STS clustering. By

definition of k-means, each cluster centre is simply the average of all the objects within that cluster (cf. step 4 of Table 2). For the case of time series, the cluster centre is just another time series, the values of which are the averages of all time series within that cluster. Apparently, because the objective of k-means is to group similar objects in the same cluster, we should expect the cluster centre to look somewhat similar to the objects in the cluster. We will demonstrate this on the classic cylinder-bell–funnel data (Keogh and Kasetty 2002). This dataset consists of random instantiations of the eponymous patterns, with Gaussian noise added. Note that this dataset has been freely available for a decade and has been referenced more than 50 times (Keogh and Kasetty 2002). While each time series is of length 128, the onset and duration of the shape is subject to random variability. Figure 7 shows one instance from each of the three patterns.

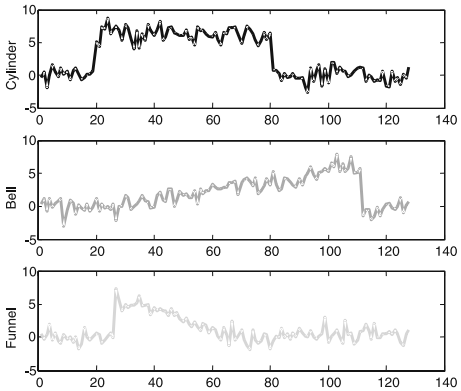


Fig. 7. Examples of cylinder, bell, and funnel patterns

We generated a dataset that contains 30 instances of each pattern and performed k-means clustering on it, with $k = 3$. The resulting cluster centres are shown in Fig. 8. As one might expect, all three clusters are successfully found. The final centres closely resemble the three different patterns in the dataset, although the sharp edges of the patterns have been somewhat softened by the averaging of many time series with some variability in the time axis.

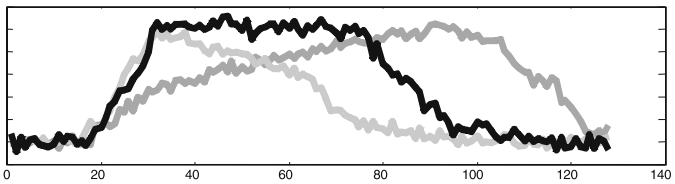


Fig. 8. The three final centres found by k-means on the cylinder–bell–funnel dataset. The shapes of the centres are close approximations of the original patterns

To compare the results of whole clustering to STS clustering, we took the 90 time series used above and concatenated them into one long time series. We then performed STS clustering with k-means. To make it simple for the algorithm, we

used the exact length of the patterns ($w = 128$) as the window length and $k = 3$ as the number of desired clusters. The cluster centres are shown in Fig. 9.

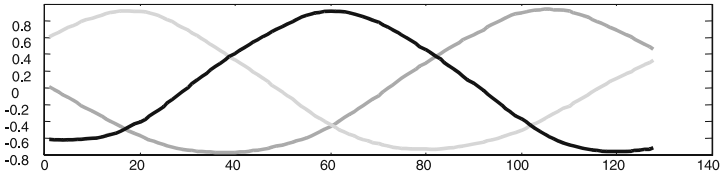


Fig. 9. The three final centres found by subsequence clustering using the sliding-window approach. The cluster centres appear to be sine waves, even though the data itself is not particularly spectral in nature. Note that, with each random restart of the clustering algorithm, the phase of the resulting sine waves changes in an arbitrary and unpredictable way

The results are extraordinarily unintuitive! The cluster centres look nothing like any of the patterns in the data; what's more, they appear to be perfect sine waves.

In fact, for $w \ll m$, we get approximate sine waves with STS clustering regardless of the clustering algorithm, the number of clusters or the dataset used! Furthermore, although the sine waves are always exactly out of phase with each other by $1/k$ period, overall, their joint phase is arbitrary and will change with every random restart of k-means.

This result explains the results from the last section. If sine waves appear as cluster centres for every dataset, then clearly it will be impossible to distinguish one dataset's clusters from another. Although we have now explained the inability of STS clustering to produce meaningful results, we have revealed a new question: Why do we always get cluster centres with this special structure?

4.1. A hidden constraint

To explain the unintuitive results above, we must introduce a new fact.

Theorem 1. For any time-series dataset T with an overall trend of 0, if T is clustered using sliding windows, and $w \ll m$, then the mean of all the data (i.e. the special case of $k = 1$), will be an approximately constant vector.

In other words, if we run STS k-means on *any* dataset, with $k = 1$ (an unusual case, but perfectly legal), we will always end up with a horizontal line as the cluster centre. The proof of this fact is straightforward but long, so we have elucidated it in a separate technical report (Truppel et al. 2003). Note that the requirement that the overall trend be 0 can be removed, in which case, the $k = 1$ cluster centre is still a straight line, but with slope greater than 0.

We content ourselves here with giving the intuition behind the proof and offering a visual proof in Fig. 10.

The intuition behind Theorem 1 is as follows. Imagine an arbitrary datapoint t_i somewhere in the time series T, such that $w \leq i \leq m - w + 1$. If the time series is much longer than the window size, then virtually all datapoints are of this type. What contribution does this datapoint make to the overall mean of the STS matrix S? As the sliding window passes by, the datapoint first appears as the rightmost value in the window, then it goes on to appear exactly once in *every* possible location within

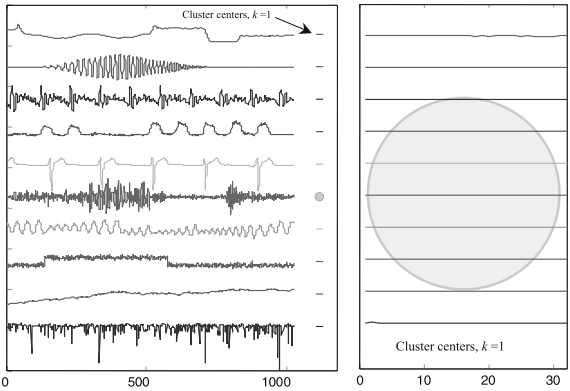


Fig. 10. A visual proof of Theorem 1. Ten time series of vastly different properties of stationarity, noise, periodicity, symmetry, autocorrelation, etc. The cluster centres for each time series, for $w = 32$, $k = 1$, are shown at right. Far right shows a zoom-in that shows just how close to a straight line the cluster centres are. While the objects have been shifted for clarity, they have *not* been rescaled in either axis; note the light gray circle in both graphs. The datasets used are, reading from top to bottom, space shuttle, flutter, speech, power_data, Koski_ecg, earthquake, chaotic, cylinder, random walk, and balloon

the sliding window. So the t_{i} datapoint contribution to the overall shape is the same everywhere and must be a horizontal line. Only those points at the very beginning and the very end of the time series avoid contributing their value to all w columns of S, but these are asymptotically irrelevant. The average of many horizontal lines is clearly just another horizontal line. Another way to look at it is that every value v_i in the mean vector, $1 \leq i \leq w$, is computed by averaging essentially every value in the original time series; more precisely, from t_{i} to $t_{\mathrm{m-w+i}}$. So for a time series of $m = 1{,}024$ and $w = 32$, the first value in the mean vector is the average of $t[1 \ldots 993]$; the second value is the average of $t[2 \ldots 994]$, and so forth. Again, the only datapoints not being included in every computation are the ones at the very beginning and at the very end, and their effects are negligible asymptotically.

The implications of Theorem 1 become clearer when we consider the following well-documented fact. For any dataset, the weighted (by cluster membership) average of k clusters must sum up to the global mean. The implication for STS clustering is profound. Because the global mean for STS clustering is a straight line, then the weighted average of k-clusters must in turn sum to a straight line. However, there is no reason why we should expect this to be true of *any* dataset, much less *every* dataset. This hidden constraint limits the utility of STS clustering to a vanishingly small set of subspace of all datasets. The out-of-phase sine waves as cluster centres that we get from the last section conforms to this theorem, because their weighted average, as expected, sums to a straight line.

4.2. The importance of trivial matches

There are further constraints on the types of datasets where STS clustering could possibly work. Consider a subsequence C_p that is a member of a cluster. If we examine the entire dataset for similar subsequences, we should typically expect to find the best matches to C_p to be the subsequences $\ldots, C_{p-2}, C_{p-1}, C_{p+1}, C_{p+2}, \ldots$. In other words, the best matches to any subsequence tend to be just slightly shifted

versions of the subsequence. Figure 11 illustrates the idea and Definition 4 states it more formally.

Definition 4. *Trivial match:* Given a subsequence C beginning at position p, a matching subsequence M beginning at q, and a distance R, we say that M is a *trivial match* to C of order R, if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.

The importance of trivial matches, in a different context, has been documented elsewhere (Lin et al. 2002).

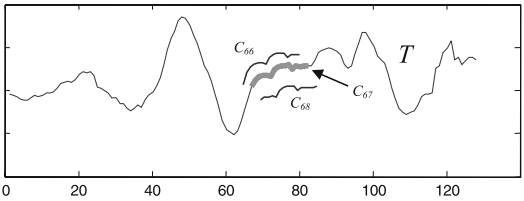


Fig. 11. For almost any subsequence C in a time series, the closest matching subsequences are the subsequences immediately to the left and right of C

An important observation is the fact that different subsequences can have vastly different numbers of trivial matches. In particular, smooth, slowly changing subsequences tend to have many trivial matches, whereas subsequences with rapidly changing features and/or noise tend to have very few trivial matches. Figure 12 illustrates the idea. The figure shows a time series that subjectively appears to have a cluster of three square waves. The bottom plot shows how many trivial matches each subsequence has. Note that the square waves have very few trivial matches, so all three taken together sit in a sparsely populated region of w-space. In contrast, consider the relatively smooth Gaussian bump centred at 125. The subsequences in the smooth ascent of this feature have more than 25 trivial matches and thus sit in a dense region of w-space; the same is true for the subsequences in the descent from the peak. So if clustering this dataset with k-means, $k = 2$, then the two cluster centres will be irresistibly drawn to these two shapes, simple ascending and descending lines.

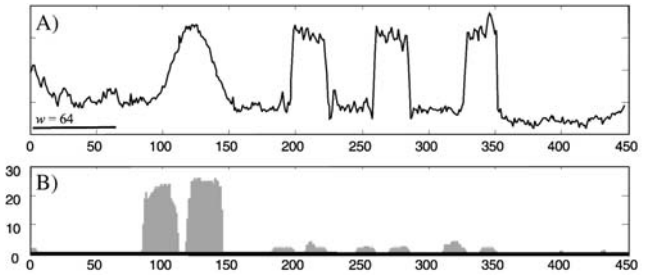


Fig. 12. A) A time series T that subjectively appears to have a cluster of three noisy square waves. B) Here the ith value is the number of trivial matches for the subsequence C_i in T, where $R = 1$, $w = 64$

The importance of this observation for STS clustering is obvious. Imagine we have a time series where we subjectively see two clusters: equal numbers of a smooth, slowing changing pattern and a noisier pattern with many features. In w-dimensional space, the smooth pattern is surrounded by many trivial matches. This dense volume will appear to any clustering algorithm an extremely promising cluster centre. In contrast, the highly featured, noisy pattern has very few trivial matches and thus sits in a relatively sparse space, all but ignored by the clustering algorithm. Note that it is not possible to simply remove or factor out the trivial matches because there is no way to know beforehand the true patterns.

We have not yet fully explained why the cluster centres for STS clustering degenerate to sine waves (cf. Fig. 9). However, we have shown that for STS clustering, algorithms do not really cluster the data. If not clustering, what are the algorithms doing? It is instructive to note that, if we perform singular value decomposition on time series, we also get shapes that seem to approximate sine waves (Keogh et al. 2001). This suggests that STS clustering algorithms are simply returning a set of basis functions that can be added together in a weighted combination to approximate the original data.

An even more tantalizing piece of evidence exists. In the 1920s, data miners were excited to find that, by preprocessing their data with repeated smoothing, they could discover trading cycles. Their joy was shattered by a theorem by Evgeny Slutsky (1880–1948), who demonstrated that any noisy time series will converge to a sine wave after repeated applications of moving window smoothing (Kendall 1976). While STS clustering is not exactly the same as repeated moving window smoothing, it is clearly highly related. For brevity, we will defer future discussion of this point to future work.

4.3. Is there a simple fix?

Having gained an understanding of the fact that STS clustering is meaningless and having developed an intuition as to why this is so, it is natural to ask if there is a simple modification to allow it to produce meaningful results. We asked this question, not just among ourselves, but also of dozens of time-series clustering researchers with whom we shared our initial results. While we considered all suggestions, we discuss only the two most promising ones here.

The first idea is to increment the sliding window by more than one unit each time. In fact, this idea was suggested by Das et al. (1998), but only as a speed-up mechanism. Unfortunately, this idea does not help. If the new step size s is much smaller than w, we still get the same empirical results. If s is approximately equal to or larger than w, we are no longer doing subsequence clustering, but whole clustering. This is not useful because the choice of the offset for the first window would become a critical parameter, and choices that differ by just one timepoint can give arbitrarily different results. As a concrete example, clustering weekly stock market data from Monday to Sunday will give completely different cluster patterns and cluster memberships from a Tuesday-to-Monday clustering.

The second idea is to set k to be some number much greater than the true number of clusters we expect to find, then do some postprocessing to find the *real* clusters. Empirically, we could not make this idea work, even on the trivial dataset introduced in the last section. We found that, even if k is extremely large, unless it is a significant fraction of T, we still get arbitrary sine waves as cluster centres. In addition, we note that the time complexity for k-means increases with k.

It is our belief that there is no simple solution to the problem of STS-clustering; the definition of the problem is itself intrinsically flawed.

4.4. Necessary conditions for STS clustering to work

We conclude this section with a summary of the conditions that must be satisfied for STS clustering to be meaningful.

Assume that a time series contains k approximately or exactly repeated patterns of length w. Further assume that we happen to know k and w in advance. A necessary (but not necessarily sufficient) condition for a clustering algorithm to discover the k patterns is that the weighted mean of the patterns must sum to a horizontal line and each of the k patterns must have approximately equal numbers of trivial matches.

It is obvious that the chances of both these conditions being met is essentially 0.

5. A case study on existing work

As we noted in the introduction, an obvious counter argument to our claim is the following. *Because many papers have been published that use time-series subsequence clustering as a subroutine and these papers produce successful results, time-series subsequence clustering must be a meaningful operation.* To counter this argument, we have reimplemented the most influential such work, the time-series rule-finding algorithm of Das et al. (1998) (the algorithm is not named in the original work, we will call it TSRF here for brevity and clarity).

5.1. (Not) finding rules in time series

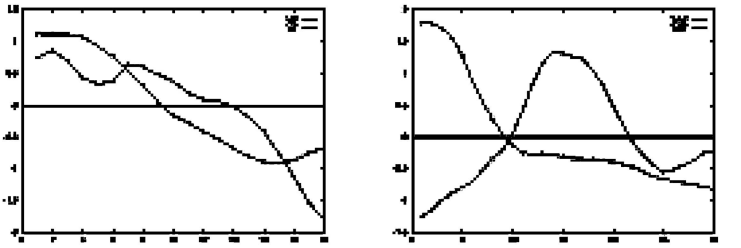
The algorithm begins by performing STS clustering. The centres of these clusters are then used as primitives to convert the real-valued time series into symbols, which are in turn fed into a slightly modified version of a classic association-rule algorithm (Agrawal et al. 1993). Finally, the rules are ranked by their J-measure, an entropy based measure of their significance.

The rule-finding algorithm found the rules shown in Fig. 13 using 19 months of NASDAQ data. The high values of support, confidence and J-measure are offered as evidence of the significance of the rules. The rules are to be interpreted as follows: In Fig. 13 (b), we see that "***if*** *stock rises then falls greatly, follow a smaller rise,* ***then*** *we can expect to see within 20 time units, a pattern of rapid decrease followed by a leveling out*." (Das et al. 1998).

What would happen if we used the TSRF algorithm to try to find rules in random-walk data, using exactly the same parameters? Because no such rules should exist by definition, we should get radically different results.[2] Figure 14 shows one such experiment; the support, confidence and J-measure values are essentially the same as in Fig. 13!

This one experiment might have been an extraordinary coincidence; we might have created a random-walk time series that happens to have some structure to it. Therefore, for every result shown in the original paper, we ran 100 recreations using

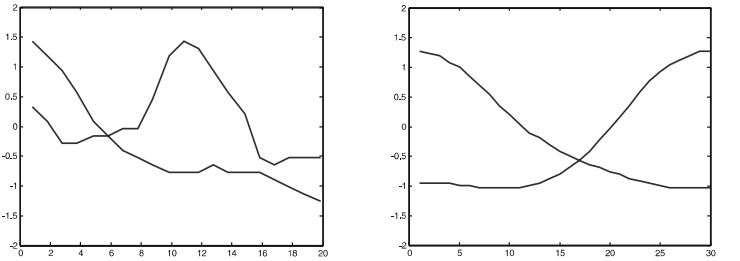
[2] Note that the shapes of the patterns in Figs. 13 and 14 are only very approximately sinusoidal. This is because the time series are relatively short compared with the window length. When the experiments are repeated with longer time series, the shapes converge to pure sine waves.



(a) (b)

w	d	Rule	Sup %	Conf %	J-Mea.	Fig
20	5.5	$7 \Rightarrow^{15} 8$	8.3	73.0	0.0036	(a)
30	5.5	$18 \Rightarrow^{20} 21$	1.3	62.7	0.0039	(b)

Fig. 13. Above, two examples of significant rules found by Das et al. (This is a capture of Fig. 4 from their paper.) Below, a table of the parameters they used and results they found



(a) (b)

w	d	Rule	Sup %	Conf %	*J*-Mea.	Fig
20	5.5	$11 \Rightarrow^{15} 3$	6.9	71.2	0.0042	(a)
30	5.5	$24 \Rightarrow^{20} 19$	2.1	74.7	0.0035	(b)

Fig. 14. Above, two examples of significant rules found in *random-walk* data using the techniques of Das et al. Below, we used identical parameters and found near identical results

different random-walk datasets, using quantum mechanically generated numbers to insure randomness (Walker 2001). In every case, the results published cannot be distinguished from our results on random-walk data.

The above experiment is troublesome, but perhaps there are simply no rules to be found in stock market data. We devised a simple experiment in a dataset that does contain known rules. In particular, we tested the algorithm on a normal, healthy electrocardiogram. Here, there is an obvious rule that one heartbeat follows another. Surprisingly, even with much tweaking of the parameters, the TSRF algorithm cannot find this simple rule.

The TSRF algorithm is based on the classic rule-mining work of Agrawal et al. (1993); the only difference is the STS step. Because the rule-mining work has been carefully vindicated in 100s of experiments on both real and synthetic datasets, it seems reasonable to conclude that the STS clustering is at the heart of the problems with the TSRF algorithm.

These results may appear surprising because they invalidate the claims of a highly referenced paper, and many of the dozens of extensions researchers have proposed (Das et al. 1998; Fu et al. 2001; Harms et al. 2002a, 2002b; Hetland and Satrom 2002; Jin et al. 2002a, 2002b; Mori and Uehara 2001; Osaki et al. 2000; Sarker et al. 2002; Uehara and Shimada 2002; Yairi et al. 2001). However, in retrospect, this result should not really be too surprising. Imagine that a researcher claims to have an algorithm that can differentiate between three types of Iris flowers (Setosa, Virginica and Versicolor) based on petal and sepal length and width[3] (Fisher 1936). This claim is not so extraordinary, given that it is well known that even amateur botanists and gardeners have this skill (British Irish Society 1997). However, the paper in question is claiming to introduce an algorithm that can find rules in stock-market time series. There is simply no evidence that any human can do this, in fact, the opposite is true: every indication suggests that the patterns much beloved by technical analysts such as the calendar effect are completely spurious (Jensen 2000; Timmermann et al. 1998).

6. A tentative solution

The results presented in this paper thus far are somewhat downbeat. In this section, we modify the tone by introducing an algorithm that *can* find clusters in some time series. This algorithm is not presented as the best way to find clusters in time series; for one thing, its time complexity is untenable for massive datasets. It is simply offered as an existence proof that such an algorithm exists and to pave the way for future research.

Our algorithm is motivated by the two observations in Sect. 4, that attempting to cluster every subsequence produces an unrealistic constraint and that considering trivial matches causes smooth, low-detail subsequences to form pseudo-clusters.

We begin by considering time-series motifs, a concept highly related to clusters. Motifs are overrepresented sequences in discrete strings, for example, in musical or DNA sequences (Reinert et al. 2000). Classic definitions of motifs require that the underlying data be discrete, but in recent work, the present authors have extended the definitions to real-valued time series (Chiu et al. 2003; Lin et al. 2002). Figure 15 illustrates a visual intuition of a motif and definition 5 defines the concept more concretely.

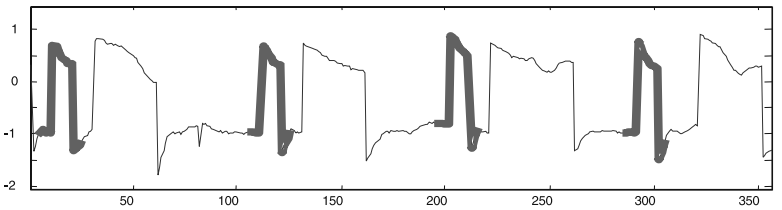


Fig. 15. An example of a motif that occurs four times in a short section of the Winding(4) dataset

[3] This of course is the famous Iris classification problem introduced by R.A. Fischer. It is probably the most referenced dataset in the world.

Definition 5. *k-motifs:* Given a time series T and a distance range R, the most significant motif in T (called *1-Motif*) is the subsequence C_1 that has the highest count of nontrivial matches. Subsequently, the Jth most significant motif in T (called *J-Motif*) is the subsequence C_J that has the highest count of nontrivial matches and satisfies $D(C_J, C_i) > 2R$, for all $1 \leq i < J$.

Figure 16 provides a visual explanation of why motifs are required to be at least $2R$ apart.

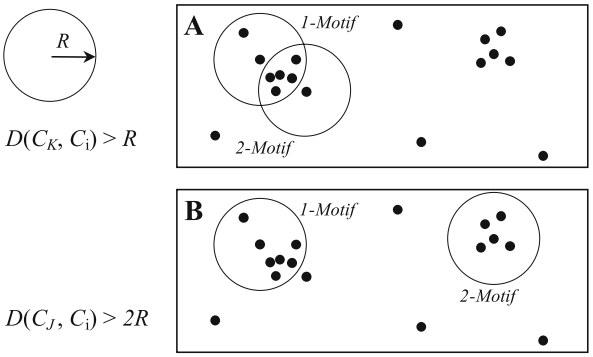


Fig. 16. A A visual explanation of why the definition of *J-Motif* requires that each motif to be at least $2R$ apart. If the motifs are only required to be R distance apart, as in **A**, then the two motifs may share the majority of their elements. In contrast, **B** illustrates that requiring the centres to be at least $2R$ apart insures the motifs are unique

Although motifs may be considered similar to clusters, there are several important differences, a few of which we enumerate here.

- When mining motifs, we must specify an additional parameter R.
- Assuming the distance R is defined as Euclidean, motifs always define circular regions in space, whereas clusters may have arbitrary shapes.[4]
- Motifs generally define a small subset of the data and not the entire dataset. Note that, in Fig. 16, there are several time series that are not included in any motif.
- The definition of motifs explicitly eliminates trivial matches.

Note that, while the first two points appear as limitations, the last two points explicitly counter the two reasons that STS clustering cannot produce meaningful results.

We cannot simply run a k-motif detection algorithm in place of STS clustering because a subset of the motifs discovered might really be a group that should be clustered together. For example, imagine a true cluster that sits in a hyperellipsoid. It might be approximated by 2 or 3 motifs that cover approximately the same volume. However, we could run a K-motif detection algorithm, with $K \gg k$ to extract promising subsequences from the data and then use a classic clustering algorithm to cluster only these subsequences. This simple idea is formalized in Table 3.

Step 2 of the algorithm requires a call to a motif discovery algorithm; an optimized exact algorithm for this appears in Lin et al. (2002) and a linear time-approximate algorithm appears in Chiu et al. (2003).

[4] It is true that k-means favors circular clusters, but more generally, clustering algorithms can define arbitrary spaces.

Table 3. An outline of the motif-based-clustering algorithm

Algorithm *motif-based-clustering*
1. Decide on a value for k.
2. Discover the K-motifs in the data, for $K = k \times c$ (c is some constant, in the region of about 2 to 30).
3. Run k-means, or k partitional hierarchical clustering, or any other clustering algorithm on the subsequences covered by K-motifs.

6.1. Experimental results

We have seen in Sect. 5 that the cluster centres returned by STS have been mistaken for meaningful clusters by many researchers. To eliminate the possibility of repeating this mistake, we will demonstrate the proposed algorithm on the dataset introduced in Sect. 4, which consists of the concatenation of 30 examples each of the cylinder, bell, funnel shapes, in random order. We would like our clustering algorithm to be able to find cluster centres similar to the ones shown in Fig. 8.

We ran the motif-based clustering algorithm with $w = 128$ and $k = 3$, which is fair because we also gave these two correct parameters to all the algorithms above. We needed to specify the value of R; we did this by simply examining a fraction of our dataset, finding ten pairs of subsequences we found to be similar, measuring the Euclidean distance between these pairs, and averaging the results. The cluster centres found are shown in Fig. 17.

These results tell use that, on at least some datasets, we can do meaningful clustering. The fact that this is achieved by working with only a subset of the data and explicitly excluding trivial matches further supports our explanations in Sects. 4.1 and 4.2 of why STS clustering is meaningless.

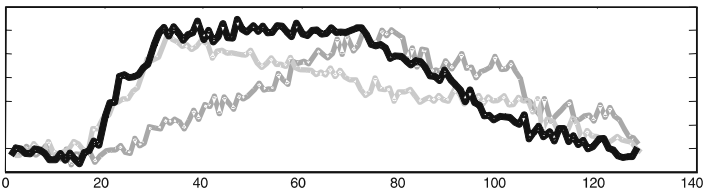


Fig. 17. The cluster centres found by the motif-based-clustering algorithm on the concatenated cylinder–bell–funnel dataset. Note the results are very similar to the prototype shapes shown in Fig. 7, and the cluster centres found by the whole matching case, shown in Fig. 8

7. Discussion and conclusions

As one might expect with such an unintuitive and surprising result, the original version of this paper caused some controversy when first published. Some suggested that the results were due to an implementation bug. Fortunately, many researchers have since independently confirmed our findings; we will note a few below.

Dr. Loris Nanni noted that she had encountered problems clustering economic times series. After reading an early draft of our paper she wrote "*At first we didn't*

understand what the problem was, but after reading your paper this fact we experimentally confirmed that (STS) clustering is meaningless!!" (Nanni 2003). Dr. Richard J. Povinelli and his student Regis DiGiacomo experimentally confirmed that STS clustering produces sine wave clusters regardless of the dataset used or the setting of any parameters (Povinelli 2003). Dr. Miho Ohsaki reexamined work she and her group had previously published and confirmed that the results are indeed meaningless in the sense described in this work (Ohsaki et al. 2002). She has subsequently been able to redefine the clustering subroutine in her work to allow more meaningful pattern discovery (Ohsaki et al. 2003). Dr. Frank Höppner noted that he had observed a year earlier than us that "...*when using few clusters the resulting prototypes appear very much like dilated and translated trigonometric functions*..." (Hoppner 2002); however, he did not attach any significance to this. Dr. Eric Perlman wrote to tell us that he had begun to scale up a project of astronomical time-series data mining (Perlman and Java 2003); however, he abandoned it after noting that the results were consistent with being meaningless in the sense described in this work. Dr. Anne Denton noted, "I've experimented myself, (and) the central message of your paper—that subsequence clustering is meaningless—is very right," and "it's amazing how similar the cluster centres for widely distinct series look!" (Denton 2003)

7.1. Conclusions

We have shown that a popular technique for data mining does not produce meaningful results. We have further explained the reasons why this is so.

Although our work may be viewed as negative, we have shown that a reformulation of the problem can allow clusters to be extracted from (streaming) time series. In future work, we intend to consider several related questions; for example, whether or not the weaknesses of STS clustering described here have any implications for model-based, streaming clustering of time series or streaming clustering of nominal data (Guha et al. 2000).

Acknowledgements. We gratefully acknowledge the following people who looked at an early draft of this work. Some of these people were justifiably critical of the work, and their comments lead to extensive rewriting and additional experiments. Their criticisms and comments greatly enhanced the arguments in this paper: Christos Faloutsos, Frank Höppner, Howard Hamilton, Daniel Barbara, Magnus Lie Hetland, Hongyuan Zha, Sergio Focardi, Xiaoming Jin, Shoji Hirano, Shusaku Tsumoto, Loris Nanni, Mark Last, Richard J. Povinelli, Zbigniew Struzik, Jiawei Han, Regis DiGiacomo, Miho Ohsaki, Sean Wang, and the anonymous reviewers of the earlier version of this paper (Keogh et al. 2003). Special thanks to Michalis Vlachos for pointing out the connection between our work and that of Slutsky.

References

Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on management of data. Washington, DC, 26–28 May, pp 207–216

Bar-Joseph Z, Gerber G, Gifford D, Jaakkola T, Simon I (2002) A new approach to analyzing gene expression time-series data. In: Proceedings of the 6th annual international conference on research in computational molecular biology. Washington, DC, 18–21 Apr, pp 39–48

Bradley PS, Fayyad UM (1998) Refining initial points for *K*-means clustering. In: Proceedings of the 15th international conference on machine learning. Madison, WI, 24–27 July, pp 91–99

British Irish Society, Species Group Staff (1997) A guide to species irises: their identification and cultivation. Cambridge University Press

Chiu B, Keogh E, Lonardi S (2003) Probabilistic discovery of time series motifs. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining. Washington, DC, USA, 24–27 Aug, pp 493–498

Cotofrei P (2002) Statistical temporal rules. In: Proceedings of the 15th conference on computational statistics —short communications and posters. Berlin, Germany, 24–28 Aug

Cotofrei P, Stoffel K (2002) Classification rules + time = temporal rules. In: Proceedings of the 2002 international conference on computational science. Amsterdam, The Netherlands, 21–24 Apr, pp 572–581

Das G, Lin K, Mannila H, Renganathan G, Smyth P (1998) Rule discovery from time series. In: Proceedings of the 4th international conference on knowledge discovery and data mining. New York, NY, 27–31 Aug, pp 16–22

Denton A (2003) Personal communication. Dec

Fisher RA (1936) The use of multiple measures in taxonomic problems. Ann of Eugen 7:179–188

Fu TC, Chung FL, Ng V, Luk R (2001) Pattern discovery from stock time series using self-organizing maps. Workshop notes of the workshop on temporal data mining at the 7th ACM SIGKDD international conference on knowledge discovery and data mining. San Francisco, CA, 26–29 Aug, pp 27–37

Gavrilov M, Anguelov D, Indyk P, Motwani R (2000) Mining the stock market: which measure is best? In: Proceedings of the 6th ACM international conference on knowledge discovery and data mining. Boston, MA, 20–23 Aug, pp 487–496

Guha S, Mishra N, Motwani R, O'Callaghan L (2000) Clustering data streams. In: Proceedings of the 41st annual symposium on foundations of computer science. Redondo Beach, CA, 12–14 Nov, pp 359–366

Halkidi M, Batistakis Y, Vazirgiannis M (2001) On clustering validation techniques. J Intellt Inf Syst 17:107–145

Harms SK, Deogun J, Tadesse T (2002a) Discovering sequential association rules with constraints and time lags in multiple sequences. In: Proceedings of the 13th international symposium on methodologies for intelligent systems. Lyon, France, 27–29 Jun, pp 432–441

Harms SK, Reichenbach S, Goddard SE, Tadesse T, Waltman WJ (2002b) Data mining in a geospatial decision support system for drought risk management. In: Proceedings of the 1st national conference on digital government. Los Angeles, CA, 21–23 May, pp 9–16

Hetland ML, Satrom P (2002) Temporal rules discovery using genetic programming and specialized hardware. In: Proceedings of the 4th international conference on recent advances in soft computing. Nottingham, UK, 12–13 Dec

Honda R, Wang S, Kikuchi T, Konishi O (2002) Mining of moving objects from time-series images and its application to satellite weather imagery. J Intell Inf Syst 19:79–93

Hoppner F (2002) Time series abstraction methods—a survey. In: Tagungsband zur 32. GI Jahrestagung 2002, Workshop on knowledge discovery in databases. Dortmund, Sept/Oct, pp 777–786

Jensen D (2000) Data snooping, dredging and fishing: the dark side of data mining. 1999 SIGKDD panel report. ACM SIGKDD Explor 1:52–54

Jin X, Lu Y, Shi C (2002a) Distribution discovery: local analysis of temporal rules. In: Proceedings of the 6th Pacific-Asia conference on knowledge discovery and data mining. Taipei, Taiwan, 6–8 May, pp 469–480

Jin X, Wang L, Lu Y, Shi C (2002b) Indexing and mining of the local patterns in sequence database. In: Proceedings of the 3rd international conference on intelligent data engineering and automated learning. Manchester, UK, 12–14 Aug, pp 68–73

Kendall M (1976) Time-series, 2nd ed. Griffin, London

Keogh E (2002a) Exact indexing of dynamic time warping. In: Proceedings of the 28th international conference on very large data bases. Hong Kong, 20–23 Aug, pp 406–417

Keogh E (2002b) The UCR time series data mining archive. Computer Science & Engineering Department, University of California, Riverside, CA. http://www.cs.ucr.edu/~eamonn/TSDMA/index.html

Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2001) Dimensionality reduction for fast similarity search in large time series databases. J Knowl Inf Syst 3:263–286

Keogh E, Kasetty S (2002) On the need for time series data mining benchmarks: a survey and empirical demonstration. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining. Edmonton, Alberta, Canada, 23–26 July, pp 102–111

Keogh E, Lin J, Truppel W (2003) Clustering of time series subsequences is meaningless: implications for past and future research. In: Proceedings of the 3rd IEEE international conference on data mining. Melbourne, FL, 19–22 Nov, pp 115–122

Li C, Yu PS, Castelli V (1998) MALM: a framework for mining sequence database at multiple abstraction levels. In: Proceedings of the 7th ACM International conference on information and knowledge management. Bethesda, MD, 3–7 Nov, pp 267–272

Lin J, Keogh E, Patel P, Lonardi S (2002) Finding motifs in time series. Workshop notes of the 2nd workshop on temporal data mining at the 8th ACM international conference on knowledge discovery and data mining. Edmonton, Alberta, Canada, 23–26 July

Mantegna RN (1999) Hierarchical structure in financial markets. Eur Physical J B 11:193–197

Mori T, Uehara K (2001) Extraction of primitive motion and discovery of association rules from human motion. In: Proceedings of the 10th IEEE international workshop on robot and human communication. Bordeaux-Paris, France, 18–21 Sept, pp 200–206

Nanni L (2003) Personal communication. 22 Apr

Oates T (1999) Identifying distinctive subsequences in multivariate time series by clustering. In: Proceedings of the 5th international conference on knowledge discovery and data mining. San Diego, CA, 15–18 Aug, pp 322–326

Ohsaki M, Sato Y, Yokoi H, Yamaguchi T (2002) A rule discovery support system for sequential medical data, in the case study of a chronic hepatitis dataset. Workshop notes of the international workshop on active mining at IEEE international conference on data mining. Maebashi, Japan, 9–12 Dec

Ohsaki M, Sato Y, Yokoi H, Yamaguchi T (2003) A rule discovery support system for sequential medical data, in the case study of a chronic hepatitis dataset. Workshop notes of discovery challenge workshop at the 14th European conference on machine learning/the 7th European conference on principles and practice of knowledge discovery in databases. Cavtat-Dubrovnik, Croatia, 22–26 Sep

Osaki R, Shimada M, Uehara K (2000) A motion recognition method by using primitive motions. In: Arisawa H, Catarci T (eds) Advances in visual information management: visual database systems. Kluwer, pp 117–127

Perlman E, Java A (2003) Predictive mining of time series data. In: Payne HE, Jedrzejewski RI, Hook RN (eds) ASP conference series, vol 295, Astronomical data analysis software and systems XII. San Francisco, pp 431–434

Povinelli R (2003) Personal communication. 19 Sept

Radhakrishnan N, Wilson JD, Loizou PC (2000) An alternative partitioning technique to quantify the regularity of complex time series. Int J Bifur Chaos 10:1773–1779

Reinert, G, Schbath, S, Waterman MS (2000) Probabilistic and statistical properties of words: an overview. J Comput Biol 7:1–46

Roddick JF, Spiliopoulou M (2002) A survey of temporal knowledge discovery paradigms and methods. Trans Data Eng 14:750–767

Sarker BK, Mori T, Uehara K (2002) Parallel algorithms for mining association rules in time series data. CS24-2002-1, Technical report

Schittenkopf C, Tino P, Dorffner G (2000) The benefit of information reduction for trading strategies. Report series for adaptive information systems and management in economics and management society. July. Report #45

Steinback M, Tan PN, Kumar V, Klooster S, Potter C (2002) Temporal data mining for the discovery and analysis of ocean climate indices. Workshop notes of the 2nd workshop on temporal data mining at the 8th ACM SIGKDD international conference on knowledge discovery and data mining. Edmonton, Alberta, Canada, 23 July

Timmermann A, Sullivan R, White H (1998) The dangers of data-driven inference: the case of calendar effects in stock returns. FMG discussion papers dp0304, Financial Markets Group and ESRC

Tino P, Schittenkopf C, Dorffner G (2000) Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: lessons learned from financial volatility trading. Report series for adaptive information systems and management in economics and management sciences. July. Report #46

Truppel W, Keogh E, Lin J (2003) A hidden constraint when clustering streaming time series. UCR technical report

Uehara K, Shimada M (2002) Extraction of primitive motion and discovery of association rules from human motion data. Progress in discovery science. Lecture notes in artificial intelligence, vol 2281. Springer, Berlin, Heidelberg, New York, pp 338–348

van Laerhoven K (2001) Combining the Kohonen self-organizing map and K-means for on-line classification of sensor data. In: Dorffner G, Bischof H, Hornik K (eds) Artificial neural networks. Lecture notes in artificial intelligence, vol 2130. Springer, Berlin, Heidelberg, New York, pp 464–470

Walker J (2001) HotBits: genuine random numbers generated by radioactive decay. http://www.fourmilab.ch/hotbits

Yairi Y, Kato Y, Hori K (2001) Fault detection by mining association rules in house-keeping data. In: Proceedings of the 6th international symposium on artificial intelligence, robotics and automation in space. Montreal, Canada, 18–21 Jun

Author biographies

Eamonn Keogh is an assistant professor of computer science at the University of California, Riverside. His research interests are in data mining, machine learning and information retrieval. Several of his papers have won best paper awards, including papers at SIGKDD and SIGMOD. Dr. Keogh is the recipient of a 5-year NSF Career Award for *Efficient Discovery of Previously Unknown Patterns and Relationships in Massive Time Series Databases*.

Jessica Lin is a Ph.D. candidate at the University of California, Riverside, where she received her B.S. and M.S. degrees in computer science. Her research interests include data mining and information retrieval.

Correspondence and offprint requests to: Eamonn Keogh, University of California–Riverside, Computer Science & Engineering Department, Riverside, CA 92521, USA. Email: eamonn@cs.ucr.edu