

Useful Clustering Outcomes from Meaningful Time Series Clustering

Jason R. Chen

Department of Information Engineering
Research School of Information Science and Engineering
College of Engineering and Computer Science
The Australian National University
Canberra, ACT, 0200, Australia
Email: Jason.Chen@anu.edu.au

Abstract

Clustering time series data using the popular subsequence (STS) technique has been widely used in the data mining and wider communities. Recently the conclusion was made that it is meaningless, based on the findings that it produces (a) clustering outcomes for distinct time series that are not distinguishable from one another, and (b) cluster centroids that are smoothed. More recent work has since showed that (a) could be solved by introducing a lag in the subsequence vector construction process, however we show in this paper that such an approach does not solve (b). Motivating the terminology that a clustering method which overcomes (a) is meaningful, while one which overcomes (a) and (b) is useful, we propose an approach that produces *useful* time series clustering. The approach is based on restricting the clustering space to extend only over the region visited by the time series in the subsequence vector space. We test the approach on a set of 12 diverse real-world and synthetic data sets and find that (a) one can distinguish between the clusterings of these time series, and (b) that the centroids produced in each case retain the character of the underlying series from which they came.

Keywords: Time Series, Clustering, Subsequence-Time-Series Clustering

1 Introduction

Clustering analysis is a tool used widely in the Data Mining community and beyond (B.S.Everitt et al. 2001). In essence, the method allows us to summarise what can be a very large data set X with a much smaller set $C = \{c_i | i = 1, \dots, k\}$ of representative points (called centroids), and a membership map $\gamma : X \rightarrow C$ relating each point in X to its representative in C . It then becomes much easier to access and manipulate the essential “information” in the data set, and this is one of the reasons why clustering is so widely used, and often as a pre-processing step for further analysis.

Time series are a special type of data set where elements have a temporal ordering. Imagine we have a system or process evolving over time, and that we take measurements x_t (scalar or vector) on a fixed periodic basis. Then our time series X is

$$X = \{x_t | t = 1, \dots, n\} \quad (1)$$

where n is the number of measurements taken, and where the subscript t reflects the temporal ordering in the set.

Given a single time series, or a number of time series from the same system or process, we often wish to summarise the series as a set of key features or shapes found in

the series. Clustering analysis seems an obvious candidate for such a purpose and, historically, the approach has been to construct a set Z of delay vectors (called “subsequences” in (E.Keogh et al. 2003), “regressors” in (G.Simon et al. 2006)) by moving a sliding window of length w across the data, where the p th delay vector is given by,

$$z_p = \{x_{p-(w-1)}, x_{p-(w-2)}, \dots, x_{p-2}, x_{p-1}, x_p\} \quad (2)$$

and where $Z = \{z_p | p = w \dots n\}$. Clustering Z in \mathbb{R}^w using one of a number of possible metrics (e.g. including those induced by the L^p norms, but usually the L^2 (Euclidean) norm, Mahalanobis distance, DTW distance, etc.) was typically conducted using one of the many possible clustering algorithms (k-means, hierarchical, EM-algorithm, Self Organising Maps, etc.) to produce a set C of representatives that was used as the set of key features in the time series. In this paper we refer to this approach as Subsequence-Time-Series (STS) clustering.

STS Clustering has been widely used, as highlighted in (E.Keogh et al. 2003). However, surprisingly, work in (E.Keogh et al. 2003) found that the outcome of clustering a time series in this way is meaningless. The two principal findings made to support this conclusion were,

- (A) that one cannot distinguish between the clustering outcomes of distinct time series, and
- (B) that cluster representatives are smoothed and generally do not look at all like any part of the original time series

Clearly, both (A) and (B) are problematic properties for any prospective “summary” of a time series to have. While (E.Keogh et al. 2003) proposed that these two problems were one and the same, i.e. that one cannot distinguish between cluster centres of different time series because they are all smoothed and hence alike, we show in this paper that this is in fact false. Problems (A) and (B) are really two separate problems that can be solved separately, and it will turn out that our contribution in this paper will be to propose a method which solves problem (B).

A number of reasons and solutions to the dilemma raised in (E.Keogh et al. 2003) have been proposed in the literature. Struzik (Z.R.Struzik 2003) proposed that the “meaningless” outcome results only in pathological cases, i.e. when the time series structure is fractal, or when the redundancy of subsequence sampling causes trivial matches to hide the underlying rules in the series. They suggested autocorrelation operations to suppress the latter, but did not confirm this with experiments. Denton (A.Denton 2005) proposed density based clustering, as opposed to k-means or hierarchical clustering, as a solution. They proposed that time series can contain significant noise, and that density based clustering identifies and removes this noise by only considering clusters rising above a preset threshold in the density landscape. Chen (J.R.Chen 2007) proposed an alternative clustering metric based on temporal and formal distances that did lead to meaningful time series clustering,

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at the Sixth Australasian Data Mining Conference (AusDM 2007), Gold Coast, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 70, Peter Christen, Paul Kennedy, Jiuyong Li, Inna Kolyshkina and Graham Williams, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

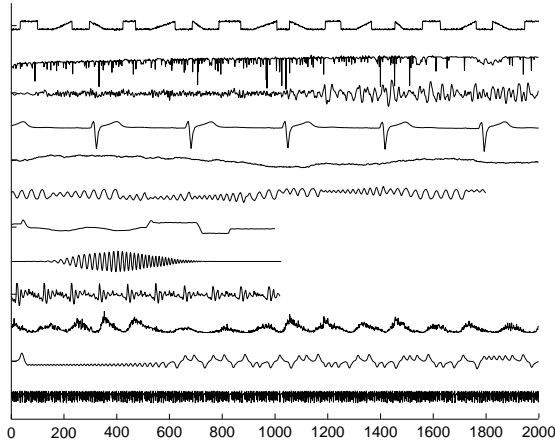


Figure 1: The time series used in experiments: CBFV time series, the Earthquake, Koski-ECG, Random Walk, Chaotic, Shuttle, Speech time series from the UCR database (E.Keogh 2002), and the Sunspot, Lorenz and Henon time series from (G.Simon et al. 2006)

however the work was limited to time series that are cyclic. Peker (K.Peker 2005) identified that clustering with a very large number of clusters leads to cluster centroids which are more representative of the signal in the original time series. Goldin et. al. (D.Goldin et al. 2006) proposed an alternative way of measuring distance between different time series clusterings, which gave quite high recognition rates with respect to the time series from which they came. Although each of these works has shown promising results, none has clearly demonstrated a method which overcomes the problems (A) and (B) identified in (E.Keogh et al. 2003).

Some work in the literature proposed that a lag should be introduced into the window formation aspect of STS-clustering (J.R.Chen 2007, G.Simon et al. 2006), i.e.

$$z_p = \{x_{p-(w-1)q}, x_{p-(w-2)q}, \dots, x_{p-2q}, x_{p-q}, x_p\} \quad (3)$$

and $Z = \{z_p | p = (w-1)q+1, \dots, n\}$. That is, delay vectors should not be formed (as in Equation 2) from w contiguous data points in the time series, but rather from w data points each separated themselves by q data points. In essence, introducing the lag q unfolds the delay vector distribution (spreads delay vectors away from the diagonal of delay space) so that the ‘‘information’’ existing in the time series is retained. Work in (G.Simon et al. 2006) went on to show that the clustering outcomes of different time series could indeed be distinguished from one another if this unfolding is carried out first. We will refer to this approach as Unfolded-Time-Series (UTS) Clustering.

While we feel that the work in (G.Simon et al. 2006) provides an important step towards the solution of the time series clustering dilemma, i.e. it solves problem (A), we propose that it is not the final one. More specifically, work in (J.R.Chen 2007) found that, even when a lag was introduced in the the delay vector construction process, smoothed cluster centroids were still produced. i.e. problem (B) still existed. Specifically, quite a number of lags were tried in the experiments involving the Cylinder-Bell-Funnel (CBF) time series in that work, however none gave the desired outcome. Hence, it seems there is some discrepancy about whether introducing a lag results in meaningful clustering, or whether something else is required. The remainder of this paper is about making sense of what is happening here.

Time Series	Lag (q)	Time Series	Lag (q)
CBFV	9	Shuttle	50
Balloon	7	Flutter	5
Earthquake	7	Speech	7
Koski ECG	18	Sunspot	20
Random Walk	100	Lorenz	14
Chaotic	7	Henon	1

Table 1: Lags selected for $w = 5$

2 Meaningful versus Useful

Let us first provide a grounding to our work here by confirming the results in (G.Simon et al. 2006). We select 12 time series (as shown in Figure 1): (1) the CBF time series used in (J.R.Chen 2007) with variability added in feature onset and duration¹, (2-9) from (E.Keogh 2002) identified in (E.Keogh et al. 2003) as having some of the most distinct characteristics of any time series in that database, and (10-12) used in (G.Simon et al. 2006). We adopt the same basic methodology as in (G.Simon et al. 2006) (and (E.Keogh et al. 2003)) of creating a set of centroid sets for each time series and measuring the difference between centroid locations for the same, and then different, time series. The steps taken were (details, where not present, can be found in (G.Simon et al. 2006)):

1. Some time series from (E.Keogh 2002) are very long. For these time series we take only the first 2000 points. This is sufficient to make our point in this paper
2. Normalise each time series to have zero mean and standard deviation one. This is mandatory if a fair comparison of distances between centroid sets of different time series pairs is to be made.
3. Choose a lag according to a maximum or plateau in the Sum-of-Distance-To-Diagonal (*SDTD*) measure, as proposed in (G.Simon et al. 2006). We in fact use the Mean-Distance-To-Diagonal (*MDTD*), calculated by dividing *SDTD* by the number of points in Z (a constant), since this measure will be useful later. That is,

¹In contrast to work in (J.R.Chen 2007), where a fixed feature onset and duration CBF time series was used, in this paper, we construct the CBF time series with features with variable onset and duration. Specifically, we take ‘‘windows’’ of data points of length 128, where each window contains one of either the Cylinder, Bell, or Funnel features, and where the onset and termination of the feature is selected with a uniform probability over the data point ranges of 16 to 48 and 80 to 112 respectively. This results in, by visual inspection, at least an equal, if not greater, variability in feature onset and duration than the original CBF time series used in (E.Keogh et al. 2003). We show the resulting time series in Figure 2. For clarity, denote the ‘‘fixed’’ CBF time series used in (J.R.Chen 2007) as the CBF time series, and the ‘‘variable’’ CBF time series we use here as the CBFV time series.

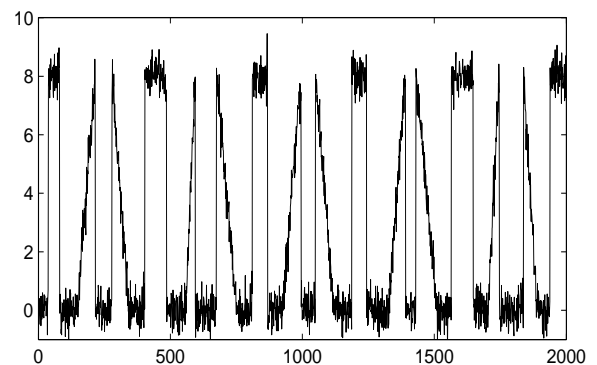


Figure 2: CBFV Time Series

$$MDTD = \frac{1}{n'} \sum_{p=1}^{n'} \sqrt{\frac{\|v_1 - z_p\|^2 \|u\|^2 - ((v_1 - z_p)^T u)^2}{\|u\|^2}} \quad (4)$$

where $n' = n - (w - 1)q$ reflects the number of delay vectors in Z , $\|\cdot\|$ is the L_2 norm, and $u = v_2 - v_1$ where v_1 and v_2 are the w -dimensional origin $(0, 0, 0, 0, \dots)$ and unit vectors $(1, 1, 1, 1, \dots)$ respectively. The idea here is to find a lag that provides a sufficiently unfolded delay vector distribution, and maxima in the $MDTD$ value reflect when this occurs. The lags selected for each time series, for $w = 5$ are shown in Table 1.

4. Cluster the normalised time series using UTS-clustering with the lag selected from Step 3. Use k-means clustering and the Euclidean metric. Repeat the clustering two lots of 100 times to create two sets of centroid sets for each time series: a base set \mathcal{B} and a comparison set \mathcal{C} .
5. For each base centroid set \mathcal{B}_r (i.e. $r = 1, \dots, 12$ for our set of 12 time series here) calculate the distance between it and every comparison set \mathcal{C}_s , $s = 1, \dots, 12$. When $r = s$, the distance is called the *within* distance (i.e reflecting that both sets of clusterings came from the same time series), while for $r \neq s$, the distance is called the *between* distance.
6. For each time series as the base, plot the within distance value, and the between distances values, for a range of number-of-cluster (k) values (from 3 to 70).

The rationale behind these steps is that, if the clustering of one time series is distinguishable from that of another, then the within distance should always be less than the between distance. In other words, the position of centroids in clusterings produced from the same time series should coincide more than those of different time series. As in (G.Simon et al. 2006), *sets* of centroid sets were used to allow for the effect of different clustering outcomes from the one time series due to differing initial seeds.

The result of the experiments is shown in Figure 3 and we see that the clustering outcome produced by different time series are indeed distinguishable if a lag is introduced, as reported by (G.Simon et al. 2006). For each time series as a base, the distance between centroids produced by clustering the same time series is always smaller than that produced by clusterings from different time series. The results shown here are for $w = 5$, however as in (G.Simon et al. 2006), this experiment was also repeated with a number of higher and lower w values, and were found to support the same conclusions in each case. Hence introducing a lag into the sliding windows part of the STS-clustering process does indeed seem to solve problem (A).

We now then return to our initial observation that introducing a lag in the CBFF experiment in (J.R.Chen 2007) did not prevent smoothed cluster centroids from being produced, i.e. that it did not solve problem (B). A number of lag values were selected in those experiments; however it may have been that an appropriate one (i.e. one obtained using the $MDTD$ criteria) was not selected. Let us repeat the experiment conducted in (J.R.Chen 2007), but this time use the $MDTD$ criteria to select the lag. UTS-cluster the CBFV time series with $w = 10$, $k = 3$ and a lag $q = 13$ selected using the $MDTD$ criteria. Figure 4 shows the result is still the same set of smoothed cluster centroids first reported in (E.Keogh et al. 2003). In the lower plot in Figure 4, we indicate the membership of the different delay vectors constructed from the time series. Each delay vector in Z has a distinct element x_t in the time series as its first component. For each delay vector, we plot in Figure 4 a coloured dot below the x_t which forms its first component,

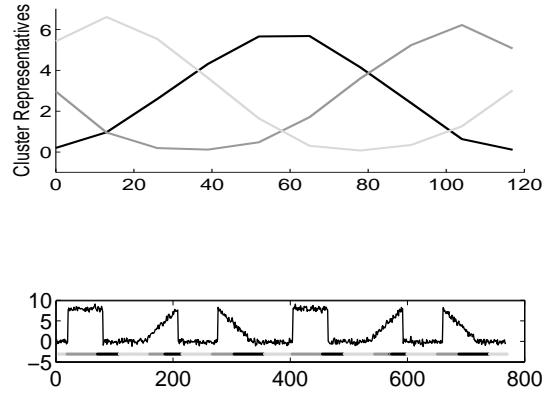


Figure 4: UTS Clustering: CBFV results

where the colour of the dot indicates the delay vector's cluster membership. In essence we have formed a membership "bar" for delay vectors that can be superimposed back on the original time series. Note that the membership outcome in Figure 4 is not what we are looking for, since it has not captured the three features into separate clusters.

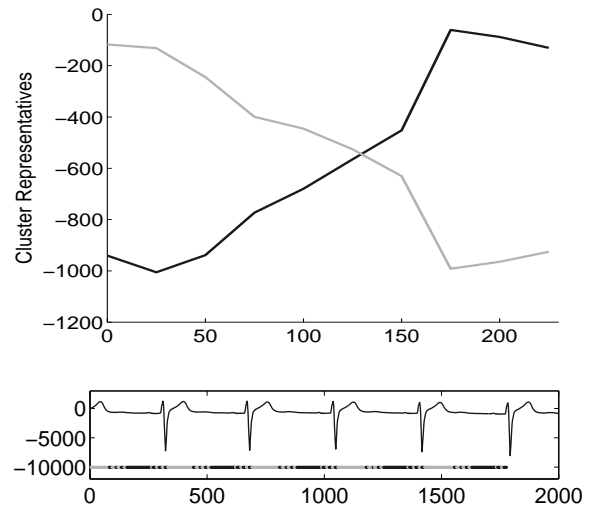


Figure 5: UTS Clustering: Koski-ECG results

Let us try UTS-clustering other time series in our data set to see if the CBFV time series represents an isolated problematic result. First, try to identify the two main features in the Koski-ECG time series: the in-beat, and the between-beat phases of the heart in this series. Figure 5 shows the centroids produced using $w = 10$, $k = 2$ and a lag $q = 25$ selected using the $MDTD$ criteria. The resulting centroids look nothing like the features in the original time series. Further, although the membership bar broadly demarcates between the in-beat and between-beat phases, a curious oscillation between memberships occurs at transitions.

Next, let us try UTS-clustering the Chaotic time series. We choose to look for 3 clusters, with $w = 10$ and a lag $q = 10$ selected using the $MDTD$ criteria. Figure 6 shows the strange result. The membership bar seems to indicate that the UTS clustering method has identified oscillations in this time series occurring at three different vertical axis levels (i.e. roughly at the values -0.25, 0 and 0.25), however the oscillations in the time series at these three levels have mysteriously been smoothed. Note that there is no part of

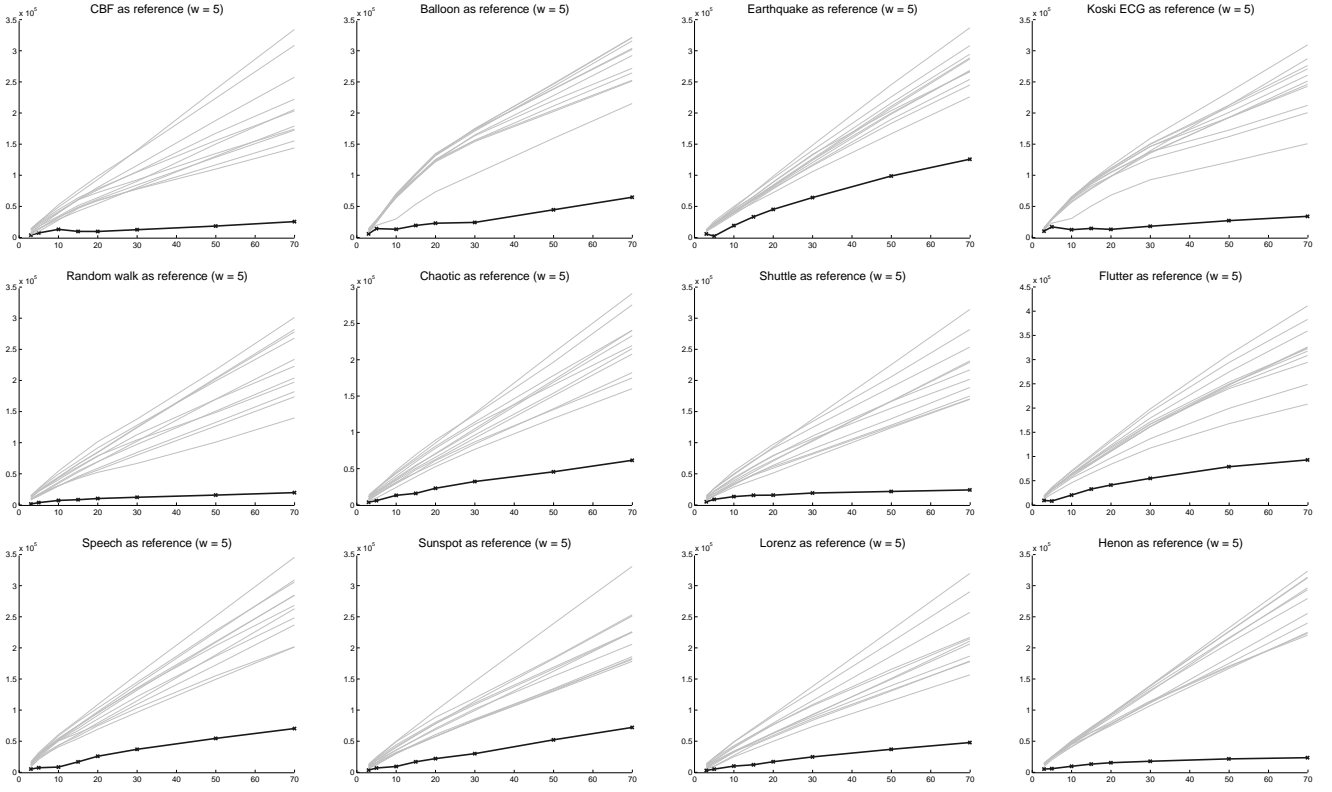


Figure 3: Results of the within-between distance experiments using UTS Clustering

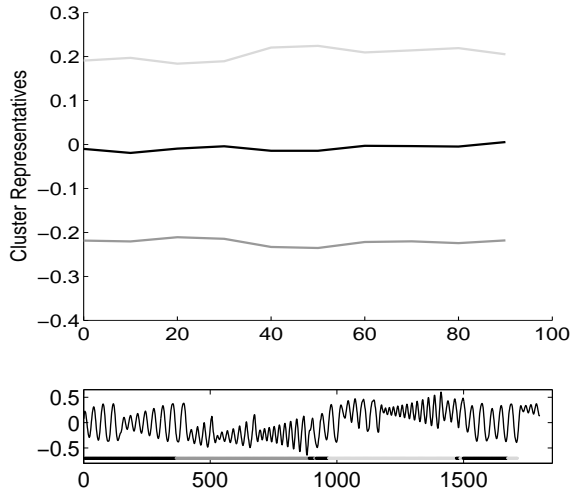


Figure 6: UTS Clustering: Chaotic results

the original time series at this time scale where the flat segments shown occurred.

It would seem, then, that introducing a lag into the sliding windows part of the STS-clustering process does not solve problem (B). However, let us conduct another experiment involving all the time series in our data set that further confirms what we have just observed. Work in (J.R.Chen 2007) proposed that the smoothing effect associated with problem (B) occurs because the STS and UTS clustering methods produce centroids that lie away from the cluster members they are meant to represent. The work there made the observations based on experiments with a small number of cyclic time series. Let us confirm this observation using the full range of general time series data sets introduced above. We propose a simple measure we term the *CP-PP*

ratio to measure the existence of this effect. Given a set $C = \{c_i | i = 1, \dots, k\}$ of centroids from a time series clustering, first, measure the average minimum centroid to data point distance as

$$MCD = \frac{1}{k} \sum_{i=1}^k \min_j d(c_i, z_{ji}), 1 \leq j \leq n_i \quad (5)$$

where z_{ji} indicates the j th data point in the i th cluster, and where $d(\cdot, \cdot)$ indicates the Euclidean Distance. That is, *MCD* gives the average distance between the centroids in C and the nearest members in their respective clusters. Next calculate, for every point in Z , the average distance between this data point and the closest data point in the same cluster as

$$MDD = \frac{1}{k} \sum_{i=1}^k \frac{1}{n_i} \sum_{p=1}^{n_i} \min_j d(z_{pi}, z_{ji}), 1 \leq j \leq n_i \quad (6)$$

Finally, take the *CP-PP* ratio simply as

$$CP-PP = \frac{MCD}{MDD} \quad (7)$$

One would envisage *CP-PP*'s generally at, or less than, unity, reflecting a deflationary influence on *MCD* of clusters centres lying in among the data points in the cluster, and the inflationary influence on *MDD* of outliers and points on the boundaries of the cluster. If we UTS-cluster each of the 12 time series introduced above and calculate *CP-PP* for a range of k values, we obtain the results shown in Figure 7 (we split these results across two plots for clarity). Note the large values of *CP-PP* for many time series, especially the Henon, Shuttle, Koski-ECG, Random Walk, Lorenz and Chaotic time series. These larger values of *CP-PP* reduce at higher k values, and this concurs with the findings of Peker (K.Peker 2005) that clustering with large k numbers produces centroids with characteristics more like those of

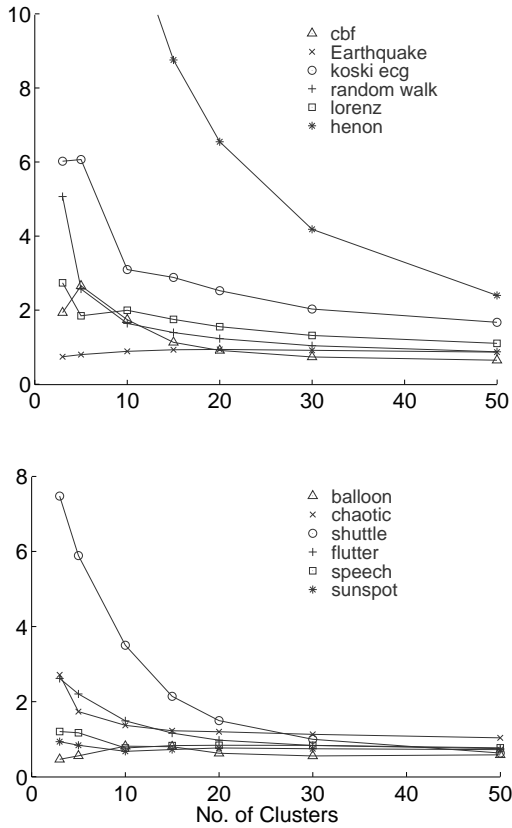


Figure 7: Ratio of centroid-to-nearest-point distance to point-to-nearest-point distance (CP-PP Ratio) for a range of k values

the original time series. Note that the results in Figure 7 are average values calculated over 100 clustering runs for each time series using random initial seeds, and so are not the result of any particular clustering initialisation used.

Let us now summarise the resulting situation. Work in (E.Keogh et al. 2003) concluded that STS clustering was not meaningful for reasons (A) and (B) above. Work in (G.Simon et al. 2006) proposed that clustering is meaningful if a lag is introduced. Our results here would suggest that clustering time series does not necessarily become meaningful (in the sense that it was used in (E.Keogh et al. 2003), i.e. (A) and (B) above) by only introducing a lag, since, although (A) is overcome, (B) is not. Hence we now introduce a more precise set of terminology to reflect this fact.

Definition 1 *The outcome of a time series clustering algorithm is meaningful if the cluster centres produced for one time series are distinguishable from those produced for other distinct time series.*

One can determine if cluster centres are distinguishable based on the experimental method outlined in Steps 1 to 6 above. We propose that *meaningful* is the correct term to use here. If the cluster centroids produced from a particular time series are distinguishable (i.e. distinct) from those of many other time series, then the clustering is meaningful since it has preserved the “information” in the original time series that made the time series distinct from all others.

Definition 2 *The outcome of a time series clustering algorithm is useful if it is (i) meaningful and (ii) if centroids well represent members in their respective clusters (in the sense that they retain the properties - e.g. sharpness, magnitudes, etc - of the signal in the original time series).*

We adopt the term *useful* here since our stated motivation at the outset of this paper was for clustering to obtain a “summarising” feature set for time series data. A clustering is

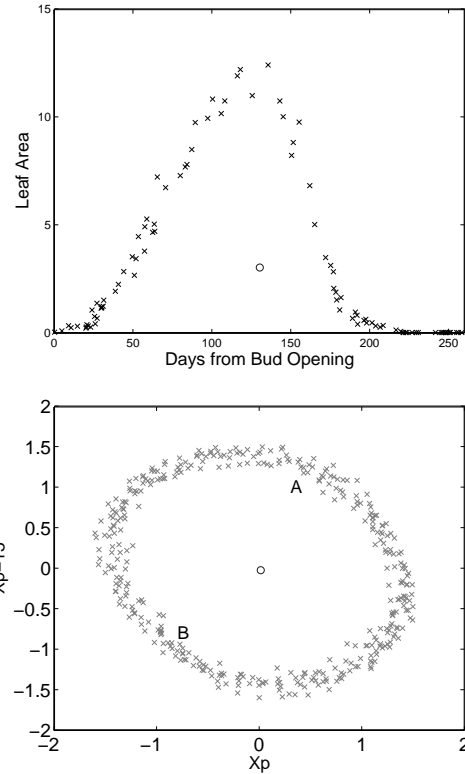


Figure 8: Hypothetical tree leaf area (top) and Undamped Pendulum delay space plots (bottom)

only useful in this context if centroids well represent their cluster members in the sense that they retain the properties of the signal in the original time series. Our aim for the remainder of this paper is to propose a method that produces useful time series clustering.

3 Asking the Right Question

So, according to the definitions in the previous section, introducing an appropriate lag leads to meaningful time series clustering, but not to useful clustering. How can the clustering process produce cluster representatives that look nothing like the cluster members they represent? We propose two possibilities; either clustering as a method is flawed, or that the clustering technique is sound, but that we are applying it in the wrong way, (i.e. we are using it in a way that is asking the wrong question). The clustering technique has a long history of successful use outside the time series domain, so it is improbable the fault lies there. We then explore if we could be applying the technique in the wrong way.

The general technique of clustering involves extracting some common pertinent features (e.g. height and weight) from a set of objects (e.g. children) and creating a clustering space formed as a coordinate system where each axis represents one of the aforementioned features. In constructing such a space, an important question to ask is what subset of this space corresponds to realisable instances or outcomes in these objects in the real world? It seems clear that we should limit any clustering process to this subset, or non-sensical clustering outcomes can result.

For example, take for illustration purposes a hypothetical data set of the total leaf area for a particular species and maturity of healthy deciduous tree during the growing season. Figure 8 (top) shows the data in a clustering space formed by capturing the two pertinent parameters from the physical system: the total leaf area in metres squared and the number of days from when the leaf emerges from the

bud. Data points are shown as crosses, and if we cluster for a single cluster, we obtain a non-sensical outcome: a healthy tree in mid-summer without (or with few) leaves. However, it is easy to see why this point was chosen as the centroid; it really is the point in the clustering space which has minimum sum of distance to all the data points. Clustering for a single cluster is an unusual, but valid, step which best highlights the point we wish to make. Clustering with more than one cluster here leads to the same effect, albeit in a less pronounced way.

The previous example was illustrative, but involved a non-time series derived data set. Lets look more closely at this issue with regard to clustering a set of time-series-derived delay vectors. In Figure 8 (bottom) we show a plot of a set of delay vectors of a simple undamped pendulum time series with measurement noise ² (ignore the labels A and B for the moment). We know that this pendulum system only lives in (i.e. it can only produce delay vectors that lie in) the annular region shown, and so we should limit any clustering process here to this subset of delay space. If we take the usual approach of clustering without restriction (e.g. k-means and the Euclidean Metric) for one centroid, we get the problematic result shown. This result is problematic since the outcome represented by this point (zero swing amplitude with zero velocity) does not occur in this system. Again, it is easy to see why this point was chosen as the centroid; it really is the point in the clustering space which has minimum sum of distance to all the data points. As with the Leaf-Area example, clustering with more than one cluster here leads to the same effect, albeit in a less pronounced way. What sense does it make to cluster in a space which includes points that cannot be produced by the underlying system? i.e. that can never appear in the time series. Note that the idea that clustering should be limited to a subset of the clustering space pertinent to the underlying physical system is not new. Work proposing methods for clustering on subspaces ((R.M.Haralick & R.Harpaz 2005) and references therein), and on manifolds (M.Breitenbach & G.Z.Grundic 2005) exists and is motivated by exactly this line of thinking, although we should not expect arbitrary time series data to generate delay vectors which exhibit a subspace or manifold structure.

So, we have proposed that one should cluster in a restricted region corresponding to plausible outcomes realisable in nature. The problem is that we have available a set of time-series-derived delay vectors that generally represent only a partial view of where in delay space the underlying system lives. We then need to make an assumption as to where our system lives in delay space. The two obvious possible approaches are: (i) cluster in delay space without restriction (the approach to date), or (ii) cluster in that part of delay space where we have evidence (from the time series) that the system lives. We propose that each is a valid assumption to make, but that each asks a different question. That is, if we want to assume that the system can produce the full range of signals corresponding to an unrestricted set of delay vectors in delay space, then the answer we get necessarily can include delay vectors as centroids that were not in (or not like any in) the original time series. As we saw in the Leaf-Area and Undamped-Pendulum examples above, these centroids really do represent the best average member of members in their cluster if we accept that the physical system can produce any delay vector in delay space. The clustering technique is providing an appropriate answer to the question we are asking here (i.e. we cannot complain about centroids being smoothed).

The other obvious approach we can take is to make assumption (ii), which corresponds to asking a different question. When we look to cluster a time series for k clusters, the question we generally want to ask is: given the features (delay vectors) that were *seen* in the time series, which k of

²This time series consists of essentially a noisy sinusoid. The plot shown is then formed by applying Equation 3 with $w = 2$ and a lag q which unfolds the data into the annular region shown

these features “summarise” (in the sense that we described how the clustering technique summarises a data set in the introduction to this paper) the time series best?, i.e. we are looking for (given Definition 2 above) a useful clustering. Assumption (ii) then makes sense since we are forming a clustering space containing only these features, and the clustering technique will return what we desire; a centroid set as the set of features *existing* in the time series that best summarise the time series. With this as motivation, we now present the details of the approach.

4 An Algorithm

Algorithm 1 TF Clustering Algorithm

- 1: Find the ED distance between all point pairs in Z , and store in A^* , i.e. $A_{ij}^* = d(z_i, z_j)$, where $d(\cdot, \cdot)$ denotes the Euclidean Distance metric
 - 2: Set main diagonal of A to zero
 - 3: Set first upper and lower diagonal of A equal to those in A^*
 - 4: For each remaining entry in A , set A_{ij} to A_{ij}^* if $A_{ij}^* < \epsilon$ or zero otherwise
 - 5: Use Floyd Warshall Algorithm or alternative to calculate a shortest path distance matrix D from A
 - 6: Cluster using the K-medoids algorithm on the space $(Z, d_{TF}(\cdot, \cdot))$ where $d_{TF}(z_i, z_j) = D_{ij}$
-

There are three main phases to our approach. The first phase (Steps 1 to 4 in Algorithm 1) is to define a clustering “space” that is restricted to the region of delay space that was visited by the time series. Our approach is to capture the geometric structure of the delay vectors in delay space in a graph \mathcal{G} , where there is a node in \mathcal{G} for each delay vector in Z , and where arcs represent the Euclidean distance between delay vectors. The key to creating a space that is restricted to the region visited by the time series using this approach is to apply an upper bound ϵ on the distance that can exist between vectors, above which the nodes in \mathcal{G} representing them are not connected by a direct arc. For example, delay vectors in localities A and B in the Pendulum delay space plot in Figure 8 will not be directly connected by an arc in \mathcal{G} if a reasonable ϵ is set. Rather, the distance between these vectors will be built up as the sum of distances along a path (in the annular region) of locally adjacent (i.e. within ϵ spaced) vectors between A and B.

It is interesting to note that the construction of our restricted region is quite similar to the first 2 steps of the seminal method by Taubenbaum et. al. (J.B.Taubenbaum et al. 2000) for identifying embedded manifolds from data. However, our situation here is slightly different to what they address since (i) for general time series we do not necessarily expect to find a manifold structure, (ii) we are not interested in applying the dimensionality reduction steps (i.e. steps subsequent to Steps 1 and 2) in that method prior to clustering since they impede the selection of cluster centroids, and (iii) there is the notion of temporal adjacency between delay vectors in Z which does not exist in general non-time-series derived data sets, and this is critical in affecting how arcs are input into the graph. Note that, by temporally adjacent vectors, we mean two vectors $z_i, z_j \in Z$ where $i = j + 1$ or $i = j - 1$.

The cost of arcs in \mathcal{G} are stored in a matrix A . The last point (iii) above hints at why we split the construction of A into two parts, i.e. in Step 3 we construct the first upper and lower diagonals (entries representing temporally adjacent vectors), and then in step 4 we do the remaining entries. A time series is usually a discrete sampling of a continuously evolving system, resulting in a discrete trajectory of delay vectors in delay space. Then, we know that temporally adjacent vectors really should be connected (since they are separated by a distance that simply reflects our sampling rate) and so we always do this (Step 3). For the remainder

of delay vector pairs (Step 4) we connect them with an arc only if the distance is below a threshold ϵ .

The second phase of our approach concerns building the metric space $(Z, d_{TF}(\cdot, \cdot))$ in which to cluster. We construct $d_{TF}(\cdot, \cdot)$ using A . This involves building a distance matrix D , where $D_{ij} = d_{TF}(z_i, z_j)$ holds the shortest path distance in \mathcal{G} between nodes i and j . There are a number of ways to construct D from A . One is the well known Floyd Warshall Algorithm. A reduced computation time approach uses Dijkstra’s algorithm with Fibonacci heap data structures (J.B.Tenenbaum et al. 2000). We used the latter for implementations in this paper. We call d_{TF} the TF-metric³. Phase 3 of the approach simply involves clustering in the space (Z, d_{TF}) using the k-medoids algorithm. Denote the above algorithm as the TF-clustering algorithm or the TF-algorithm.

As in (J.B.Tenenbaum et al. 2000) the setting of the threshold ϵ is an integral part of our approach. When we cluster a new time series, we need a way to set ϵ . An appropriate value will ensure we don’t capture in d_{TF} the straight line distance between points on diverse and unrelated parts of the delay vector distribution in delay space. Roughly speaking, we don’t want to fill in “holes” or voids in this distribution when constructing our restricted region. Clearly then the value of ϵ should not be anywhere near the extent of the distribution (this would just give UTS-clustering) but rather some fraction of it. Note that the *MDTD* value provides a measure of the extent of the distribution of delay vectors in delay space. For the purposes of our experiments in this paper, we used ϵ set at 10 percent of the *MDTD* value. Intuitively, this represented a reasonable value given our motivation above of not filling in “holes”, and it also translated into the good membership and centroid results we present later. Investigating the result of setting ϵ differently is outside the scope of the work at this time.

Prior to presenting experimental results for the TF-algorithm, we introduce a variation of it which can be useful for very large datasets. The idea is to approximate the restricted region by the union of smaller convex regions, with a centroid determined for each. This can be done using standard UTS-clustering with a large⁴ number of clusters (“mini-clusters”). We then take the centroids from this clustering as input into what is basically the TF-algorithm above. Denote this as the TF-Minicluster algorithm, where the details are shown in Algorithm 2. Note that this algorithm can be computationally advantageous compared to the standard TF-algorithm, since Step 5 of Algorithm 1 is now required on a much smaller set of points. However, its disadvantage is that we must choose the number of clusters p , with the risk of not approximating the restricted region well.

Algorithm 2 TF-Minicluster Algorithm

- 1: Perform Steps 1 to 4 in Algorithm 1 on the raw point set Z to give A as before
 - 2: Cluster Z using UTS-clustering with a large number (p) of clusters to give a centroid point set \bar{Z} and the raw point set partitioned into mini-clusters
 - 3: Follows steps 1 to 2 in Algorithm 1 on the centroid point set \bar{Z} to give an initial adjacency matrix \bar{A}^*
 - 4: Initialise \bar{A} to zero. Construct \bar{A} by checking, for each non zero element A_{ij} , whether $z_i, z_j \in Z$ are in different mini-clusters. If so, set \bar{A}_{ij} to \bar{A}_{ij}^*
 - 5: Perform Steps 5 and 6 in Algorithm 1 on \bar{A}
-

³Note that the TF-metric really is a metric, since (a) $d_e(z_i, z_j) \geq 0$, (b) $d_e(z_i, z_j) = 0$ only when $z_i = z_j$, and (c) the triangle inequality holds since $d_e(z_i, z_j)$ is the shortest path, so that $d_e(z_i, z_k) + d_e(z_k, z_j)$ must necessarily represent a longer (or at most equal length) path. Further, existence of such a path is guaranteed, since z_i, z_j are guaranteed to be connected by an intermediate path of temporally adjacent points.

⁴If too few clusters are used then the union of small regions will not approximate the restricted region well

5 Results

We saw with the *CP-PP* ratio in Figure 7 that UTS clustering results in centroids lying away from data points. We would hope that this is rectified by our approach, and clearly it will be the case. Since we cluster using k-medoids on (Z, d_{TF}) , we know that each centroid must coincide with a data point and so the *CP-PP* ratio must be zero. This is a desirable outcome, but let us clarify. In the TF-algorithm, we have (i) proposed to restrict the region to that visited by the time series in order to produce useful clustering and (ii) proposed an approach that achieves (i) by using k-medoids on (Z, d_{TF}) . Let us confirm that the desirable outcome has been achieved due to (i) and not (ii). We can do this by presenting the *CP-PP* ratio results for the TF-miniclustering approach, where (ii) is then not applicable.

The *CP-PP* ratio results for the TF-miniclustering approach are shown in Figure 9. Note how the ratios for all the time series are now down around or below unity. This is in contrast to the results shown in Figure 7 for the UTS-clustering algorithm, where very large *CP-PP* ratio values were observed. Only the Henon time series still has a higher value of around 3 for small k values, however this is much smaller than its respective value in Figure 7, and it could be brought down by selecting a greater number p of mini-clusters⁵. Hence, unlike for the UTS (and STS) clustering algorithms, the TF-clustering algorithm produces centroids that sit in among their clusters’ members. This is a requirement if the useful clustering outcome we desire is to be achieved.

The placement of cluster centroids by our method would then seem, at the “macro” level, to be correct. Lets us look now at the TF-clustering outcomes for specific time series in our data set. We first show the result of TF clustering the CBFV time series in Figure 10 (top). What is immediately obvious is that the three distinct shapes (the Cylinder, Bell and Funnel) now get returned as centroids, as desired, rather than the three sine-type waves observed in Figure 4 for UTS-clustering. This is confirmed in by the membership bars shown in Figure 10 (bottom), i.e. each shape now populates a distinct cluster⁶. We show in Figure 10 (middle) plots of the full window of points over which the delay vectors chosen as centroids were constructed. One of the advantages of the TF algorithm compared to the UTS algorithm is that, since it selects members of Z as cluster centroids, we have available for our centroid representation the full window of time series data points over which the delay vector spanned. This circumvents any centroid discretisation issues caused by introducing a lag.

Next, we show in Figure 11 the result of TF-clustering the Koski-ECG time series. Recall from Figure 5 how UTS clustering returns very strange centroids that look nothing like any part of the time series. Figure 11 shows that the TF-algorithm correctly returns the in-beat and between-beat

⁵For this experiment we chose p as the number of points in the time series divided by 15 (i.e. so that on average there were 15 points in each mini-cluster) where 15 reflected more or less the greatest value we could select before insufficient points remained for clustering at the higher k numbers for the smaller data sets.

⁶Note that the range of time series represented by each cluster is the bar shown plus the length of the delay vector, i.e. q times w , which is why the bars don’t seem to be centred properly. This factor must also be taken into account in the membership plots for experiments which follow

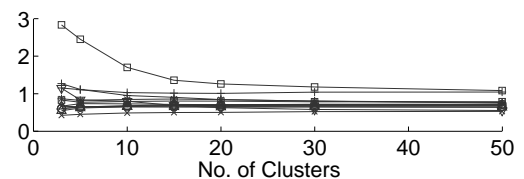


Figure 9: *CP - PP* Ratio: TF-Miniclustering

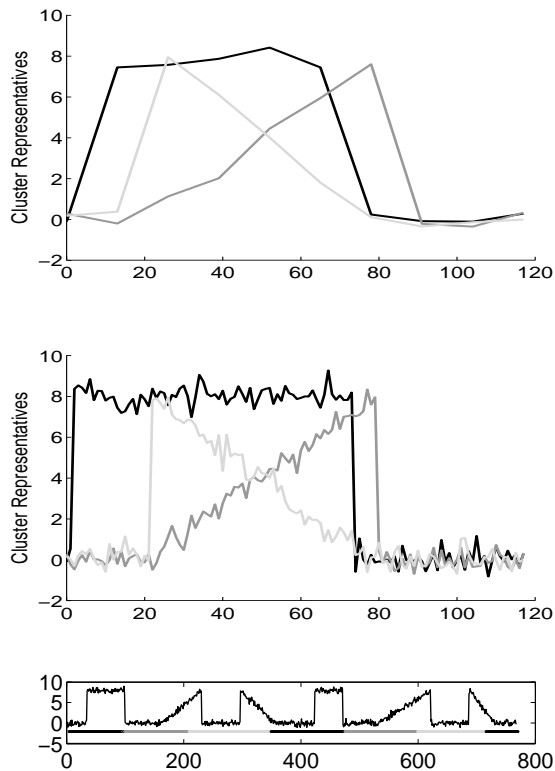


Figure 10: TF Clustering: CBFV results

phases in this time series. This is confirmed by the membership bars shown in the bottom plot ⁶, where homogeneous bars without the high frequency oscillation between membership evident in Figure 5 have been returned.

Finally, Figure 12 shows the result of TF-clustering the Chaotic Time Series. Recall from Figure 6 how the UTS Clustering algorithm seemed to identify three distinct levels in this time series, but how the oscillations in the time series at these levels disappeared. Figure 12 shows that the TF-algorithm includes these oscillations in its centroids. In essence, it has picked out the three best features to summarise this time series as an oscillatory signal at each of these three levels, and this seems an intuitively appropriate “summary” of this time series.

It would seem that our method provides a means for the proper selection of cluster centroids, and indeed produces clustering membership outcomes more in line with what one would intuitively expect. However, recall that a useful clustering must be meaningful. The final step in our experiments is to show that the TF-algorithm produces meaningful clusterings. We repeat Steps 1 to 6 of Section 2 above, where in Step 4, we cluster with the TF-algorithm in place of UTS-clustering. Figure 13 shows the resulting within versus between distance of clusters for all twelve time series over a range of k values. We can see that, for each time series, the between distances are always greater than the within distance, suggesting that the TF algorithm indeed produces meaningful clustering outcomes.

6 Conclusion

The finding was made in (E.Keogh et al. 2003) that clustering time series using the traditional STS approach was meaningless since (A) the clustering outcomes of different time series were not distinguishable from one another, and (B) that cluster centroids were smoothed. While (E.Keogh et al. 2003) proposed that these two problems were one and the same, i.e. that one could not distinguish between cluster centres of different time series because they were all

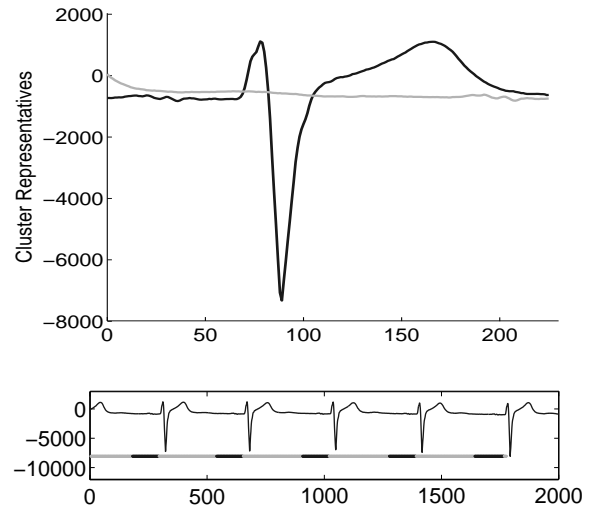


Figure 11: TF Clustering: Koski-ECG results

smoothed and hence alike, we showed in this paper that this is in fact false. Problems (A) and (B) are really two separate problems that can be solved separately. Work in (G.Simon et al. 2006) showed that (A) could be solved by introducing a lag in the delay vector construction process. In this paper we showed, however, that such an approach does not solve (B). Introducing the alternative terminology that a clustering method overcoming problem (A) is meaningful, while one overcoming (A) and (B) is useful, we proposed an approach that produces useful time series clustering. The two key elements of the approach were, (I) unfolding the delay vector distribution by using a lag, and then, (II) clustering only in the “subset” of delay space visited by the time series. We proposed that one should cluster in the subset of delay space on which the underlying physical system that produced the data lives, although, we noted that one does not typically know the extent of this region a-priori. Two choices then seem possible: to cluster in all of delay space as has been the case to date, or to cluster in the subset of delay space visited by the time series. Both lead to meaningful clustering, however we proposed that the question we are typically asking when looking to cluster a time series corresponds to the latter. In essence, we want a summary of the time series as the k features existing in the time series which best summarise it. Finding the low dimensional subset of higher

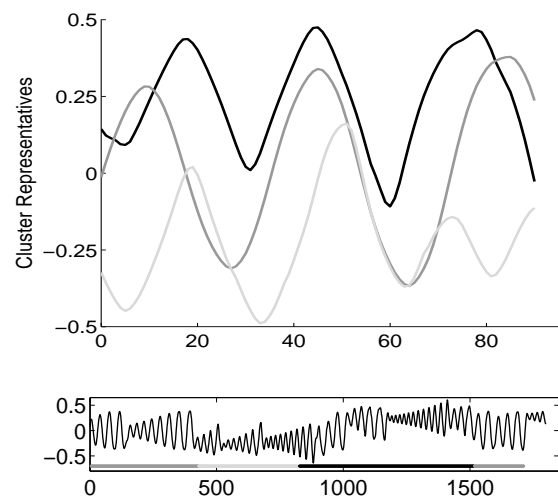


Figure 12: TF Clustering: Chaotic results

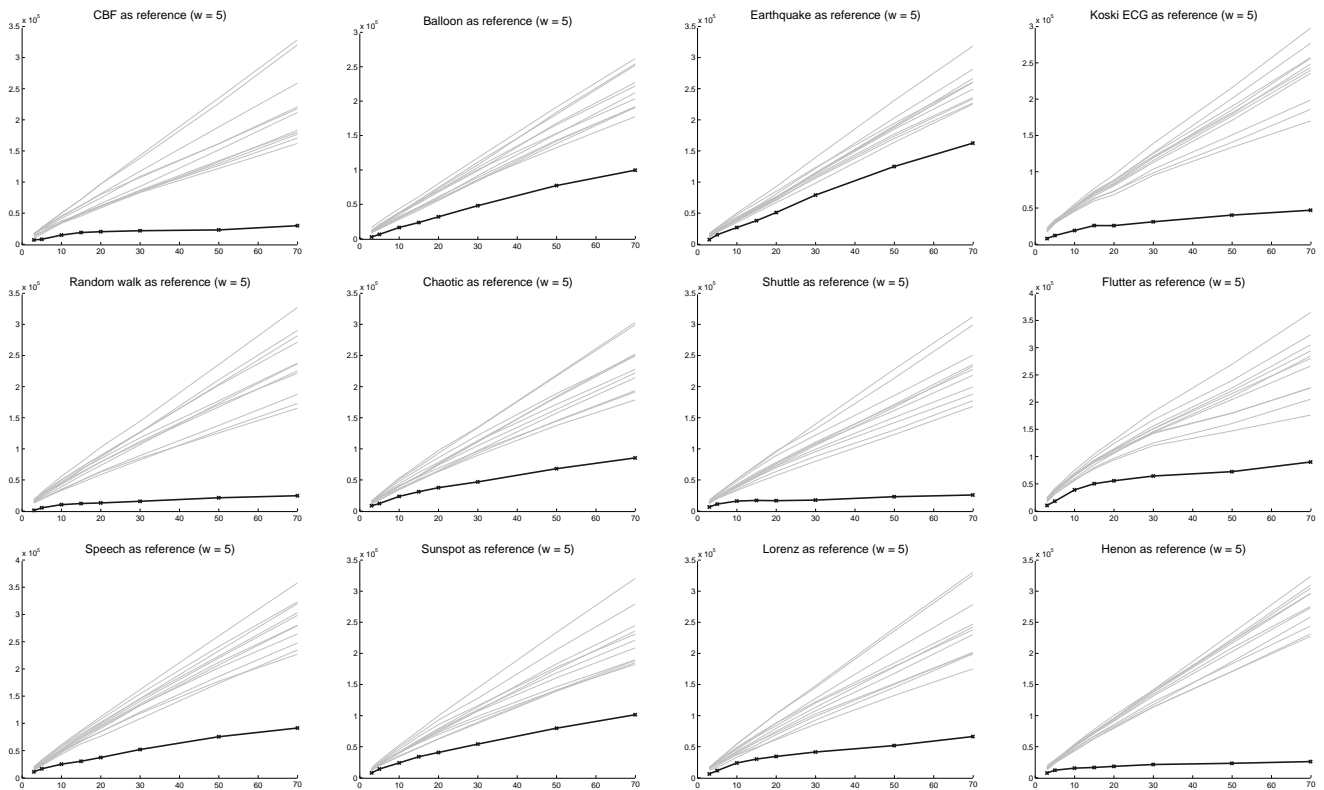


Figure 13: Results of the within-between distance experiments using TF Clustering

dimensional spaces on which a data set lives is not new in the literature, although the fact that time series data sets have temporal ordering, and that we are looking to cluster the data set once the relevant subset is found, adds novelty here. We noted the close parallels of seminal work in the dimensionality-reduction-of-data-sets literature and the approach proposed in this paper. Experiments to validate our approach were conducted on 12 real life and synthetic data sets. These experiments indicated that our approach could overcome both problems (A) and (B); it produced meaningful clusterings, and it produced cluster representatives that well represented (were located in among) the data points in their respective clusters. Quite a number of solutions have been suggested to the STS-clustering dilemma since it was first identified in (E.Keogh et al. 2003). However, to our knowledge, this is the first method that directly addresses both the problems (A) and (B) identified there. As such, we propose the TF-algorithm as a means for obtaining useful clustering outcomes when the clustering of time series is required.

References

- A.Denton (2005), Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model, *in* 'Proceedings of IEEE International Conference on Data Mining', Houston, USA.
- B.S.Everitt, S.Landau & M.Leese (2001), *Clustering Analysis*, Wiley.
- D.Goldin, R.Mardales & G.Nagy (2006), In search of meaning for time series subsequence clustering: matching algorithms based on a new distance measure, *in* 'Proceedings of Conference of Information and Knowledge Management', Arlington, USA.
- E.Keogh (2002), 'The ucr time series data mining archive', <http://www.cs.ucr.edu/~eamonn/TSDMA/index.html>.
- E.Keogh, J.Lin & W.Truppel (2003), Clustering of time series subsequences is meaningless: Implications for previous and future research, *in* 'Proceedings of the International Conference of Data Mining'.
- G.Simon, J.A.Lee & M.Verleysen (2006), 'Unfolding preprocessing for meaningful time series clustering', *Neural Networks* **19**, 877–888.
- J.B.Tenenbaum, Silva, V. & J.C.Langford (2000), 'A global geometric framework for nonlinear dimensionality reduction', *Science* **290**, 2319–2322.
- J.R.Chen (2007), 'Making clustering in delay vector space meaningful', *Knowledge and Information Systems, an International Journal* **11**(3), 369–385.
- K.Peker (2005), Subsequence time series (sts) clustering techniques for meaningful pattern discovery, *in* 'International Conference Integration of Knowledge Intensive Multi-Agent Systems (KIMAS)', Waltham, USA.
- M.Breitenbach & G.Z.Grundic (2005), Clustering through ranking on manifolds, *in* 'Proceedings of the 22nd International Conference on Machine Learning', Bonn, Germany.
- R.M.Haralick & R.Harpaz (2005), Linear manifold clustering, *in* 'Proceedings of the International Conference on Machine Learning and Data Mining', Leipzig.
- Z.R.Struzik (2003), *Foundations of Intelligent Systems*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, chapter Time Series Rule Discovery: Tough, Not Meaningless, pp. 32–39.

