In Search of Meaning for Time Series Subsequence Clustering: Matching Algorithms Based on a New Distance Measure

Dina Goldin
Brown University
Providence, RI, USA
dqg@cs.brown.edu

Ricardo Mardales
University of Connecticut
Storrs, CT, USA
mardales@engr.uconn.edu

George Nagy
Rensselaer Polytechnic Inst.
Troy, NY, USA
nagy@ecse.rpi.edu

ABSTRACT

Recent papers have claimed that the result of K-means clustering for time series subsequences (STS clustering) is *independent* of the time series that created it. Our paper revisits this claim. In particular, we consider the following question: *Given several time series sequences and a set of STS cluster centroids from one of them (generated by the K-means algorithm), is it possible to reliably determine which of the sequences produced these cluster centroids?* While recent results suggest that the answer should be *NO*, we answer this question in the *affirmative*.

We present *cluster shape distance*, an alternate distance measure for time series subsequence clusters, based on *cluster shapes*. Given a set of clusters, its *shape* is the sorted list of the pairwise Euclidean distances between their centroids. We then present two algorithms based on this distance measure, which match a set of STS cluster centroids with the time series that produced it. While the first algorithm creates smaller "fingerprints" for the sequences, the second is more accurate. In our experiments with a dataset of 10 sequences, it produced a correct match 100% of the time.

Furthermore, we offer an analysis that explains why our cluster shape distance provides a reliable way to match STS clusters to the original sequences, whereas cluster set distance fails to do so. Our work establishes for the first time a strong relation between the result of K-means STS clustering and the time series sequence that created it, despite earlier predictions that this is not possible.

1. INTRODUCTION

It has been claimed in recent literature [18] (preliminary version [19]) that the result of K-means clustering for time series subsequences (STS clustering) is *independent* of the time series that created it:

> "As we prove in this paper, the output of STS clustering does not depend on input, and is therefore meaningless". [18]

Specifically, it was shown that when measuring the distance between sets of cluster centroids (*cluster set distance*), this distance is on average no smaller for cluster sets from the same time series, obtained by using different random seeds in the clustering algorithm, than for cluster sets from different time series.

The claim of *meaninglessness* for STS subsequence clustering "has cast a shadow over the emerging discipline of time series data mining" [28]. This work has also generated a flurry of follow-up research activity, including the current paper.

Our paper focuses on the issue of *dependence* between the output of STS clustering (i.e., the set of clusters) and its input (i.e. a time series sequence). Specifically, we revisit the following question, which was believed to be solved in the negative by [18]:

> *Given several time series sequences and a set of STS cluster centroids from one of them (generated by the K-means algorithm), is it possible to reliably determine which of the sequences produced these cluster centroids?*

To accomplish this task, we first define an alternate distance measure for time series subsequence clusters, *cluster shape distance*. Given a cluster set, its *shape* consists of the pairwise Euclidean distances between the cluster centroids in the set; cluster shape distance is the Euclidean distance between the shapes of two cluster sets. Our distance measure can be contrasted with the older measure of *cluster set distance*. Unlike the latter, our measure is invariant under translations and rotations of the cluster sets.

When using the new distance measure, we find that the input and output of subsequence clustering are no longer independent. In fact, we provide two algorithms for matching an STS cluster set with the original time series that generated it. These algorithms take as input a *dataset* consisting of multiple time series sequences, and a set of *queries*, where each query is a set of STS cluster centroids that are to be matched to one of the sequences in the dataset.

Given a dataset $D = \{S_1, S_2, ...S_n\}$ and a query Q (i.e. the centroids of an STS cluster set generated by some dataset from D), our algorithms return some index j into D; this means that S_j is believed to be the time series that generated Q (i.e. its match). If this is correct, then we say that

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.

our algorithm has produced a correct match.

Both matching algorithms contain two phases, where phase I *preprocesses* the time series sequences in our dataset, and phase II uses the results of phase I to *match* a query with some sequence in the dataset. The separation of matching algorithms into two phases allows them to be easily adapted to a *database setting*, where the dataset of multiple time series sequences is given ahead of time, and the queries are supplied in an ad-hoc fashion one at a time. This setting allows preprocessing to be done off-line, and ensures a very quick answer for each query.

In phase I, both algorithms compute a list of cluster shapes for each sequence, by performing multiple clustering runs with different initial seeds. The first algorithm outputs the *average* of these cluster shapes, which we call a *cluster constellation*, whereas the second algorithm outputs *the whole list*. While the first algorithm creates smaller "fingerprints" for the sequences, the second is more accurate. In our experiments with a dataset of 10 sequences, it produced a correct match 100% of the time!

These results establish a strong dependence between the result of K-means STS clustering and the time series sequence that created it. As far as we know, ours is the only work that achieves this while using the *same* K-means STS clustering algorithm as in [18], thereby disproving their claim. Furthermore, we are able to distinguish between many sequences at a time, whereas other work, such as [18, 8, 27], only distinguished between two sequences at a time.

In addition, we offer an analysis that explains why our cluster shape distance provides a reliable way to match STS clusters to the original sequences, whereas cluster set distance fails to do so.

Outline. We start by providing some background in Section 2. We describe our algorithm for STS clustering, which is the same as that in [18]. In Section 3 we define the two notions of distance between cluster sets, *cluster set distance* and *cluster shape distance*. The latter is a new approach to measuring distances between sets of cluster centroids. In Section 4, we provide an algorithm for matching an STS cluster set with the original time series, which is based on cluster shape distance. In Section 5, we present accuracy tests for our matching algorithm, and discuss their results. In Section 6, we consider the notion of *sequence smoothness*, and observe its correlation with certain aspects of the sequence's cluster shapes, such as their number. In Section 7, we explain why the shape of the cluster centroids depends on the sequence, whereas the actual *position* of the cluster centroids does not; we also explain the correlation between sequence smoothness and the number of shapes. We conclude with Section 8.

2. BACKGROUND

In this section, we provide the background information about the K-means subsequence clustering algorithm.

2.1 K-means STS Clustering

Our algorithm for subsequence clustering is the same as that in [18]; we describe and discuss it in this section.

The input to the STS clustering algorithm consists of a time series sequence of length m, a window size w, and the number of clusters K. We slice the original time series into $m-w+1$ subsequences, each of length w and we put these into a subsequence matrix $M[m-w+1][w]$. We then normalize each subsequence, using the standard technique that was first introduced to time-series similarity querying in [13]. Normalization is performed by subtracting the subsequence average from each value in the subsequence, and dividing it by the standard deviation. As a result, each subsequence has an average of 0 and a standard deviation of 1. We then cluster the normalized subsequences from M using the K-means clustering algorithm introduced in [20], as follows:

STS Clustering Algorithm.

1. Randomly choose K subsequences indices in the range $[1, m-w+1]$, to be used as seeds into the algorithm. The subsequences corresponding to these indices will serve as the initial set of K cluster centroids.

2. For each subsequence in M, calculate its Euclidean Distance to the K cluster centroids and assign it a value in the range $[1, K]$, corresponding to the closest cluster center.

3. After each subsequence has been assigned to a cluster, recalculate the center of each cluster, taking the *mean* (average) of all the subsequences in that cluster.

4. Repeat steps 2-3 until the cluster centroids remain unchanged. The output is a matrix $C[K][w]$, which contains the final set of K cluster centroids.

We refer to one run of this algorithm as a *clustering run*. Given a data sequence S, the number of clusters K, and the window length w, a clustering run returns some set of cluster centroids for S.

2.2 K-means clustering

We now give a perspective on K-means clustering, which is a subroutine of K-means STS clustering. It is a technique for discovering groups of similar patterns in a large collection of unlabeled vectors. It was used by one of the authors as early as 1964 for finding similar Han (Chinese) characters for the first level of a two-level classifier [4], and later for sorting unlabeled printed glyphs into alphabetic classes in a cryptographic approach to optical character recognition [5, 6, 24], for feature extraction [23], and for unsupervised crop classification by remote sensing [25].

The algorithm was popularized as a general method for "exploratory" multivariate analysis of unlabeled data [2, 20]. A variation that addressed some of the shortcomings of the elementary algorithm by splitting and merging classes was called *Isodata* [3]. In the communications community, iterative minimization of the sum-of-squared-error criterion became known as *Vector Quantization* [12, 29]. Examples of easy- and difficult-to-cluster pattern configurations were presented in [22] and in [15].

Among the first attempts at evaluating its effectiveness was [9]. Variations of the method with respect to initialization, cost function, splitting and merging clusters, and distance metrics, are described in [16, 29, 10]. Current research focuses on combining multiple cluster configurations obtained by different algorithms, i.e., *clustering ensembles* [30].

It is easy to see that both the sample re-assignment step and the recomputation of the cluster centroids decrease the sum of the distances of the samples to their cluster centroids. Therefore starting with any set of seed points (initial cluster centroids), K-means always converges to a (local) minimum.

Different seeds may lead to different final cluster configurations with different sums of squared error. Because the sum of the within-cluster and between-cluster scatter is a constant, the algorithm simultaneously maximizes the latter.

The final number of clusters is not necessarily equal to the number of initial seeds: some clusters may become unpopulated, but neither step forms new clusters. Finally, and for our analysis most importantly, multiple global minima, each reachable by a different initialization, may exist.

In STS analysis, the K-means algorithm is used to find the averages of overlapping subsequences. The K cluster centroids are representative of the different subsequences, and may therefore be considered as a condensed representation of the entire sequence (much like the eigenvectors in Karhunen-Loeve Transform or Principal Component Analysis). Different sequences typically yield different centroids. It appears, however, that with different initializations, even the same sequence may lead to different centroids.

3. MEASURING CLUSTER DISTANCE

Our ability to correctly match K-means STS clusters to the sequences that produced them hinges on the issue of cluster distance. While it has been shown impossible to accomplish this matching by measuring the Euclidean distances between sets of cluster centroids, we avoid this problem by defining an alternate distance measure for sets of centroids. This section discusses both distance measures.

3.1 Cluster Set Distance

In [18], it was shown that the *cluster set distance* between two sets of cluster centroids from the same time series, obtained by using different random seeds in the clustering algorithm, is on average no smaller than that between two sets of cluster centroids from different time series.

DEFINITION 1. **(Cluster Set Distance)** *Given two sets of K cluster centroids , $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K)$ and Y, their* cluster set distance *is the sum of minimal Euclidean distances:*

$$\textit{cluster-set-dist}(X, Y) = \textit{dist}(\mathbf{x}_1, \textit{closest}(\mathbf{x}_1, Y)) + \\ + \textit{dist}(\mathbf{x}_2, \textit{closest}(\mathbf{x}_2, Y)) + \ldots + \textit{dist}(\mathbf{x}_K, \textit{closest}(\mathbf{x}_K, Y)),$$

where dist is the Euclidean distance, and closest$(\mathbf{x}_i, Y)$ denotes the cluster center in Y with the smallest Euclidean distance to $\mathbf{x}_i$ (nearest neighbor).

Figure 1 illustrates the definition. In this case, the cluster set distance between X and Y is the sum of $AD+BD+CD$; the cluster distance between Y and X is $DB + EC + FB$. Note that this distance measure is not commutative.

We have implemented subsequence clustering and cluster set distance, and confirmed the results observed in [18], using both Matlab and Java. We agree that the cluster set distance between the two sets of clusters for the same time series, obtained by using different random seeds in the clustering algorithm, is no smaller than for two sets of clusters from different time series. In addition, we have confirmed another observation in [18], that the plots of the cluster centroids resemble sine waves, and that the mean of the cluster centroids, when weighed by cluster size, is a line or very close to it.

While it is clear that the *cluster set distance* measure does not produce meaningful results, we will present an alternate measure for comparing time series subsequence clusters.

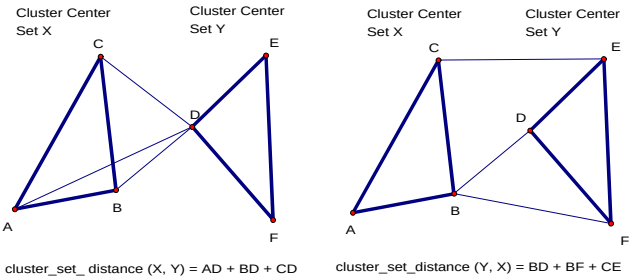

Figure 1: Examples of cluster set distance.

run	w	seeds($K=3$)			$D_{1,2}$	$D_{1,3}$	$D_{2,3}$	Avg
1	8	226	549	82	2.7771	4.4644	2.4942	3.2452
2	8	902	7	171	4.4855	2.7416	2.9164	3.3812
3	8	525	751	820	4.4928	2.5958	2.8388	3.3091
4	16	226	549	82	5.1477	6.5741	3.1317	4.9512
5	16	902	7	171	6.6168	3.6607	4.8998	5.0591
6	16	525	751	820	6.5801	3.2478	5.0325	4.9535
7	32	226	549	82	8.3883	9.2677	3.7964	7.1508
8	32	902	7	171	6.9156	9.4574	5.9889	7.4539
9	32	525	751	820	9.2988	4.9502	7.4518	7.2336

Figure 2: different clustering runs for *ocean* series

This new measure, called *cluster shape distance* will allow us to create algorithms that definitively establish a dependence between the input of STS clustering and its output. We discuss it next.

3.2 Cluster Shapes

In this section, we introduce the notion of *cluster shapes*, that will later be used for an alternate distance measure between sets of cluster centroids. They are defined as follows:

DEFINITION 2. **(Cluster Shape)** *Given a set S of K clusters with centroids $C_1, \ldots, C_k$, the shape of S is the sorted sequence of all the pairwise Euclidean distances between the cluster centroids in S; it contains $K * (K-1)/2$ numbers.*

To illustrate this definition, imagine that after clustering a set of time-series subsequences using the algorithm in Section 2.1, we have obtained K cluster centroids of length w. We then calculate the Euclidean distances for all pairs of these centroids; there are $K * (K-1)/2$ such pairs. For example, when $K = 3$, we obtain 3 distances $D_{1,2}, D_{1,3}, D_{2,3}$, where $D_{i,j}$ represents the Euclidean distance between cluster centroids i and j.

Figure 2 illustrates the results for 9 different clustering runs for the *ocean* series from [17].[1] It shows the distances between cluster centroids , as well as their sum and average. These results produced by clustering runs with $K = 3$ and $w = 8, 16, 32$; three different runs were performed for each combination of K and w.

It is easy to see that in Figure 2, when we consider different runs with the same K and w, the sum and average are very close regardless of the initial choice of cluster centroids. While their distances (columns 4-6) are not necessarily close, the insight here is that the order of the distances is

[1] All data sequences used in this paper were of length 1000.

run	δ_1	δ_2	δ_3
1	2.4942	2.7771	4.4644
2	2.7416	2.9164	4.4855
3	2.5958	2.8388	4.4928
4	3.1317	5.1477	6.5741
5	3.6607	4.8998	6.6168
6	3.2478	5.0325	6.5801
7	3.7964	8.3883	9.2677
8	5.9889	6.9156	9.4574
9	4.9502	7.4518	9.2988

Figure 3: different shapes for the *ocean* series

arbitrary. By sorting the set of distances $D_{i,j}$ for each run, the resulting lists of numbers $\delta_1, \delta_2, \delta_3$ are also similar for different initial seeds, as can be seen in Figure 3. This list of numbers is the *shape* of the cluster set.

3.3 Cluster Shape Distance

In this section, we introduce a new approach to measuring distances between sets of cluster centroids, called *cluster shape distance*. With this new approach, given n series from different sources and m sets of cluster centroids, these sets can be mapped back to their original series with high accuracy.

Cluster shape distance is just the distance between the shapes of two sets of cluster centroids:

DEFINITION 3. **(Cluster Shape Distance)** *Given two sets of clusters, X and Y, with cluster shapes $\{\delta_1, \delta_2, \delta_3, \ldots\}$ and $\{\delta'_1, \delta'_2, \delta'_3, \ldots\}$ respectively, their cluster shape distance is the Euclidean distance between their shapes:*

$$\textit{cluster-shape-dist}(X, Y) = \\ = \textit{dist}(\{\delta_1, \delta_2, \delta_3, \ldots\}, \{\delta'_1, \delta'_2, \delta'_3, \ldots\})$$

Note that the storage requirements are much smaller for cluster shape distance than for cluster set distance. In order to compute the cluster set distance between a given set of cluster centroids S and another one that we will be receiving in the future (our *query*), one needs to store S in its entirety; for K centroids of length w, this means $K*w$ values. On the other hand, to compute the cluster shape distance between S and a future set, one only needs to store the shape of S, which means $K*(K-1)/2$ values. Since K is typically much smaller than $2w$, the amount of storage is greatly reduced.

Cluster constellations are simply the result of averaging together many cluster shapes for the same series:

DEFINITION 4. **(Cluster Constellation)** *A cluster constellation is the average of cluster shapes resulting from multiple clustering runs over the same series with the same w and K. Cluster constellations are denoted by $(\Delta_1, \Delta_2, \Delta_3, \ldots)$.*

EXAMPLE 1. *Let A, B, C, and D be sets of cluster centroids produced by four clustering runs over the same time series, with $K = 3$. Let a_1, a_2, a_3 be the shape of A, where a_1 is the shortest length between any two cluster centroids in A, a_2 is the second shortest, and a_3 is the longest. Similarly, we compute b_1, b_2, b_3, c_1, c_2, c_3, and d_1, d_2, d_3 as the shapes of B, C, and D respectively. Then, $(\Delta_1, \Delta_2, \Delta_3)$ is the cluster constellation for (A, B, C, D), where $\Delta_i = avg(a_i, b_i, c_i, d_i)$.*

As can be seen in Figure 3, for a given series with any given combination of K and w, these shapes for various clustering runs tend to be very similar to each other. While all these shapes have $K = 3$, this is not necessary. These shapes remained similar when we varied K and w, or when we tried other series.

	Δ_1	Δ_2	Δ_3
ocean	2.3598	3.0464	4.4583
packet	2.1712	2.2315	2.3619
soil	1.9434	2.0073	2.0582
sp	2.5302	2.9574	3.7740
tide	2.7175	3.3878	3.7050

Figure 4: cluster constellations for $K = 3, w = 8$

data	Δ_1	Δ_2	Δ_3
ocean	3.3018	5.1779	6.5902
packet	2.3881	2.4878	2.6392
soil	2.0046	2.3572	2.5495
sp	3.6240	4.1210	5.5512
tide	3.7356	4.2792	4.6382

Figure 5: cluster constellations for $K = 3, w = 16$

On the other hand, shapes for clustering runs from different time series, tend to be very different. This is illustrated in Figures 4, 5, and 6. Figure 4 shows cluster constellations for five sequences from [17], with $K = 3$ and $w = 8$. Figure 5 is for the same sequences, with $K = 3$ and $w = 16$. Figure 6 is with $K = 4$ and $w = 8$. Each of these cluster constellations were computed by performing 100 clustering runs and averaging the resulting shapes together.

On the basis of these empirical observations, one can conjecture that cluster shape distance is a more appropriate distance for STS clusters than cluster set distance. Our conjecture will be confirmed in the next section by showing that cluster shape distance can serve as a basis for an algorithm that matches STS cluster sets back to their original time series.

4. CLUSTER MATCHING ALGORITHM

We now provide an algorithm for matching an STS cluster set with the original time series, which uses the distances between cluster shapes (and constellations), as opposed to cluster set distances. There are two variations of this algorithm, which we treat as separate algorithms.

The input to both algorithms consists of:

- a *dataset* consisting of N time series sequences
- one or more *queries* consisting of STS cluster set centroids that are to be matched to one of the sequences in the dataset
- fixed values for the number of clusters K and the win-

data	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6
ocean	2.4787	2.5844	2.9337	2.9778	3.2012	4.7337
packet	2.2235	2.2518	2.3438	2.5314	2.5647	2.6129
soil	2.0568	2.0874	2.1464	2.1803	2.2105	2.2533
sp	2.1239	2.4650	2.9448	3.1700	3.4565	4.1346
tide	2.4059	2.4585	3.3746	3.4247	4.0473	4.1776

Figure 6: cluster constellations for $K = 4, w = 8$

dow size w, that were used to create the queries

Both algorithms consist of two phases, where phase I *preprocesses* all the time series sequences in the input dataset, and phase II uses the results of phase I to match each query with some sequence in the dataset.

STS Cluster Matching Algorithm I

I-a For each sequence in the dataset, perform Q different clustering runs with given K and w, compute the shape of each resulting cluster set, and average all the shapes to obtain the cluster constellation for that sequence.

I-b Store the resulting N constellations in a *master table* M.

II-a For each query, compute its shape, and find the Euclidean distance to each of the constellations in M.

II-b The sequence with the smallest distance is the answer to the query.

In phase I, both algorithms compute a number of cluster shapes for each sequence, by performing multiple clustering runs, and create a master table M. However, the first algorithm stores in M, for each input sequence, the *average* of these cluster shapes (a cluster constellation), whereas the second algorithm stores *all* these shapes in an *individual table*.

STS Cluster Matching Algorithm II

I-a For each sequence in the dataset, perform Q different clustering runs with given K and w and compute the shape of each resulting cluster set; store all the shapes into an *individual table* consisting of the resulting Q constellations.

I-b Store the resulting N individual tables in a *master table* M.

II-a For each query, compute its shape, and find the Euclidean distance to each of the shapes in each of the individual tables in M.

II-b The sequence with the smallest distance is the answer to the query.

Figure 7 shows a set of 10 cluster shapes $(\delta_1, \delta_2, \delta_3)$. Each of these shapes came from a different clustering run, two runs for each of the five series listed in Figure 4. The information about the sequence that produced each shape was then hidden, and the cluster matching algorithm was used to match these shapes (queries) with the original sequences (dataset). In this case, the assignments are all correct, and appear in the rightmost column of Figure 7.

Note that both matching algorithms can be easily adapted to a *database setting*, where the dataset of multiple time series sequences is given ahead of time, and the queries consisting of STS cluster set centroids are supplied in an ad-hoc fashion one at a time. This setting allows preprocessing to be done off-line, and ensures a very quick answer for each query.

5. EXPERIMENTAL EVALUATION

We now present our accuracy tests for the two matching algorithms defined above. These tests used a dataset of ten

δ_1	δ_2	δ_3	Assignment
2.6517	2.9498	3.7824	sp
2.5873	3.5066	3.6869	tide
2.5958	2.8388	4.4928	ocean
2.1594	2.2460	2.3478	packet
1.9323	2.0474	2.0711	soil
2.1960	2.2640	2.3352	packet
2.4942	2.7771	4.4644	ocean
2.5529	2.8036	3.7939	sp
1.9481	2.0417	2.0672	soil
2.8821	3.1982	3.7473	tide

Figure 7: sample shapes and their assignment for $K = 3, w = 8$

sequences from [17], listed in Figure 8. While such a dataset seems small, it must be remembered that earlier work has predicted that reliable matching would be impossible even in the case of only two sequences! Ten sequences is therefore more than sufficient for our purposes.

Accuracy test for STS cluster matching

1. We varied K and w, so K took values between 3 and 5, and w took one of the values in 8, 16, 32, 64. For each of the 12 combinations of K and w, we ran the algorithm 100 times. Therefore, the algorithm was run a total of 12 * 100 = 1,200 times.
2. There were 10 queries for each invocation of the algorithm. These were computed afresh each time from the data sequences, by running the clustering algorithm once on each of the 10 sequences in the dataset.
3. Q, the number of clustering runs per sequence (see step **I-a** of the matching algorithm), was fixed at 100.
4. Each invocation of the algorithm produced 10 potential matches, one for each sequence (i.e. for the query that came from that sequence). As a result, for each combination of K and w, we obtained 1000 potential matches, 100 for each sequence.
5. We then computed the percentage of correct matches for each sequence; we will refer to it as the *accuracy score*. Ideally, it should be close to 100%.

The results of our test for Algorithm I appear in Figure 8. Overall there is an accuracy average of over 90%. This means that, given a set of cluster centroids, Algorithm I will correctly identify which series produced it in over 90% of the cases.

The outcome for Algorithm II is more dramatic. *All the scores are 100%*! These results show that it is possible to tell which STS cluster set comes from which input sequence, by using the cluster shape distance as a similarity measure. If we are able to identify the original series that produced the cluster centroids, this in turn means that the output STS clustering does depend on input, despite earlier claims to the contrary.

6. THE EFFECT OF SMOOTHNESS

In this section, we consider the notion of *smoothness* for data sequences, and its effect on the number of unique shapes for the sequences, as well as on those shapes themselves. We observe that there is a strong correlation between them.

K	3				4				5			
w	8	16	32	64	8	16	32	64	8	16	32	64
burstin	89	100	100	100	89	100	100	100	97	100	100	100
robot	100	100	100	100	99	100	100	100	96	100	100	100
soil	80	100	94	100	85	100	99	95	95	100	100	97
tide	98	100	100	100	99	100	98	100	100	100	97	100
infra	100	100	100	100	85	100	100	100	100	100	100	100
packet	100	100	87	100	100	100	100	100	100	100	100	100
sp	90	100	100	100	75	100	100	100	98	90	100	100
sensorA	100	100	100	100	100	100	100	100	98	100	100	96
ocean	100	100	93	100	95	100	100	100	93	100	85	95
leleccum	100	100	100	83	97	92	91	99	98	93	100	83

Figure 8: accuracy scores

	sequence	b	
1	burstin	-1.1187	
2	robot	-0.5442	
3	soil	-0.4324	less smooth
4	tide	-0.1331	
5	packet	0.9389	
6	infra	0.9756	
7	sp	2.0663	
8	sensorA	2.1822	more smooth
9	ocean	2.3444	
10	leleccum	2.9033	

Figure 9: The smoothness of the data sequences

	δ_1	δ_2	δ_3	skew
1	3.1317	5.1477	6.5741	2.8626
2	3.1601	5.1581	6.5680	2.8977
3	3.1851	5.3072	6.5946	2.9209
4	3.1995	5.4227	6.5815	2.9487
5	3.2478	5.0325	6.5801	2.9705
6	3.3854	5.2233	6.5960	3.1462
7	3.4312	4.9095	6.5661	3.1684
8	3.6607	4.8998	6.6168	3.4080
9	3.9748	4.4091	6.6150	3.5316

Figure 10: unique shapes for ocean ($K = 3, w = 16$)

6.1 Sequence Smoothness

First, we define the notion of sequence *smoothness*. If we look at the *DFT* of a sequence, the energy for a large class of signals (*colored noises*) is concentrated in the first few coefficients. These signals have a skewed energy (or power) spectrum, that drops as $O(f^{-b})$, where f is the frequency. [11]

DEFINITION 5. **(Smoothness Coefficient)** *Given a time series sequence σ, its smoothness coefficient b is measured as follows: $|DFT_m(\sigma)| = O(f^{-b})$, where $DFT_m(\sigma)$ represents the first m coefficients of the Discrete Fourier Transform of σ for some small value of m. [11]*

From the definition, it follows that the smaller the exponent of f, the fewer coefficients of the DFT are needed to store most of the amplitude of the time series. The actual value strongly depends on the nature of the data sequences. For example, for $b = 2$, we have *brown noise*, or a random walk, which models stock movements and exchange rates; its energy spectrum follows $O(f^{-2})$. Informally, the larger the b, the smoother the time series.

For our set of 10 sequences, the values of b are shown in Figure 9. These were computed with the value of 8 for m; since the DFT coefficients are complex, this represents 16 real numbers.

In the next section, we show a correlation between the b value of a sequences and some properties of its shape. For that purpose, we classify the sequences in Figure 9 into two groups: *less smooth* (sequences 1-6) and *more smooth* (sequences 7-10).

6.2 Differences between the two groups

When examining the individual tables in Algorithm II, we often find that after many clustering runs on any one series, only a few *unique shapes* are returned; that is, all other shapes are copies of one of these (up to some precision factor).

For instance, after 100 clustering runs for the *ocean* series, with $K = 3$ and $w = 16$, only 9 unique shapes (up to precision of 4 decimal digits) were found; Figure 10 shows these shapes. Note that all three shapes for the ocean series from Figure 3 with $w = 16$ have exact matches in Figure 10, namely shapes 1, 8, and 5.

However, other series produced more unique shapes, such as 82 for the *burstin* sequence ($K = 3, w = 16$). Are there any differences between these sequences that can account for this disparity? The explanation seems to be in their smoothness coefficient b.

In general, we have observed a strong correlation between sequence smoothness and the number of unique shapes for it. Figure 11 shows the maximum number of unique shapes for the sequences in the two groups from Figure 9, for various combinations of K and w. It is clear that the more smooth sequences have fewer unique shapes than the less smooth ones.

While there are more shapes for the less smooth sequences, they tend to be "closer" to each other. That is, when we use precision 3 rather than 4, the number of less smooth shapes decreases significantly whereas the number of the more smooth shapes does not (Figure 12).

Furthermore, there is a strong correlation between sequence smoothness and the shapes themselves. Specifically, we consider the *shape skew*, defined as follows:

DEFINITION 6. **(Shape Skew)** *Given a shape* $(\delta_1, \delta_2, \ldots)$,

K	3				4			
w	8	16	32	64	8	16	32	64
less smooth	70	82	79	77	81	87	89	84
more smooth	14	9	16	28	29	27	18	54

Figure 11: number of unique shapes for the two groups (precision 4)

K	3				4			
w	8	16	32	64	8	16	32	64
less smooth	45	46	59	60	53	80	67	63
more smooth	14	9	14	25	25	25	17	49

Figure 12: number of unique shapes for the two groups (precision 3)

its skew is the standard deviation of δ_i's, divided by their average:

$$skew(\delta_1, \delta_2, \ldots) = stddev(\delta_1, \delta_2, \ldots) \;/\; avg(\delta_1, \delta_2, \ldots)$$

For equilateral triangles, skew is 0; the further a triangle is from an equilateral, the greater is its skew.

Figure 13 shows the average skew of the sequences in the two groups from Figure 9. It is clear that the shapes for the more smooth sequences are more skewed than for the less smooth ones.

7. CAUSE OF EQUIVALENT CONFIGURATIONS IN STS CLUSTERING

We will now explain why the shape, or *structure* of the configuration of STS cluster centroids depends on the sequence (Section 5), whereas the actual *position* of the cluster centroids does not. We also explain why the number of shapes depends on the smoothness coefficient (Section 6).

7.1 K-means optimization as a function of pairwise distances

In order to explain the experimental results presented in Section 5, we first reformulate the usual K-means objective function or optimization criterion, i.e., the *Sum-of-Squared-Errors* (SSE), in terms of the scalar products (pairwise distances) between pattern vectors.

Let N d-dimensional row vectors, labeled $x_1, \ldots, x_i, \ldots, x_N$, be clustered into K clusters with centroids $\mu_1, \ldots, \mu_k, \ldots, \mu_K$. Let $I(k)$ be the index of the cluster to which x_i was assigned, and $n(k)$ be the number of patterns in cluster k. Then, the following is the formula for SSE:

$$SSE = \min \sum_{k=1}^{K} \sum_{i \in I(k)} (x_i - \mu_k)(x_i - \mu_k)'$$

$$= \min \sum_{k=1}^{K} \left(\sum_{i \in I(k)} x_i - \frac{1}{n(k)} \sum_{j \in I(k)} x_j \right) \left(\sum_{i \in I(k)} x_i - \frac{1}{n(k)} \sum_{j \in I(k)} x_j \right)'$$

$$= \min \left(\sum_{i=1}^{N} x_i x_i' - \sum_{k=1}^{K} \frac{1}{n(k)} \sum_{i \in I(k)} \sum_{j \in I(k)} x_i x_j' \right)$$

$$= const + \max \left(\sum_{k=1}^{K} \frac{1}{n(k)} \sum_{i \in I(k)} \sum_{j \in I(k)} x_i x_j' \right)$$

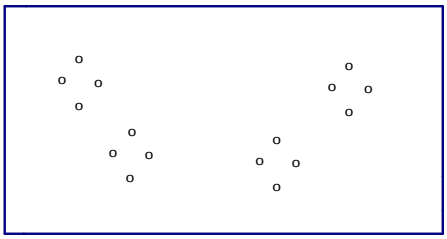

Figure 14: A set of 16 points in 2-D space.

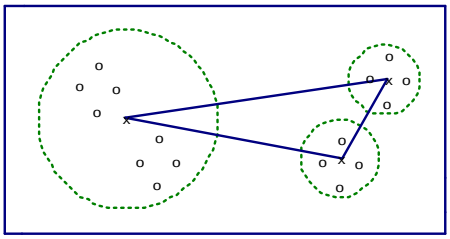

Figure 15: Cluster configuration, with centroids shown with crosses, found by K-means algorithm with $K = 3$.

Therefore the conventional objective function SSE is equivalent to maximizing the sum of the intra-cluster scalar products

$$\sum_{k=1}^{K} \frac{1}{n(k)} \sum_{i \in I(k)} \sum_{j \in I(k)} x_i x_j',$$

where

$$x_i x_j' = ||x_i|| + ||x_j|| - \frac{1}{2}(x_i - x_j)(x_i - x_j)'.$$

We conclude that if all the vectors have the same length, then two cluster configurations with identical intra-cluster pairwise scalar products $x_i x_j'$ (or alternately identical pairwise distances $(x_i - x_j)(x_i - x_j)'$, must yield the same value of SSE. However, the cluster centroids of the two configurations will, in general, be different.

7.2 Multiple global maxima

Figure 14 shows a set of points in 2-D vector space that exhibit multiple global maxima of the SSE. When clustered with the K-means algorithm with $K = 3$ these samples will

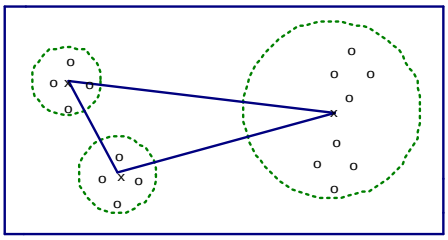

Figure 16: Centroids found with different initial cluster centroids. The two cluster configurations have the same shape and the same SSE, but the centroids cannot be mapped into one another one another simply by renumbering them.

K	3				4			
w	8	16	32	64	8	16	32	64
less smooth	0.0788	0.0834	0.1111	0.0821	0.1256	0.1087	0.1500	0.1475
more smooth	0.2541	0.2520	0.2125	0.3330	0.2440	0.2489	0.2253	0.2840

Figure 13: average skew for the two groups

be grouped either as indicated in Figure 15 or as in Figure 16. The two configurations have the same SSE (the sum of the distances from the patterns to their cluster centroids), and the same shape (here, congruent triangles), but evidently not the same position; that is, the cluster centroids cannot be mapped into one another by renumbering.

The existence of multiple equivalent configurations for the points in Figure 14 is due to symmetries in the configurations of the subsequence vectors. We will show that approximate symmetries inevitably arise in STS clustering because many pairs of shifted subsequences have almost the same Euclidean distance between them. This results in multiple cluster configurations with nearly identical SSEs.

7.3 Cyclic shifts

We will first show that periodic signals exhibit many pairs of equidistant shifted subsequences and, consequently, multiple global minima of the SSE. Overlapped subsequences of a periodic sequence can be conveniently represented as *cyclic shifts* of a single subsequence (provided that the window length is equal to the period (or to a multiple thereof).

Consider a sequence S consisting of many repetitions of a single subsequence s of length w, i.e., $S = s, s, s, s, s, \ldots$. With infinite repetition, all of the shifted subsequences of length w of s can be obtained from the *circulant matrix* M. For example, if $w = 5$, $s = 2, 3, 8, 4, 7$, and $S = 2, 3, 8, 4, 7, 2, 3, 8, 4, 7, 2, 3, 8, 4, 7, \ldots$ then

$$\mathrm{M}= \begin{vmatrix} 2 & 3 & 8 & 4 & 7 \\ 3 & 8 & 4 & 7 & 2 \\ 8 & 4 & 7 & 2 & 3 \\ 4 & 7 & 2 & 3 & 8 \\ 7 & 2 & 3 & 8 & 4 \end{vmatrix}$$

Because all of the cyclic-shifted subsequences contain the same values, their lengths are also the same. Now consider two pairs of rows that correspond to the same shift, for instance rows 1 & 4, and rows 2 & 5. The scalar product of both pairs will be the same (113 in this case). Therefore the Euclidean distance between row vectors 1 & 4, and between rows 2 & 5, is also the same. The dot product and the distance between the vectors corresponding to rows 3 & 1 is also the same (because 6 mod 5 = 1): they are *shift equivalent* to row vectors 1 & 4.

In fact, there can be at most $w/2$ distinct distance values (aside from 0) for w even and $(w-1)/2$ distinct values for w odd. (The factor of 2 arises because a shift forward or backward is the same.) Therefore the scalar products among the (5 x 4)/2 = 10 pairs of vectors created by cyclic shifts of 2, 3, 8, 4, 7, will have only two distinct values: 104 and 113. So the five row vectors lie in 5-D space at the vertices of a polyhedron with only two distinct edge lengths (i.e., an *isosceles simplex*).

If we self-concatenate the above subsequence many times, different random initializations will form three clusters populated either by rows 1 & 4; 3 & 5; and 2, or by rows 1 & 3; 2 & 5; and 4. The SSE for the two configurations is the same: 58.0. So are their shapes, or lengths of the sides of the two triangles formed by the two sets of cluster centroids. However, the coordinates of the centroids of the two configurations are different.

	1	2	3	4	5	6	7	8	9	10	11	12	13
S:	2	3	8	4	7	1	6	5	9	4	1	3	8
(1)	2	*3*	*8*	*4*	*7*	*1*	*6*	*5*					
(3)	8	*4*	*7*	*1*	*6*	*5*	*9*	*4*					
(2)	*3*	*8*	*4*	*7*	*1*	*6*	*5*	9					
(4)	*4*	*7*	*1*	*6*	*5*	*9*	*4*	1					

Figure 17: Similar distance components in shifted subsequences.

The shape of the triangles depends on the two distinct values of pairwise distances in 5-D space. We saw that for 2, 3, 8, 4, 7, all the scalar products between vectors were either 104 or 113. For a different sequence, 1, 2, 4, 8, 16, they are 124 or 186. The ratios of the sides of the triangles of the optimal cluster configurations (which form isosceles triangles) are 0.86 and 0.97: therefore their shapes are different.

7.4 Arbitrary sequences

We are not clustering periodic sequences, but the number of dissimilar distance values between shifted subsequences in many real problems is also lower than what might be expected. Consider the arbitrary sequence

$$S = 2, 3, 8, 4, 7, 1, 6, 5, 9, 4, 1, 3, 8, \ldots$$

shown in Figure 17. Examine two pairs of subsequences, with window length $w = 8$, beginning at positions 1 & 3, and 2 & 4 respectively. If the window were shifted cyclically, then the two pairwise distances would be exactly equal. Even as it is, however, we see that seven pairs of component values (shown in italics) are identical. If we shifted the second sequence one more position, then six of the eight pairs of dot-product components would still be the same. We would therefore expect that the corresponding distances between the shifted subsequences would also be very similar.

This multiplicity of almost identical distances gives rise to alternative cluster configurations with almost identical SSEs. With different random initializations, K-means lands on one of these local extrema. The average value of each distinct group of distances depends, of course, on the underlying sequence, and therefore so will the shape of the resulting cluster configuration. (Note that Figure 14 had many pairs of patterns with almost identical distances.) As in the case of cyclic shifting, the centroids of the different configurations cannot be mapped onto one another by renumbering.

If the same subsequence (a *pattern*) occurs several times in the sequence, then the number of identical pairwise distance values will increase correspondingly. Strict periodicity is the extreme case. Smooth sequences have more similar shift-equivalent pairs of subsequences, and therefore give rise to

more congruent and less skewed cluster configurations. We note, however, that *exact* equivalences arise only with periodic sequences, and only if the window length is equal to an integer multiple of the period.

7.5 Summary

In STS, the K-means sum-of-squared-error optimization criterion is a function of the scalar product of pairs of subsequence. Pairs of subsequences with the same displacement (*shift-equivalent pairs*) are almost equidistant (exact equality holds only for strictly periodic sequences). The multiplicity of similar pairwise distance values gives rise to approximately equal extrema of the optimization criterion (the SSE) for several cluster configurations, any of which may be reached with some random initialization. The centroids of these configurations have the same structure, but the clusters contain different vectors, hence their centroids cannot be mapped into one another by renumbering. The distinctive shape of the configurations depends on the distances between tight groups of shift-equivalent subsequences (windows). The cluster configurations of smooth, repetitive sequences display the most symmetry.

8. DISCUSSION

The recent claim of meaninglessness for STS subsequence clustering [18] (preliminary version [19]) "has cast a shadow over the emerging discipline of time series data mining" [28]. It has also generated a flurry of follow-up research activity. As a result of this claim, many have moved away from STS subsequence clustering, such as [26, 1, 21]. Others tried to find "meaningful" results by using alternate clustering methods, such as *density-based clustering* [8] and *self-organizing maps* [27], or by limiting themselves to cyclic data [7], or by suggesting the use of medoids rather than means for clustering [14].

As far as we know, ours is the only work that succeeds to demonstrate a strong dependence of output on input for the K-means STS clustering algorithm. This is despite the fact that our matching criteria are tougher than elsewhere, such as [18, 8, 27] – we are not restricted to just two sequences. We therefore hope that this work will help to lift some of the shadow that has been cast over time-series data mining.

Our research raises several questions, such as:

1. Why does cluster shape distance provide a reliable way to match clusters to the original sequences for STS clustering, where cluster set distance fails to do so?

2. How does sequence shape depend on various properties of the sequence? Specifically, why are the number of unique shapes, and the shape skew, correlated with sequence smoothness?

3. Given some value of Q (number of shapes), what is the expected level of accuracy of our matching algorithms for a given dataset? Conversely, how large should the value of Q be to achieve some desired expectation of matching accuracy?

We have provided the answer to Question 1, and partially to Question 2, in Section 7. Question 3 remains open; we now discuss it further.

We found that $Q = 100$ was sufficient for 85% accuracy for algorithm I and 100% accuracy of Algorithm II. However, this clearly depends on the nature of the sequences. Would $Q = 25$ suffice? Will there be cases where $Q = 100$ is not enough? How can we tell, for a given dataset and a given degree of accuracy, what the appropriate value of Q should be?

There is a direct connection between the number of unique shapes and the value of Q. Clearly, if there are at most j unique shapes for any sequence in the input dataset, and we only store those, then $Q = j$ suffices to achieve 100% accuracy with Algorithm II. We therefore hope that our research into shapes, and specifically the two correlations we have identified in Section 6, will prove a good starting point for answering Question 3.

In general, relating the error rate to the size of the training or reference set has been the objective of long-standing research in pattern recognition. Only asymptotic results have been obtained, the best known of which is the bound of the Nearest Neighbor classifier at twice the Bayes Error Rate [10]. There are also no general results on reaching local vs. global optima with different initializations in gradient descent algorithms. We therefore expect that strong assumptions about the nature of the sequences in the dataset will be necessary to provide a full answer to Question 3.

Another question that arises naturally is whether the sensitivity of our clustering algorithm to different initializations is simply due to the fact that it finds local, rather than global, minima. We believe that this is not the primary cause. We chose to use the clustering algorithm from [18] so as to duplicate their setting. However, we have experimented with other K-means STS clustering algorithms, that attempted to get closer to the global minimum, and the results where the same. We also saw no improvement in our matching algorithm when we used K-medoids in place of K-means.

9. ACKNOWLEDGMENTS

George Nagy gratefully acknowledges the support of NSF grant #0414854.

10. REFERENCES

[1] A. J. Bagnall, G. Janakec, B. de la Iglesia, and M. Zhang. Clustering time series from mixture polynomial models with discretised data. In *Proc. 2nd Australasian Data Mining Workshop*, Dec. 2003.

[2] G. Ball. Data analysis in the social sciences: What about the details? In *Procs. Fall Joint Computer Conference. Spartan Books.*, pages 533–560, 1965.

[3] G. H. Ball and D. J. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, March 1967.

[4] R. Casey and G. Nagy. Recognition of printed chinese characters. *IEEE Trans. Electronic Computers*, 15(1):91–101, February 1966.

[5] R. Casey and G. Nagy. Autonomous reading machine. *IEEE Transactions on Computers*, 17(5):492–503, May 1968.

[6] R. Casey and G. Nagy. Advances in pattern recognition. *Scientific American*, 224(4):56–71, 1971.

[7] J. Chen. Making subsequence time series clustering meaningful. In *Proc. IEEE Int'l Conf. on Data Mining. Houston, Texas.*, Nov. 2005.

[8] A. Denton. Density-based clustering of time series subsequences. In *Proc. 3rd Workshop on Mining Temporal and Sequential Data (TDM 04) in conjunction with The Tenth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Aug. 2004.

[9] R. Dubes and A. Jain. Validity studies in clustering methodologies. *Pattern Recognition*, 11:235–254, 1979.

[10] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley Publishers, 2001.

[11] C. Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, Boston, 1996.

[12] A. Gersho and R. Gray. Vector quantization and signal compression. *The International Series in Engineering and Computer Science*, 1991.

[13] D. Goldin and P. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proc. 1st Int'l Conf. on the Principles and Practice of Constraint Programming. CP'95, Cassis, France*, pages 137–153, Sep. 1995.

[14] T. Ide. Why does subsequence time-series clustering produce sine waves? In *Proc. PKDD, Sep. 18-22, 2006, LNAI 4213*, pages 311–322. Springer, 2006.

[15] A. Jain. *Cluster Analysis (ch.2) in Handbook of Pattern Recognition and Image Processing*. Academic Press, New York, 1986.

[16] A. Jain and D. R. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[17] E. Keogh and T. Folias. Data mining archive [http://www.cs.ucr.edu/ eamonn/tsdma/index.html]. cse dept., univ. of california-riverside. 2002.

[18] E. Keogh and J. Lin. Clustering of time series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8(2):154–177, Aug. 2005.

[19] E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Proc. 3rd IEEE Int'l Conf. on Data Mining*, pages 115–122. IEEE, Nov. 2003.

[20] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Statist.*, pages 281–297, 1967.

[21] M. Mahoney and P. Chan. Learning rules for time series anomaly detection. *Tech. Rep. CS-2005-04. Florida Inst. of Technology. Melbourne, FL.*, 2005.

[22] G. Nagy. State of the art in pattern recognition. In *Proc. IEEE, 56:5*, pages 336–362, May 1968.

[23] G. Nagy. Feature extraction on binary patterns. *IEEE Trans. System Science and Cybernetics*, 5(4):273–278, Oct. 1969.

[24] G. Nagy, S. Seth, and K. Einspahr. Decoding substitution ciphers by means of word matching with application to ocr. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(5):710–715, Sep. 1987.

[25] G. Nagy and J. Tolaba. Nonsupervised crop classification through airborne multispectral observations. *IBM Journal of Research and Development*, 16(2):138–153, March 1972.

[26] P. Rodrigues, J. Gama, and P. Pedroso. Hierarchical time-series clustering for data streams. In *Proc. 1st Int'l Workshop on Knowledge Discovery in Data Streams.*, Sep. 2004.

[27] G. Simon, J. Lee, and M. Verleysen. On the need of unfolding preprocessing for time series clustering. In *Proc. of WSOM'05, Workshop on Self-Organizing Maps. Paris, France.*, pages 251–258, Sep. 2005.

[28] Z. Struzik. Time series rule discovery: Tough, not meaningless. In *Proc. Int'l Symp. on Methologies for Intelligent Systems (ISMIS)*, pages 32–39, Oct. 2003.

[29] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1999.

[30] A. Topchy, A. Jain, and W. Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(12):1866–1881, Dec. 2005.