Chapter 15

Time Series Data Clustering

Dimitrios Kotsakos
Dept. of Informatics and Telecommunications
University of Athens
Athens, Greece
dimkots@di.uoa.gr

Goce Trajcevski
Dept. of EECS, Northwestern University
Evanston, IL, USA
goce@eecs.northwestern.edu

Dimitrios Gunopulos
Dept. of Informatics and Telecommunications
University of Athens
Athens, Greece
dg@di.uoa.gr

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
charu@us.ibm.com

15.1 Introduction

Time-series data is one of the most common forms of data encountered in a wide variety of scenarios such as the stock markets, sensor data, fault monitoring, machine state monitoring, environmental applications or medical data. The problem of clustering finds numerous applications in the time series domain, such as the determination of groups of entities with similar trends. Time-series clustering has numerous applications in diverse problem domains:

- *Financial Markets:* In financial markets, the values of the stocks represent time-series which continually vary with time. The clustering of such time-series can provide numerous insights into the trends in the underlying data.
- *Medical Data:* Different kinds of medical data such as EEG readings are in the form of time-series. The clustering of such time-series can provide an understanding of the common shapes in the data. These common shapes can be related to different kinds of diseases.
- *Earth Science Applications:* Numerous applications in earth science, such as temperature or pressure, correspond to series, which can be mined in order to determine the frequent trends in the data. These can provide an idea of the common climactic trends in the data.
- *Machine State Monitoring:* Numerous forms of machines create sensor data, which provides a continuous idea of the states of these objects. These can be used in order to provide an idea of the underlying trends.
- *Spatio-temporal Data:* Trajectory data can be considered a form of multi-variate time-series data, in which the X-coordinate and Y-coordinates of objects correspond to continuously varying series. The trends in these series can be used in order to determine the important trajectory clusters in the data.

Time-series data falls within the class of *contextual data* representations. Many kinds of data such as time-series data, discrete sequences, and spatial data fall in this class. Contextual data contains two kinds of attributes:

- *Contextual Attribute:* For the case of time-series data, this corresponds to the time-dimension. These provide the reference points at which the behavioral values are measured. The time-stamps could correspond to actual time values at which the data points are measured, or they could correspond to *indices* at which these values are measured.
- *Behavioral Attribute:* This could correspond to any kind of behavior which is measured at the reference point. Some examples include stock ticker values, sensor measurements such as temperature, or other medical time series.

The determination of clusters of time series is extremely challenging because of the difficulty in defining similarity across different time series which may be scaled and translated differently both on the temporal and behavioral dimensions. Therefore, the concept of similarity is a very important one for time-series data clustering. Therefore, this chapter will devote a section to the problem of time series similarity measures. Note that once a similarity measure has been defined for time-series

data, it can be treated as an abstract object on which a variety of similarity-based methods such as spectral methods or partitioning methods can be used.

Time-series data allows diverse formulations for the clustering process, depending upon whether the series are clustered on the basis of their online correlations, or whether they are clustered on the basis of their shapes. The former methods are usually performed with an online approach based on a small past window of history, whereas the latter sets of methods are typically performed with an offline approach on the entire series. Therefore, this chapter will also carefully study the diverse formulations which arise in the context of time-series data clustering, map them to different application-specific scenarios, and also discuss the techniques for the different formulations.

Clustering of time series data, like clustering for all types of data, has the goal of producing clusters with high intra-cluster similarity and low inter-cluster similarity. Specifically, objects belonging to the same cluster must exhibit high similarity to each other, while objects belonging to different clusters must exhibit low similarity, thus high distance to each other. However, some specific properties that are part of the nature of the time-series data – such as the *high dimensionality*, the presence of *noise* and the *high feature correlation* [53, 25] – pose unique challenges for designing effective and efficient clustering algorithms. In time series clustering it is crucial to decide what kind of similarity is important for the clustering application. Accordingly, an appropriate clustering algorithm and an appropriate distance measure should be chosen. For example, Euclidean distance reflects similarity in time, while dynamic time warping (DTW) reflects similarity in shape. Other approaches, like model-based clustering methods such as Hidden Markov Models (HMM) or ARMA processes [56] are followed when similarity in change matters.

A significant difference between time-series data clustering and clustering of objects in Euclidean space is that the time series to be clustered may not be of equal length. When this is not the case, so all time series are of equal length, standard clustering techniques can be applied by representing each time series as a vector and using a traditional L_p-norm distance [18]. With such an approach, only similarity in time can be exploited, while similarity in shape and similarity in change are disregarded. In this study we split the clustering process into two basic steps. The first one is the choice of the similarity measure that will be employed, while the second one concerns the grouping algorithm that will be followed. In the section discussing the first step, we classify the proposed approaches into two categories: one for one-dimensional time series and one for multidimensional time series. In the section discussing the clustering algorithms, we present them following the traditional classification scheme that defines three main classes of clustering algorithms: (a) *k*-means and its variants, (b) hierarchical approaches and (c) density-based clustering. In addition to these three main classes, in subsection 15.5.4 we discuss methods for trajectory clustering, as we consider this field to be of individual interest.

15.2 The Diverse Formulations for Time-series Clustering

Time-series clustering lends itself to diverse formulations, which are highly application-specific in nature. For example, do we wish to determine sets of time-series with similar online trends, or do we wish to determine time-series with similar shapes? The objective function criteria in these cases are clearly very different. Furthermore, the application-specific goals in these cases are also quite different, and the corresponding similarity functions used in the analytical process are also therefore quite different.

Broadly speaking, two main formulations exist for time-series data clustering:

- *Correlation-based Online Clustering:* In this case, a bunch of time-series are clustered in *real-time* on the basis of the correlations among the different time series. Such applications

are common in scenarios such as financial markets, where it is desirable to determine groups of stocks which move together over time. Such methods can also be used in order to find time-series which have similar trends, or at least correlated trends over time. In these cases, a short window of history is used for the clustering process, and the similarity functions across the different series are based on inter-attribute correlations. These methods are also intimately related to the problems of time-series forecasting and regression. In fact, a few such clustering methods such as *Selective Muscles* [58] use clustering methods for stream selection and efficient forecasting. Other methods in the literature use such techniques for sensor selection [1, 2]. An important aspect of such methods is that they often need to be performed in real time, as the streams are evolving over time. In many cases, it is also desirable to detect the significant changes in the clustering behavior. This provides insights into how the stream correlation trends have changed significantly.

- *Shape-based Offline Clustering:* In these cases, time-series of similar *shapes* are determined from the data. The main challenge here is to define an appropriate shape-based similarity function. This can be extremely challenging because it requires the definition of similarity functions which can capture the similarities in the shapes of the time-series. Depending upon the domain, the time-series may be translated, scaled and time-warped. Therefore, numerous similarity functions such as the Euclidean function or dynamic time warping are used for time-series clustering. Clearly, the appropriateness of a similarity function depends upon the underlying data domain. Furthermore, some of the similarity functions are computationally quite expensive, since they require the use of dynamic programming methods for computation. Depending upon the application, the exact values on the time-stamps may sometimes not be important, as long as the shapes of the time-series are similar. For example, consider the case, where the EEG readings of different patients are used to create a database of multiple time-series. In such cases, the exact value of the time-stamp at which the EEG reading was measured is not quite as important. This is very different from online correlation-based clustering, where all the series are evaluated over approximately the same time-stamps. Some versions of shape-based clustering are also used for online-clustering, such as the methods which perform subsequence clustering.

Clearly, the precise choice of model and similarity function depends upon the specific problem domain. Therefore, this chapter will examine the different formulations in the context of different application domains.

The problem becomes more complex, as one moves to the multi-variate scenario. The simplest multi-variate scenario is that of trajectory clustering, which can be considered a form of bivariate or trivariate time-series data, depending upon whether 2- or 3-dimensional trajectories are considered. The afore-mentioned cases also apply to the case of trajectories, where it may be desirable to either determine objects which move together (online correlation methods), or determine trajectories which have similar shape (offline shape methods). In the latter case, the exact time at which the trajectories were created may be of less importance. This chapter will carefully address the case of multi-variate time-series clustering, and will use spatio-temporal data as specific application domain of interest. While spatio-temporal data is used as a specific case in this chapter, because it provides the only known set of multi-variate clustering methods, the techniques for spatio-temporal data clustering can be generalized to any bivariate and possibly non-spatial time-series data, by "pretending" that the two series correspond to a trajectory in 2-dimensional space. Furthermore, the core techniques for trajectory-based methods can also be generalized to higher dimensional multivariate time-series, by treating them as n-dimensional trajectories. Therefore, the existing research on trajectory clustering provides some of the most powerful results on how multi-variate time series clustering can be properly addressed.

15.3 Online Correlation-based Clustering

Online correlation-based clustering methods are closely related to the problem of forecasting. Such methods are typically based on clustering the streams on the basis of their correlations with one another in their past window of history. Thus, the similarity function between the different series uses an intra-stream regression function in order to capture the correlations across different streams.

These methods are typically based on windows of the immediate history. Specifically, a window of length p is used for regression analysis, and the different series are clustered on the basis of these trends. Some of the most common methods for segmenting such streams define *regression-based similarity functions* on the previous window of history. Note that two streams in the same cluster need not be positively correlated. In fact, two streams with perfect negative correlation may also be assumed to belong to the same cluster, as long as the *predictability* between the different streams is high. This is quite often the case in many real scenarios in which some streams can be predicted almost perfectly from others.

15.3.1 Selective Muscles and Related Methods

The *Selective Muscles* method [58] is designed to determine the k best representatives from the current time series which can be used in order to predict the other series. This approach can be considered a version of the k-medoid clustering for online predictability-based clustering of time series. One important aspect of the original *Selective Muscles* approach is that it was designed for finding the k best representatives which predict *one particular* stream in the data. On the other hand, in unsupervised correlation clustering, it is desirable to determine the best set of representatives which can predict all the streams in the data. However, the two problems are almost exactly the same in principle, since the same approach in *Selective Muscles* can be used with an aggregated function over the different time series. The problem can be posed as follows:

Problem 15.3.1 *Given a dependent variable $\overline{X_i}$ and $d-1$ independent variables $\{\overline{X_j} : j \neq i\}$, determine the top $b < d$ streams to pick, so as to minimize the expected estimation error.*

Note that the estimation is performed with a standard model such as the *Autoregressive (AR)*, or the *Autoregressive Integrated Moving Average (ARIMA)* models. A more general version of the problem does not treat any particular variable as special, and may be defined as follows:

Problem 15.3.2 *Given d variables $\overline{X_1} \ldots \overline{X_d}$, determine the top $b < d$ streams to pick so as to minimize the average estimation error over all d streams.*

Note that while the second problem is slightly different from the first, and more relevant to online clustering, it is no different in principle.

The approach in [58] uses a greedy method in order to select the k representatives for optimizing the predictability of the other streams. The approach for selecting the representatives is as follows. In each iteration, a stream is included in the representative set, which optimizes the estimation error. Subsequently, the next stream which is picked is based on maximizing the aggregate impact on the estimation error, considering the streams which have already been picked. In each iteration, the stream is added to the set of representatives, for which the incremental impact on the estimation error is as large as possible.

However, the technique in [58] is optimized in order to pick the k streams which optimize a specific dependent variable. A method in [2] uses the greedy approach in order to prick the streams with the use of the formulation discussed in Problem 15.3.2. A graph-based representation is used to model the dependencies among the streams with the use of a linkage structure. A greedy algorithm

```
Algorithm CorrelationCluster(Time Series Streams: [1...n]
     NumberOfStreams: k;
begin
  J = Randomly sampled set of k time series streams;
  At the next time stamp do
  repeat
    Add a stream to J, which leads to
      maximum decrease in regression error;
    Drop the stream from J which leads to
      least increase of regression error;
  until(J did not change in last iteration)
end
```

FIGURE 15.1: Dynamically Maintaining Cluster Representatives

is used in order to pick the optimal set of representatives. It has been shown in [2] that the approach has an approximation bound of $(e-1)/e$ over the optimal choice of representatives.

A method in [1] performs the clustering directly on the streams by defining a *regression-based similarity function*. In other words, two streams are deemed to be similar, if it is possible to predict one stream from the other. Otherwise the immediate window of history is used in order to cluster the streams. The work in [1] is a general method, which also associates a cost of selecting a particular representative. In general, for the clustering problem, it is not necessary to model costs. A simplified version of the algorithm, in which all costs are set to the same value is provided in Figure 15.1. The similarity between two streams i and j is equal to the regression error in predicting stream j from stream i with the use of any linear model. A particular form of this model has been discussed in [1]. Note that the similarity function between streams i and j is not symmetric, since the error of predicting stream i from stream j is different from the error of predicting stream j from stream i.

These methods for online correlation based stream clustering are very useful in many applications, since it is possible to select small subsets of streams, from which all the other streams can be effectively predicted. A number of other methods, which are not necessarily directly related to muscles select representatives from the original data streams. Such methods are typically used for sensor selection.

15.3.2 Sensor Selection Algorithms for Correlation Clustering

Sensor selection algorithms are naturally related to correlation clustering. This is because such methods typically pick a representative set of streams which can be used in order to predict the other streams in the data. The representative streams are used as a proxy for collecting streams from all the different streams. Such an approach is used in order to save energy. This naturally creates clusters in which each stream belongs to the cluster containing a particular representative. A number of techniques have been designed in recent years in order to determine correlations between multiple streams in real time [43, 48, 58, 59]. The techniques in [43, 48] use statistical measures in order to find lag-correlations and forecast the behavior of the underlying data stream. The works in [2, 16] propose methods for sensor selection with the use of domain-specific linkage knowledge and utility-feedback, respectively. Methods for observation selection are proposed in [27], when the importance of a sensor-set can be *pre-modeled* as a submodular function. Methods for using adaptive models for time-series prediction were proposed in [57]. The method in [58] uses linear regression in order to determine the correlations in the underlying data and use them for forecasting. The technique in

[59] proposes general monitoring techniques to determine statistical correlations in the data in real time.

Correlation clustering algorithms in time series can also be used in order to monitor key *changes* in the correlation trends in the underlying stream. This is because when the correlations between the different streams change over time, this leads to changing membership of streams in different clusters in the data. Such changes are relevant to determining key temporal anomalies in the data, and are discussed in [4].

15.4 Similarity and Distance Measures

For more conventional *shape-based* clustering algorithms, the use of similarity and distance measures is crucial. In [34] Liao classifies time series distance measures into three classes:

(a) feature-based

(b) model-based

(c) shape-based

While this classification makes sense and analyzes in depth the proposed time series distance measures and their differences, it lacks information about whether each distance measure is appropriate for comparing multi-variate time series as well or not. Over the last few years, multi-variate time series clustering has attracted the interest of the time series research community as it has a variety of interesting applications. In this sense, our study of similarity / distance measures that are used in time series clustering is supplementary to that of Liao [34] and provides another interesting insight. One of the most important decisions when performing time series clustering is the similarity / distance measure that will be chosen in order to compare the time series. Given a time series similarity measure, time series clustering can be performed with one of the several proposed techniques that will be reviewed in the next section. Time series similarity has been a long and deeply studied problem for the past two decades. In the literature one can find many proposed distance measures, with each of them being appropriate for different applications and having specific advantages and disadvantages. Many classification schemes have been proposed regarding time series similarity measures. A popular classification criterion is the time series representation scheme that is assumed by the various measures, as the time series can be represented either by the raw data or by transformations of the original data such as the frequency transformation representations (Discrete Fourier Transformation (DFT), Discrete Wavelet Transformation (DWT)) or other representation models (Landmark and Important Points Representation, ARIMA Model and LPC Cepstral Coefficient Representation, Symbolic Representation, Signature Representation [11]). In this section we employ the raw data representation, where a time series T of length n is represented as an ordered sequence of values $T = [t_1, t_2, \ldots, t_n]$. T is called raw representation of the time series data. Another important feature and classification criterion when classifying similarity / distance measures for time series is whether a measure is appropriate for univariate or multi-variate time series. In this section we review the most commonly used similarity / distance measures when it comes to time series clustering and distinguish them to univariate and multi-variate measures.

15.4.1 Univariate Distance Measures

In this subsection, we shortly present some similarity / distance measures that are widely used by the majority of time series clustering methods proposed in the literature and reviewed in this

chapter. The distance/similarity measures presented here handle one-dimensional time series and most of them are extended by many multidimensional time series similarity / distance approaches that will be described in the next subsection.

15.4.1.1 L_p Distance

The L_p-norm is a distance metric, since it satisfies all of the non-negativity, identity, symmetry and triangle inequality conditions. An advantage of the L_p-norm is that it can be computed in linear time to the length of the trajectories under comparison, thus its time complexity is O(n), n being the length of the time series. In order to use the L_p-norm, the two time series under comparison must be of the same length.

The Minkowski of order p or the L_p-norm distance, being the generalization of Euclidean distance, is defined as follows:

$$L_p - \text{norm}(T_1, T_2) = D_{M,p}(T_1, T_2) = \sqrt[p]{\sum_{i=1}^{n} (T_{1i} - T_{2i})^p} \tag{15.1}$$

The Euclidean distance between two one-dimensional time series T_1 and T_2 of length n is a special case of the L_p-norm for $p = 2$ and is defined as:

$$D_E(T_1, T_2) = L_2 - \text{norm}(T_1, T_2) = \sqrt{\sum_{i=1}^{n} (T_{1i} - T_{2i})^2} \tag{15.2}$$

L_1-norm (p=1) is named the Manhattan distance or city block distance.

15.4.1.2 Dynamic Time Warping Distance

Dynamic Time Warping (DTW) is a well known and widely used shape-based distance measure. DTW computes the warping path $W = w_1, w_2, ..., w_K$ with $\max(m,n) \leq K \leq m+n-1$ of minimum distance for two time series of lengths m and n.

DTW stems from the speech processing community [45] and has been very popular in the literature of time series distance measures [6]. Moreover, it has been extended by many approaches to handle the multi-dimensional case of trajectory matching. With use of dynamic programming DTW between two one-dimensional time series T_1 and T_2 of length m and n respectively can be computed as follows:

(a) $D_{\text{DTW}}(T_1, T_2) = 0$, if $m = n = 0$

(b) $D_{\text{DTW}}(T_1, T_2) = \infty$, if $m = n = 0$

(c) $D_{\text{DTW}}(T_1, T_2) = \text{dist}(T_{11}, T_{21}) + \textit{minFactor}$, otherwise

where *minFactor* is computed as:

$$\textit{minFactor} = \min \begin{cases} D_{\text{DTW}}(\text{Rest}(T_1), \text{Rest}(T_2)) \\ D_{\text{DTW}}(\text{Rest}(T_1), T_2) \\ D_{\text{DTW}}(T_1, \text{Rest}(T_2)) \end{cases}$$

where $\text{dist}(T_{1,1}, T_{2,1})$ is typically the L_2-norm. The Euclidean distance, as long as all L_p-norms, described in 15.4.1.1, performs an one-to-one mapping between the data points of the time series under comparison. Thus, it can be seen as a special case of the DTW distance, which performs an one-to-many mapping. DTW is a more robust distance measure than L_p-norm because it allows time shifting and thus matches similar shapes even if they have a time phase difference. An important point is that DTW does not satisfy the triangular inequality which could be a problem while indexing

time series. However, in the literature there are several lower bounds that serve as solutions for indexing DTW offering faster performance [54]. Another advantage of DTW over L_p-norm is that DTW can handle different sampling intervals in the time series. This is a very important feature especially for long time series that span many years as the data sampling strategy may change over a long time.

15.4.1.3 EDIT Distance

EDIT distance (ED) comes from the field of string comparison and measures the number of insert, delete and replace operations that are needed to make two strings of possibly different lengths identical to each other. More specifically, the EDIT distance between two strings S_1 and S_2 of length m and n respectively is computed as follows:

(a) $D_{ED}(S_1, S_2) = m$, if $n = 0$

(b) $D_{ED}(S_1, S_2) = n$, if $m = 0$

(c) $D_{ED}(S_1, S_2) = D_{ED}(\text{Rest}(S_1), \text{Rest}(S_2))$, if $S_{1_1} = S_{2_1}$

(d) $D_{ED}(S_1, S_2) = \min \begin{cases} D_{ED}(\text{Rest}(S_1), \text{Rest}(S_2)) + 1 \\ D_{ED}(\text{Rest}(S_1), S_2) + 1 \\ D_{ED}(S_1, \text{Rest}(S_2)) + 1 \end{cases}$, otherwise

Although ED for strings is proven to be a metric distance, the two ED-related time series distance measures DTW and Longest Common Subsequence (LCSS) that will be described in the next subsection are proven not to follow the triangle inequality. Lei Chen [10] proposed two extensions to EDIT distance, namely Edit distance with Real Penalty (ERP) to support local time shifting and Edit Distance on Real sequence (EDR) to handle both local time shifting and noise in time series and trajectories. Both extensions have a high computational cost, so Chen proposes various lower bounds, indexing and pruning techniques to retrieve similar time series more efficiently. Both ERP and DTW can handle local time shifting and measure the distance between two out of phase time series effectively. An advantage that ERP has over DTW is that the former is a metric distance function, whereas the latter is not. DTW does not obey triangle inequality, and therefore traditional index methods cannot be use to improve efficiency in DTW-based applications. On the other hand, ERP is proved to be a metric distance function [10], and therefore traditional access methods can be used. EDR, as a distance function, proves to be more robust than Euclidean distance, DTW and ERP are more accurate than LCSS that will be described in the next subsection. EDR is not a metric, thus the author proposes three non-constraining pruning techniques (mean value Q-grams, near triangle inequality and histograms) to improve retrieval efficiency.

15.4.1.4 Longest Common Subsequence

The Longest Common Subsequence (LCSS) distance is a variation of EDIT distance described in 15.4.1.3 [51]. LCSS allows time series to stretch in the time axis and does not match all elements, thus being less sensitive to outliers than L_p-norms and DTW. Specifically, the LCSS distance between two real-valued sequences S_1 and S_2 of length m and n respectively is computed as follows:

(a) $D_{\text{LCSS},\delta,\varepsilon}(S_1, S_2) = 0$, if $n = 0$ or $m = 0$

(c) $D_{\text{LCSS},\delta,\varepsilon}(S_1, S_2) = 1 + D_{\text{LCSS},\delta,\varepsilon}(\text{HEAD}(S_1), \text{HEAD}(S_2))$
if $|S_{1,m} - S_{2,m}| < \varepsilon$ and $|m - n| \leq \delta$

(d) $\max \begin{cases} D_{\text{LCSS},\delta,\varepsilon}(\text{HEAD}(S_1), S_2) \\ D_{\text{LCSS},\delta,\varepsilon}(S_1, \text{HEAD}(S_2)) \end{cases}$, otherwise

where HEAD(S_1) is the subsequence $[S_{1,1}, S_{1,2}, ..., S_{1,m-1}]$, δ is an integer that controls the maximum distance in the time axis between two matched elements and ε is a real number $0 < \varepsilon < 1$ that controls the maximum distance that two elements are allowed to have to be considered matched, as depicted in Figure 15.2.

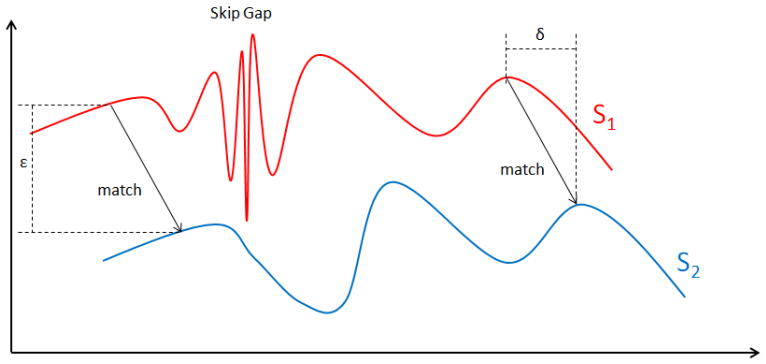

FIGURE 15.2: LCSS distance, thresholds δ and ε

Apart from being used in time series clustering, LCSS distance is often used in domains like speech recognition and text pattern mining. Its main drawback is that often it is needed to scale ore transform the one sequence to the other. A detailed study of LCSS variations and algorithms is presented in [5].

15.4.2 Multivariate Distance Measures

A metric distance function, called Edit distance with Real Penalty (ERP), is proposed that can support local time shifting in time series and trajectory data. A second distance function, Edit Distance on Real sequence (EDR) is proposed to measure the similarity between time series or trajectories with local time shifting and noise [11].

15.4.2.1 Multidimensional L_p Distance

The L_p-norm between two d-dimensional time series T_1 and T_2 of length n extends the L_p-norm for the one-dimensional case and is defined as:

$$L_p - \text{norm}(T_1, T_2) = D_{M,p}(T_1, T_2) = \sqrt[p]{\sum_{i=1}^{n} (T_{1i} - T_{2i})^p} = \sqrt[p]{\sum_{i=1}^{n} \sum_{j=1}^{d} (T_{1ij} - T_{2ij})^p} \tag{15.3}$$

As in the one-dimensional case, multidimensional L_p-norm has been proven to be very sensitive to noise and to local time shifting [29]. A wide variety of methods described in this section, either use the Euclidean distance or expand it, in order to define new distance measures. Lee et al. use the multidimensional Euclidean distance, namely the L_2-norm, to compare multidimensional time series [19]. They define the distance between two multi-variate sequences of equal length as the mean Euclidean distance among all corresponding points in the sequences. For the case where the two compared sequences are not of same length, their approach slides the shorter one over the longer one, and the overall distance is defined as the minimum of all mean distances, computed as described above. Lin and Su [37] use the Euclidean distance in order to define the distance from a point p to a trajectory T as follows: $D_{\text{point}}(p, T) = \min_{q \in T} ED(p, q)$ where $ED(p, q)$ represents the Euclidean distance between points p and q. D_{point} is used to define the one way distance (OWD) from one trajectory T_1 to another trajectory T_2 as the integral of the distance from points of T_1 to trajectory

T_2, divided by the length of T_1. OWD is not symmetric, so the distance between trajectories T_1 and T_2 is the average of their one way distances: $D(T_1,T_2) = \frac{1}{2} \cdot (D_{\text{OWD}}(T_1,T_2) + D_{\text{OWD}}(T_2,T_1))$. Similarly, using an alternate grid representation for trajectories, Lin and Su define the distance between two grid cells as their Euclidean distance, and through it they define the distance between two trajectories [37]. Moreover, they provide a semi-quadratic algorithm with complexity $O(mn)$ for grid trajectory distance computation, where n is the length of trajectories and m is the number of local min points. The grid OWD computation algorithm turns out to be faster than the quadratic complexity needed to compute the DTW between two trajectories. The experimental evaluation proves that OWD outperforms DTW in accuracy and performance. However, OWD does not take into account the time information in trajectories, so no discussion about trajectories with different sampling rates can be made. Frentzos et al. [14] focus on the problem of identifying spatio-temporally similar trajectories, by taking into account the time information in trajectories, except for their spatial shapes. They introduce a dissimilarity metric, DISSIM, between two trajectories Q and R by integrating their Euclidean distance over a definite time interval when both Q and R are valid. This way, DISSIM takes into account the time dimension in both trajectories. Moreover, DISSIM can be used for trajectories with different sampling rates, if the non-recorded data points are approximated by linear interpolation, assuming that the objects follow linear movements. The linear-interpolation technique for missing values can be applied to LCSS and EDR measures too, as pointed out in [14]. Lee et al. solve the problem of searching for similar multidimensional sequences in a database by computing the distance between two sequences through their MBRs [29]. The database sequences as well as the query sequence are partitioned into optimal subsequences that are represented by their MBR. The query processing is based on these MBRs, so scanning and comparing the entire data sequences is avoided. The distance between two MBRs is defined as the minimum Euclidean distance between the two corresponding hyper-rectangles. Based on this distance, the authors introduce two lower-bounding distance metrics and propose a pruning algorithm to efficiently process similarity queries.

15.4.2.2 Multidimensional DTW

DTW (Dynamic Time Warping) between two d-dimensional time series T_1 and T_2 of length m and n respectively is defined as the one-dimensional case, that is described in 15.4.1.2.

Just like in the one-dimensional case, multidimensional DTW allows stretching in time axis, matches all elements and is extensively used in speech recognition domain. DTW, in contrast to Euclidean distance, does not require the two time series under comparison to be of the same length and is not sensitive to local time shifting. However, DTW is not a metric, since it doesn't follow triangle inequality and its time complexity is $O(mn)$, which means that it is computationally expensive for long time series and is useful only for short ones, comprised of a few thousand points. Moreover, DTW, like Euclidean distance, has been proven to be sensitive to noise. [49] contains a very detailed description of the computation and the semantics of DTW. Vlachos et al. apply DTW on handwriting data [54]. Before comparing two trajectories, these are transformed into a rotation invariant Angle/Arc-Length space in order to remove the translation, rotation and scaling components. In the new space, the technique of warped matching is used, in order to compensate for shape variations. Salvador and Chan propose FastDTW, an approximation of DTW in linear time and space [49]. Their approach creates multiple resolutions of the compared time series, coarsening them and representing them with fewer points. Then the standard DTW is run over the lowest resolution and the produced wrap path is passed over to the next higher resolution. Finally the path is refined and this process continues until the original resolution of the time series is reached. FastDTW, being a suboptimal approximation of DTW and producing errors up to 19.2%, is faster because the number of cells it evaluates scales linearly with the length of the time-series. Through experimental evaluation, the authors prove the accuracy and efficiency improvements over Sakoe-Chiba bands and data abstraction, which are two other popular DTW approximations [46]. However, the authors do not

report results on multidimensional time series, so the performance of FastDTW when applied on this type of data has to be examined.

15.4.2.3 Multidimensional LCSS

Vlachos et al. proposed two non-metric distance functions as an extension of LCSS for multidimensional time series. The method proved to be robust to noise, especially when compared to DTW and ERP [51]. LCSS does not depend on continuous mapping of the time series, thus the approaches using or extending it tend to focus on the similar parts between the examined sequences. LCSS, in contrast to the Euclidean distance, does not take into account unmatched elements and matches only the similar parts. It therefore allows trajectories to stretch in the time axis. DTW and Euclidean distance try to match every element, so they are more sensitive to outliers. However, when using LCSS the time series under comparison must have the same sampling rates. In [51] the two-dimensional LCSS between two two-dimensional trajectories T_1 and T_2 of length m and n respectively is computed as follows:

(a) $D_{\text{LCSS},\delta,\varepsilon}(T_1, T_2) = 0$, if $n = 0$ or $m = 0$

(c) $D_{\text{LCSS},\delta,\varepsilon}(T_1, T_2) = 1 + D_{\text{LCSS},\delta,\varepsilon}(\text{HEAD}(T_1), \text{HEAD}(T_2))$
if $|T_{1,m,x} - T_{2,n,x}| < \varepsilon$ and $|T_{1,m,y} - T_{2,n,y}| < \varepsilon$ and $|i - j| \leq \delta$

(d) $$\max \begin{cases} D_{\text{LCSS},\delta,\varepsilon}(\text{HEAD}(T_1), T_2) \\ D_{\text{LCSS},\delta,\varepsilon}(T_1, \text{HEAD}(T_2)) \end{cases}, \text{otherwise}$$

where $\text{HEAD}(T_1)$, δ and ε are defined as in 15.4.1.4. Two-dimensional LCSS can easily be extended to d dimensions.

15.4.2.4 Multidimensional Edit Distance

In 2005, Chen et al. proposed EDR, Edit Distance on Real sequence, in order to address the problem of comparing real noisy trajectories with accuracy and robustness, claiming that EDR is more robust and accurate than DTW, LCSS, ERP and Euclidean distance [10]. EDR is defined as the number of insert, delete or replace operations to convert a trajectory T_1 into another T_2. Specifically, applying the Edit Distance on sequences of real numbers, rather than strings as it was originally proposed in [11] by Levenshtein, the authors define EDR as follows:

(a) $\text{EDR}(T_1, T_2) = m$, if $n = 0$

(b) $\text{EDR}(T_1, T_2) = n$, if $m = 0$

(d) $$\text{EDR}(T_1, T_2) = \min \begin{cases} \text{EDR}(\text{Rest}(T_1), \text{Rest}(T_2)) + sc \\ \text{EDR}(\text{Rest}(T_1), T_2) + sc \\ \text{EDR}(T_1, \text{Rest}(T_2)) + sc \end{cases}, \text{otherwise}$$

where $sc = 0$ if T_{11} and T_{21} match, and $sc = 1$ otherwise. Elements T_{1i} and T_{2i} are supposed to match if the distance between them in all dimensions is below a threshold ε, similarly to the way LCSS distance described in 15.4.1.4 and 15.4.2.3 defines matching. This way, EDR manages to cope with noisy multi-variate time series by not being affected by outliers and to handle shifting in the time axis like ERP distance. In their experimental evaluation, Chen et al. prove their claims about the improvements of EDR over DTW, ERP and LCSS when applied on noisy sequences.

15.4.2.5 Multidimensional Subsequence Matching

There are a variety of methods proposed for multidimensional subsequence matching that can be used for data mining tasks such as clustering and classification. SPRING is a dynamic-programming based method that identifies the subsequences of evolving numerical streams that are closest to a query in constant space and linear time in the dataset size [47].

Kotsifakos et al. have introduced SMBGT, a subsequence matching method, that allows for gaps in both the query and the target sequences and constrains the maximum match length between the two [26]. In their study, they apply SMBGT to build a Query-By-Humming system that given a hummed query song, retrieves the top K most similar songs in a database. The proposed similarity measure, SMBGT, given a query Q and a target sequence X, finds the subsequence of X that best matches Q. The experimental evaluation of the proposed similarity measure was performed on 2-dimensional time series of notes of arbitrary length. Given sequences Q and X and their respective subesequences $Q[qs,qe]$ and $X[xs,xe]$ of equal length, SMBGT(Q,X) is defined as follows. Let G_Q and G_X be the indices of $Q[qs,qe]$ and $X[xs,xe]$ in Q and X. If $q_{\pi_i} \approx_\varepsilon x_{\gamma_i}, \forall \pi_i \in G_Q, \forall \gamma_i \in G_X, i = 1,\ldots,|G_Q|$, and

$$\pi_{i+1} - \pi_i - 1 \leq \beta, \gamma_{i+1} - \gamma_i - 1 \leq \alpha, \tag{15.4}$$

then, the pair $\{Q[qs,qe], X[xs,xe]\}$ is a common bounded-gapped subsequence of Q and X. The longest such subsequence with $xs - xe \leq r$ is called SMBGT(Q,X). ε controls tolerance, α and β control allowed gaps in sequences X and Q respectively, and r controls the maximum alignment length. A graphical example of SMBGT distance in two dimensions is illustrated in Figure 15.3.

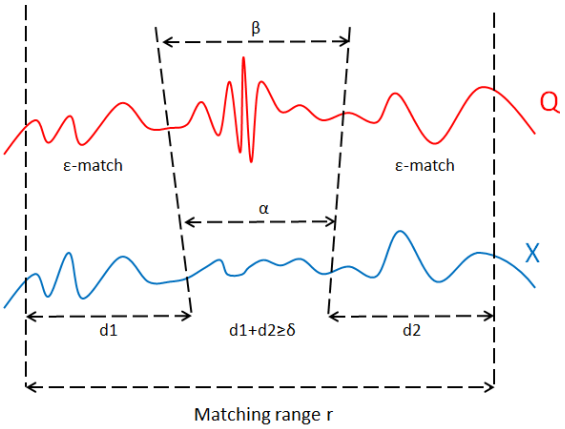

FIGURE 15.3: SMBGT distance between a query sequence Q and a database sequence X

In the experimental evaluation is shown that the main advantage of SMBGT over compared subsequence matching methods (SPRING, Edit distance and DTW) is that it can handle high noise levels better. In some applications, like the studied Query-By-Humming problem, this is extremely important. Other approaches that perform subsequence matching are described in [20, 5, 7].

15.5 Shape-based Time Series Clustering Techniques

In this section we review the major and most widely used techniques for shape-based time series clustering. The two most popular approaches are k-means clustering and hierarchical clustering.

Most techniques either extend or use one of these two clustering methods, thus we classify the reviewed approaches accordingly.

15.5.1 *k*-means Clustering

One of the most widely used clustering techniques is *k*-means clustering and this fact holds for time series data clustering as well. *k*-means is a simple partitioning clustering algorithm, as it groups similar objects in the same cluster and, using an iterative refinement technique, it minimizes an objective error function. A general description of the algorithm is the following:

1. Find *k* initial cluster centers by selecting *k* random objects.
2. Assign each object to the most similar cluster. The most similar cluster is the cluster with the closest center, according to some distance function, e.g. Euclidean, DTW, etc.
3. Recalculate the *k* cluster centers by averaging all the assigned objects for each cluster.
4. Repeat steps 2. and 3. until cluster centers no longer move. The objective error function, which is the sum of squared errors among each cluster center and its assigned objects has been minimized.

The complexity of *k*-means algorithm is $O(k \cdot N \cdot r \cdot D)$, where *k* is the number of desired clusters, *N* is the number of objects to be clustered (which equals the size of the dataset), *r* is the number of iterations until convergence is reached and *D* is the dimensionality of the object space [38]. In a slight modification of *k*-means algorithm, called *k*-medoids clustering, in Step 3, each cluster center is represented by the cluster object that is located nearest to the cluster center. For clustering large datasets of time series, *k*-means and *k*-medoids are preferred over other clustering methods, due to their computational complexity. However, both *k*-means and *k*-medoids require an initial cluster center selection which affects the clustering results, as both are hill-climbing algorithms, converging on a local and not a global optimum. The main disadvantage of *k*-means is that the number *k* of clusters must be specified a priori. This imposes the possibility that the optimal number of clusters for a specific dataset is not known before the clustering process, so *k*-means will produce a sub-optimal clustering result.

Many *k*-means time series clustering approaches use Euclidean distance as a distance metric and a corresponding averaging technique to compute new cluster centers at each step. However, DTW distance is considered a better distance for most time series data mining applications. Until very recently and the work of Meesrikamolkul et al. [39] there was no DTW-based *k*-means clustering approach with satisfying performance. In step 3 of the algorithm an averaging approach is needed in order to calculate the *k* new cluster centers. Unfortunately, DTW averaging produces sequences of equal or greater length than the original ones, thus decreasing a clustering system's accuracy, because the new cluster centers do not preserve the characteristics of the cluster objects. In [39], the authors propose a shape-based *k*-means clustering technique that uses DTW as distance measure and improves the time complexity of DTW averaging. In their approach, called Shape-based Clustering for Time Series (SCTS), they propose a DTW averaging method they call Ranking Shape-based Template Matching Framework (RSTMF), where a cluster center is calculated by averaging a pair of time series with Cubic-spline Dynamic Time Warping (CSDTW) averaging. RSTMF computes an approximate ordering of the sequences before averaging, instead of calculating the DTW distance between all pairs of sequences within each cluster before selecting the most similar pair. DTW is a computationally expensive distance and therefore DTW-based *k*-means clustering using DTW averaging can become also computationally expensive.

Vlachos et al. proposed an anytime variation of the *k*-means algorithm that is based on an initial approximation of the raw data by wavelets [53]. As the process is repeated, the approximation be-

comes finer and the process stops when the approximation resembles the original data or the clustering results do not change. Their approach reduces the running time of original *k*-means and improves the quality of clustering results. Moreover, they demonstrate that time series can be effectively approximated by higher level representations while still preserving their shape characteristics useful to classification or clustering tasks. According to the Liao algorithm classification, the algorithm of Vlachos et al. is placed in the feature-based category, as it operates on a reduced-dimensionality approximation of the original time series using the Haar wavelet basis. In [36] the same team of researchers describe an extension to the work presented in [53], where the same approach is followed, in this work with an Expectation Maximization (EM) method serving the clustering process. EM is rather a soft version of *k*-means than a fundamentally different approach, in the sense that each data object has a degree of membership in each cluster, whereas in *k*-means each object must belong to exactly one cluster. The major difference between EM and *k*-means is that EM produces a richer variety of cluster shapes than *k*-means, which favors spherical clusters.

15.5.2 Hierarchical Clustering

There are two types of hierarchical clustering, *agglomerative* and *divisive*. *Agglomerative* hierarchical clustering starts by regarding each data object as a different cluster and continues by searching the most similar pair of clusters. Then the most similar pair is merged into one cluster and the process continues until the desired number of clusters is reached.

Agglomerative hierarchical clustering has a variety of options for choosing which two clusters are the closest to each other and thus should be merged in the current step. Some of them are listed below:

- Single linkage: In single linkage selection, the distance between two clusters is defined as the shortest distance among all their member objects. Specifically, the single-link distance between clusters C_i and C_j is the following:

$$D_{SL}(C_i, C_j) = \min x \in C_i, y \in C_j(\text{dist}(x,y)) \tag{15.5}$$

  where *dist* is the chosen distance measure.

- Complete linkage: In complete linkage selection, the distance between two clusters is defined as the longest distance among all their member objects. Specifically, the complete-link distance between clusters C_i and C_j is the following:

$$D_{CL}(C_i, C_j) = \max_{x \in C_i, y \in C_j}(\text{dist}(x,y)) \tag{15.6}$$

  where *dist* is the chosen distance measure.

- Average linkage: In average linkage selection, the distance between two clusters is defined as the average distance among all their member objects. Specifically, the average-link distance between clusters C_i and C_j is the following:

$$D_{AV}(C_i, C_j) = \text{avg } x \in C_i, y \in C_j(\text{dist}(x,y)) \tag{15.7}$$

  where *dist* is the chosen distance measure.

Divisive hierarchical clustering is the inverse process of agglomerative hierarchical clustering in that it starts by regarding the whole dataset as a single cluster and continues by recursively dividing it into smaller ones.

Hierarchical clustering has better visualization capabilities than *k*-means, as the clustering process forms a dendrogram. In contrast to *k*-means, it takes no parameters and can be stopped and

traced back at any point to the desired clustering level. Its major drawback is its quadratic computational complexity which makes hierarchical clustering practically useful only for small datasets. As opposed to *k*-means clustering, the hierarchical algorithm is a deterministic algorithm, which means that when applied on the same dataset it provides the same clustering results in every run.

In [19] the authors perform an evaluation of clustering methods applied on a long time series of medical data and implement agglomerative hierarchical clustering. Their experimental results show that complete-linkage cluster selection produces more reasonable clusters and better formed dendrograms, in that the input data sequences are more uniformly distributed in the output clusters. However, the superiority of complete-linkage selection over other methods is not proven nor believed to hold on all datasets and clustering applications.

In [18] the authors propose a hierarchical clustering method followed by a *k*-means fine-tuning process using DTW distance, where the objective function that is minimized is a sum of DTW distances from each object to a prototype of the cluster it belongs to. The cluster prototype can be the DTW average of the objects belonging to the cluster, which has been proven to be inaccurate [42], the cluster medoid or a locally optimal prototype that has been computed with a warping path based local search.

15.5.3 Density-based Clustering

In [13] Ester et al. propose DBSCAN as a way to identify clusters of points utilizing the fact that inter-cluster density is higher than that among points that belong to different clusters. The intuition behind their approach is that objects belonging to a cluster must be surrounded by a minimum number of objects at a small distance, thus defining the notion of neighborhood density. In the DBSCAN algorithm, points that are located in a neighborhood of high density are defined as *core* points, whereas points that do not have a core point in their neighborhood are defined as *noise* points and are discarded. Clusters are formed around *core* points and clusters that are in the same neighborhood are merged.

Ertöz et al. argue that traditional DBSCAN cannot be used effectively in high dimensional data such as time series, because the notion of Euclidean density is meaningless as the number of dimensions increases [12]. Instead, they propose the use of the *k*-nearest neighbor approach to multivariate density estimation, where a point is considered to be in a region with high probability density if it has a lot of highly similar neighbors. Using this notion, they eliminate noise and outliers, by identifying dense clusters in the data.

15.5.4 Trajectory Clustering

While multidimensional time series can be used to represent trajectories[1], another commonly accepted interpretation is that a trajectory is a data type representing the movement of an object. In the past couple of decades, Moving Objects Databases have become a research trend of their own, and various representation methods, storage and indexing techniques, along with spatio-temporal queries processing methodologies have been introduced [17, 61]. Clustering and mining of spatio-temporal trajectories is of interest in various application domains such as traffic management, transportation optimizations, ecological studies of animals motions/migrations, etc.

Vlachos et al. define a trajectory as set of positional information of a moving object ordered by time [54]. Lin and Su [37] disregard the time information in trajectories and focus on the shape information only. They point out that continuous representation of trajectories is usually costly and difficult to compute, so they represent trajectories in terms of line segment sequences. Specifically, each trajectory is defined as an ordered sequence of points $T = [p_1, p_2, ..., p_n]$ that are connected with straight line segments. Alternatively, Chen et al. propose to explicitly incorporate the time-

[1] For that matter, a trajectory can be perceived as a special case of multidimensional time series.

values in the representation – hence, a trajectory S in the two-dimensional plane is defined as a sequence of triples $S = [(t_1; s_{1_x}; s_{1_y}), \ldots, (t_n; s_{n_x}; s_{n_y})]$ [10].

A fair amount of work has been devoted to trajectory clustering, where not only shape but also speed and direction are important. Some methods [44] also discuss the online correlation aspect of trajectory clustering, though these methods are not discussed in detail in this chapter. Globally, the efforts can be grouped in several categories: *relative motion patterns, flocks, convoys, moving clusters and swarms* (cf. [21]). Going in great details about the peculiarities of each category of works is beyond the scope of this chapter, therefore, in the sequel we provide an overview of a few techniques in order to illustrate some specific issues (and solutions) arising in the domain of trajectories clustering.

Lee et al. [30] describe a trajectory clustering approach that belongs to the category of density-based clustering approaches. For a given collection of trajectories, the proposed method can operate on trajectories of different lengths and produce a set of clusters and a representative trajectory for each cluster. The authors argue on the meaningfulness of subtrajectory clustering, therefore the clusters they generate are sets of trajectory partitions. A trajectory partition is a line segment $p_i p_j$, where p_i and p_j ($i < j$) are points from the trajectory. The representative trajectory for each cluster is a trajectory partition that is common to all the trajectories that belong to that cluster. Trajectory partitions that belong to the same cluster have a relatively small distance to each other, according to the respective distance function which operates on line segments and corresponds to the weighted sum of the: *perpendicular distance*, *parallel distance* and *angle distance* (cf. [30]). After partitioning the trajectories, the method proceeds to the line segment clustering process based on DBSCAN (discussed in [13]). The main difference to DBSCAN is that in line segment clustering which is applied here, not all density connected groups of line segments can become clusters, because many line segments can belong to the same trajectory. Thus, each cluster that contains line segments from less than a desired number of trajectories is discarded. For the computation of the representative trajectory for each cluster, the authors use an average direction vector and sweep a vertical line across the line segments in its direction. The complexity of the proposed algorithm is $O(n^2)$ if no spatial index is used, and $O(n \log n)$ otherwise, with n being the total number of line segments that are produced by the partitioning process. In the subsequent work [31], trajectory-based and region-based clustering were combined to build a feature generation framework for trajectory classification. Region-based clustering disregards movement information and clusters regions of trajectories that are mostly of one class. The trajectory-based clustering extends the previous work [30] by using class label information in the clustering process. After trajectory partitioning, region-based clustering is performed and the partitions that cannot be represented by homogeneous regions are the input of the trajectory-based clustering module. The proposed framework generates a hierarchy of features in a top-down approach, namely features produced by region-based clustering do not include movement patterns and are thus of higher level whereas trajectory-clustering features are less general.

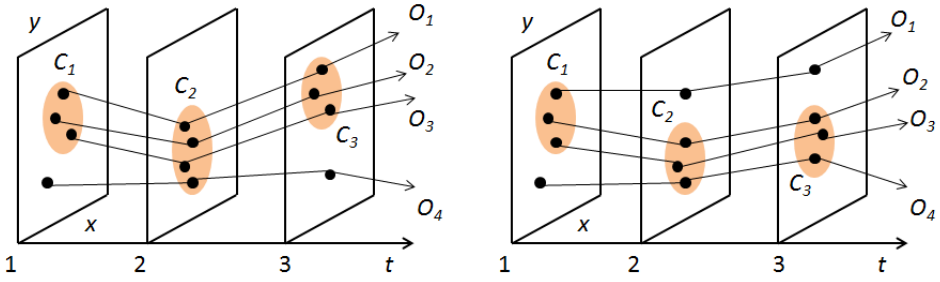

FIGURE 15.4: Trajectories Grouping: Moving Clusters and Convoys

One observation regarding the above approaches is that they do not properly incorporate the tem-

poral dimension of the trajectories (i.e., they work with *routes*). One of the first works that brought the temporal awareness in the realm of clustering spatio-temporal trajectories clustering was [24], which introduced the concept of a *moving cluster* – set of objects that move close to each other for a given time duration. One can think of it as a temporal sequence of spatial clusters, preserving the property that the number of common objects among consecutive clusters is maintained above a certain threshold Θ. An example of a moving cluster with $\Theta \geq 75\%$ (i.e., 3/4 of the objects are within a cluster at each time-instant) is shown in the left portion of Figure 15.4. A concept that uses different criteria for grouping the trajectories is the one of *convoys* [22] – which corresponds to a group that has *at least m* objects who are density-connected with respect to *distance e* and *cardinality m* during *k consecutive time-instants*. For comparison, the right portion of Figure 15.4 shows the formation of a convoy of three trajectories over three consecutive time-instants, illustrating the difference with the moving clusters.

Many other criteria for grouping trajectories have been proposed, generating different corresponding models like, for example, dynamic/evolving convoys, flocks, swarms – and we refer the reader to [21] for a recent survey. We close this section with *data-driven* observations regarding trajectories clustering – given that in the recent years, the GPS traces and sensor-based location data are becoming widely available:

- The sheer volume of the *(location,time)* data may become large-enough to incur high computation cost for clustering trajectories. Hence, often times, some *simplification* techniques may be employed to reduce the size of the data [8], before proceeding with the clustering.
- The data streams are often subject to imprecision in the measurements as well as noise during communication/transmission. Hence, in order to improve the efficiency and effectiveness of the clustering techniques, it may be desirable to apply *trajectory smoothing* techniques as a pre-processing step to their clustering [9].

15.6 Time Series Clustering Applications

Time series clustering is a very interesting domain and has increasingly many applications. The widespread smartphone networks and mobile computing environments as long as the corresponding active communities present a field where multidimensional spatio-temporal trajectory clustering is important and necessary. Location-oriented applications and services can use trajectory clustering techniques to improve query evaluation performance. One example is route recommendations, as touristic guides can benefit from identifying similar user routes to recommend places or tours to users. In Figure 15.5 is illustrated an example of how a route recommendation touristic application would utilize trajectory clustering information to classify a new user and recommend paths to follow.

The online scenario is particularly common in financial markets, machine monitoring and anomaly detection. In fact, since outliers and clusters are connected together by a complementary relationship, multi-variate regression models are often used in order to identify broad trends in the data. Data points which do not match this broad trend are declared as outliers [4].

In another example, mobile social networking applications can avoid controversial privacy concerns by using distributed techniques to identify and use similar trajectories in their network without disclosing the traces, as the SmartTrace system [28]. Such systems provide the functionality of nearest neighbor search, where a user can determine other users that have exposed similar spatio-temporal behavior, like visiting the same places, without knowing the exact trajectories or revealing their trajectory either. Both centralized and distributed approaches have been proposed to evaluate

trajectory similarity queries, while the former serve applications where the transfer of data to the central site is inexpensive and the latter are appropriate for environments with expensive or not always connected mediums, like wireless sensor networks [60] or smartphone networks.

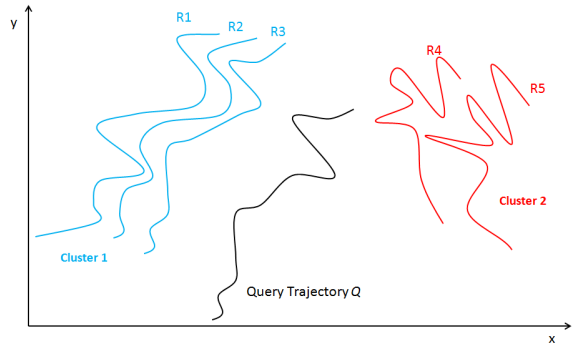

FIGURE 15.5: Query Trajectory Q represents the route of a user of a mobile route-recommendation application. The application performs trajectory clustering and classifies the user to Cluster 1 (containing trajectories $R1$, $R2$ and $R3$), thus it recommends paths similar to that users of Cluster 1 followed.

It becomes apparent that trajectory clustering is suitable for applications where privacy and anonymity are needed. A classic example of such cases is social sensing applications [44].

In another example, video surveillance and tracking systems can largely benefit from trajectory and time series clustering, both due to the insight in scene monitoring that movement clustering provides and to privacy and security issues that have arisen. Abnormal events like pedestrians crossing the street or dangerous vehicle movements can be represented as outliers to clusters of normal movement patterns [23]. In [23] the authors use a hierachical clustering method to overcome the overfitting in HMM trajectory similarity that is often used in surveillance video analysis, where video events are represented as object trajectories.

Automatic counting of pedestrians in detection and tracking systems is another application example. In [3] the authors propose a method to reduce the difference between the number of tracked pedestrians and the real number of individuals, as most detection and tracking systems overestimate the number of targets. The authors apply agglomerative hierarchical trajectory clustering, assuming that trajectories belonging to the same human body are more similar to each other than trajectories produced by the movement of different individuals. In this process they employ different trajectory representation schemes, including time series and independent component analysis representation, and different distance/similarity measures, including Hausdorff Distance and LCSS.

Time series clustering is necessary in a variety of other domains, including music retrieval [26, 32, 40], speech recognition [55, 50] and financial and socio-economic time series applications [25, 15].

15.7 Conclusions

Time series data has diverse formulations because of the variety of applications in which it can be used. The two primary formulations for time series clustering use online and offline analysis. The application domains for these cases are quite different. The online formulation is often used for real time analysis and applications such as financial markets or sensor selection. The online scenario is

also relevant to social sensing applications. The offline scenario is more useful for applications in which the key shapes in the data need to be discovered for diagnostic purposes.

Bibliography

[1] C. Aggarwal, Y. Xie, and P. Yu. On Dynamic Data-driven Selection of Sensor Streams, *KDD Conference*, 2011.

[2] C. Aggarwal, A. Bar-Noy, S. Shamoun. On Sensor Selection in Linked Information Networks, *DCOSS Conference*, 2011.

[3] G. Antonini and J.-P. Thiran. Counting Pedestrians in Video Sequences Using Trajectory Clustering. *IEEE Trans. Circuits Syst. Video Techn*, 16(8): 1008-1020, 2006.

[4] S. Bay, K. Saito, N. Ueda, and P. Langley. A Framework for Discovering Anomalous Regimes in Multivariate Time-series Data with Local Models. *Technical report, Center for the Study of Language and Information*, Stanford University, 2004.

[5] L. Bergroth, H. Hakonen and T. Raita. A Survey of Longest Common Subsequence Algorithms. *SPIRE*, pp. 39–48, 2000.

[6] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. *KDD workshop*, Vol. 10, No. 16, pp. 359-370, 1994.

[7] B. Bollobás, G. Das, D. Gunopulos and H. Mannila. Time-Series Similarity Problems and Well-Separated Geometric Sets. Nord. J. Comput., 8(4): 409–423, 2001.

[8] H. Cao, O. Wolfson and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB J.*,15:3, 2006.

[9] F. Chazal, D. Chen, L. Guibas, X. Jiang and C. Sommer. Data-driven trajectory smoothing. *GIS*, 2011.

[10] L. Chen, M. Tamer Ozsu and V. Oria. Robust and Fast Similarity Search for Moving Object Trajectories. *SIGMOD Conference*, 2005: 491–502.

[11] L. Chen. Similarity Search Over Time Series and Trajectory Data. *PhD Thesis*, 2005.

[12] L. Ertöz, M. Steinbach and V. Kumar. Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. *SDM*, 2003.

[13] M. Ester, H.-P. Kriegel, J. Sander and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *KDD*, pp. 226-231, 1996.

[14] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. *In Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE)*, 816–825, 2007.

[15] T.-C. Fu, C. Law, K. Chan, K. Chung and C. Ng. Stock Time Series Categorization and Clustering Via SB-Tree Optimization. *FSKD*, pp. 1130–1139, 2006.

[16] D. Golovin, M. Faulkner, A. Krause. Online distributed sensor selection. *IPSN Conference*, pp. 220–231, 2010.

[17] R. Güting and M. Schneider. Moving Objects Databases. *Morgan Kaufmann*, 2005.

[18] V. Hautamaki, P. Nykanen, P. Franti. Time-series clustering by approximate prototypes. *ICPR*, pp. 1–4, 2008.

[19] S. Hirano, S. Tsumoto Empirical Comparison of Clustering Methods for Long Time-Series Databases. *Active Mining*, 2003: 268-286.

[20] N. Hu, R. Dannenberg and A. Lewis. A Probabilistic Model of Melodic Similarity. Proceedings of the 2002 International Computer Music Conference., pp. 509-515, 2002.

[21] H. Jeung, M. Yiu and C. Jensen. Trajectory Pattern Mining. *Computing with Spatial Trajectories*, 2011.

[22] H. Jeung, M. Yiu, X. Zhou, C. Jensen and H. Shen. Discovery of convoys in trajectory databases. *PVLDB*, 2008.

[23] F. Jiang, Y. Wu and A. Katsaggelos. Abnormal Event Detection from Surveillance Video by Dynamic Hierarchical Clustering. *ICIP*, 145–148, 2007.

[24] P. Kalnis, N. Mamoulis and S. Bakiras. On Discovering Moving Clusters in Spatio-temporal Data. *SSTD*, pp. 364–381, 2005.

[25] K. Kalpakis, D. Gada, V. Puttagunta. Distance Measures for Effective Clustering of ARIMA Time-Series. *ICDM*, 2001: 273-280.

[26] A. Kotsifakos, P. Papapetrou, J. Hollmen and D. Gunopulos. A Subsequence Matching with Gaps-Range-Tolerances Framework: A Query-By-Humming Application. *PVLDB*, 4(11): 761-771, 2011.

[27] A. Krause, C. Guestrin, Near-optimal observation selection using submodular functions, *AAAI Conference*, pp. 1650–1654, 2007.

[28] C. Laoudias, M. Andreou and D. Gunopulos. Disclosure-Free GPS Trace Search in Smartphone Networks. *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management*, pp 78–87, 2011.

[29] S. Lee, S. Chun, D. Kim, J. Lee, and C. Chung. Similarity search for multidimensional data sequences. *Proceedings of 16th International Conference on. IEEE Data Engineering*, 2000, 599-608.

[30] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. *SIGMOD Conference*, 2007, 593–604.

[31] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering. *VLDB Conference*, 2008.

[32] K. Lemström and E. Ukkonen. Including interval encoding into edit distance based music comparison and retrieval. *AISB*, pp. 53–60, 2000.

[33] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *In Soviet physics doklady*, Vol. 10:707–710, 1966.

[34] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition 38(11): 1857-1874 (2005)*.

[35] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Multi-Resolution K-Means Clustering of Time Series and Applications to Images. *Workshop on Multimedia Data Mining (MDM)*,SIGKDD, Washington DC, 2003.

[36] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative Incremental Clustering of Time Series. *EDBT*, 2004: 106-122.

[37] B. Lin, and J. Su. Shapes based trajectory queries for moving objects. *In Proceedings of the 13th annual ACM international workshop on Geographic information systems*, ACM, 2005:21–30.

[38] J. MacQueen. Some methods for classification and analysis of multi-variate observations. *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob*, 1967, 281-297.

[39] W. Meesrikamolkul, V. Niennattrakul, C. Ratanamahatana. Shape-Based Clustering for Time Series Data. *PAKDD*, 530–541, 2012.

[40] M. Mongeau and D. Sankoff. Comparison of Musical Sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[41] M. Munich, P. Perona. Continuous Dynamic Time Warping for Translation-Invariant Curve Alignment with Applications to Signature Verification. *ICCV*,1999: 108-115.

[42] V. Niennattrakul, C. Ratanamahatana Inaccuracies of Shape Averaging Method Using Dynamic Time Warping for Time Series Data. *International Conference on Computational Science*, 2007: 513-520.

[43] S. Papadimitriou, J. Sun, C, Faloutsos. Dimensionality Reduction and Forecasting of Time-Series Data Streams. *Data Streams: Models and Algorithms, Ed. Charu Aggarwal, Springer*, Chapter 12, pp. 261–288, 2007.

[44] G. Qi, C. Aggarwal, and T. Huang. Online Community Detection in Social Sensing, *WSDM Conference*, 2013.

[45] N. Roussopoulos, S. Kelley and Fr. Vincent. Nearest Neighbor Queries. *SIGMOD Conference*, 71–79, 1995.

[46] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Ac., Sph., and Sig. Proc.*, 26:143-165, 1978.

[47] Y. Sakurai, C. Faloutsos and M. Yamamuro. Stream Monitoring under the Time Warping Distance. *ICDE 2007*, pp. 1046–1055, 2007.

[48] Y. Sakurai, S. Papadimitriou, C. Faloutsos. BRAID: Stream Mining through Group Lag Correlations, *ACM SIGMOD Conference*, pp. 599–610, 2005.

[49] S. Salvador, and P. Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. *In KDD workshop on mining temporal and sequential data*, 70–80, 2004.

[50] D. Tran and M. Wagner. Fuzzy C-Means Clustering-Based Speaker Verification. *Proceedings of the 2002 AFSS International Conference on Fuzzy Systems*, pp. 318–324, 2002.

[51] M. Vlachos, D. Gunopulos, and G. Kollios. Robust similarity measures for mobile object trajectories. *13th International Workshop on Database and Expert Systems Applications*, 2002, 721-726.

[52] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time-Series. *Workshop on Clustering High-Dimensionality Data and its Applications*, SIAM Datamining, San Francisco, 2003.

[53] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh. Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures. *In Proc. of 9th International Conf. on Knowledge Discovery & Data Mining (SIGKDD)*,Washington, DC, 2003.

[54] M. Vlachos, D. Gunopulos, and G. Das. Rotation invariant distance measures for trajectories. *In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004: 707–712.

[55] J. Wilpon and L. Rabiner. A modified k-means clustering algorithm for use in isolated work recognition. *IEEE transactions on acoustics, speech and signal processing*, ASSP, (33):587–594, 1985.

[56] Y. Xiong and D.-Y. Yeung. Mixtures of ARMA Models for Model-Based Time Series Clustering. *ICDM 2002*, pp. 717–720, 2002.

[57] L. Yann-Ael, S. Santini, G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing*, Vol. 87, pp. 3010–3020, 2007.

[58] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. *ICDE Conference*, 2000.

[59] Y. Zhu, and D. Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. *VLDB Conference*, 2002.

[60] D. Zeinalipour-Yazti, S. Lin and D. Gunopulos. Distributed spatio-temporal similarity search. *CIKM*, pp. 14–23, 2006.

[61] Y. Zheng and X. Zhou. Computing with Spatial Trajectories. *Springer*, 2011.