Chapter 1

MINING TIME SERIES DATA

Chotirat Ann Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh
University of California, Riverside

Michail Vlachos
IBM T.J. Watson Research Center

Gautam Das
University of Texas, Arlington

Abstract Much of the world's supply of data is in the form of time series. In the last decade, there has been an explosion of interest in mining time series data. A number of new algorithms have been introduced to classify, cluster, segment, index, discover rules, and detect anomalies/novelties in time series. While these many different techniques used to solve these problems use a multitude of different techniques, they all have one common factor; they require some high level representation of the data, rather than the original raw data. These high level representations are necessary as a feature extraction step, or simply to make the storage, transmission, and computation of massive dataset feasible. A multitude of representations have been proposed in the literature, including spectral transforms, wavelets transforms, piecewise polynomials, eigenfunctions, and symbolic mappings. This chapter gives a high-level survey of time series data mining tasks, with an emphasis on time series representations.

Keywords: Data Mining, Time Series, Representations, Classification, Clustering, Time Series Similarity Measures

1. Introduction

Time series data accounts for an increasingly large fraction of the world's supply of data. A random sample of 4,000 graphics from 15 of the world's

newspapers published from 1974 to 1989 found that more than 75% of all graphics were time series (Tufte, 1983). Given the ubiquity of time series data, and the exponentially growing sizes of databases, there has been recently been an explosion of interest in time series data mining. In the medical domain alone, large volumes of data as diverse as gene expression data (Aach and Church, 2001), electrocardiograms, electroencephalograms, gait analysis and growth development charts are routinely created. Similar remarks apply to industry, entertainment, finance, meteorology and virtually every other field of human endeavour. Although statisticians have worked with time series for more than a century, many of their techniques hold little utility for researchers working with massive time series databases (for reasons discussed below).

Below are the major task considered by the time series data mining community.

- **Indexing** (Query by Content): Given a query time series Q, and some similarity/dissimilarity measure $D(Q, C)$, find the most similar time series in database *DB* (Chakrabarti *et al.*, 2002; Faloutsos *et al.*, 1994; Kahveci and Singh, 2001; Popivanov *et al.*, 2002).

- **Clustering**: Find natural groupings of the time series in database *DB* under some similarity/dissimilarity measure $D(Q, C)$ (Aach and Church, 2001; Debregeas and Hebrail, 1998; Kalpakis *et al.*, 2001; Keogh and Pazzani, 1998).

- **Classification**: Given an unlabeled time series Q, assign it to one of two or more predefined classes (Geurts, 2001; Keogh and Pazzani, 1998).

- **Prediction** (Forecasting): Given a time series Q containing n data points, predict the value at time $n + 1$.

- **Summarization**: Given a time series Q containing n data points where n is an extremely large number, create a (possibly graphic) approximation of Q which retains its essential features but fits on a single page, computer screen, etc. (Indyk *et al.*, 2000; Wijk and Selow, 1999).

- **Anomaly Detection** (Interestingness Detection): Given a time series Q, assumed to be normal, and an unannotated time series R, find all sections of R which contain anomalies or "surprising/interesting/unexpected" occurrences (Guralnik and Srivastava, 1999; Keogh *et al.*, 2002; Shahabi *et al.*, 2000).

- **Segmentation**: (**a**) Given a time series Q containing n data points, construct a model $\bar{Q}$, from K piecewise segments ($K << n$), such that $\bar{Q}$ closely approximates Q (Keogh and Pazzani, 1998). (**b**) Given a time

series Q, partition it into K internally homogenous sections (also known as change detection (Guralnik and Srivastava, 1999)).

Note that indexing and clustering make *explicit* use of a distance measure, and many approaches to classification, prediction, association detection, summarization, and anomaly detection make *implicit* use of a distance measure. We will therefore take the time to consider time series similarity in detail.

2. Time Series Similarity Measures

2.1 Euclidean Distances and L_p Norms

One of the simplest similarity measures for time series is the Euclidean distance measure. Assume that both time sequences are of the same length n, we can view each sequence as a point in n-dimensional Euclidean space, and define the dissimilarity between sequences C and Q and $D(C, Q) = L_p(C, Q)$, i.e. the distance between the two points measured by the L_p norm (when $p = 2$, it reduces to the familiar Euclidean distance). Figure 1.1 shows a visual intuition behind the Euclidean distance metric.

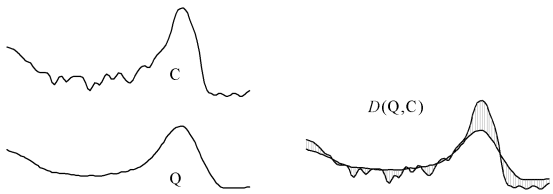

Figure 1.1. The intuition behind the Euclidean distance metric

Such a measure is simple to understand and easy to compute, which has ensured that the Euclidean distance is the most widely used distance measure for similarity search (Agrawal *et al.*, 1993; Chan and Fu, 1999; Faloutsos *et al.*, 1994). However, one major disadvantage is that it is very brittle; it does not allow for a situation where two sequences are alike, but one has been "stretched" or "compressed" in the Y-axis. For example, a time series may fluctuate with small amplitude between 10 and 20, while another may fluctuate in a similar manner with larger amplitude between 20 and 40. The Euclidean distance between the two time series will be large. This problem can be dealt with easily with offset translation and amplitude scaling, which requires normalizing the sequences before applying the distance operator[1].

In Goldin and Kanellakis (1995) , the authors describe a method where the sequences are normalized in an effort to address the disadvantages of the L_p as a similarity measure. Figure 1.2 illustrates the idea.

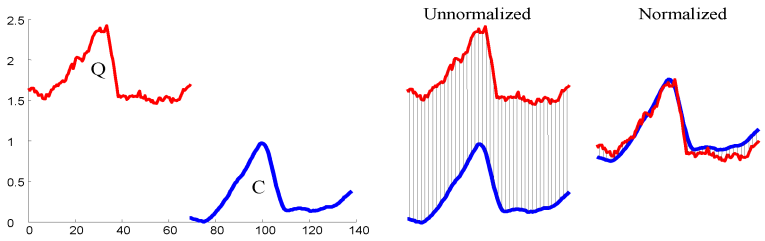

Figure 1.2. A visual intuition of the necessity to normalize time series before measuring the distance between them. The two sequences Q and C appear to have approximately the same shape, but have different offsets in Y-axis. The unnormalized data greatly overstate the subjective dissimilarity distance. Normalizing the data reveals the true similarity of the two time series.

More formally, let $\mu(C)$ and $\sigma(C)$ be the mean and standard deviation of sequence $C = \{c_1, \ldots, c_n\}$. The sequence C is replaced by the normalized sequences C', where

$$c'_i = \frac{c_i - \mu(C)}{\sigma(C)}$$

Even after normalization, the Euclidean distance measure may still be unsuitable for some time series domains since it does not allow for acceleration and deceleration along the time axis. For example, consider the two subjectively very similar sequences shown in Figure 1.3A. Even with normalization, the Euclidean distance will fail to detect the similarity between the two signals. This problem can generally be handled by Dynamic Time Warping distance measure, which will be discussed in the next section.

2.2 Dynamic Time Warping

In some time series domains, a very simple distance measure such as the Euclidean distance will suffice. However, it is often the case that the two sequences have approximately the same overall component shapes, but these shapes do not line up in X-axis. Figure 1.3 shows this with a simple example. In order to find the similarity between such sequences or as a preprocessing step before averaging them, we must "warp" the time axis of one (or both) sequences to achieve a better alignment. Dynamic Time Warping (DTW) is a technique for effectively achieving this warping.

In Berndt and Clifford (1996) , the authors introduce the technique of dynamic time warping to the data mining community. Dynamic time warping is

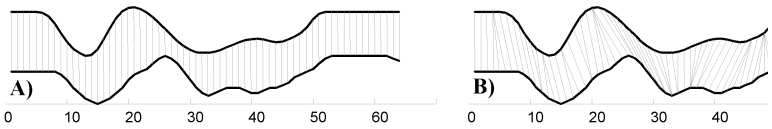

Figure 1.3. Two time series which require a warping measure. Note that while the sequences have an overall similar shape, they are not aligned in the time axis. Euclidean distance, which assumes the i^{th} point on one sequence is aligned with i^{th} point on the other (A), will produce a pessimistic dissimilarity measure. A nonlinear alignment (B) allows a more sophisticated distance measure to be calculated.

an extensively used technique in speech recognition, and allows acceleration-deceleration of signals along the time dimension. We describe the basic idea below.

Consider two sequence (of possibly different lengths), $C = \{c_1, \ldots, c_m\}$ and $Q = \{q_1, \ldots, q_n\}$. When computing the similarity of the two time series using Dynamic Time Warping, we are allowed to extend each sequence by repeating elements.

A straightforward algorithm for computing the Dynamic Time Warping distance between two sequences uses a bottom-up dynamic programming approach, where the smaller sub-problems $D(i, j)$ are first determined, and then used to solve the larger sub-problems, until $D(m, n)$ is finally achieved, as illustrated in Figure 1.4 below.

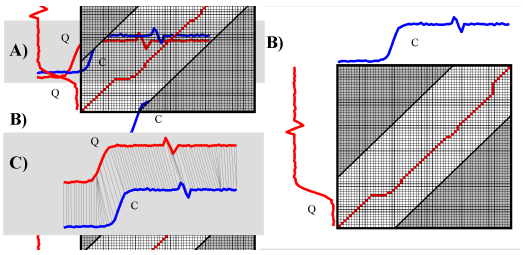

Figure 1.4. A) Two similar sequences Q and C, but out of phase. B) To align the sequences, we construct a warping matrix, and search for the optimal warping path, shown with solid squares. Note that the "corners" of the matrix (shown in dark gray) are excluded from the search path (specified by a warping window of size w) as part of an Adjustment Window condition. C) The resulting alignment

Although this dynamic programming technique is impressive in its ability to discover the optimal of an exponential number alignments, a basic implementation runs in $O(mn)$ time. If a warping window w is specified, as shown

in Figure 1.4B, then the running time reduces to $O(nw)$, which is still too slow for most large scale application. In (Ratanamahatana and Keogh, 2004), the authors introduce a novel framework based on a learned warping window constraint to further improve the classification accuracy, as well as to speed up the DTW calculation by utilizing the lower bounding technique introduced in (Keogh, 2002).

2.3 Longest Common Subsequence Similarity

The longest common subsequence similarity measure, or LCSS, is a variation of edit distance used in speech recognition and text pattern matching. The basic idea is to match two sequences by allowing some elements to be unmatched. The advantage of the LCSS method is that some elements may be unmatched or left out (e.g. outliers), where as in Euclidean and DTW, all elements from both sequences must be used, even the outliers. For a general discussion of string edit distances, see (Kruskal and Sankoff, 1983).

For example, consider two sequences: C = {1,2,3,4,5,1,7} and Q = {2,5,4,5,3,1,8}. The longest common subsequence of C and Q is {2,4,5,1}.

More formally, let C and Q be two sequences of length m and n, respectively. As was done with dynamic time warping, we give a recursive definition of the length of the longest common subsequence of C and Q. Let $L(i, j)$ denote the longest common subsequences $\{c_1, \ldots, c_i\}$ and $\{q_1, \ldots, q_j\}$. $L(i, j)$ may be recursively defined as follows:

if $a_i = b_j$, then
$L(i, j) = 1 + L(i - 1, j - 1)$
else
$L(i, j) = \max\{D(i - 1, j), D(i, j - 1)\}$

We define the dissimilarity between C and Q as

$$LCSS(C, Q) = \frac{m + n - 2l}{m + n}$$

where l is the length of the longest common subsequence. Intuitively, this quantity determines the minimum (normalized) number of elements that should be removed from and inserted into C to transform C to Q. As with dynamic time warping, the LCSS measure can be computed by dynamic programming in $O(mn)$ time. This can be improved to $O((n + m)w)$ time if a matching window of length w is specified (i.e. where $|i - -j|$ is allowed to be at most w).

With time series data, the requirement that the corresponding elements in the common subsequence should match exactly is rather rigid. This problem is addressed by allowing some tolerance (say ε ¿ 0) when comparing elements. Thus, two elements a and b are said to match if $a(1 - \varepsilon)$ ¡ b¡ $a(1 + \varepsilon)$.

In the next two subsections, we discuss approaches that try to incorporate local scaling and global scaling functions in the basic LCSS similarity measure.

2.3.1 Using local Scaling Functions. In (Agrawal *et al.*, 1995), the authors develop a similarity measure that resembles LCSS-like similarity with local scaling functions. Here, we only give an intuitive outline of the complex algorithm; further details may be found in the paper.

The basic idea is that two sequences are similar if they have enough non-overlapping time-ordered pairs of contiguous subsequences that are similar. Two contiguous subsequences are similar if one can be scaled and translated appropriately to approximately resemble the other. The scaling and translation function is local, i.e. it may be different for other pairs of subsequences.

The algorithmic challenge is to determine how and where to cut the original sequences into subsequences so that the overall similarity is minimized. We describe it briefly here (refer to (Agrawal *et al.*, 1995) for further details). The first step is to find all pairs of atomic subsequences in the original sequences A and Q that are similar (atomic implies subsequences of a certain small size, say a parameter w). This step is done by a spatial self-join (using a spatial access structure such as an R-tree) over the set of all atomic subsequences. The next step is to "stitch" similar atomic subsequences to form pairs of larger similar subsequences. The last step is to find a non-overlapping ordering of subsequence matches having the longest match length. The stitching and subsequence ordering steps can be reduced to finding longest paths in a directed acyclic graph, where vertices are pairs of similar subsequences, and a directed edge denotes their ordering along the original sequences.

2.3.2 Using a global scaling function. Instead of different local scaling functions that apply to different portions of the sequences, a simpler approach is to try and incorporate a single global scaling function with the LCSS similarity measure. An obvious method is to first normalize both sequences and then apply LCSS similarity to the normalized sequences. However, the disadvantage of this approach is that the normalization function is derived from all data points, including outliers. This defeats the very objective of the LCSS approach which is to ignore outliers in the similarity calculations.

In (Bollobas *et al.*, 2001), an LCSS-like similarity measure is described that derives a global scaling and translation function that is independent of outliers in the data. The basic idea is that two sequences C and Q are similar if there exists constants a and b, and long common subsequences C' and Q' such that Q' is approximately equal to *aC'* + b. The scale+translation linear function (i.e. the constants a and b) is derived from the subsequences, and not from the original sequences. Thus, outliers cannot taint the scale+translation function.

Although it appears that the number of all linear transformations is infinite, Bollobas *et al.* (2001) shows that the number of different unique linear transformations is $O(n^2)$. A naive implementation would be to compute LCSS on all transformations, which would lead to an algorithm that takes $O(n^3)$ time. Instead, in (Bollobas *et al.*, 2001), an efficient randomized approximation algorithm is proposed to compute this similarity.

2.4 Probabilistic methods

A different approach to time-series similarity is the use of a probabilistic similarity measure. Such measures have been studied in (Ge and Smyth, 2000; Keogh and Smyth, 1997). While previous methods were "distance" based, some of these methods are "model" based. Since time series similarity is inherently a fuzzy problem, probabilistic methods are well suited for handling noise and uncertainty. They are also suitable for handling scaling and offset translations. Finally, they provide the ability to incorporate prior knowledge into the similarity measure. However, it is not clear whether other problems such as time-series indexing, retrieval and clustering can be efficiently accomplished under probabilistic similarity measures.

Here, we briefly describe the approach in (Ge and Smyth, 2000). Given a sequence C, the basic idea is to construct a probabilistic generative model M_C, i.e. a probability distribution on waveforms. Once a model M_C has been constructed for a sequence C, we can compute similarity as follows. Given a new sequence pattern Q, similarity is measured by computing $p(Q|M_C)$, i.e. the likelihood that M_C generates Q.

2.5 General Transformations

Recognizing the importance of the notion of "shape" in similarity computations, an alternate approach was undertaken by Jagadish *et al.* (1995) . In this paper, the authors describe a general similarity framework involving a transformation rules language. Each rule in the transformation language takes an input sequence and produces an output sequence, at a cost that is associated with the rule. The similarity of sequence C to sequence Q is the minimum cost of transforming C to Q by applying a sequence of such rules. The actual rules language is application specific.

3. Time Series Data Mining

The last decade has seen the introduction of hundreds of algorithms to classify, cluster, segment and index time series. In addition, there has been much work on novel problems such as rule extraction, novelty discovery, and dependency detection. This body of work draws on the fields of statistics, machine learning, signal processing, information retrieval, and mathematics. It is in-

teresting to note that with the exception of indexing, researches in the tasks enumerated above predate not only the decade old interest in data mining, but in computing itself. What then, are the essential differences between the classic and the data mining versions of these problems? The key difference is simply one of size and scalability; time series data miners routinely encounter datasets that are gigabytes in size. As a simple motivating example, consider hierarchical clustering. The technique has a long history and well-documented utility. If however, we wish to hierarchically cluster a mere million items, we would need to construct a matrix with 10^{12} cells, well beyond the abilities of the average computer for many years to come. A data mining approach to clustering time series, in contrast, must explicitly consider the scalability of the algorithm (Kalpakis *et al.*, 2001).

In addition to the large volume of data, most classic machine learning and data mining algorithms do not work well on time series data due to their unique structure; it is often the case that each individual time series has a very high dimensionality, high feature correlation, and large amount of noise (Chakrabarti *et al.*, 2002), which present a difficult challenge in time series data mining tasks. Whereas classic algorithms assume relatively low dimensionality (for example, a few measurements such as "height, weight, blood sugar, etc."), time series data mining algorithms must be able to deal with dimensionalities in the hundreds or thousands. The problems created by high dimensional data are more than mere computation time considerations; the very meanings of normally intuitive terms such as "similar to" and "cluster forming" become unclear in high dimensional space. The reason is that as dimensionality increases, all objects become essentially equidistant to each other, and thus classification and clustering lose their meaning. This surprising result is known as the "curse of dimensionality" and has been the subject of extensive research (Aggarwal *et al.*, 2001). The key insight that allows meaningful time series data mining is that although the actual dimensionality may be high, the *intrinsic* dimensionality is typically much lower. For this reason, virtually all time series data mining algorithms avoid operating on the original "raw" data; instead, they consider some higher-level representation or abstraction of the data.

Before giving a full detail on time series representations, we first briefly explore some of the classic time series data mining tasks. While these individual tasks may be combined to obtain more sophisticated data mining applications, we only illustrate their main basic ideas here.

3.1 Classification

Classification is perhaps the most familiar and most popular data mining technique. Examples of classification applications include image and pattern recognition, spam filtering, medical diagnosis, and detecting malfunctions in

industry applications. Classification maps input data into predefined groups. It is often referred to as supervised learning, as the classes are determined prior to examining the data; a set of predefined data is used in training process and learn to recognize patterns of interest. Pattern recognition is a type of classification where an input pattern is classified into one of several classes based on its similarity to these predefined classes. Two most popular methods in time series classification include the Nearest Neighbor classifier and Decision trees. Nearest Neighbor method applies the similarity measures to the object to be classified to determine its best classification based on the existing data that has already been classified. For decision tree, a set of rules are inferred from the training data, and this set of rules is then applied to any new data to be classified. Note that even though decision trees are defined for real data, attempting to apply raw time series data could be a mistake due to its high dimensionality and noise level that would result in deep, bushy tree. Instead, some researchers suggest representing time series as Regression Tree to be used in Decision Tree training (Geurts, 2001).

The performance of classification algorithms is usually evaluated by measuring the accuracy of the classification, by determining the percentage of objects identified as the correct class.

3.2 Indexing (Query by Content)

Query by content in time series databases has emerged as an area of active interest since the classic first paper by Agrawal et al. (1993) . This also includes a sequence matching task which has long been divided into two categories: whole matching and subsequence matching (Faloutsos *et al*., 1994; Keogh *et al*., 2001).

Whole Matching: a query time series is matched against a database of individual time series to identify the ones similar to the query

Subsequence Matching: a short query subsequence time series is matched against longer time series by sliding it along the longer sequence, looking for the best matching location.

While there are literally hundreds of methods proposed for whole sequence matching (See, e.g. (Keogh and Kasetty, 2002) and references therein), in practice, its application is limited to cases where some information about the data is known *a priori*.

Subsequence matching can be generalized to whole matching by dividing sequences into non-overlapping sections by either a specific period or, more arbitrarily, by its shape. For example, we may wish to take a long electrocardiogram and extract the individual heartbeats. This informal idea has been used by many researchers.

Most of the indexing approaches so far use the original GEMINI framework (Faloutsos *et al.*, 1994) but suggest a different approach to the dimensionality reduction stage. There is increasing awareness that for many data mining and information retrieval tasks, very fast approximate search is preferable to slower exact search (Chang *et al.*, 2002). This is particularly true for exploratory purposes and hypotheses testing. Consider the stock market data. While it makes sense to look for approximate patterns, for example, "*a pattern that rapidly decreases after a long plateau*", it seems pedantic to insist on *exact* matches. Next we would like to discuss similarity search in some more detail.

Given a database of sequences, the simplest way to find the closest match to a given query sequence Q, is to perform a *linear* or *sequential* scan of the data. Each sequence is retrieved from disk and its distance to the query Q is calculated according to the pre-selected distance measure. After the query sequence is compared to all the sequences in the database, the one with the smallest distance is returned to the user as the closest match.

This brute-force technique is costly to implement, first because it requires many accesses to the disk and second because it operates or the raw sequences, which can be quite long. Therefore, the performance of linear scan on the raw data is typically very costly.

A more efficient implementation of the linear scan would be to store two levels of approximation of the data; the raw data and their compressed version. Now the linear scan is performed on the compressed sequences and a *lower bound* to the original distance is calculated for all the sequences. The raw data are retrieved in the order suggest by the lower bound approximation of their distance to the query. The smallest distance to the query is updated after each raw sequence is retrieved. The search can be terminated when the lower bound of the currently examined object exceeds the smallest distance discovered so far.

A more efficient way to perform similarity search is to utilize an *index structure* that will cluster similar sequences into the same group, hence providing faster access to the most promising sequences. Using various pruning techniques, indexing structures can avoid examining large parts of the dataset, while still guaranteeing that the results will be identical with the outcome of linear scan. Indexing structures can be divided into two major categories: vector based and metric based.

3.2.1 Vector Based Indexing Structures. Vector based indices work on the compressed data dimensionality. The original sequences are compacted using a dimensionality reduction method, and the resulting multi-dimensional vectors can be grouped into similar clusters using some vector-based indexing technique, as shown in Figure 1.5.

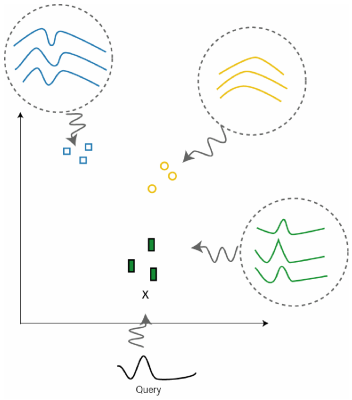

Figure 1.5. Dimensionality reduction of time-series into two dimensions

Vector-based indexing structures can also appear in two flavors; hierarchical or non-hierarchical. The most common hierarchical vector based index is the R-tree or some variants. The R-tree consists of multi-dimensional vector on the leaf levels, which are organized in the tree fashion using hyper-rectangles that can potentially overlap, as illustrated in Figure 1.6.

In order to perform the search using an index structure, the query is also projected in the compressed dimensionality and then probed on the index. Using the R-tree, only neighboring hyper-rectangles to the query's projected location need to be examined.

Other commonly used hierarchical vector-based indices are the kd-B-trees (Robinson, 1981) and the quad-trees (Tufte, 1983). Non-hierarchical vector based structures are less common and are typically known as grid files (Nievergelt *et al*., 1984). For example, grid files have been used in (Zhu and Shasha, 2002) for the discovery of the most correlated data sequences.

However, such types of indexing structures work well only for low compressed dimensionalities (typically¡5). For higher dimensionalities, the pruning power of vector-based indices diminishes exponentially. This can be experimentally and analytically shown and it is coined under the term 'dimensionality curse' (Zhu and Shasha, 2002). This inescapable fact suggests that even when using an index structure, the complete dataset would have to be retrieved from disk for higher compressed dimensionalities.

3.2.2 Metric Based Indexing Structures. Metric based structures can typically perform much better than vector based indices, even for higher dimensionalities (up to 20 or 30). They are more flexible because they require

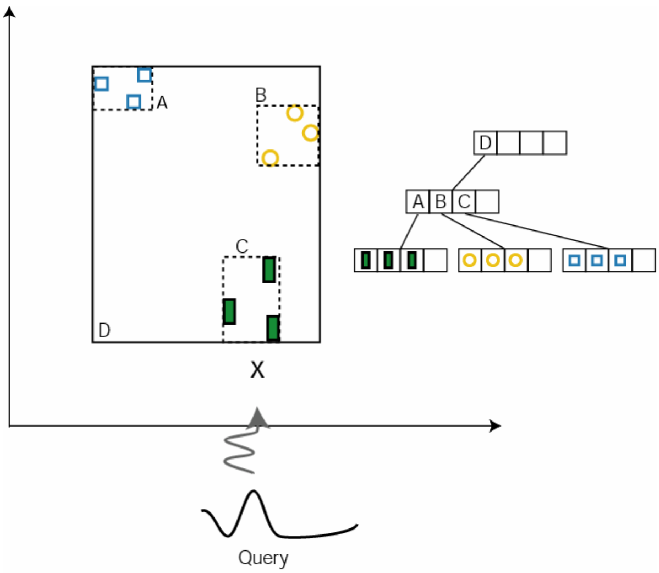

Figure 1.6. Hierarchical organization using an R-tree

only distances between objects. Thus, they do not cluster objects based on their compressed features but based on relative object distances. The choice of reference objects, from which all object distances will be calculated, can vary in different approaches. Examples of metric trees include the Vantage Point (VP) tree (Yianilos, 1992), M-tree (Ciaccia *et al.*, 1997) and GNAT (Brin, 1995). All variations of such trees, exploit the distances to the reference points in conjunction with the triangle inequality to prune parts of the tree, where no closer matches (to the ones already discovered) can be found. A recent use of VP-trees for time-series search under Euclidean distance using compressed Fourier descriptors can be found in (Vlachos *et al.*, 2004).

3.3 Clustering

Clustering is similar to classification that categorizes data into groups; however, these groups are not predefined, but rather defined by the data itself, based on the similarity between time series. It is often referred to as unsupervised learning. The clustering is usually accomplished by determining the similarity among the data on predefined attributes. The most similar data are grouped into clusters, but the clusters themselves should be very dissimilar. And since the clusters are not predefined, a domain expert is often required to interpret the meaning of the created clusters. The two general methods of time series clustering are Partitional Clustering and Hierarchical Clustering. Hierarchical Clustering computes pairwise distance, and then merges similar clusters in a

bottom-up fashion, without the need of providing the number of clusters. We believe that this is one of the best (subjective) tools to data evaluation, by creating a dendrogram of several time series from the domain of interest (Keogh and Pazzani, 1998), as shown in Figure 1.7. However, its application is limited to only small datasets due to its quadratic computational complexity.

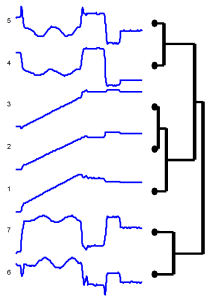

Figure 1.7. A hierarchical clustering of time series

On the other hand, Paritional Clustering typically uses the K-means algorithm (or some variant) to optimize the objective function by minimizing the sum of squared intra-cluster errors. While the algorithm is perhaps the most commonly used clustering algorithm in the literature, one of its shortcomings is the fact that the number of clusters, K, must be pre-specified.

Clustering has been used in many application domains including biology, medicine, anthropology, marketing, and economics. It is also a vital process for condensing and summarizing information, since it can provide a synopsis of the stored data. Similar to query by content, there are two types of time series clustering: whole clustering and subsequence clustering. The notion of whole clustering is similar to that of conventional clustering of discrete objects. Given a set of individual time series data, the objective is to group similar time series into the same cluster. On the other hand, given a single (typically long) time series, subsequence clustering is performed on each individual time series (subsequence) extracted from the long time series with a sliding window. Subsequence clustering is a common pre-processing step for many pattern discovery algorithms, of which the most well-known being the one proposed for time series rule discovery. Recent empirical and theoretical results suggest that subsequence clustering may not be meaningful on an entire dataset (Keogh *et al*., 2003), and that clustering should only be applied to a subset of the data. Some feature extraction algorithm must choose the subset of data, but we cannot use clustering as the feature extraction algorithm, as this would open the possibility of a chicken and egg paradox. Several researchers

have suggested using time series motifs (see below) as the feature extraction algorithm (Chiu *et al.*, 2003).

3.4 Prediction (Forecasting)

Prediction can be viewed as a type of clustering or classification. The difference is that prediction is predicting a future state, rather than a current one. Its applications include obtaining forewarning of natural disasters (flooding, hurricane, snowstorm, etc), epidemics, stock crashes, etc. Many time series prediction applications can be seen in economic domains, where a prediction algorithm typically involves regression analysis. It uses known values of data to predict future values based on historical trends and statistics. For example, with the rise of competitive energy markets, forecasting of electricity has become an essential part of an efficient power system planning and operation. This includes predicting future electricity demands based on historical data and other information, e.g. temperature, pricing, etc. As another example, the sales volume of cellular phone accessories can be forecasted based on the number of cellular phones sold in the past few months. Many techniques have been proposed to increase the accuracy of time series forecast, including the use neural network and dimensionality reduction techniques.

3.5 Summarization

Since time series data can be massively long, a summarization of the data may be useful and necessary. A statistic summarization of the data, such as the mean or other statistical properties can be easily computed even though it might not be particularly valuable or intuitive information. Rather, we can often utilize natural language, visualization, or graphical summarization to extract useful or meaningful information from the data. Anomaly detection and motif discovery (see the next section below) are special cases of summarization where only anomalous/repeating patterns are of interest and reported. Summarization can also be viewed as a special type of clustering problem that maps data into subsets with associated simple (text or graphical) descriptions and provides a higher-level view of the data. This new simpler description of the data is then used in place of the entire dataset. The summarization may be done at multiple granularities and for different dimensions.

Some of popular approaches for visualizing massive time series datasets include *TimeSearcher*, *Calendar-Based Visualization*, *Spiral* and *VizTree*.

TimeSearcher (Hochheiser and Shneiderman, 2001) is a query-by-example time series exploratory and visualization tool that allows user to retrieve time series by creating queries, so called TimeBoxes. Figure 1.8 shows three TimeBoxes being drawn to specify time series that start low, increase, then fall once

more. However, some knowledge about the datasets may be needed in advance and users need to have a general idea of what to look for or what is interesting.

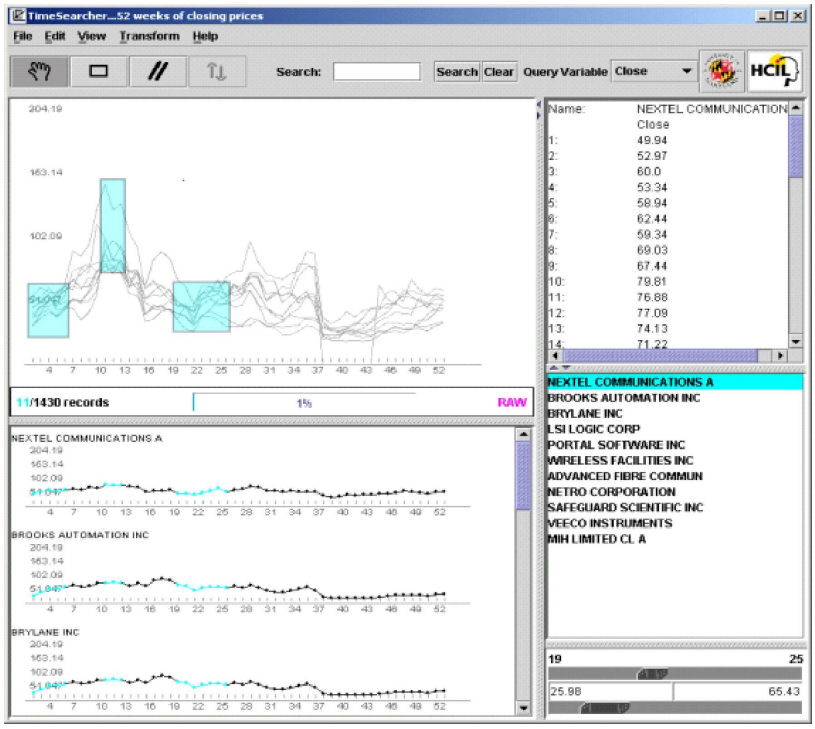

Figure 1.8. The TimeSearcher visual query interface. A user can filter away sequences that are not interesting by insisting that all sequences have at least one data point within the query boxes

Cluster and Calendar-Based Visualization (Wijk and Selow, 1999) is a visualization system that 'chunks' time series data into sequences of day patterns, and these day patterns are clustered using a bottom-up clustering algorithm. The system displays patterns represented by cluster average, along with a calendar with each day color-coded by the cluster it belongs to. Figure 1.9 shows an example view of this visualization scheme. From viewing patterns which are linked to a calendar we can potentially discover simple rules such as: "*In the winter months the power consumption is greater than in summer months*".

Spiral (Weber *et al.*, 2000) maps each periodic section of time series onto one "ring" and attributes such as color and line thickness are used to characterize the data values. The main use of the approach is the identification of

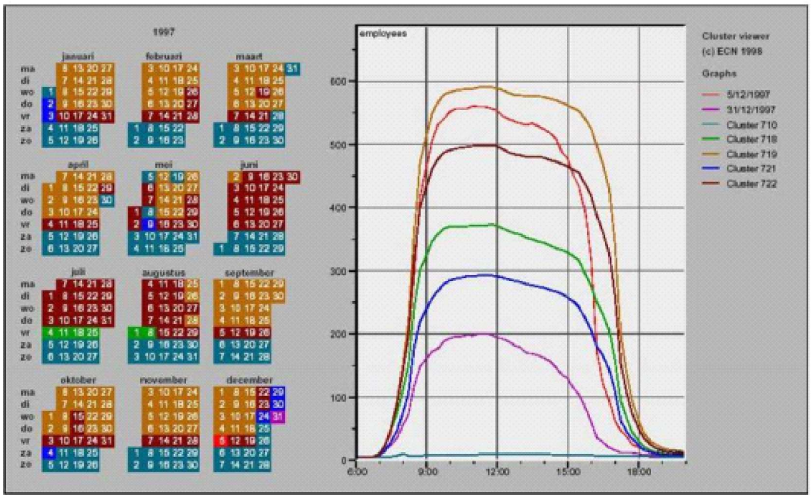

Figure 1.9. The cluster and calendar-based visualization on employee working hours data. It shows six clusters, representing different working-day pattern

periodic structures in the data. Figure 1.10 displays the annual power usage that characterizes the normal "9-to-5" working week pattern. However, the utility of this tool is limited for time series that do not exhibit periodic behaviors, or when the period is unknown.

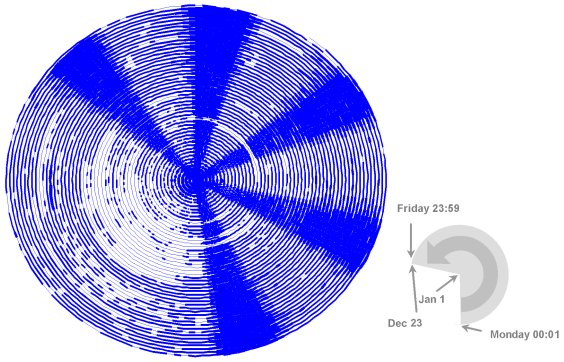

Figure 1.10. The Spiral visualization approach applied to the power usage dataset

VizTree (Lin *et al.*, 2004) is recently introduced with the aim to discover previously *unknown* patterns with little or no knowledge about the data; it provides an overall visual summary, and potentially reveal hidden structures in the

data. This approach first transforms the time series into a symbolic representation, and encodes the data in a modified suffix tree in which the frequency and other properties of patterns are mapped onto colors and other visual properties. Note that even though the tree structure needs the data to be discrete, the original time series data is not. Using a time-series discretization introduced in (Lin *et al.*, 2003), continuous data can be transformed into discrete domain, with certain desirable properties such as lower-bounding distance, dimensionality reduction, etc. While frequently occurring patterns can be detected by thick branches in VizTree, simple anomalous patterns can be detected by unusually thin branches. Figure 1.11 demonstrates both motif discovery and simple anomaly diction on ECG data.

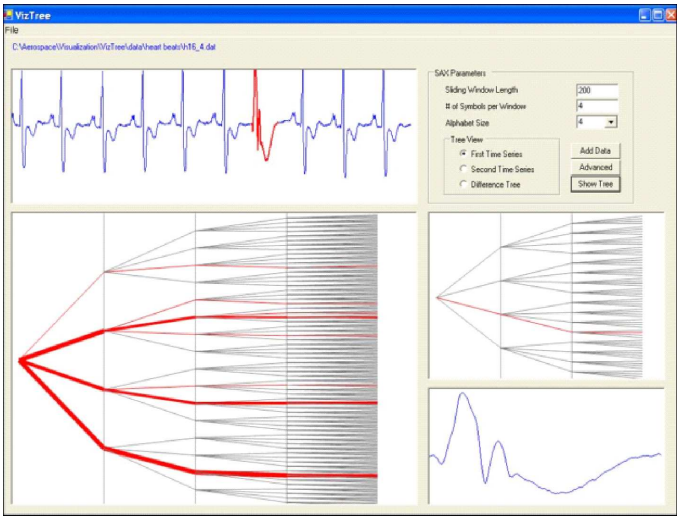

Figure 1.11. ECG data with anomaly is shown. While the subsequence tree can be used to identify motifs, it can be used for simple anomaly detection as well

3.6 Anomaly Detection

In time series data mining and monitoring, the problem of detecting anomalous/surprising/novel patterns has attracted much attention (Dasgupta and Forrest, 1999; Ma and Perkins, 2003; Shahabi *et al.*, 2000). In contrast to subsequence matching, anomaly detection is identification of previously *unknown* patterns. The problem is particularly difficult because what constitutes an anomaly can greatly differ depending on the task at hand. In a general sense, an anomalous behavior is one that deviates from “normal” behavior. While there have been numerous definitions given fro anomalous or surprising behaviors,

the one given by (Keogh *et al.*, 2002) is unique in that it requires no explicit formulation of what is anomalous. Instead, the authors simply define an anomalous pattern as on "*whose frequency of occurrences differs substantially from that expected, given previously seen data*". The problem of anomaly detection in time series has been generalized to include the detection of surprising or interesting patterns (which are not necessarily anomalies). Anomaly detection is closely related to Summarization, as discussed in the previous section. Figure 1.12 illustrates the idea.

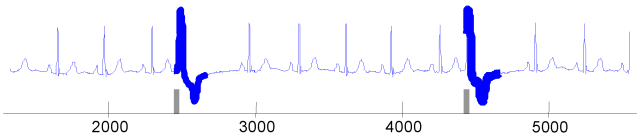

Figure 1.12. An example of anomaly detection from the MIT-BIH Noise Stress Test Database. Here, we show only a subsection containing the two most interesting events detected by the compression-based algorithm (Keogh et al., 2004) (the thicker the line, the more interesting the subsequence). The gray markers are independent annotations by a cardiologist indicating Premature Ventricular Contractions.

3.7 Segmentation

Segmentation in time series is often referred to as a dimensionality reduction algorithm. Although the segments created could be polynomials of an arbitrary degree, the most common representation of the segments is of linear functions. Intuitively, a Piecewise Linear Representation (PLR) refers to the approximation of a time series Q, of length n, with K straight lines. Figure 1.13 contains an example.

Figure 1.13. An example of a time series segmentation with its piecewise linear representation

Because K is typically much smaller than n, this representation makes the storage, transmission, and computation of the data more efficient.

Although appearing under different names and with slightly different implementation details, most time series segmentation algorithms can be grouped into one of the following three categories.

- **Sliding-Windows (SW)**: A segment is grown until it exceeds some error bound. The process repeats with the next data point not included in the newly approximated segment.
- **Top-Down (TD)**: The time series is recursively partitioned until some stopping criteria is met.
- **Bottom-Up (BU)**: Starting from the finest possible approximation, segments are merged until some stopping criteria are met.

We can measure the quality of a segmentation algorithm in several ways, the most obvious of which is to measure the reconstruction error for a fixed number of segments. The reconstruction error is simply the Euclidean distance between the original data and the segmented representation. While most work in this area has consider static cases, recently researchers have consider obtaining and maintaining segmentations on streaming data sources (Palpanas *et al*., 2004)

4. Time Series Representations

As noted in the previous section, time series datasets are typically very large, for example, just eight hours of electroencephalogram data can require in excess of a gigabyte of storage. Rather than analyzing or finding statistical properties on time series data, time series data miners' goal is more towards discovering useful information from the massive amount of data efficiently. This is a problem because for almost all data mining tasks, most of the execution time spent by algorithm is used simply to move data from disk into main memory. This is acknowledged as the major bottleneck in data mining because many naïve algorithms require multiple accesses of the data. As a simple example, imagine we are attempting to do k-means clustering of a dataset that does not fit into main memory. In this case, every iteration of the algorithm will require that data in main memory to be swapped. This will result in an algorithm that is thousands of times slower than the main memory case.

With this in mind, a generic framework for time series data mining has emerged. The basic idea can be summarized as follows.

As with most problems in computer science, the suitable choice of representation/approximation greatly affects the ease and efficiency of time series data mining. It should be clear that the utility of this framework depends heavily on the quality of the approximation created in Step 1). If the approximation is very faithful to the original data, then the solution obtained in main memory is likely to be the same as, or very close to, the solution we would have obtained on the original data. The handful of disk accesses made in Step 2)

Table 1.1. A generic time series data mining approach.

1)	Create an approximation of the data, which will fit in main memory, yet retains the essential features of interest.
2)	Approximately solve the problem at hand in main memory.
3)	Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data.

to confirm or slightly modify the solution will be inconsequential, compared to the number of disks accesses required if we had worked on the original data. With this in mind, there has been a huge interest in approximate representation of time series, and various solutions to the diverse set of problems frequently operate on high-level abstraction of the data, instead of the original data. This includes the Discrete Fourier Transform (DFT) (Agrawal *et al.*, 1993), the Discrete Wavelet Transform (DWT) (Chan and Fu, 1999; Kahveci and Singh, 2001; Wu *et al.*, 2000), Piecewise Linear, and Piecewise Constant models (PAA) (Keogh *et al.*, 2001; Yi and Faloutsos, 2000), Adaptive Piecewise Constant Approximation (APCA) (Keogh *et al.*, 2001), and Singular Value Decomposition (SVD) (Kanth *et al.*, 1998; Keogh *et al.*, 2001; Korn *et al.*, 1997).

Figure 1.14 illustrates a hierarchy of the representations proposed in the literature.

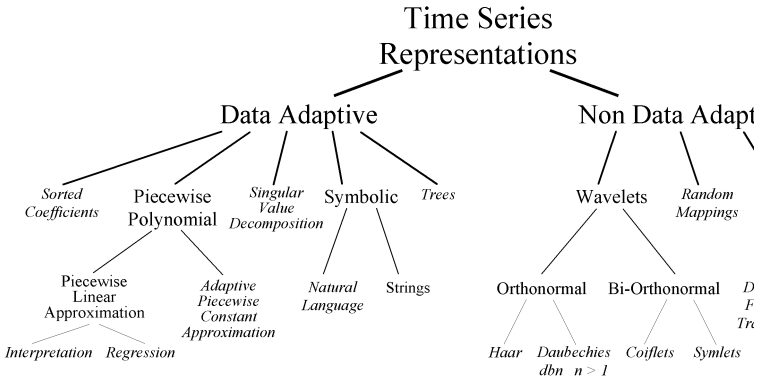

Figure 1.14. A hierarchy of time series representations

It may seem paradoxical that, after all the effort to collect and store the precise values of a time series, the exact values are abandoned for some high level approximation. However, there are two important reasons why this is so.

We are typically not interested in the exact values of each time series data point. Rather, we are interested in the trends, shapes and patterns contained within the data. These may best be captured in some appropriate high-level representation.

As a practical matter, the size of the database may be much larger than we can effectively deal with. In such instances, some transformation to a lower dimensionality representation of the data may allow more efficient storage, transmission, visualization, and computation of the data.

While it is clear no one representation can be superior for all tasks, the plethora of work on mining time series has not produced any insight into how one should choose the best representation for the problem at hand and data of interest. Indeed the literature is not even consistent on nomenclature. For example, one time series representation appears under the names Piecewise Flat Approximation (Faloutsos *et al.*, 1997), Piecewise Constant Approximation (Keogh *et al.*, 2001) and Segmented Means (Yi and Faloutsos, 2000).

To develop the reader's intuition about the various time series representations, we have discussed and illustrated some of the well-known representations in the following subsections below.

4.1 Discrete Fourier Transform

The first technique suggested for dimensionality reduction of time series was the Discrete Fourier Transform (DFT) (Agrawal *et al.*, 1993). The basic idea of spectral decomposition is that any signal, no matter how complex, can be represented by the super position of a finite number of sine/cosine waves, where each wave is represented by a single complex number known as a Fourier coefficient. A time series represented in this way is said to be in the frequency domain. A signal of length n can be decomposed into $n/2$ sine/cosine waves that can be recombined into the original signal. However, many of the Fourier coefficients have very low amplitude and thus contribute little to reconstructed signal. These low amplitude coefficients can be discarded without much loss of information thereby saving storage space.

To perform the dimensionality reduction of a time series C of length n into a reduced feature space of dimensionality N, the Discrete Fourier Transform of C is calculated. The transformed vector of coefficients is truncated at $N/2$. The reason the truncation takes place at $N/2$ and not at N is that each coefficient is a complex number, and therefore we need one dimension each for the imaginary and real parts of the coefficients.

Given this technique to reduce the dimensionality of data from n to N, and the existence of the lower bounding distance measure, we can simply "slot in" the DFT into the GEMINI framework. The time taken to build the entire index

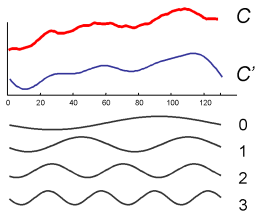

Figure 1.15. A visualization of the DFT dimensionality reduction technique

depends on the length of the queries for which the index is built. When the length is an integral power of two, an efficient algorithm can be employed.

This approach, while initially appealing, does have several drawbacks. None of the implementations presented thus far can guarantee no false dismissals. Also, the user is required to input several parameters, including the size of the alphabet, but it is not obvious how to choose the best (or even reasonable) values for these parameters. Finally, none of the approaches suggested will scale very well to massive data since they require clustering all data objects prior to the discretizing step.

4.2 Discrete Wavelet Transform

Wavelets are mathematical functions that represent data or other functions in terms of the sum and difference of a prototype function, so called the "analyzing" or "mother" wavelet. In this sense, they are similar to DFT. However, one important difference is that wavelets are localized in time, i.e. some of the wavelet coefficients represent small, local subsections of the data being studied. This is in contrast to Fourier coefficients that always represent global contribution to the data. This property is very useful for multiresolution analysis of the data. The first few coefficients contain an overall, coarse approximation of the data; addition coefficients can be imagined as "zooming-in" to areas of high detail, as illustrated in Figure 1.16.

Recently, there has been an explosion of interest in using wavelets for data compression, filtering, analysis, and other areas where Fourier methods have previously been used. Chan and Fu (1999) produces a breakthrough for time series indexing with wavelets by producing a distance measure defined on wavelet coefficients which provably satisfies the lower bounding requirement. The work is based on a simple, but powerful type of wavelet known as the Haar Wavelet. The Discrete Haar Wavelet Transform (DWT) can be calculate efficiently and an entire dataset can be indexed in $O(mn)$.

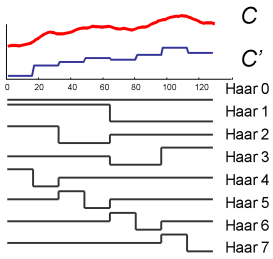

Figure 1.16. A visualization of the DWT dimensionality reduction technique

DTW does have some drawbacks, however. It is only defined for sequence whose length is an integral power of two. Although much work has been undertaken on more flexible distance measures using Haar wavelet (Huhtala *et al.*, 1995; Struzik and Siebes, 1999), none of those techniques are indexable.

4.3 Singular Value Decomposition

Singular Value Decomposition (SVD) has been successfully used for indexing images and other multimedia objects (Kanth *et al.*, 1998; Wu *et al.*, 1996) and has been proposed for time series indexing (Chan and Fu, 1999; Korn *et al.*, 1997).

Singular Value Decomposition is similar to DFT and DWT in that it represents the shape in terms of a linear combination of basis shapes, as shown in 1.17. However, SVD differs from DFT and DWT in one very important aspect. SVD and DWT are local; they examine one data object at a time and apply a transformation. These transformations are completely independent of the rest of the data. In contrast, SVD is a global transformation. The entire dataset is examined and is then rotated such that the first axis has the maximum possible variance, the second axis has the maximum possible variance orthogonal to the first, the third axis has the maximum possible variance orthogonal to the first two, etc. The global nature of the transformation is both a weakness and strength from an indexing point of view.

SVD is the optimal transform in several senses, including the following: if we take the SVD of some dataset, then attempt to reconstruct the data, SVD is the (linear) transform that minimizes reconstruction error (Ripley, 1996). Given this, we should expect SVD to perform very well for the indexing task.

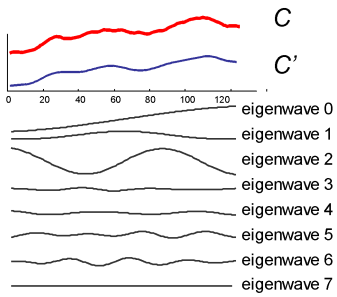

Figure 1.17. A visualization of the SVD dimensionality reduction technique.

4.4 Piecewise Linear Approximation

The idea of using piecewise linear segments to approximate time series dates back to 1970s (Pavlidis and Horowitz, 1974). This representation has numerous advantages, including data compression and noise filtering. There are numerous algorithms available for segmenting time series, many of which were pioneered by (Pavlidis and Horowitz, 1974). Figure 1.18 shows an example of a time series represented by piecewise linear segments.

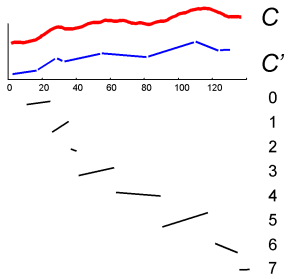

Figure 1.18. A visualization of the PLA dimensionality reduction technique

An open question is how to best choose K, the "optimal" number of segments used to represent a particular time series. This problem involves a trade-off between accuracy and compactness, and clearly has no general solution.

4.5 Piecewise Aggregate Approximation

The recent work (Keogh *et al.*, 2001; Yi and Faloutsos, 2000) (independently) suggest approximating a time series by dividing it into equal-length segments and recording the mean value of the data points that fall within the

segment. The authors use different names for this representation. For clarity here, we refer to it as Piecewise Aggregate Approximation (PAA). This representation reduces the data from n dimensions to N dimensions by dividing the time series into N equi-sized 'frames'. The mean value of the data falling within a frame is calculated, and a vector of these values becomes the data reduced representation. When $N = n$, the transformed representation is identical to the original representation. When $N = 1$, the transformed representation is simply the mean of the original sequence. More generally, the transformation produces a piecewise constant approximation of the original sequence, hence the name, Piecewise Aggregate Approximation (PAA). This representation is also capable of handling queries of variable lengths.

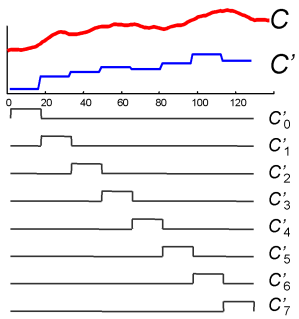

Figure 1.19. A visualization of the PAA dimensionality reduction technique

In order to facilitate comparison of PAA with other dimensionality reduction techniques discussed earlier, it is useful to visualize it as approximating a sequence with a linear combination of box function. Figure 1.19 illustrates this idea.

This simple technique is surprisingly competitive with the more sophisticated transform. In addition, the fact that each segment in PAA is of the same length facilitates indexing of this representation.

4.6 Adaptive Piecewise Constant Approximation

As an extension to the PAA representation, Adaptive Piecewise Constant Approximation (APCA) is introduced (Keogh *et al.*, 2001). This representation allows the segments to have arbitrary lengths, which in turn needs two numbers per segment. The first number records the mean value of all the data points in segment, and the second number records the length of the segment.

It is difficult to make any intuitive guess about the relative performance of the two techniques. On one had, PAA has the advantage of having twice as

many approximating segments. On the other hand, APCA has the advantage of being able to place a single segment in an area of low activity and many segments in areas of high activity. In addition, one has to consider the structure of the data in question. It is possible to construct artificial datasets, where one approach has an arbitrarily large reconstruction error, while the other approach has reconstruction error of zero.

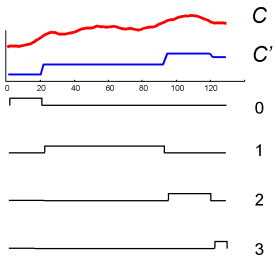

Figure 1.20. A visualization of the APCA dimensionality reduction technique

In general, finding the optimal piecewise polynomial representation of a time series requires a $O(Nn^2)$ dynamic programming algorithm (Faloutsos *et al.*, 1997). For most purposed, however, an optimal representation is not required. Most researchers, therefore, use a greedy suboptimal approach instead (Keogh and Smyth, 1997). In (Keogh *et al.*, 2001), the authors utilize an original algorithm which produces high quality approximations in $O(nlog(n))$. The algorithm works by first converting the problem into a wavelet compression problem, for which there are well-known optimal solutions, then converting the solution back to the APCA representation and (possible) making minor modification.

4.7 Symbolic Aggregate Approximation (SAX)

Symbolic Aggregate Approximation is a novel symbolic representation for time series recently introduced by (Lin *et al.*, 2003), which has been shown to preserve meaningful information from the original data and produce competitive results for classifying and clustering time series.

The basic idea of SAX is to convert the data into a discrete format, with a small alphabet size. In this case, every part of the representation contributes about the same amount of information about the shape of the time series. To convert a time series into symbols, it is first normalized, and two steps of discretization will be performed. First, a time series T of length n is divided into w equal-sized segments; the values in each segment are then approximated and

replaced by a single coefficient, which is their average. Aggregating these w coefficients form the Piecewise Aggregate Approximation (PAA) representation of T. Next, to convert the PAA coefficients to symbols, we determine the breakpoints that divide the distribution space into α equiprobable regions, where α is the alphabet size specified by the user (or it could be determined from the Minimum Description Length). In other words, the breakpoints are determined such that the probability of a segment falling into any of the regions is approximately the same. If the symbols are not equi-probable, some of the substrings would be more probable than others. Consequently, we would inject a probabilistic bias in the process. In (Crochemore *et al.*, 1994), Crochemore et al. show that a suffix tree automation algorithm is optimal if the letters are equiprobable.

Once the breakpoints are determined, each region is assigned a symbol. The PAA coefficients can then be easily mapped to the symbols corresponding to the regions in which they reside. The symbols are assigned in a bottom-up fashion, i.e. the PAA coefficient that falls in the lowest region is converted to "a", in the one above to "b", and so forth. Figure 1.21 shows an example of a time series being converted to string *baabccbc*. Note that the general shape of the time series is still preserved, in spite of the massive amount of dimensionality reduction, and the symbols are equiprobable.

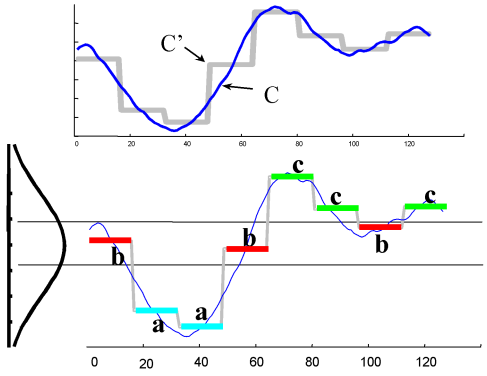

Figure 1.21. A visualization of the SAX dimensionality reduction technique

To reiterate the significance of time series representation, Figure 1.22 illustrates four of the most popular representations.

Given the plethora of different representations, it is natural to ask which is best. Recall that the more faithful the approximation, the less clarification disks

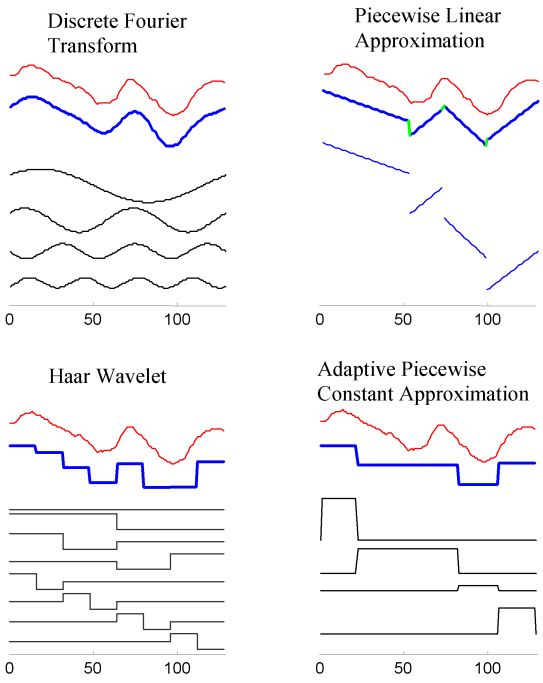

Figure 1.22. Four popular representations of time series. For each graphic, we see a raw time series of length 128. Below it, we see an approximation using 1/8 of the original space. In each case, the representation can be seen as a linear combination of basis functions. For example, the Discrete Fourier representation can be seen as a linear combination of the four sine/cosine waves shown in the bottom of the graphics.

accesses we will need to make in Step 3 of Table 1.1. In the example shown in Figure 1.22, the discrete Fourier approach seems to model the original data the best. However, it is easy to imagine other time series where another approach might work better. There have been many attempts to answer the question of which is the best representation, with proponents advocating their favorite technique (Chakrabarti *et al.*, 2002; Faloutsos *et al.*, 1994; Popivanov *et al.*, 2002; Rafiei *et al.*, 1998). The literature abounds with mutually contradictory statements such as "*Several wavelets outperform the . . . DFT*" (Popivanov *et al.*, 2002), "*DFT-base and DWT-based techniques yield comparable results*" (Wu *et al.*, 2000), "*Haar wavelets perform . . . better than DFT*" (Kahveci and Singh, 2001). However, an extensive empirical comparison on 50 diverse datasets suggests that while some datasets favor a particular approach, overall, there is little difference between the various approaches in terms of their ability to approximate the data (Keogh and Kasetty, 2002). There are however, other important differences in the usability of each approach (Chakrabarti *et al.*, 2002). We will consider some representative examples of strengths and weaknesses below.

The wavelet transform is often touted as an ideal representation for time series data mining, because the first few wavelet coefficients contain information about the overall shape of the sequence while the higher order coefficients contain information about localized trends (Popivanov *et al.*, 2002; Shahabi *et al.*, 2000). This multiresolution property can be exploited by some algorithms, and contrasts with the Fourier representation in which every coefficient represents a contribution to the global trend (Faloutsos *et al.*, 1994; Rafiei *et al.*, 1998). However, wavelets do have several drawbacks as a data mining representation. They are only defined for data whose length is an integer power of two. In contrast, the Piecewise Constant Approximation suggested by (Yi and Faloutsos, 2000), has exactly the fidelity of resolution of as the Haar wavelet, but is defined for arbitrary length time series. In addition, it has several other useful properties such as the ability to support several different distance measures (Yi and Faloutsos, 2000), and the ability to be calculated in an incremental fashion as the data arrives (Chakrabarti *et al.*, 2002). One important feature of all the above representation is that they are real values. This somewhat limits the algorithms, data structures, and definitions available for them. For example, in anomaly detection, we cannot meaningfully define the probability of observing any particular set of wavelet coefficients, since the probability of observing any real number is zero. Such limitations have lead researchers to consider using a symbolic representation of time series (Lin *et al.*, 2003).

5. Summary

In this chapter, we have reviewed some major tasks in time series data mining. Since time series data are typically very large, discovering information from these massive data becomes a challenge, which leads to the enormous research interests in approximating the data in reduced representation. The dimensionality reduction of the data has now become the heart of time series data mining and is the primary step to efficiently deal with data mining tasks for massive data. We review some of important time series representations proposed in the literature. We would like to emphasize that the key step in any successful time series data mining endeavor always lies in choosing the right representation for the task at hand.

Notes

1. In unusual situations, it might be more appropriate not to normalize the data, e.g. when offset and amplitude changes are important.

References

Aach, J. and Church, G. Aligning gene expression time series with time warping algorithms. Bioinformatics; 2001, Volume 17, pp. 495-508.

Aggarwal, C., Hinneburg, A., Keim, D. A. On the surprising behavior of distance metrics in high dimensional space. In proceedings of the 8th International Conference on Database Theory; 2001 Jan 4-6; London, UK, pp 420-434.

Agrawal, R., Faloutsos, C., Swami, A. Efficient Similarity Search in Sequence Data bases. International Conference on Foundations of Data Organization (FODO); 1993.

Agrawal, R., Lin, K.-I., Sawhney, H.S., Shim, K. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Trime-Series Databases. Proceedings of 21^{st} International Conference on Very Large Databases (VLDB); 1995 Sep; Zurich, Switzerland, pp. 490-500.

Berndt, D.J., Clifford, J. Finding Patterns in Time Series: A Dynamic Programming Approach. In Advances in Knowledge Discovery and Data Mining AAAI/MIT Press, Menlo Park, CA, 1996, pp. 229-248.

Bollobas, B., Das, G., Gunopulos, D., Mannila, H. Time-Series Similarity Problems and Well-Separated Geometric Sets. Nordic Jour. of Computing 2001; 4.

Brin, S. Near neighbor search in large metric spaces. Proceedings of 21^{st} VLDB; 1995.

Chakrabarti, K., Keogh, E., Pazzani, M., Mehrotra, S. Locally adaptive dimensionality reduction for indexing large time series databases. ACM Transactions on Database Systems. Volume 27, Issue 2, (June 2002). pp 188-228.

Chan, K., Fu, A.W. Efficient time series matching by wavelets. Proceedings of 15^{th} IEEE International Conference on Data Engineering; 1999 Mar 23-26; Sydney, Australia, pp. 126-133.

Chang, C.L.E., Garcia-Molina, H., Wiederhold, G. Clustering for Approximate Similarity Search in High-Dimensional Spaces. IEEE Transactions on Knowledge and Data Engineering 2002; Jul – Aug, 14(4): 792-808.

Chiu, B.Y., Keogh, E., Lonardi, S. Probabilistic discovery of time series motifs. Proceedings of ACM SIGKDD; 2003, pp. 493-498.

Ciaccia, P., Patella, M., Zezula, P. M-tree: An efficient access method for similarity search in metric spaces. Proceedings of 23^{rd} VLDB; 1997, pp. 426-435.

Crochemore, M., Czumaj, A., Gasjeniec, L, Jarominek, S., Lecroq, T., Plandowski, W., Rytter, W. Speeding up two string-matching algorithms. Algorithmica; 1994; Vol. 12(4/5), pp. 247-267.

Dasgupta, D., Forrest, S. Novelty Detection in Time Series Data Using Ideas from Immunology. Proceedings of 8^{th} International conference on Intelligent Systems; 1999 Jun 24-26; Denver, CO.

Debregeas, A., Hebrail, G. Interactive interpretation of kohonen maps applied to curves. In proceedings of the 4^{th} Int'l Conference of Knowledge Discovery and Data Mining; 1998 Aug 27-31; New York, NY, pp 179-183.

Faloutsos, C., Jagadish, H., Mendelzon, A., Milo, T. A signature technique for similarity-based queries. Proceedings of the International Conference on Compression and Complexity of Sequences; 1997 Jun 11-13; Positano-Salerno, Italy.

Faloutsos, C., Ranganathan, M., Manolopoulos, Y. Fast subsequence matching in time-series databases. In proceedings of the ACM SIGMOD Int'l Conference on Management of Data; 1994 May 25-27; Minneapolis, MN, pp 419-429.

Ge, X., Smyth, P. Deformable Markov Model Templates for Time-Series Pattern Matching. Proceedings of 6^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2000 Aug 20-23; Boston , MA, pp. 81-90.

Geurts, P. Pattern extraction for time series classification. Proceedings of Principles of Data Mining and Knowledge Discovery, 5^{th} European Conference; 2001 Sep 3-5; Freiburg, Germany, pp 115-127.

Goldin, D.Q., Kanellakis, P.C. On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. Proceedings of the 1^{st} International Conference on the Principles and Practice of Constraint Programming; 1995 Sep 19-22; Cassis, France, pp. 137-153.

Guralnik, V., Srivastava, J. Event detection from time series data. In proceedings of the 5th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining; 1999 Aug 15-18; San Diego, CA, pp 33-42.

Huhtala, Y., Karkkainen, J, Toivonen, H. Mining for similarities in aligned time series using wavelet. Data mining and Knowledge Discovery: Theory, Tools, and Technology, SPIE Proceedings Series 1995; Orlando, FL, Vol. 3695, pp. 150-160.

Hochheiser, H., Shneiderman,, B. Interactive Exploration of Time-Sereis Data. Proceedings of 4^{th} International conference on Discovery Science; 2001 Nov 25-28; Washington, DC, pp. 441-446.

Indyk, P., Koudas, N., Muthukrishnan, S. Identifying representative trends in massive time series data sets using sketches. In proceedings of the 26th Int'l Conference on Very Large Data Bases; 2000 Sept 10-14; Cairo, Egypt, pp 363-372.

Jagadish, H.V., Mendelzon, A.O., and Milo, T. Similarity-Based Queries. Proceedings of ACM PODS; 1995 May; San Jose, CA, pp. 36-45.

Kahveci, T., Singh, A. Variable length queries for time series data. In proceedings of the 17th Int'l Conference on Data Engineering; 2001 Apr 2-6; Heidelberg, Germany, pp 273-282.

Kalpakis, K., Gada, D., Puttagunta, V. Distance measures for effective clustering of ARIMA time-series. Proceedings of the IEEE Int'l Conference on Data Mining; 2001 Nov 29-Dec 2; San Jose, CA, pp 273-280.

Kanth, K.V., Agrawal, D., Singh, A. Dimensionality reduction for similarity searching in dynamic databases. Proceedings of ACM SIGMOD International Conference; 1998, pp. 166-176.

Keogh, E. Exact indexing of dynamic time warping. Proceedings of 28^{th} Internation Conference on Very Large Databases; 2002; Hong Kong, pp. 406-417.

Keogh, E., Chakrabarti, K., Mehrotra, S., Pazzani, M. Locally adaptive dimensionality reduction for indexing large time series databases. Proceedings of ACM SIGMOD International Conference; 2001.

Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S. Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems 2001; 3: 263-286.

Keogh, E., Lin, J., Truppel, W. Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research. Proceedings of ICDM; 2003, pp. 115-122.

Keogh, E., Lonardi, S., Chiu, W. Finding Surprising Patterns in a Time Series Database In Linear Time and Space. In the 8^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2002 Jul 23 – 26; Edmonton, Alberta, Canada, pp 550-556.

Keogh, E., Lonardi, S., Ratanamahatana, C.A. Towards Parameter-Free Data Mining. Proceedings of 10^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2004 Aug 22-25; Seattle, WA.

Keogh, E., Pazzani, M. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Proceedings of the 4^{th} Int'l Conference on Knowledge Discovery and Data Mining; 1998 Aug 27-31; New York, NY, pp 239-241.

Keogh, E. and Kasetty, S. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2002 Jul 23 – 26; Edmonton, Alberta, Canada, pp 102-111.

Keogh, E., Smyth, P. A Probabilistic Approach to Fast Pattern matching in Time Series Databases. Proceedings of 3^{rd} International conference on Knowledge Discovery and Data Mining; 1997 Aug 14-17; Newport Beach, CA, pp. 24-30.

Korn, F., Jagadish, H., Faloutsos, C. Efficiently supporting ad hoc queries in large datasets of time sequences. Proceedings of SIGMOD International Conferences 1997; Tucson, AZ, pp. 289-300.

Kruskal, J.B., Sankoff, D., Editors. Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley, 1983.

Lin, J., Keogh, E., Lonardi, S., Chiu, B. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. Workshop on Research Issues in Data Mining and Knowledge Discovery, 8^{th} ACM SIGMOD; 2003 Jun 13; San Diego, CA.

Lin, J., Keogh, E., Lonardi, S., Lankford, J. P., Nystrom, D. M. Visually Mining and Monitoring Massive Time Series. Proceedings of the 10^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2004 Aug 22-25; Seattle, WA.

Ma, J., Perkins, S. Online Novelty Detection on Temporal Sequences. Proceedings of 9^{th} International Conference on Knowledge Discovery and Data Mining; 2003 Aug 24-27; Washington DC.

Nievergelt, H., Hinterberger, H., Sevcik, K.C. The grid file: An adaptable, symmetricmultikey file structure. ACM Trans. Database Systems; 1984; 9(1): 38-71.

Palpanas, T., Vlachose, M, Keogh, E., Gunopulos, D., Truppel, W. Online Amnestic Approximation of Streaming Time Series. Proceedings of 20^{th} International Conference on Data Engineering; 2004, Boston, MA.

Pavlidis, T., Horowitz, S. Segmentation of plane curves. IEEE Transactions on Computers; 1974 August; Vol. C-23(8), pp. 860-870.

Popivanov, I., Miller, R. J. Similarity search over time series data using wavelets. In proceedings of the 18^{th} Int'l Conference on Data Engineering; 2002 Feb 26-Mar 1; San Jose, CA, pp 212-221.

Rafiei, D., Mendelzon, A. O. Efficient retrieval of similar time sequences using DFT. In proceedings of the 5^{th} Int'l Conference on Foundations of Data Organization and Algorithms; 1998 Nov 12-13; Kobe, Japan.

Ratanamahatana, C.A., Keogh, E. Making Time-Series Classification More Accurate Using Learned Constrints. Proceedings of SIAM International Conference on Data Mining; 2004 Apr 22-24; Lake Buena Vista, FL, pp.11-22.

Ripley, B.D. Pattern recognition and neural networks. Cambridge University Press, Cambridge, UK, 1996.

Robinson, J.T. The K-d-b-tree: A search structure for large multidimensional dynamic indexes. Proceedings of ACM SIGMOD; 1981.

Shahabi, C., Tian, X., Zhao, W. TSA-tree: a wavelet based approach to improve the efficiency of multi-level surprise and trend queries. In proceedings of the 12^{th} Int'l Conference on Scientific and Statistical Database Management; 2000 Jul 26-28; Berlin, Germany, pp 55-68.

Struzik, Z., Siebes, A. The Haar wavelet transform in the time series similarity paradigm. Proceedings of 3^{rd} European Conference on Principles and Practice of Knowledge Discovery in Databases; 1999; Prague, Czech Republic, pp. 12-22.

Tufte, E. The visual display of quantitative information. Graphics Press, Cheshire, Connecticut, 1983.

Tzouramanis, T., Vassilakopoulos, M., Manolopoulos, Y. Overlapping Linear Quadtrees: A Spatio-Temporal Access Method. ACM-GIS; 1998, pp. 1-7.

Guralnik, V., Srivastava, J. Event Detection from Time Series Data. Proceedings of ACM SIGKDD; 1999, pp 33-42.

Vlachos, M., Gunopulos, D., Das, G. Rotation Invariant Distance Measures for Trajectories. Proceedings of 10^{th} International Conference on Knowledge Discovery and Data Mining; 2004 Aug 22-25; Seattle, WA.

Vlachos, M., Meek, C., Vagena, Z., Gunopulos, D. Identification of Similarities, Periodicities & Bursts for Online Search Queries. Proceedings of International Conference on Management of Data; 2004; Paris, France.

Weber, M., lexa, M., Muller, W. Visualizing Time Series on Spirals. Proceedings of IEEE Symposium on Information Visualization; 2000 Oct 21-26; San Diego, CA, pp. 7-14.

Wijk, J.J. van, E. van Selow. Cluster and calendar-based visualization of time series data. Proceedings of IEEE Symposium on Information Visualization; 1999 Oct 25-26, IEEE Computer Society, pp 4-9.

Wu, D., Agrawal, D., El Abbadi, A., Singh, A, Smith, T.R. Efficient retrieval for browsing large image databases. Proceedings of 5^{th} International Conference on Knowledge Information; 1996; Rockville, MD, pp. 11-18.

Wu, Y., Agrawal, D., El Abbadi, A. A comparison of DFT and DWT based similarity search in time-series databases. In proceedings of the 9^{th} ACM

CIKM Int'l Conference on Information and Knowledge Management; 2000 Nov 6-11; McLean, VA, pp 488-495.

Yi, B., Faloutsos, C. Fast time sequence indexing for arbitrary lp norms. Proceedings of the 26th Int'l Conference on Very Large Databases; 2000 Sep 10-14; Cairo, Egypt, pp 385-394.

Yianilos, P. Data structures and algorithms for nearest neighbor search in general metric spaces. Proceedings of 3^{rd} SIAM on Discrete Algorithms; 1992.

Zhu, Y., Shasha, D. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time, Proceedings of VLDB; 2002, pp. 358-369.