Similarity Search in Time Series Databases

Maria Kontaki Apostolos N. Papadopoulos Yannis Manolopoulos

Data Enginering Lab, Department of Informatics,
Aristotle University, 54124 Thessaloniki, Greece

INTRODUCTION

In many application domains, data can be represented as a series of values (*time series*). Examples include stocks, seismic signals, audio and many more. Similarity search in time series databases is an important research direction. Several methods have been proposed in order to provide algorithms for efficient query processing in the case of static time series of fixed length. Research in this field has focused on the development of effective transformation techniques, the application of dimensionality reduction methods and the design of efficient indexing schemes. These tools enable the process of *similarity queries* in time series databases. In the case where time series are continuously updated with new values (*streaming time series*), the similarity problem becomes even more difficult to solve, since we must take into consideration the new values of the series. The dynamic nature of streaming time series make the methods proposed for the static case inappropriate. To attack the problem, significant research has been performed towards the development of effective and efficient methods for streaming time series processing. In this article, we introduce the most important issues concerning similarity search in static and streaming time series databases, presenting fundamental concepts and techniques that have been proposed by the research community.

BACKGROUND

Time series are used in a broad range of applications, modeling data that change over time. For example, stock changes, audio signals, seismic signals, electrocardiograms, can be represented as time series data. In fact, any measurement that changes over time can be represented as a time series. Two simple time series examples are depicted in Figure 1.

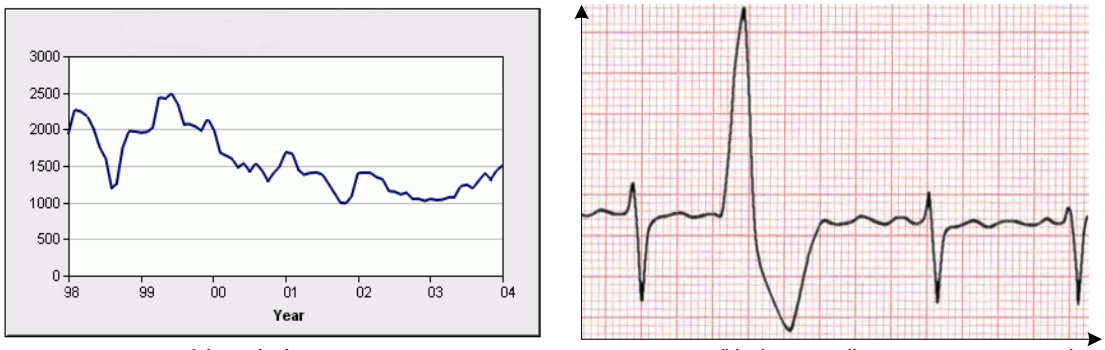


Figure 1: *Examples of time series data.*

We differentiate between two types of time series, namely: 1) *static time series*, and 2) *streaming time series*. In the first case, we assume that the time series is composed of a finite number of sample values, whereas in the second case the size of the series is increasing since new values are appended. For example, if the data correspond to stock prices for the year 2004, then we can use static time series to capture the stock prices of the time period of interest. On the other hand, if there is a need for continuous stock monitoring as time progresses, then streaming time series are more appropriate.

Streaming time series is a special case of streaming data, which nowadays are considered very important and there is an increasing research interest in the area. Traditional database methods can not be applied directly to data streams. Therefore, new techniques and algorithms are required in order to guarantee efficient and effective query processing in terms of the CPU time and the number of disk accesses. The most important difficulty that these techniques must address is the continuous change, which poses serious restrictions.

The purpose of a time series database is to organize the time series in such a way that user queries can be answered efficiently. Although user queries may vary according to the application, there are some fundamental query types that are supported:

- *whole-match queries*, where all time series have the same length, and
- *subsequence-match queries*, where the user's time series is smaller than the time series in the database, and therefore we are interested in time series which contain the user's time series.

In contrast to traditional database systems, time series databases may contain erroneous or noisy data. This means that the probability that two time series have exactly the same values in the same time instances is very small. In such a case, *exact search* is not very useful, and therefore *similarity search* is more appropriate. There are three basic types of similarity queries:

- *similarity range query*: given a user time series Q and a distance e, this query retrieves all time series that are within distance e from Q.
- *similarity nearest-neighbor query*: given a user time series Q and a integer k, this query retrieves the k series that are closer to Q.

- *similarity join query*: given two sets of times series U,V and a distance e, this query retrieves all pairs (u,v) $u \in U$ and $v \in V$ such that the distance between u and v is less or equal to e.

It is evident from the above definitions, that in order to express similarity between two time series objects, a distance measure D is required. This distance measure usually ranges between 0 and 1. If two time series u and v are similar, then the value $D(u,v)$ should be close to 1, whereas if they are dissimilar then $D(u,v)$ should be close to 0. Similarity search can be applied for whole-match queries and subsequence-match queries as well, for static or streaming time series.

SIMILARITY SEARCH IN TIME SERIES

We begin our study with methods proposed for static time series. Streaming time series are considered later in this section. The efficient processing of similarity queries requires the addressing of the following important issues:

- the definition of a meaningful distance measure D in order to express the similarity between two time series objects,
- the efficient representation of time series data, and
- the application of an appropriate indexing scheme in order to quickly discard database objects that can not contribute to the final answer.

Assuming that each time series has a length of m, then it is natural to think that each time series is represented as a vector in the m-dimensional space. In such a case, the similarity between two time series u and v can be expressed as the Euclidean distance:

$$D(u,v) = \sqrt{\sum_{i=1}^{m} (u[i] - v[i])^2}$$

where $u[i]$, $v[i]$ is the value of u and v for the i-th time instance. The Euclidean distance has been widely used as a similarity measure in time series literature (Agrawal 1993, Faloutsos 1994, Chan 1999, Kontaki 2004), because of its simplicity.

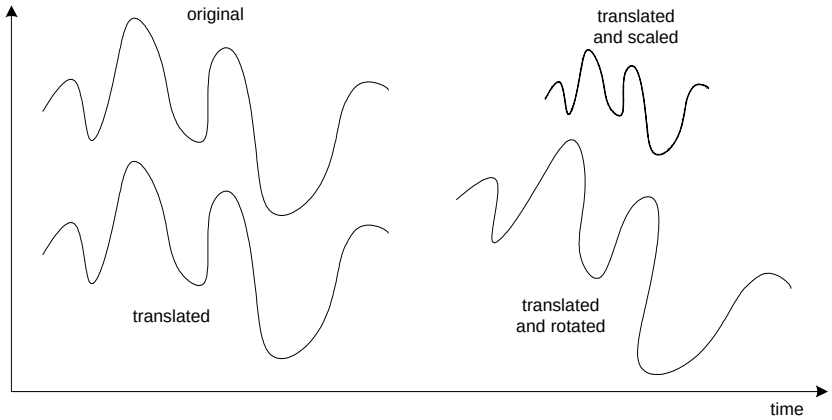


Figure 2: *Examples of translation, rotation and scaling of time series.*

Several alternative distance functions have been proposed in order to allow translation, rotation and scaling invariance. Consider for example the time series depicted in Figure 2. Note that although all time series have the same shape, they will be considered different if the Euclidean distance is used to express similarity. Translation, rotation and scaling invariance is studied in (Agrawal 1995, Yi 1998, Chan 1999, Yi 2000).

The main shortcoming of the Euclidean distance is that all time series must have equal length, which is a significant restriction. If time series are sampled using different time intervals, then their length will not be the same, and therefore the Euclidean distance can not be applied. In order to express similarity between time series of different lengths, other more sophisticated distance measures have been proposed (Yi 1998, Park 2000). One such distance measure is *Time Warping* (TW) that allows time series to be stretched along the time axis. The time warping distance maps each element of a time series *u* to one or more elements of another time series *v*. Given two time series *u* and *v* the time warping distance $D_{TW}(u,v)$ is defined as follows (several variations have been proposed):

$$D_{TW}(u,v) = |u[1]-v[1]| + \min\begin{cases} D_{TW}(u, v[2:*]) \\ D_{TW}(u[2:*], v) \\ D_{TW}(u[2:*], v[2:*]) \end{cases}$$

where *u*[2:*] and *v*[2:*] denote the suffix of *u* and *v* respectively. The TW distance has been used extensively in pattern matching, for voice, audio and other types of signals (e.g., electrocardiograms). The computation cost of the TW is much higher than that of the Euclidean distance.

Many string similarity functions as the Edit Distance (Bozkaya 1997) and the Longest Common Subsequence have been modified in order to match similar time series. These similarity functions can be used in time series with different lengths or different sampling rates because some elements may be unmatched.

The number of samples of each time series may range from a few to hundreds or thousands. Therefore, representing each time series as an *m*-dimensional vector may result in performance degradation during query processing, due to high computation costs of the distance function. It is preferable to compute the distance as efficiently as possible, towards increased processing performance. To attack this problem, *dimensionality reduction* is applied to the time series, in order to transform them to a more manageable representation. Figure 3 illustrates an example.

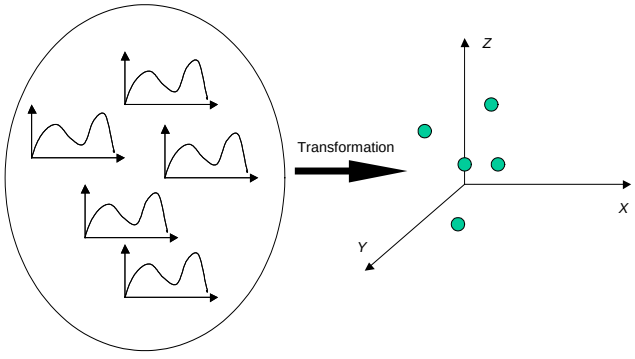


Figure 3: *Applying dimensionality reduction using transformation.*

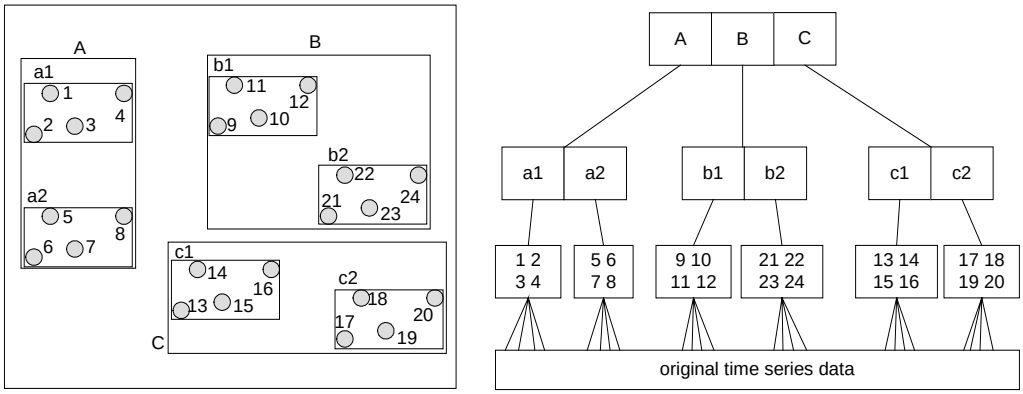
One of the first dimensionality reduction techniques applied to time series is the *Discrete Fourier Transform* (DFT), which transforms a time series to the frequency domain. For many real time series (e.g. stock changes) it is observed that most of the information is concentrated in the first few DFT coefficients. Therefore, each time series can be represented by using only a few real numbers, instead of using all the values in the time domain. DFT has been successfully used for whole-match and subsequence-match in static or streaming time series (Agrawal 1993, Faloutsos 1994, Yi 2000, Kontaki 2004). Among other dimensionality reduction techniques we note the *Singular Value Decomposition* (SVD), the *Discrete Wavelet Transform* (DWT), and *FastMap* which have been successfully applied in time series databases.

Note that dimensionality reduction is a lossy operation, since after the transformation of the time series some information is lost. This means that the transformed data is an approximation of the original data, and the latter must be retained in order to be available for reference. The original and the transformed data are used for query processing by means of the *filter-refinement* processing technique:

- During the *filter step*, the approximations (transformed data) are investigated in order to quickly discard time series that can not contribute to the final answer. The result of this step is a set of *candidate* time series that may be part of the final answer.
- Candidates are investigated further in the *refinement step*, by accessing the original time series in the time domain. Candidate objects that do not satisfy query constraints are characterized as *false alarms*, and are discarded.

It is important that the filter step be very efficient with respect to the processing performance and the number of candidates determined. The number of candidates depends on the transformation technique used to produce the approximations, whereas the processing performance depends heavily on the *indexing scheme* applied. If the time series approximations are vectors in a multi-dimensional space, then *spatial access methods* can be used to organize data hierarchically. One of the most influential spatial access methods is the R-tree (Guttman 1984) and the R*-tree (Bechmann 1990), which is one of its successful variations. The multi-dimensional approximations are organized in an efficient way, in order to speed-up query processing. The R*-tree manages to discard quickly a large portion of the database, and therefore helps significantly in the efficient processing of the filter step. Efficient algorithms have been proposed for similarity range search, similarity nearest-neighbor search and similarity join. An R*-tree example for a 2-dimensional point dataset is illustrated in Figure 4.

If the dimensionality of the transformed data is still large after the application of the dimensionality reduction method, then indexing schemes for high-dimensional data can be used. These schemes are influenced by the R*-tree access method and they use several optimization techniques in order to attack the dimensionality curse problem. Among these sophisticated access methods we highlight the TV-tree (Lin 1995) and the X-tree (Berchtold 1996).



(a) 2-dimensional approximations (b) R-tree example

Figure 4: *An R*-tree example for 2-dimensional approximations.*

Although a significant amount of research work has been performed for static time series, the field of streaming time series is quite immature, since the data stream model has been recently introduced (Babou 2001, Babcock 2002, Gilbert 2003). The difficulty in a streaming time series database is that new values for the time series are continuously arrive. This characteristic yields existing techniques for static time series inefficient, because the index must be updated continuously for every new value.

Similarity queries in streaming time series have been studied in (Gao 2002) where whole-match queries are investigated by using the Euclidean distance as the similarity measure. A prediction-based approach is used for query processing. The distances between the query and each data stream are calculated using the predicted values. When the actual values of the query are available, the upper and lower bound of the prediction error are calculated and the candidate set is formed using the predicted distances. Then, false alarms are discarded. The same authors have proposed two different approaches, based on pre-fetching (Gao 2002b, Gao 2002c).

The aforementioned research efforts examine the case of whole-match queries, where the data are static time series and the query is a streaming time series. In (Liu 2003) the authors present a method for query processing in streaming time series where both the query object and the data are streaming time series. The VA-stream and VA+-stream access methods have been proposed, which are variations of the VA-file (Weber 1998). These structures are able to generate a summarization of the data and enable the incremental update of the structure every time a new value arrives. The performance of this approach is highly dependent on the number of bits associated with each dimension.

In (Kontaki 2004) a different approach is followed in order to provide a flexible technique for similarity range queries when both data and queries are streaming time series. The proposed method (IDC-Index) is based on the R-tree which is used as the indexing scheme for the underlying time series approximations. The dimensionality reduction technique applied to the original time series is based on an incremental computation of the DFT which avoids re-computation. Moreover, the R-tree is equipped by a deferred update policy in order to avoid index adjustments every time a new value for a streaming time series is available. Experiments performed on synthetic random walk time series

and on real time series data have shown that the proposed approach is very efficient in comparison to previously proposed methods.

FUTURE TRENDS

The research interest in the last years has focused to the streaming time series. Apart from the investigation of more efficient techniques for similarity search, there is significant work performed towards *data mining* of streaming data. The challenge is to overcome the difficulty of continuous data change, and apply *clustering* algorithms to streaming time series. Some interesting results have been reported (Guha 2003).

Another important research direction is the management of *continuous queries* over streaming time series. In this case users pose queries that must be continuously evaluated for a time interval. Therefore, when a new value for a time series arrives, the value must be used to determine which queries are satisfied. Continuous queries over data streams are studied in (Chandrasekaran 2002, Gao 2002, Gao 2002b, Gao 2002c, Babcock 2002).

So far we have focused on 1-dimensional time series, since there is only one measurement that changes over time. However, there are applications that require the manipulation of multi-dimensional time series. For example, consider an object that changes location and we are interested in tracking its position. Assuming that the object moves in the 2-dimensional space, there are two values (x and y coordinates) that change over time. Some interesting research proposals for multi-dimensional time series can be found in (Vlachos 2003).

CONCLUSION

Time series data are used to model values that change over time, and they are successfully applied to diverse fields such as online stock analysis, computer network monitoring, network traffic management, seismic wave analysis. In order to manipulate time series effectively and efficiently, sophisticated processing tools are required from the database viewpoint.

Time series are categorized as static or streaming. In static time series each sequence has a static length, whereas in the streaming case new values are continuously appended. This dynamic characteristic of streaming time series poses significant difficulties and challenges in storage and query processing.

A fundamental operation in a time series database system is the processing of similarity queries. To achieve this goal, there is a need for a distance function, an appropriate representation and an efficient indexing scheme. By using the filter-refinement processing technique, similarity range queries, similarity nearest-neighbor queries and similarity join queries can be answered very efficiently.

REFERENCES

Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient Similarity Search In Sequence Databases'. *Proceedings of FODO*, 69-84, Evanston, Illinois, USA.

Agrawal, R., Lin, K.-I., Sawhney, H.S. and Swim, K. (1995). Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. *Proceedings of VLDB*, Zurich, Switzerland.

Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. (2002). Models and Issues in Data Stream Systems. *Proceedings of ACM PODS*, 1-16, Madison, Wisconsin.

Berchtold, S., Keim, D., and Kriegel H.-P. (1996). The X-tree: An Index Structure for High-Dimensional Data. *Proceedings of VLDB*, Bombay, India.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *Proceedings of ACM SIGMOD*, 322-331, Atlantic City, NJ.

Babu, S. and Widom, J. (2001). Continuous Queries over Data Streams. *SIGMOD Record*, 30(3), 109-120.

Bozkaya, T., Yazdani, N., and Ozsoyoglu, M. (1997). Matching and Indexing Sequences of Different Lengths. *Proceedings of CIKM*, Las Vegas, NV, USA.

Chan, K. and Fu, A. W. (1999). Efficient Time Series Matching by Wavelets. *Proceedings of IEEE ICDE*, 126-133.

Chandrasekaran, S., and Franklin, M.J. (2002). Streaming Queries over Streaming Data. *Proceedings of VLDB*, Hong Kong, China.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y.(1994). Fast Subsequence Matching in Time-Series Databases. *Proceedings of ACM SIGMOD*, 419-429, Minneapolis, Minnesota, USA.

Gao, L. and Wang, X. S.(2002). Continually Evaluating Similarity-Based Pattern Queries on a Streaming Time Series. *Proceedings of ACM SIGMOD*, Madison, Wisconsin.

Gao, L.and Wang, X. S.(2002b). Improving the Performance of Continuous Queries on Fast Data Streams: time series case. *Proceedings of SIGMOD/DMKD* Workshop, Madison, Wisconsin.

Gao, L., Yao, Z. and Wang, X. S.(2002c). Evaluating Continuous Nearest Neighbor Queries for Streaming Time Series via Pre-fetching. *Proceedings of VLDB*, Hong Kong, China.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S. and Strauss, M. J. (2003). One-Pass Wavelet Decompositions of Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3), 541-554.

Guha, S., Meyerson, A., Mishra, N., Motwani, R. and O'Callaghan, L. (2003). Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3), 515-528.

Keogh, E. J. and Pazzani, M. J. (1999). An Indexing Scheme for Fast Similarity Search in Large Time Series Databases. Proceedings of SSDBM, Clevelant, Ohio.

Kontaki, M. and Papadopoulos, A. N. (2004). Similarity Search in Streaming Time Sequences. *Proceedings of SSDBM (to appear)*, Santorini, Greece.

Lin, K., Jagadish, H. V. and Faloutsos, C. (1995). The TV-tree: An Index Structure for High Dimensional Data. *The VLDB Journal*, 3, 517-542.

Liu, X., and Ferhatosmanoglu, H. (2003). Efficient k-NN Search on Streaming Data Series. *Proceedings of SSTD*, Santorini, Greece.

Park, S., Chu, W. W., Yoon, J. and Hsu, C. (2000). Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases. *Proceedings of IEEE ICDE*.

Vlachos, M., Hatjieleftheriou, M., Gunopoulos, D. and Keogh, E. (2003). Indexing Multidimensional Time Series with Support for Multiple Distance Measures. Proceedings of ACM SIGKDD, Washington, DC, USA.

Weber, R., Schek, H.-J. and Blott, S. (1998). A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. Proceedings of VLDB, 194-205, New York City, New York.

Yi, B.-K. and Faloutsos, C. (2000). Fast Time Sequence Indexing for Arbitrary Lp Norms. *Proceedings of VLDB*, Cairo, Egypt.

Yi, B.-K., Jagadish, H. V. and Faloutsos, C. (1998). Efficient Retrieval of Similar Time Sequences Under Time Wraping. *Proceedings of IEEE ICDE*. 201-208, Orlando, Florida.

Terms and Definitions

Time Series: It is composed of a sequence of values, where each value corresponds to a time instance. The length remains constant.

Streaming Time Series: It is composed of a sequence of values, where each value corresponds to a time instance. The length changes, since new values are appended.

Distance Function: It is used to express the similarity between two objects. It is usually normalized in the range between 0 to 1. Examples of distance functions used for time series data are the Euclidean distance and the Time Warping distance.

Dimensionality Reduction: It is a technique that is used to lower the dimensionality of the original dataset. Each object is transformed to another object which is described by less

information. It is very useful for indexing purposes, since it increases the speed of the filtering step.

Similarity Queries: These are queries that retrieve objects which are similar to a query object. There are three basic similarity query types, namely, similarity range, similarity nearest-neighbor and similarity join.

Filter-Refinement Processing: A technique used in query processing, which is composed of the filter step and the refinement step. The filter step discards parts of the database that can not contribute to the final answer, and determines a set of candidate objects, which are then processed by the refinement step. Filtering is usually enhanced by efficient indexing schemes for improved performance.

Data Mining: A research field which investigates the extraction of useful knowledge from large datasets. Clustering and association rule mining are two examples of data mining techniques.