TOWARDS A FASTER SYMBOLIC AGGREGATE APPROXIMATION METHOD

Muhammad Marwan Muhammad Fuad, Pierre-François Marteau

TOWARDS A TIGHTER, MORE INTUITIVE SYMBOLIC AGGREGATE APPROXIMATION METHOD

Keywords: Time Series Information Retrieval, Symbolic Representation of Time Series, Symbolic Aggregate Approximation, Updated Minimum Distance.

Abstract: The similarity search problem is one of the main problems in time series data mining. Traditionally, this problem was tackled by sequentially comparing the given query against all the time series in the database, and returning all the time series that are within a predetermined threshold of that query. But the large size and the high dimensionality of time series databases that are in use nowadays make that scenario inefficient to tackle this problem. There are many representation techniques that aim at reducing the dimensionality of time series so that the search can be handled faster at a lower-dimensional space level. Symbolic representation is one of the promising techniques, because symbolic representation methods try to benefit from the wealth of search algorithms used in bioinformatics and text mining communities. The symbolic aggregate approximation (SAX) is one of the most competitive methods in the literature. SAX utilizes a similarity measure, which is easy to compute because it is based on pre-computed distances obtained from lookup tables. In this paper we present a new similarity distance that is as easy to compute as the original similarity distance, it is also tighter because it uses updated lookup tables. In addition, it is more intuitive. We conduct several experiments, which show that this new similarity distance gives better results than the original one.

1 INTRODUCTION

Similarity search is a fundamental problem in computer science. This problem has many applications in multimedia databases, bioinformatics, pattern recognition, text mining, computer vision, medicine, data mining, machine learning and so on. With the advert of the internet and the increasing use of it, this problem has received more attention from researchers. The richness of information available on the internet could not be manageable if search engines were not present. The usefulness of information depends highly not only on its quality, but also on the speed at which it is retrieved, which, in turn, depends upon the way it is represented and indexed. This all poses questions on indexing, representation and retrieval methods. Small databases that contain simple data objects can be handled easily. But managing large databases, like many of the databases in use today, requires serious effort, especially when they contain complex data types.

Another substantial change is the kind of queries launched, which is, somehow, related to the two latter changes; searching for data objects that are identical to a given query in unstructured, weakly-structured, or imprecise data bases, like many databases in use today, may not be very meaningful. Besides, in many cases the user may not be sure of what they are looking for when they launch the initial query. Hence *range query*, in which the user is interested in retrieving all the data objects that are within a predefined threshold of a given query, or *k-nearest neighbor*, in which the user tries to retrieve the *k* closest data objects to the query, have become popular. All these problems make traditional retrieving techniques inadequate that it is inevitable to think of new ways to handle these databases.

Time series are data types that appear in many applications, which vary from medicine through science and technology to business and economics. Time series data mining includes many tasks such as classification, clustering, similarity search, motif discovery, anomaly detection, and others. One key to performing these tasks successfully is representation methods that can represent the time series efficiently and effectively. Another key is indexing time series in appropriate structures, which direct the query process towards regions in the search space, where similar time series to the query are likely to be found, which makes the retrieving process faster.

Time series are high dimensional data types, so even indexing structures can suffer from the so-called "dimensionality curse". One of the best solutions to deal with this problem is to utilize a dimensionality reduction technique to reduce the dimensionality of these data objects, then to utilize a suitable indexing structure on the reduced space.

There have been different suggestions to represent time series. To mention a few; DFT (Agrawal *et al* . 1993) and (Agrawal *et al* . 1995), DWT (Chan and Fu 1999), SVD (Korn *et al* . 1997), APCA (Keogh *et al* . 2001), PAA (Keogh *et al* . 2000) and (Yi and Faloutsos 2000), PLA(Morinaka et al . 2001)...etc.

Among dimensionality reduction techniques, symbolic representation has several advantages, because it allows researchers to benefit from text-retrieval algorithms and techniques (Keogh *et al* . 2001).

Similarity between two data objects can be computed by means of a similarity distance. There are quite a large number of similarity distances; some are applied to a particular data type, while others can be applied to different data types. Among the different similarity distances, there are those that can be used on symbolic data types. At first they were available for data types whose representation is naturally symbolic (DNA and proteins sequences, textual data…etc). But later these symbolic similarity distances were also applied to other data types that can be transformed into strings by using some symbolic representation technique.

Of all the symbolic representation methods in the times series data mining literature, the symbolic aggregate approximation method (SAX) (Jessica *et al* . 2003) stands out as one of the most powerful methods. The main advantage of this method is that the similarity distance that it uses is easy to compute, because it uses statistical lookup tables. In this paper we present an improved similarity distance to be used with SAX. It has the same advantages as the original similarity distance used in SAX. But our new similarity distance gives better results in different time series mining tasks

The rest of this paper is organized as follows: in section 2 we present background on dimensionality reduction, and on symbolic representation of time series in general, and SAX in particular. The proposed similarity distance is presented in section 3. In section 4 we present some of the results of the different experiments we conducted. The conclusion is presented in section 5.

2 BACKGROUND

2.1 Dimensionality Reduction

Handling high-dimensional data is difficult to achieve. One of the main paradigms to overcome this problem is to embed the data objects of the original space into a lower dimensional space. Time series are highly correlated data, so representation methods that aim at reducing dimensionality by projecting the original data onto lower dimensional spaces and processing the query in those reduced spaces is a scheme that is widely used in time series data mining community.

When embedding the original space into a lower dimensional space and performing the similarity query in the transformed space, two main side-effects may be encountered; *false alarms*, also called *false positivity,* and *false dismissals*. False alarms are data objects that belong to the response set in the transformed space, but do not belong to the response set in the original space. False dismissals are data objects that the search algorithm excluded in the transformed space, although they are answers to the query in the original space. Generally, false alarms are more tolerated than false dismissals, because a post-processing scan is usually performed on the results of the query in the transformed space to filter out these data objects that are not valid answers to the query in the original space. However, false alarms can slow down the search time if they are too many. False dismissals are a more serious problem and they need more sophisticated procedures to avoid them.

False alarms and false dismissals are dependent on the transformation used in the embedding. If f is a transformation from the original space $(S_{original}, d_{original})$ into another space

$(S_{transformed}, d_{transformed})$ then in order to guarantee no false dismissals this transform should satisfy:

$$d_{transformed}(f(u_1), f(u_2)) \le d_{original}(u_1, u_2), \quad \forall u_1, u_2 \in S_{original} \qquad (1)$$

The above condition is known as the *lower-bounding lemma.* (Yi and Faloutsos 2000)

If a transformation can make the two above distances equal for all the data objects in the original space, then similarity search produces no false alarms or false dismissals. Unfortunately, such an ideal transformation is very hard to find. Yet, we try to make the above distances as close as possible. The above condition can be written as:

$$0 \le \frac{d_{transformed}(f(u_1), f(u_2))}{d_{original}(u_1, u_2)} \le 1 \qquad (2)$$

A *tight* transformation is one that makes the above ratio as close as possible to 1.

2.2 Symbolic Representation

One of the dimensionality reduction schemes in time series data mining is symbolic representation. Symbolic representation of time series uses an alphabet A (usually finite) to reduce the dimensionality of the time series. This can be defined formally as follows: Given a time series $TS = t_1, t_2, ..., t_n$. The symbolic representation scheme can be considered as a map

$$f : [t_i, t_j] \rightarrow \alpha_k \qquad \alpha_k \in A \qquad (3)$$

Symbolic representation of time series has been a hot research topic, because by using this scheme we can not only reduce the dimensionality of time series, but can also benefit from the numerous algorithms used in bioinformatics and text data mining. However, first symbolic representation methods were ad hoc and did not give satisfactory results. But later more sophisticated methods emerged.

The symbolic aggregate approximation method (SAX) is one of the most powerful methods of symbolic representation of time series. SAX is based on the fact that normalized time series have highly Gaussian distribution (Larsen and Marx 1986), so by determining the breakpoints that correspond to the alphabet size, one can obtain equal sized areas under the Gaussian curve.

SAX is applied in the following steps: in the first step the time series are normalized. In the second step, the dimensionality of the time series is reduced. This is obtained by using the PAA (Piecewise Aggregate Approximation) (Keogh, et al . 2000). In PAA the times series is divided into equal sized frames and the mean value of the points within the frame is computed. The lower dimensional vector of the original time series is the vector whose components are the means of all successive frames. In the third step, the PAA representation of the time series is discretized. This is achieved by determining the number and the location of the breakpoints. The number of breakpoints is related to the desired alphabet size (which is chosen by the user), i.e. *alphabet_size=number(breakpoints)+1* .
Their locations are determined by statistical lookup tables, so that these breakpoints produce equal-sized areas under the Gaussian curve. The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of the PAA that lies within that interval is discretized by that symbol. The last step of SAX is using the following similarity distance;

$$MINDIST(\hat{S}, \hat{T}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (dist(\hat{s}_i, \hat{t}_i))^2} \qquad (4)$$

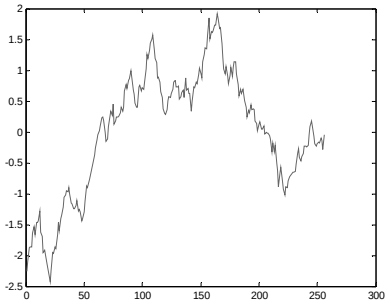
Where n is the length of the original time series, w is the length of the strings (the number of the frames), $\hat{S}$ and $\hat{T}$ are the symbolic representations of the two time series S and T respectively, and where the function $dist(\ )$ is implemented by using the appropriate lookup table.

We also need to mention that the similarity distance used in PAA is:

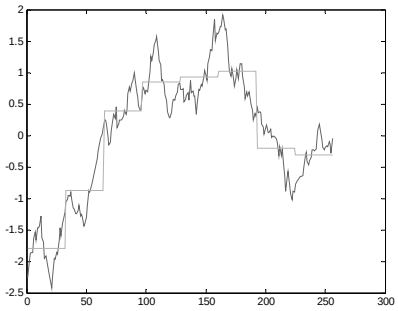
$$d(X, Y) = \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (\bar{x} - \bar{y})^2} \qquad (5)$$

Where n is the length of the time series, N is the number of frames. It is proven in (Keogh, et al . 2000) and (Yi and Faloutsos 2000) that the above similarity distance is a lower bounding of the Euclidean distance applied in the original space of time series . This results in the fact that MINDIST

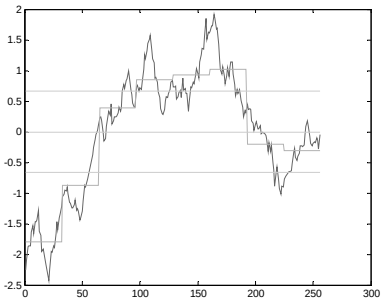
(1)The original time series



(2) Converting the time series to PAA



(3) Choosing the breakpoints



(4) Discretization the PAA

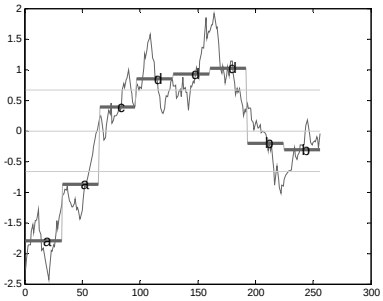


Figure 1: The different steps of SAX

is also a lower bounding of the Euclidean distance, because it is a lower bounding of the similarity distance used in PAA. This guarantees no false dismissals. Figure.1 illustrates the different steps of SAX.

3 THE UPDATED MINIMUM DISTANCE (UMD)

The main advantage of SAX, which makes it fast to apply, is that the similarity distance it uses is easy to compute, because it is based on pre-computed distances obtained from corresponding lookup tables. However, MINDIST has a main drawback; in order to be lower bounding this similarity distance ignores all the distances between any successive symbols of the alphabet and considers them to be zero. For instance, the lookup table of the MINDIST for an alphabet size of 3 is the one shown in Table 1.

Table 1: The lookup table of MINDIST for alphabet size =3. All values between any successive symbols are equal to zero. The breakpoints in this case (obtained from statistical tables) are: -0.43 and 0.43. The distance between them is 0.86

	a	**b**	**c**
a	0	0	0.86
b	0	0	0
c	0.86	0	0

This has two consequences: the first is that MINDIST is not tight enough, which produces many false alarms. The second consequence can be shown by the following example. Let the symbolic representing of the five time series $TS1, TS2$, $TS3$, $TS4$, $TS5$ using SAX with alphabet size =4 be : $TS1 = aabdd$, $TS2 = bacdc$, $TS3 = abbcd$, $TS4 = bacdd$, $TS5 = bbbdc$. The MINDIST between any two of these five times series is zero, which is not only unintuitive, since no two time series of these five are identical, but this also produces many false alarms.

In this section we present a modified minimum distance, which remedies the above problems. The new minimum distance has all the advantages of the original distance, in that it is also a lower bounding of the Euclidean distance, and it is also fast to compute, as it uses pre-computed distances. But the new minimum distance is tighter. It is also intuitive, in that it does not ignore the distances between

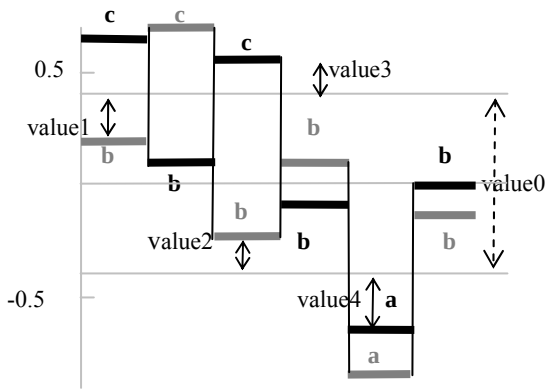


Figure 2: The PAA representation of two time series: TS1 =cbcbab (bold black) and TS2=bcbbab (bold grey). The solid arrows show the ignored distances and the dashed arrow shows the only distance considered by MINDIST : *dist (a,c)=value0* (0.86)

successive symbols.

The principal of our new minimum distance, which we call the *updated minimum distance* (UMD) is to recover the distances between any successive symbols, which were ignored in MINDIST. Figure 2 shows an example of the ignored distances in the case of alphabet size =3, and which are recovered in UMD.

Figure 2 shows that in the case of alphabet size =3 the breakpoints are -0.43 and 0.43. In this case the only non-zero distance according to MINDIST is *dist (a,c)* which is equal to 0.86 (the distance indicated by the dashed arrow). The distances represented by the solid arrows are the distances between the minima or the maxima of all the symbols of the alphabet and the corresponding breakpoint. These distances are ignored in MINDIST, but as we can see they are not equal to zero. So *dist(a,b)*, which was zero in MINDIST can be updated to become *value2+value4* , and *dist (b,c)* which was also zero in MINDIST can be updated to become *value1+value3,* and even *dist (a,c)* is updated to become *value4+value 3+value0* (value 0 is the original value). Lookup tables of different alphabet sizes are updated in a like manner. Obviously, this update of lookup tables results in a tighter similarity distance. For instance, the lookup table shown at the beginning of this section can be updated to become the one shown in Table2.

We can easily notice that this new distance is a lower bounding of the PAA distance presented in (5) in section 2.2, since we take the closest distance between two successive symbols among all the distances of all the PAA segments of these two symbols. As a result, our new distance is also a lower bounding of the Euclidean distance (c.f section 2.2). This is the same property that MINDIST has.

Table 2: The updated lookup table for alphabet size =3. We can see that the distances between successive symbols are no longer equal to zero. And the distance *dist (a,c)* is tighter

	a	**b**	**c**
a	0	value2+ value4	0.86+ value4+ value3
b	value2+ value4	0	Value1+ Value3
c	0.86+ value4+ value3	Value1+ Value3	0

The other consequence of this update is that UMD, which is based on the updated lookup tables, is intuitive, because it gives non-zero values to successive symbols, so the UMD of any two of the five time series presented at the beginning of this section is not zero, which is what we expect from any similarity distance applied to these time series because they are not identical.

In order to obtain the minimum and the maximum values of each symbol, the SAX algorithm is modified so that at the step where the different segments of the PAA are compared against the breakpoints to decide what symbol is used to discretize that segment, at that step we modify SAX so that it keeps a record of the minimum and maximum values of each segment of that time series. This is performed offline, so it does not include any extra cost at query time. Then when comparing two time series, we take the minimum (maximum) that corresponds to the same symbol of the two times series to find the mutual minimum (maximum, respectively) that corresponds to each symbol. These minima and maxima are used to update the lookup tables. The update process includes very few addition operations (three for alphabet size= 3, for instance), whose cost is very low compared with the cost of computing the distance. So UMD is as fast as MINDIST.

So, as we can see, the cost of UMD is a little bit higher at the indexing phase, which is not important, but it has the same complexity as MINDIST at the retrieval phase

4 EXPERIMENTS

We conducted extensive experiments on the proposed similarity distance. In our experiments we tested UMD on all the 20 data sets available at UCR (UCR Time Series datasets) and for all alphabet sizes, which vary between 3 (the least possible size that was used to test MINDIST) to 20 (the largest possible alphabet size). The size of these data sets varies between 28 (Coffee) and 6164 (wafer). The length of the time series varies between 60 (Synthetic Control) and 637 (Lightning-2). So these data sets are very diverse. In all our experiments we took the Euclidean distance as the reference distance, because this distance is widely used in the time series data mining community (Keogh and Kasetty 2002 and Reinert *et al* . 2000), even though it is has a few inconveniences; it is sensitive to noise and to shifts on the time axis. It is also applied to series of identical lengths only (Megalooikonomou *et al* . 2005).

4.1 Tightness

As mentioned in section 2.1, tightness of the similarity distances enhances the search process, because it minimizes the number of false alarms. As a result, it decreases the post processing time.

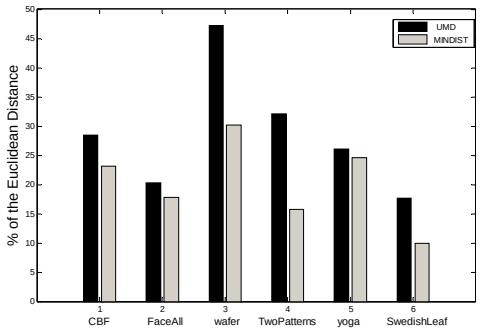


Figure 3: Comparison of the tightness of UMD with the tightness of MINDIST in 6 data sets and for alphabet size=3. The figure shows that UMD is tighter than MINDIST.

We compared the tightness of UMD with the tightness of MINDIST, for all the datasets and for all values of the alphabet size. In all the experiments, UMD was tighter than MINDIST. In Figure 3 and Figure 4, we present some of the results we obtained for alphabet size=3 and for alphabet size=10, respectively. We chose to report these data sets because they are the largest data sets in the UCR archive, so these results are the most significant statistically.

The experiments conducted on other data sets and for all values of the alphabet size, in addition to the other values of the alphabet size on the data sets presented in Figures 3 and 4, all gave similar results.

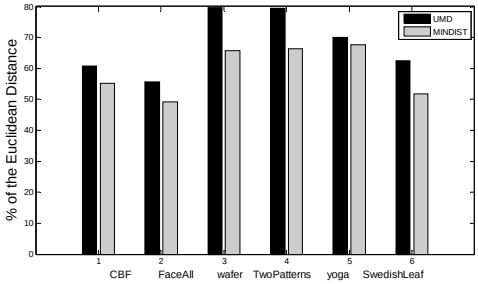


Figure 4: Comparison of the tightness of UMD with the tightness of MINDIST in 6 data sets and for alphabet size=10. The figure shows that UMD is tighter than MINDIST.

4.2 Classification

Classification is one of the main tasks in time series data mining. We tested the proposed similarity distance in a classification task based on the first nearest-neighbor rule on all the data sets available at UCR. We used leaving-one-out cross validation.

In order to make a fair comparison, we used the same compression ratio (the number of points used to represent one segment in PAA) that was used to test SAX with MINDIST (i.e.1 to 4).

4.2.1 The First Experiment

The first version of SAX used alphabet size that varies between 3 and 10. In the first classification experiment we conducted we used an alphabet size that varies between 3 and 10. We tested all the data sets in the UCR archive . We start by varying the alphabet size between 3 and 10 on the training set of each data set to find the optimal value of the alphabet size of that data set; i.e. the value that minimizes the classification error rate. Then we use that optimal alphabet size on the testing set of that data set. Table 3 shows the results of our experiment (There is no training phase for the Euclidian distance). The best result between UMD and MINDIST is highlighted. The results show that for this range of alphabet size UMD outperforms MINDIST in 14 data sets, and MINDIST outperforms UMD in 5 data sets, and in one case they both give the same result.

Table 3: The error rate of UMD and MINDIST for α (the alphabet size) between 3 and 10. Column 2 shows the error rate of the Euclidean distance

	1-NN Euclidean Distance	**UMD** (α between 3 and 10)	**MINDIST** (α between 3 and 10)
Synthetic Control	0.12	0.007	0.033
Gun-Point	0.087	0.213	0.233
CBF	0.148	0.131	0.104
Face (all)	0.286	0.306	0.319
OSULeaf	0.483	0.471	0.475
Swedish-Leaf	**0.213**	0.291	0.490
50words	0.369	0.338	0.327
Trace	0.24	0.34	0.42
Two_ Patterns	0.09	0.076	0.081
Wafer	0.005	0.004	0.004
Face (four)	0.216	0.273	0.239
Lighting-2	0.246	0.230	0.213
Lighting-7	0.425	0.411	0.493
ECG200	0.12	0.11	0.09
Adiac	0.389	0.634	0.903
Yoga	0.170	0.193	0.199
Fish	0.217	0.366	0.514
Beef	0.467	0.367	0.533
Coffee	0.25	0.179	0.464
OliveOil	**0.133**	0.367	0.833
MEAN	0.234	0.265	0.348
STD	**0.134**	0.158	0.247

It is worth to mention that for the Euclidian distance, there is no compression of information, so in some cases it may give better results than symbolic, compressed distances.

The average error of UMD over all the datasets and for this range of alphabet size is smaller than that of MINDIST. The standard deviation for UMD is also smaller than that of MINDIST. The significance of this statistical parameter is that when the standard deviation is small, the similarity distance is more robust, and can be applied to different kinds of datasets

4.2.2 The Second Experiment

SAX appeared in two versions; in the first one the average size varied in the interval (3:10), and in the second one the alphabet size varied in the interval (3:20). So we also conducted another experiment, where the alphabet size ranged in the interval (3:20), and on all the datasets in the UCR archive. We proceeded in the same way; we start by varying the alphabet size between 3 and 20 on the training set to find the optimal value of the alphabet size of that data set. Then we use this optimal alphabet size on the testing set. The results of our second experiment are shown in table 4 (We did not report the results of the Euclidean distance, because they are the same as in table 3). The results in table 4 show that UMD outperforms MINDIST in 14 datasets and MINDIST outperforms UMD in 4 datasets, and in 2 data sets they both give the same results.

In this experiment too, the mean and the standard deviation of UMD is better than that of MINDIST

The results of the two experiments show that the general performance of UMD is better than that of MINDIST

5 CONCLUSION AND PERSPECTIVES

In this paper we presented a new similarity distance to be used with the symbolic aggregate approximation (SAX). The new distance UMD improves the performance of SAX compared with the original similarity distance MINDIST used with SAX. We conducted several experiments of times series data mining tasks. The results obtained show that SAX with UMD gives better results than SAX with MINDIST. Another interesting feature of the new similarity distance is that it as fast to compute as the original one

Table 4: The error rate of UMD and MINDIST for α (the alphabet size) between 3 and 20.

	UMD (α between 3 and 20)	**MINDIST** (α between 3 and 20)
Synthetic Control	0.003	0.023
Gun-Point	0.14	0.127
CBF	0.054	0.073
Face (all)	0.305	0.305
OSULeaf	0.471	0.475
Swedish-Leaf	0.242	0.253
50words	0.345	0.334
Trace	0.27	0.35
Two_ Patterns	0.065	0.066
Wafer	0.004	0.004
Face (four)	0.273	0.239
Lighting-2	0.229	0.148
Lighting-7	0.411	0.425
ECG200	0.11	0.13
Adiac	0.494	0.867
Yoga	0.172	0.181
Fish	0.257	0.263
Beef	0.333	0.433
Coffee	0.071	0.143
OliveOil	0.3	0.833
MEAN	0.227	0.284
STD	0.147	0.237

The future work can be improving this distance by tracing other values in the original time series. This can make the distance even tighter, which is what we are working on now.

Another direction of future work focuses on modifying SAX to benefit more from UMD. We think this can be achieved by using a representation method other than PAA. This may include more calculations or more storage space at indexing time, but it could give better results

REFERENCES

Agrawal, R., Faloutsos, C., & Swami, A. 1993: Efficient similarity search in sequence databases". Proceedings of the 4th Conf. on Foundations of Data Organization and Algorithms.

Agrawal, R., Lin, K. I., Sawhney, H. S. and Shim. 1995, K,.: Fast similarity search in the presence of noise, scaling, and translation in time-series databases, in Proceedings of the 21st Int'l Conference on Very Large Databases. Zurich, Switzerland, pp. 490-501.

Chan, K. & Fu, A. W. 1999: Efficient Time Series Matching by Wavelets. In proc. of the 15th IEEE Int'l Conf. on Data Engineering. Sydney, Australia, Mar 23-26. pp 126-133..

Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, Bill Yuan-chi Chiu. 2003: A symbolic representation of time series, with implications for streaming algorithms. DMKD 2003: 2-11.

Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra. 2000: Dimensionality reduction for fast similarity search in large time series databases. J. of Know. and Inform. Sys.

Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra. 2001: Locally adaptive dimensionality reduction for similarity search in large time series databases. SIGMOD pp 151-162 .

Keogh, E. & Kasetty, S. 2002.. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In proceedings of the 8th ACM SIGKDD International Conference on

Knowledge Discovery and Data Mining. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 102-111.

Korn, F., Jagadish, H & Faloutsos. C. 1997: Efficiently supporting ad hoc queries in large datasets of time sequences. Proceedings of SIGMOD '97, Tucson, AZ, pp 289-300.

Larsen, R. J. & Marx, M. L. 1986. An Introduction to Mathematical Statistics and Its Applications. Prentice Hall, Englewood, Cliffs, N.J. 2nd Edition.

Megalooikonomou, V., Wang, Q., Li, G. & Faloutsos, C 2005. Multiresolution Symbolic Representation of Time Series. In proceedings of the 21st IEEE International Conference on Data Engineering (ICDE). Tokyo, Japan. Apr 5-9. .

Morinaka, Y., Yoshikawa, M. , Amagasa, T., and Uemura, S 2001.: The L-index: An indexing structure for efficient subsequence matching in time sequence databases. In Proc. 5th PacificAisa Conf. on Knowledge Discovery and Data Mining, pages 51-60 .

Reinert, G., Schbath, S. & Waterman, M. S. 2000 Probabilistic and Statistical Properties of Words: An Overview. Journal of Computational. Biology. Vol. 7, pp 1-46.

Yi, B,K., & Faloutsos, C. 2000: Fast time sequence indexing for arbitrary Lp norms. Proceedings of the 26st International Conference on Very Large Databases, Cairo, Egypt .

UCR Time Series datasets
http://www.cs.ucr.edu/~eamonn/time_series_data/