HAL
open science

Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables

Muhammad Marwan Muhammad Fuad, Pierre-François Marteau

To cite this version:

Muhammad Marwan Muhammad Fuad, Pierre-François Marteau. Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables. Knowledge-Based and Intelligent Information and Engineering Systems (KES'2010), Jul 2010, Cardiff, Wales, UK, United Kingdom. pp.420-431, 10.1007/978-3-642-15387-7_46 . hal-00689890

HAL Id: hal-00689890

https://hal.science/hal-00689890v1

Submitted on 20 Apr 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables

Muhammad Marwan Muhammad Fuad and Pierre-François Marteau

VALORIA, Université de Bretagne Sud, Université Européenne de Bretagne
BP. 573, 56017 Vannes, France
{marwan.fuad,pierre-francois.marteau}@univ-ubs.fr

Abstract. Similarity search in time series data mining is a problem that has attracted increasing attention recently. The high dimensionality and large volume of time series databases make sequential scanning 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, since 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 that is easy to compute because it is based on pre-computed distances obtained from lookup tables. In this paper we present a new similarity measure that is almost as easy to compute as the original similarity measure, but it is tighter because it uses updated lookup tables. In addition, the new similarity measure is more intuitive than the original one. We conduct several experiments which show that the new similarity measure gives better results than the original one.

Keywords: Time Series Data Mining, Symbolic Representation, Symbolic Aggregate Approximation, Updated Minimum Distance.

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, data mining, machine learning and others.

Time series are data types that appear in many medical, scientific and financial applications. 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 processing 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, so even indexing structures can suffer from the so-called "dimensionality curse". One of the best solutions to deal with this

R. Setchi et al. (Eds.): KES 2010, Part I, LNAI 6276, pp. 420–431, 2010.
© Springer-Verlag Berlin Heidelberg 2010

phenomenon is to utilize a dimensionality reduction technique to reduce the dimensionality of time series, 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 [1,2], DWT [3], SVD [8], APCA [6], PAA [5,13], PLA [11], etc.

Among dimensionality reduction techniques, symbolic representation has several advantages, because it allows researchers to benefit from text-retrieval algorithms and techniques [6].

Similarity between two data objects can be depicted by means of a similarity measure, or a distance metric, which is a stronger mathematical concept. There are quite a large number of similarity measures; some are applied to a particular data type, while others can be applied to different data types. Among the different similarity measures, 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 measures were also applied to other data types that can be transformed into strings by using different symbolic representation techniques.

Of all the symbolic representation methods in the times series data mining literature, the symbolic aggregate approximation method (SAX) [4] stands out as one of the most powerful methods. The main advantage of this method is that the similarity measure that it uses is easy to compute, because it uses statistical lookup tables. In this paper we present an improved similarity measure to be used with SAX. It has the same advantages as the original similarity measure used in SAX. But our new similarity measure gives better results in different time series data mining tasks, in return of a small additional computational cost that does not affect the overall complexity which remains $O(N)$ for both measures.

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

2 Related Work

2.1 Dimensionality Reduction

Time series are generally highly correlated data, so representation methods that aim at reducing the dimensionality of these data by projecting the original space onto a lower dimensional space and processing the query in the reduced space 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 scanning is usually performed on the candidate answer set using the original data objects and the original distance to filter out the data objects that are not valid answers to the query. However, false alarms can slow down the search time if they are too many. False dismissals are a more serious problem and avoiding them requires more sophisticated procedures.

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 transformation should satisfy:

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

The above condition is known as the *lower-bounding lemma.* [13]

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:

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

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

2.2 Symbolic Representation

One of the dimensionality reduction techniques in time series data mining is symbolic representation. Symbolic representation of time series uses an alphabet Σ (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 is a map:

$$[t_i, t_j] \overset{f}{\mapsto} \alpha_k \quad ; \alpha_k \in \Sigma \tag{3}$$

Symbolic representation of time series has attracted much attention, 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.

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" (according to the authors of [4]), so by determining the breakpoints that correspond to the alphabet size, one can obtain equal sized areas under the Gaussian curve.

SAX is applied as follows: in the first step the time series are normalized. In the second step, the dimensionality of the time series is reduced by using PAA (Piecewise Aggregate Approximation) [5]. 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 time series is the vector whose components are the means

of the successive frames. In the third step, the PAA representation of the time series is discretized. This is achieved by determining the number and 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 measure:

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

Where n is the length of the original time series, N 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.

Notice that MINDIST is a similarity measure and not a distance metric since it violates two conditions of distance metrics which are the identity condition: ($x = y \Leftrightarrow d(x,y) = 0$), and the triangular inequality condition:

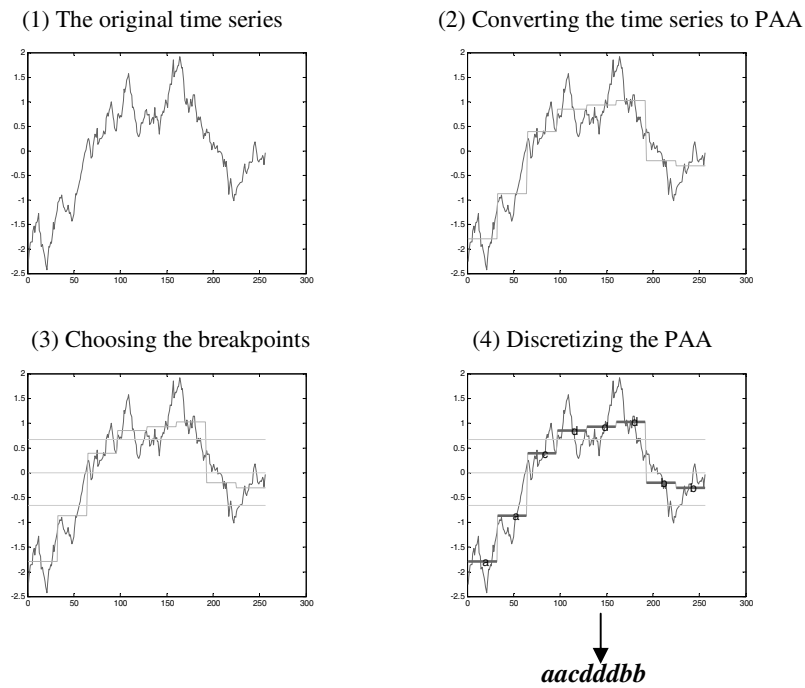

Fig. 1. The different steps of SAX

($d(x,z) \le d(x,y) + d(y,z)$), and it respects only one condition which is the symmetry condition: ($d(x,y) = d(y,x)$)

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

$$d(X,Y) = \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (\overline{x_i} - \overline{y_i})^2} \tag{5}$$

Where n is the length of the time series, N is the number of frames. It is proven in [5] and [13] 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 is also a lower bounding of the Euclidean distance, because it is a lower bounding of the similarity measure 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 measure 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 measure 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.

This drawback 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 time 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 successive symbols.

Table 1. The lookup table of MINDIST for alphabet size equals to 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

The principle 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 equals to 3, and which are recovered in UMD. 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* (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 similarity measure is lower bounding of the PAA similarity measure 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 similarity measure is also a lower bounding of the Euclidean distance (c.f section 2.2). This is the same property that MINDIST has.

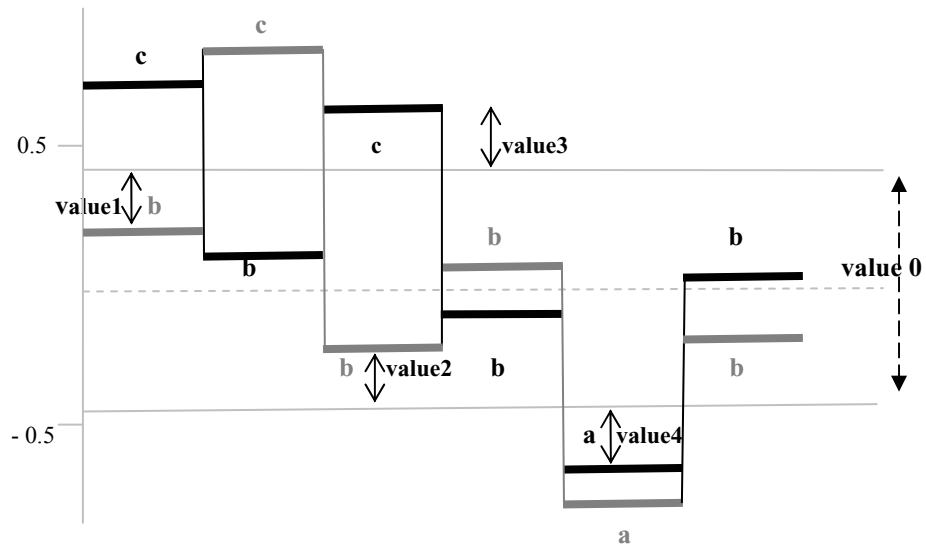

Fig. 2. The PAA representation of two time series: TS1 =cbcbab (boldface black) and TS2=bcbbab (boldface grey). The solid arrows show the ignored distances and the dashed arrow shows the only distance considered by MINDIST : *dist (a,c)=value0* (0.86).

Table 2. The updated lookup table for alphabet size equals to 3. We can see that the distances between successive symbols are no longer equal to zero, and the distance *dist (a,c)* is tighter than *dist (a,c)* in Table 1.

	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 measure 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 at indexing

```
procedure dist=UMD(TS1,TS2,TS1MinMax,TS2MinMax,
   Alphabet, LookupTable)
// INPUT : TS1,TS2; two input times series presented
// symbolically
// INPUT : TS1MinMax,TS2MinMAx; The Min and Max
// distances between TS1 (TS2) and the corresponding
// breakpoints
// INPUT : Alphabet
// INPUT : LookupTable is the look up table used with
// MINDIST
// OUTPUT : RETURN dist, the UMD distance between TS1,
// TS2
   for α_k ∈ Alphabet
          mn= min (TS1MinMax(α_k), TS2MinMAx(α_k))
      mx= min (TS1MinMax(α_k), TS2MinMax(α_k))
//update the lookup table for symbol α_k
    UpdateTable ← Update(Lookuptable, α_k, mn, mx)

   end
  return dist=MINDIST(TS1, TS2, UpdateTable)
end procedure
```

Fig. 3. Pseudo Code for the UMD distance

time, 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) that corresponds to each symbol. These minima and maxima, which are computed at indexing time, are used to update the lookup tables. The update process includes very few addition operations (three for alphabet size= 3, for instance), whose computational cost is very low compared with the cost of computing the similarity measure. So UMD has the same complexity as that of MINDIST. The pseudo code for the UMD is shown in Figure 3.

So, as we can see, the computational cost of UMD is a little bit higher at indexing time, but it has the same complexity as MINDIST at query time.

We also have to mention that UMD is also a similarity measure and not a distance metric. However, it is one-step closer to being distance metric since it violates only one condition of the distance metric which is the triangular inequality condition (c.f section 2.2).

4 Empirical Evaluation

We conducted extensive experiments on the proposed similarity measure. In our experiments we tested UMD on all the 20 datasets available at UCR [14] 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 datasets 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. We also tested these datasets using the Euclidean distance, because this distance is widely used in the time series data mining community [7, 12].

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.

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 4, we present some of the results we obtained for alphabet size equals to 3 and 10, respectively. We chose to report these datasets because they are the largest data sets in the UCR archive, thus the tightness, which is the average of the rate of the corresponding similarity measure to the Euclidean distance between all pairs of time series in the dataset, is more accurate statistically.

The experiments conducted on these datasets using other values of the alphabet size, in addition to the experiments on the other datasets in the UCR archive for all values of the alphabet size, all gave similar results.

4.2 Classification

Classification is one of the main tasks in time series data mining. We tested the proposed similarity measure 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.

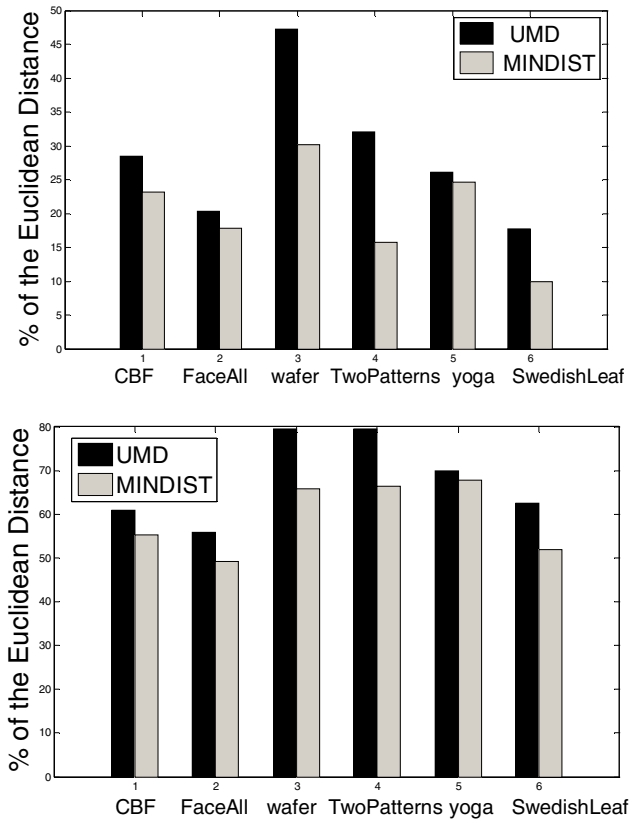

Fig. 4. Comparison of the tightness of UMD with the tightness of MINDIST in 6 data sets and for alphabet size=3 (above) and alphabet size=10 (below). The figure shows that UMD is tighter than MINDIST.

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). The first version of SAX used alphabet size that varies between 3 and 10. Then in a later version the alphabet size varied between 3 and 20. We conducted two main classification experiments; the first one is for alphabet size (3:10), and the second is for alphabet size (3:20). Each experiment tested all the datasets. In each experiment we start by varying the alphabet size (3 through 10 in the first experiment and 3 through 20 in the second one) on the training set of each dataset to find the optimal value of the alphabet size of that dataset; i.e. the value that minimizes the classification error rate. Then we use that optimal alphabet size on the testing set of that dataset. Table 3 shows the results of our experiments (There is no training phase for the Euclidian distance). Columns 3 and 4 contain the results of the first experiment of UMD and MINDIST, respectively, where the alphabet size varies between 3 and 10. The best result between UMD and MINDIST for that range of alphabet size is highlighted. Columns 5 and 6 contain the results of the second experiment

Table 3. The rate error of UMD and MINDIST for α between 3 and 10 (columns 3 and 4), and for α between 3 and 20(columns 5 and 6). 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)	**UMD** (α between 3 and 20)	**MINDIST** (α between 3 and 20)
Synthetic Control	0.12	0.007	0.033	0.003	0.023
Gun-Point	**0.087**	0.213	0.233	0.14	0.127
CBF	0.148	0.131	0.104	0.054	0.073
Face (all)	**0.286**	0.306	0.319	0.305	0.305
OSU Leaf	0.483	0.471	0.475	0.471	0.475
Swedish Leaf	**0.213**	0.291	0.490	0.242	0.253
50words	0.369	0.338	0.327	0.345	0.334
Trace	**0.24**	0.34	0.42	0.27	0.35
Two Patterns	0.09	0.076	0.081	0.065	0.066
Wafer	0.005	0.004	0.004	0.004	0.004
Face (four)	**0.216**	0.273	0.239	0.273	0.239
Lighting-2	0.246	0.230	0.213	0.229	0.148
Lighting-7	0.425	0.411	0.493	0.411	0.425
ECG200	0.12	0.11	0.09	0.11	0.13
Adiac	**0.389**	0.634	0.903	0.494	0.867
Yoga	**0.170**	0.193	0.199	0.172	0.181
Fish	**0.217**	0.366	0.514	0.257	0.263
Beef	0.467	0.367	0.533	0.333	0.433
Coffee	0.25	0.179	0.464	0.071	0.143
Olive Oil	**0.133**	0.367	0.833	0.3	0.833
MEAN	0.234	0.265	0.348	0.227	0.284
STD	**0.134**	0.158	0.247	0.147	0.237

(*: α is the alphabet size)

of UMD and MINDIST, respectively, where the alphabet size varies between 3 and 20. The best result between UMD and MINDIST for the same range of alphabet size is also highlighted.

The results show that for alphabet size in the interval (3:10), UMD outperforms MINDIST in 14 datasets, and MINDIST outperforms UMD in 5 datasets, and in one case they both give the same result. For alphabet size that varies in the interval (3:20) UMD outperforms MINDIST in 14 datasets and MINDIST outperforms UMD in 4 datasets, and in 2 datasets they both give the same results. The average error of UMD over all the datasets and for both ranges of alphabet size is smaller than that of MINDIST. The standard deviation for UMD is also smaller than that of MINDIST and for both ranges of the alphabet size. The significance of this statistical parameter is that

when the standard deviation is small, the similarity measure is more robust, and can be applied to different kinds of datasets.

The results obtained show that the general performance of UMD is better than that of MINDIST.

It is worth to mention that for the Euclidian distance, there is no compression of data, thus no loss of information, so in some cases it may give better results than symbolic, compressed distances. However, using the Euclidean distance as a reference should not be taken for granted because this distance has a few inconveniences; it is sensitive to noise and to shifts on the time axis [10].

5 Conclusion and Future Work

In this paper we presented a new similarity measure to be used with the symbolic aggregate approximation (SAX). The new similarity measure UMD improves the performance of SAX compared with the original similarity measure 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 measure is that it has the same complexity as that of MINDIST. Other experiments on the CPU time of both MIDIST and UMD may also be conducted to support the results presented in this paper.

The future work can focus on improving this similarity measure by tracing other values in the original time series. This can make the proposed similarity measure 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.

UMD can also be used as a fast-and-dirty filter to speed up the nearest neighbor search by quickly filtering out the time series which are not answers to the query and terminating the search using the Euclidean distance.

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730. Springer, Heidelberg (1993)
2. Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, 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 (1995)
3. Chan, K., Fu, A.W.: Efficient Time Series Matching by Wavelets. In: Proc. of the 15th IEEE Int'l Conf. on Data Engineering, Sydney, Australia, March 23-26, pp. 126–133 (1999)
4. Lin, J., Keogh, E.J., Lonardi, S., Chiu, B.Y.-c.: A symbolic representation of time series, with implications for streaming algorithms. In: DMKD 2003, pp. 2–11 (2003)
5. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra: Dimensionality reduction for fast similarity search in large time series databases. J. of Know. and Inform. Sys. (2000)

6. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra: Locally adaptive dimensionality reduction for similarity search in large time series databases. In: SIGMOD, pp. 151–162 (2001)
7. Keogh, E., Kasetty, S.: 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, Edmonton, Alberta, Canada, July 23 - 26, pp. 102–111 (2002)
8. Korn, F., Jagadish, H., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. In: Proceedings of SIGMOD 1997, Tucson, AZ, pp. 289–300 (1997)
9. Larsen, R.J., Marx, M.L.: An Introduction to Mathematical Statistics and Its Applications, 2nd edn. Prentice Hall, Englewood Cliffs (1986)
10. Megalooikonomou, V., Wang, Q., Li, G., Faloutsos, C.: Multiresolution Symbolic Representation of Time Series. In: proceedings of the 21st IEEE International Conference on Data Engineering (ICDE), Tokyo, Japan, April 5-9 (2005)
11. Morinaka, Y., Yoshikawa, M., Amagasa, T., Uemura, S.: The L-index: An indexing structure for efficient subsequence matching in time sequence databases. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS (LNAI), vol. 2035, pp. 51–60. Springer, Heidelberg (2001)
12. Reinert, G., Schbath, S., Waterman, M.S.: Probabilistic and Statistical Properties of Words: An Overview. Journal of Computational. Biology 7, 1–46 (2000)
13. Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary Lp norms. In: Proceedings of the 26st International Conference on Very Large Databases, Cairo, Egypt (2000)
14. UCR Time Series datasets, `http://www.cs.ucr.edu/~eamonn/time_series_data/`