HOT *a*SAX: A Novel Adaptive Symbolic Representation for Time Series Discords Discovery

Ninh D. Pham, Quang Loc Le, Tran Khanh Dang

Faculty of Computer Science and Engineering,
HCM University of Technology, Vietnam National University of HoChiMinh City, Vietnam
ninhpham@contentinterface.com, {locle, khanh}@cse.hcmut.edu.vn

Abstract. Finding discords in time series database is an important problem in the last decade due to its variety of real-world applications, including data cleansing, fault diagnostics, and financial data analysis. The best known approach to our knowledge is HOT SAX technique based on the equiprobable distribution of SAX representations of time series. This characteristic, however, is not preserved in the reduced-dimensionality literature, especially on the lack of Gaussian distribution datasets. In this paper, we introduce a *k*-means based algorithm for symbolic representations of time series called ***a***daptive **S**ymbolic **A**ggregate appro**X**imation (*a*SAX) and propose HOT *a*SAX algorithm for time series discords discovery. Due to the clustered characteristic of *a*SAX words, our algorithm produces greater pruning power than the previous approach. Our empirical experiments with real-world time series datasets confirm the theoretical analyses as well as the efficiency of our approach.

Keywords: Time Series Data Mining, SAX, Anomaly Detection, Clustering.

1 Introduction

The last decade has seen an increasing level of interest in finding time series discords due to its variety of uses for data mining, including improving the quality of clustering, data cleaning, summarization, and anomaly detection, etc. However, time series are essentially high dimensional data and directly dealing with such data in its raw format is very expensive in terms of processing and storage cost. Because of this fact, most of time series representations proposed in the literature are based on reduced-dimensionality techniques, but still preserving its fundamental characteristics, such as Discrete Fourier Transform (DFT) [3], Discrete Wavelet Transform (DWT) [2], Piecewise Aggregate Approximation (PAA) [4], and Symbolic Aggregate approXimation (SAX) [6], etc. However, there is only SAX approach allowing time series discords discovery based on a heuristic algorithm, called HOT SAX [5], which produces greater pruning power than brute force, while guaranteed to produce identical results.

HOT SAX approach is based on the symbolic representations of time series to improve the quality of the brute force algorithm. In particular, SAX approach discretizes and maps reduced-dimension time series into SAX symbols based on the

"breakpoints" which produce the equal-sized areas under Gaussian curve. Due to the equiprobable distribution of SAX representations of time series, the two heuristic searching orders in outer and inner loop of brute force were proposed for greater pruning power. Because of this fact, the performance of HOT SAX highly depends on the two major factors: (1) the Gaussian distributed feature of time series, and (2) the clustered feature of SAX words.

In this paper, we introduce a simple, but highly adaptive symbolic approach, ***a**daptive **S**ymbolic **A**ggregate appro**X**imation (aSAX)* for time series representations and propose a heuristic algorithm called HOT *a*SAX for time series discords discovery. In particular, our technique is based on the original SAX, but adaptive vector of "breakpoints". These adaptive "breakpoints" are determined by a preprocessing phase using *k*-means algorithm. Due to the clustered feature of the adaptive "breakpoints", the better heuristic searching orders in outer and inner loop of brute force could be obtained in HOT *a*SAX. That leads to greater pruning power on both the highly Gaussian distribution datasets and the lack of Gaussian distribution datasets for various discord lengths and database sizes.

The rest of this paper is organized as follows. In Section 2, we provide the problem definitions and briefly review the background material and related work. Section 3 introduces the novel adaptive representation called *a*SAX and HOT *a*SAX approach for time series discords discovery. In Section 4, we show experimental evaluations of the proposed approach with real-world datasets. Finally, in Section 5, we conclude and give some future research directions.

2 Background and Related Work

Intuitively, time series discords are the subsequences that have the least similarity to all other subsequences. However, "the best matches to a subsequence tent to be located one or two points to the left or the right of the subsequence in question" [5]. Such matches are called trivial matches and should be excluded to obtain the true discords. In this section, we will therefore formally define the problem definitions, including non-self match, top-*K* discords and then review the basic heuristic algorithm as well as HOT SAX approach for time series discords discovery.

2.1 Problem definitions

As noted above, it is obvious and intuitive to exclude trivial matches when finding time series discords; otherwise, almost all real datasets have degenerate and unintuitive solutions. We therefore need to formally define a non-self match:

Definition 1. *Non-self Match: Given a time series T, subsequence C of length n beginning at position p and a matching subsequence M beginning at position q, we say that M is a non-self match to C if* $|p - q| \geq n$.

We now use the definition of non-self match to define time series discord:

Definition 2. *Time Series Discord: Given a time series T, the subsequence D of length n beginning at position p is said to be the* ***top-1 discord*** *of T if D has the largest distance to its nearest non-self match.*

Note that we may have more than one unusual pattern in a given time series. We are thus interested in examining the top-*K* discords.

Definition 3. *Top-K Discord: Given a time series T, the subsequence D of length n beginning at position p is said to be the* ***top-K discord*** *of T if D has the* K^{th} *largest distance to its nearest non-self match, with no overlapping region to the* i^{th} *discord, for all* $1 \le i < K$.

The problem of locating top-*1* discord was first proposed in [5] by a heuristic algorithm which compares the distance from each subsequence to its nearest non-self match, as shown in Table 1.

Table 1. Heuristic Discord Discovery.

```
1.  discord_distance = 0
2.  discord_position = 0
3.  for each p in T ordered by heuristic Outer
4.     nearest_neighbor_distance = infinity
5.     for each q in T ordered by heuristic Inner
6.        if |p – q| ≥ n
7.           dist = Dist(t_p,...,t_p+n-1, t_q,...,t_q+n-1)
8.           if dist < nearest_neighbor_distance
9.              nearest_neighbor_distance = dist
10.          endif
11.          if dist < discord_distance
12.             break
13.          endif
14.       endif
15.    endfor
16.    if nearest_neighbor_distance > discord_distance
17.       discord_distance = nearest_neighbor_distance
18.       discord_position = p
19.    endif
20. endfor
21. return (discord_distance, discord_position)
```

In this algorithm, each possible candidate subsequence is extracted in the heuristic outer loop to find the nearest non-self match for each candidate subsequence in the heuristic inner loop. The candidate subsequence which has the largest distance to its nearest non-self match is the top-*1* discord. Note that the algorithm's running time has been highly dependent on the two heuristics for outer and inner loop. These heuristics determine the orders in which both outer loop and inner loop visit the subsequences in the attempt to find the discord in first few subsequences of outer loop. That leads to an early termination of algorithm and considerably speeds up the algorithm. For simplicity, we have only discussed top-*1* discords discovery. Extensions to top-*K* discords are trivial and obvious, and are omitted for brevity.

2.2 HOT SAX for Finding Time Series Discords

Keogh at al. [5] proposed the first heuristic algorithm for finding time series discords, called HOT SAX with the two heuristic orders in outer loop and inner loop based on the equiprobable distribution of SAX representations of subsequences. Time series subsequences extracted by a sliding window are converted into SAX representations and inserted into two augmented data structures to support the outer and inner heuristics. In particular, an array of SAX words containing a count of how often each word occurs in the array determines the searching order for outer loop. An augmented trie with leaves containing a list of all array indices with the same SAX encoding determines the searching order for inner loop.

Heuristic Outer Loop. Based on the intuition that unusual subsequences are very likely to map to the unique or rare SAX words, the first few subsequences which have minimum times are given to the outer loop to search over first. The rest of the candidates are visited in random order. By considering these candidate subsequences first, there is an increasing chance of giving a large value to the *discord_distance* variable early on, thus allowing more early terminations of the inner loop.

Heuristic Inner Loop. In the inner loop, all subsequences which have the same SAX encoding as the candidate subsequence of the outer loop are visited first. The rest of the subsequences are visited in random order. We thus only need to find one such subsequence that is similar enough to give a small value to the *dist* variable in order to terminate the inner loop more early.

As noted above, it is clear that HOT SAX performance is highly dependent on the clustered feature of SAX words. The highly clustered feature of SAX words will produce the better heuristics in outer and inner loop; otherwise, the speedup of HOT SAX will be reduced.

3 The Proposed Algorithm: HOT *a*SAX

It is important to note that the pruning power of HOT SAX is highly dependent on the quality of SAX approximation. In classic SAX approach, there is an assumption that most of time series have the highly Gaussian distribution because each dimension is discretized into SAX symbols based on predetermined "breakpoints" under Gaussian curve with equiprobability. Also, the discretization is often applied on the reduced-dimensionality data (e.g., PAA representation); however, no specific algorithm for that option was proposed. Finally, the two heuristic searching orders in outer and inner loop of HOT SAX are based on the equiprobable distribution of SAX representations of time series subsequences. As a consequence, HOT SAX produces higher performance on the highly Gaussian distribution datasets, but lower performance on the lack of Gaussian distribution datasets. In this section, we introduce a novel adaptive symbolic representation (*a*SAX) for time series and propose a heuristic algorithm, called HOT *a*SAX for time series discords discovery. The proposed approach produces the greater pruning power than the classic technique on both the highly Gaussian distribution datasets and the lack of Gaussian distribution datasets for various discord lengths and database sizes.

3.1 Adaptive Symbolic Representation: *a*SAX

Our proposed approach, ***a**daptive **S**ymbolic **A**ggregate appro**X**imation (aSAX)* is based on the classic SAX but with the adaptive "breakpoints". These adaptive "breakpoints" are predetermined by a pre-processing phase based on Lloyd's algorithm [7], which is nothing but the *k*-means algorithm [8] for clustering, specialized to one dimension. In particular, we use a small set of the normalized time series as the training set, the alphabet size of symbols as parameter *k* in *k*-means algorithm and the "breakpoints" under Gaussian curve as the clever initialization of cluster intervals. We note that the original normalized time series can be used for training but for the best performance we suggest the PAA representations of the training time series.

Given a training set, we first convert all time series into PAA representations and store them in a PAA array. Denote by t_n the value of the n^{th} PAA point in PAA array. Start with the given set of intervals $[\beta_i, \beta_{i+1})$ under Gaussian curve, for $i = 0,\ldots,a\text{-}1$ $\beta_0 = -\infty$ and $\beta_a = \infty$. Set $\Delta = \infty$, and fix $\gamma > 0$. Denote by r_i as the representative value for the interval *i*. The training algorithm to achieve the "adaptive" breakpoints for *a*SAX representation is derived based on *k*-means algorithm as followed:

Table 2. The Training Algorithm for Adaptive "Breakpoints".

1. For $i = 0,\ldots,a\text{-}1$, compute the new representative value as the center of mass of all PAA points in the interval $[\beta_i, \beta_{i+1})$, by $r_i = \frac{1}{N_i} \sum_{t_n \in [\beta_i, \beta_{i+1})} t_n$, where N_i is the total number of PAA points in the interval $[\beta_i, \beta_{i+1})$.
2. Compute the new intervals by $\beta_i = (r_{i-1} + r_i) / 2$, for $i = 1,\ldots,a\text{-}1$.
3. Compute the total representation error as $\Delta' = \sum_{i=1}^{k} \sum_{t_n \in [\beta_i, \beta_{i+1})} (t_n - r_i)^2$. If $\frac{\Delta - \Delta'}{\Delta} < \gamma$, then STOP. Otherwise, set $\Delta = \Delta'$, and go to step 1.

The above algorithm is guaranteed to converge rapidly because of the clever initialization with the equiprobability of the classic SAX symbols under Gaussian curve and the reduced-dimensionality PAA points. The approximation quality of the symbols is improved in each step of algorithm. By doing so, we obtained a resultant vector of new "breakpoints" for the adaptive approach, as shown in Table 3. After the third step, the error introduced by approximating the PAA points by the

corresponding symbols is further reduced. Since the representation quality of each symbol directly affects the quality of symbolic representation, this leads to better both tightness of lower bounding function and pruning power in indexing and querying. Furthermore, the clustered feature of the adaptive "breakpoints" has produced better heuristic searching orders and thus greater pruning power in finding discords on not only the highly Gaussian distribution datasets but also the lack of Gaussian distribution datasets.

Table 3. *a*SAX "breakpoints" of the random walk dataset of length 128, converted into PAA with 4 segments and SAX words of cardinality from 3 to 8.

βi \ *a*	3	4	5	6	7	8
β1	-0.46	-0.70	-0.86	-0.95	-1.03	-1.07
β2	0.45	0.00	-0.29	-0.45	-0.62	-0.71
β3		0.70	0.28	0.03	-0.21	-0.36
β4			0.85	0.51	0.20	0.00
β5				0.98	0.59	0.35
β6					1.01	0.72
β7						1.09

The astute reader will have noticed that we can easily achieve the more adaptive "breakpoints" by clustering each PAA segment separately. That means we have each "breakpoints" for each PAA segment, thus suffer more processing and storage cost but greater pruning power. In the next section, we introduce HOT *a*SAX algorithm based on the clustered feature of *a*SAX words for finding discords.

3.2 Heuristic Time Series Discords Discovery: HOT *a*SAX

Given the length of the discords *n*, the cardinality of the *a*SAX alphabet size *a* and *a*SAX word size *w*, we begin by creating an *a*SAX representation for each subsequence of time series by sliding a window of length *n* across time series. We use two data structures like HOT SAX approach to support our heuristics. These are (1) an array with the rightmost column containing a count of times each word occurs and (2) a trie with leaves containing a list of indices that have the same *a*SAX encoding.

Heuristic Outer Loop. The searching order in outer loop is based on the rightmost column of the array. The indices of all *a*SAX words that have the smallest count are given to the outer loop to search over first. The rest of the candidates are visited in random order. By considering the candidate subsequences mapping to unique or rare *a*SAX words early in the outer loop, there is an increasing chance to get the large value to the *discord_distance* variable early on, thus allowing more early terminations of the inner loop.

Heuristic Inner Loop. The searching order in inner loop is based on the leaves of trie. The indices of the time series subsequences which have the same *a*SAX encoding as the candidate subsequence are given to the inner loop to search over first. The rest

of the candidates are visited in random order. Since we discretized each subsequence based on the adaptive "breakpoints" with the highly clustered feature, the candidates of the same *a*SAX words are more likely to be highly similar. That leads to an excellent chance of giving smaller value to the *dist* variable in order to terminate the inner loop more early.

Recall that due to the highly clustered feature of *a*SAX words, the subsequences of the same *a*SAX words are more similar and the subsequences with different *a*SAX words are more separate. The speedup of HOT *a*SAX therefore increases and even greater speedup can be expected as the datasets get larger. Furthermore, it is easy to apply some minor optimizations in [5] to HOT *a*SAX for dramatic speedups.

4 Empirical Evaluation

In this section, we illustrate through extensive empirical experiments the efficiency of HOT *a*SAX with the classic technique HOT SAX in terms of the number of times the Euclidean distance function is called. In our experiments, we tested on 4 datasets, including Random Walk and Tickwise for the highly Gaussian distribution datasets, Koski ECG and Muscle Activation for the lack of Gaussian distribution datasets with the randomly extracted time series of lengths 1k, 2k, 4k and 8k. For the sake of fair comparisons, we hard code the alphabet size of 4, the word size of 4 for the lengths of time series discords [64, 128, 256] and target the top-*1* discord. Each of the experiment is repeated 10 times and the average of the results is taken.

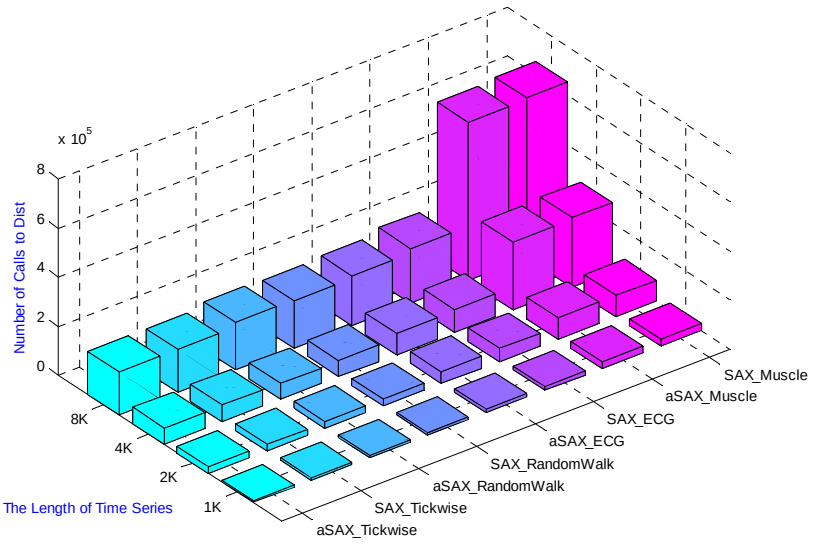

Fig. 4. Pruning power comparison (HOT *a*SAX and HOT SAX) for the discord length 128.

Figure 4 illustrates a competitive result of the two techniques, including HOT *a*SAX and HOT SAX in finding discords of length 128 over a range of data sizes for 4 representative datasets. It is clear that HOT *a*SAX approach produces better pruning power than the classic technique, especially on the lack of Gaussian distribution datasets. This is because of the clustered feature of the adaptive "breakpoints", as confirmed in Section 3.2.

To make sure that the efficiency of HOT *a*SAX for various discord lengths, we repeat such experiment on the lengths of discord to [64, 128, 256] with the time series database size of 8k on 4 datasets above, as shown in Figure 5.

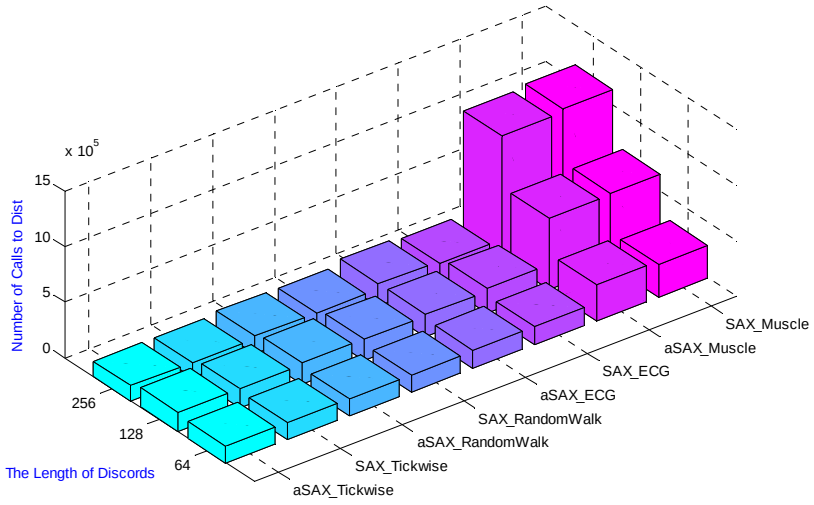

Fig. 5. Pruning power comparison (HOT *a*SAX and HOT SAX) for various discord lengths.

In general, the results show that the pruning power of HOT *a*SAX greatly outperforms or at least is comparable to HOT SAX approach, expressing the superiority in the discords of length 128 and 256 on the lack of Gaussian distribution datasets. Likewise, again, these experiments indicate that we should refer to the clustered feature more than equiprobable feature of real time series data for finding discords.

5 Conclusion

In this paper, we introduced a novel adaptive symbolic representation *a*SAX based on the clustered feature of real datasets and propose HOT *a*SAX algorithm for time series discords discovery. We have shown that our heuristic algorithm is competitive

with, or superior to, the classic approach for finding discords in the large scale time series. Note that we can easily improve the quality of *a*SAX representation by clustering each PAA segment separately. That means we have each adaptive "breakpoints" for each PAA segment, thus suffer more processing and storage cost but greater pruning power. Furthermore, it is easy to apply *a*SAX to another technique such as WAT [1] for larger speedups in time series discords discovery.

There are a number of challenging time series data mining tasks such as motif discovery, clustering, and classification which we intend to extend in future work. In addition, for the lack of Gaussian distribution datasets, the large speedups obtained may be still insufficient. We therefore plan to investigate an anytime heuristic algorithm for real time interaction.

References

1. Bu, Y., Leung, T.-W, Fu, A., Keogh, E., Pei, J., Meshkin, S.: WAT: Finding Top-*K* Discords in Time Series Databases. In: Proceedings of the 7th SIAM International Conference on Data Mining, pp. 449–454, USA (2007)
2. Chan, K., Fu, A.: Efficient time series matching by wavelets. In: Proceedings of ICDE, pp. 126-133, Australia (1999)
3. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time series databases. In: Proceedings of ACM SIGMOD, pp. 419-429, USA (1994)
4. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. In: Journal of Knowledge and Information System, pp. 263-286 (2000)
5. Keogh, E., Lin, J., Fu, A.: HOT SAX: Efficiently finding the most unusual time series subsequence. In: Proceedings of ICDM, pp. 226–233, USA (2005).
6. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. In: Journal of Data Mining Knowledge Discovery, pp. 107-144 (2007)
7. Lloyd, S. P.: Least squares quantization in PCM. In: Proceedings of IEEE Transaction on Information Theory, pp. 129-137 (1982)
8. MacQueen, J. B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297 (1967)