Discriminative Motif Analysis for Time Series Classification

Chaw Thet Zan and Hayato YAMANA
Graduate School of Fundamental Science and Engineering
Waseda University
Tokyo, Japan
E-mail: chawtzan@fuji.waseda.jp, yamana@waseda.jp,

Abstract Time series classification is one of the major works in data mining community. Classification generally works on original real-valued time series or transformed time series. The main issue of time series classification is computational complexity handling with massive amount of time series. In this work, time series are classified using motifs as feature vectors. Variable length motifs are discovered on symbolic representation with our proposed positional inverted index. We further investigated on the candidate motifs with Information Gain (IG) measure for its discriminative features. Then, the classification accuracy of motifs with and without its discriminative features on UCR benchmark datasets are analyzed. As experimental evaluation, motif with its discriminative features achieved on 7 out of 11 datasets.

Keyword Motif discovery, Time series classification, Symbolic representation

1. Introduction

Massive amounts of time series data are generated day by day in most of the real-world application domains such as financial assessment, weather monitoring, medical data examination, and multimedia systems. Handling such kind of data efficiently and finding hidden knowledge from long time series have gotten much attention in data mining community in these two decades. Discovery of repeated patterns, known as motifs, from long time series can provide not only hidden knowledge to each domain but also be used as features for other time series tasks such as clustering, classification, anomaly detection and so on.

As the time series classification is one of the major task in data mining community, many researchers proposed different classification algorithms in different perspectives. One nearest neighbor (1NN) classification with a kind of distance measure is popular choice for its simplicity and robustness [19]. On the one hand, whenever test data comes, it requires to compute for all trained data so that it becomes computationally expensive. As the alternatives, shapelet-based [1,6,13,15,21] and dictionary-based [10,16,17] time series classification methods were proposed. Shapelet is the time series subsequences which are maximally representative of a class [21]. Shapelet-based method classifies unlabeled time series by computing its similarity to the candidate shapelets. Though it is interpretable and accurate, the original shapelet-based classification is computationally expensive. In the dictionary-based classifiers, SAX-VSM [16] classifies unlabeled time series by calculating tf*idf weight vectors for each class. Since there are no pruning techniques for weight vectors in trained data, it has large sparse matrix to consume large space and time.

In this paper, we propose a motif based time series classification method based on the following ideas.

1. Variable length motifs are discovered on the Symbolic representation with our proposed positional inverted index [23].
2. Candidate motifs are analyzed with information gain (IG) for its discriminative features.
3. Discriminative motif patterns are classified with MNB and SVM classifiers on UCR time series benchmark.

The rest of the paper is organized as follows. Section 2 briefly discusses related work for motif discovery and time series classification. Section 3 describes background of our approach and its notations. Section 4 explains our motif discovery approach, i.e., how it works and how to apply motifs as feature vector for time series classification. In section 5, we perform the evaluation of classification accuracy on UCR [2] time series data and compare with 1NN (Nearest Neighbor) classification result. Finally, we conclude our work by discussing future directions in section 6.

2. Related Work

2.1 Motif Discovery

In the time series domain, algorithms for motif

discovery are classified into two types: exact and approximate. Solving a problem exactly is always more desirable than solving it approximately [12]. Exact motif discovery performs on real data directly with pre-defined motif length which requires significant domain knowledge. Even we can find out the different length of motifs iteratively by varying the length, the computational complexity for such kind of solution is considerable. On the other hand, Approximation methods transform the original time series data into low dimensional space to speed up the discovery of motifs.

Mueen et al. [14] proposed efficient exact motif discovery algorithm called "MK" based on Brute Force (BF) motif discovery. Brute force (BF) calculates similarity distance between two-time series and maintains the distance. It is updated whenever the algorithm finds out the smaller distance of another pair of time series. As BF calculates every possible combination of time series pairs, its time complexity is quadratic. MK speeds-up the BF by randomly picks up multiple time series called reference (*refs*) points among a set of time series. Distances between *refs* and a set of time series are pre-computed and stored as an array. Even MK reduced the computation time, its performance varies on data distribution and numbers of reference points. Dau and Keogh proposed motif discovery in cooperation with domain knowledge called Matrix Profile V [5] in recent. They claimed that there is a gap between user expectations and the outcomes of motif discovery. The reasons behind are explained and proposed a guided motif search framework based on [22]. Annotation Vector (AV) is created for a guided motif search which encodes the user's domain-dependent bias. Even it achieves to discover the meaningful motifs as user's expected outcomes, respective domain knowledge is required.

Chiu et al. [4] discovered time series motifs using SAX with Random Projection (RP). Its time complexity is O(n) where n is the length of a given time series. Li et al. [8] developed motif visualization system using grammar induction. Even through this visualization tool discovered variable length motifs, it cannot guarantee the motifs as significant because of weakness in their rule of ranking and pruning technique. Besides, the discovered motifs are fixed length, not variable length motifs. [11] and [18] discovered motifs on multi-dimensional time series data using SAX with Minimum Description Length (MDL) principle and "Trie" data structure. Although different lengths of motifs are found out on real applications such as weather prediction and human motion detection, their approaches do not guarantee the discovered outcomes.

2.2 Time Series Classification

Time series classification generally works on real-valued time series or transformed time series. Most of the classification works on real-valued time series performed on 1NN classifier with different types of distance measures. As an alternative way, features are extracted on real-valued data or transformed data and classification works on extracted features vectors. Shapelet-based classification [13,15,21] works on time series subsequences as feature vectors and applied it on classifying multidimensional time series [1]. Bag-of-word patterns [10] works on SAX word array by window-size, and constructs a histogram over the word distribution. Classification of unlabeled time series is done by 1NN with Euclidean distance with histogram.

3. Background and Notations

In this section, we briefly describe the background knowledge of our approach, notations and definitions used in our proposed approach.

3.1 Symbolic Aggregate Approximation

Symbolic Aggregate approXimation (SAX) is the first proposed powerful dimensionality/numerosity reduction and lower bounding approach in the time series domain. Because of the main characteristics of time series data, i.e., curse of dimensionality, it is indispensable to transform raw time series data into another low dimensional representation for minimizing computational cost. SAX transforms a time series of length n into the string of arbitrary length by using a represented symbol size a ($a \geq 2$).

SAX operates in two main steps. In the first step, a given time series is normalized with mean of zero and standard deviation of one. The normalized time series is transformed into Piecewise Aggregate Approximation (PAA) [7]. In PAA representation, time series T of length n (n-dimensions) is divided into w-dimensional equal-sized segments where ($w << n$). In brief, PAA reduces the data from n-dimensional space to w-dimensional space represented by segment-wise mean value called PAA coefficient. In the second step, PAA coefficients are mapped into alphabetic symbol a of arbitrary size, where $a \geq 2$, using a lookup table that contains "breakpoints" for separating the symbols as shown in Table 1 where the symbol size is 3 to 7. Breakpoints are a sorted list of numbers $B=<\beta_1, \ldots, \beta_{a-1}>$ such that the area under a N(0,1) Gaussian curve β_i to $\beta_{i+1} = 1/a$ (β_0 to β_a are defined as $-\infty$

and ∞, respectively) [9]. Figure 1 illustrates the mapping of the 2-breakpoints with 3 alphabetic symbols where PAA coefficients that are below the smallest breakpoints are mapped with symbol "a", all coefficients greater than or equal to the smallest breakpoints and less than the second smallest breakpoints are mapped to the symbol "b" and all the coefficients greater than or equal to the second smallest breakpoint are mapped to the symbol "c".

Table 1: Lookup table for breakpoints for separating the symbols (symbol size is from 3 to 7)

β/a	3	4	5	6	7
β1	-0.43	-0.67	-0.84	-0.97	-1.07
β2	0.43	0	-0.25	-0.43	-0.57
β3		0.67	0.25	0	-0.18
β4			0.84	0.43	0.18
β5				0.97	0.57
β6					1.07

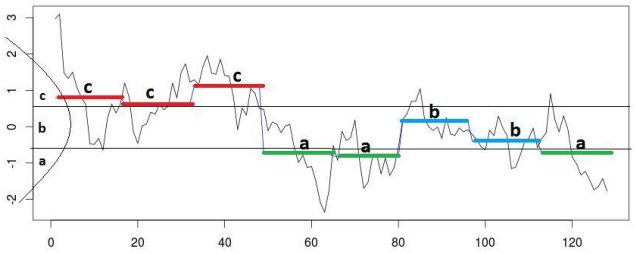


Figure 1. An Example of SAX Representation (A time series of length 128 is mapped to the word "cccaabba" where the segment size (*w*) is 8 and symbol size (*a*) is 3.)

3.2 SAX Numerosity Reduction

One of the main characteristics of SAX is numerosity reduction [8]. We explain briefly how it works here. There is a long-time series *T* of length *n*, and the proper subsequences of length *m* where ($m \leq n$) are extracted using sliding window technique. Each subsequence is discretized into a SAX word using two parameters (segment size *w* and symbol size *a*). During the discretization process, if a word occurs repeated consecutively, instead of recording every same word, the first occurrence of the word and its offset is only recorded [8]. The word array *SA* and how it performs the numerosity reduction are illustrated in Figure 2.

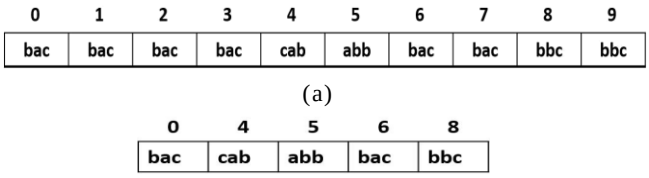


Figure 2. SAX numerosity reduction (a) original word array SA (b) numerosity reduction of (a)

3.3 Notations

Definition 1. Time Series: A time series *T* of length *n*, $T=t_1,\ldots,t_n$,is an ordered set of *n* real-values.

Most of the time series data is very long due to in order with real-time time stamps. We put our attention on local properties of time series data that means dividing sequence by sequence rather than the entire time series. Finding out subsequence matching is more interesting than the whole matching as hidden knowledge can be extracted interval by interval. To find out the subsequence matching, we define the term sliding window and time series subsequence which is extracted by using sliding window shifted by one by one.

Definition 2. Subsequence: Given a time series *T* of length n, a **subsequence** *S* of length *m* of *T*, where $m \leq n$, is a contiguous position starts at p, that is $S=t_p,\ldots,t_{p+m-1}$ for $1 \leq p \leq n-m+1$.

Definition 3. Sliding window: Given a time series *T* of length *n* and the sliding window of length *m*, all possible subsequences can be extracted by sliding a window of length *m* across *T* and considering each subsequence S_P for $1 \leq p \leq n-m+1$.

In this work, each subsequence is transformed into SAX representation using two parameters: word size (or segment size) (w) and symbol size (a).

Definition 4. Word: Each subsequence of sliding window is divided into *w* segments each of which is represented as an alphabetic symbol. Then a set of alphabetic symbols that organizes a subsequence of sliding window is called a word. Thus the number of segments in a subsequence is equal to the size of word, i.e., the number of alphabets in a word.

In the definition of matching, there are exact and approximate matching schemes especially in symbolic representation. SAX representation uses minimum distance function [9] between the original time series of two words, which we call approximate matching.

Definition 5. Match: Given a string array *SA* containing two subsequences $p_{i,l}$ and $q_{j,l}$ of the same length *l*, starting

from different positions (i≠j) and is not allowed overlapping positions each other, it is called non-overlapping. If both words are exactly same with the above non-overlapping condition, we called it as exact match or non-trivial match.

On the contrary, trivial match is defined as follows:

Definition 6. Trivial Match: Given a string array *SA* containing two subsequences $p_{i,l}$ and $q_{j,l}$ of the same length *l*, there is overlapping each other on different positions once or more even their starting positions are different (i≠j), it is called overlapping. If both words are exactly same with above overlapping condition, we called it as trivial match.

Definition 7. Word Motif: Given a string array *SA* and minimum frequency threshold *minfreq* (at least 2), a subsequence *p* of length of *l*, where $l \leq (|SA|)/2$, is defined as motif if the occurrence frequency of *p* in *SA* is *minfreq* and above.

4. Our Approach

There are two steps called variable length motifs discovery and classification with candidate motifs as feature vectors. Our idea is to find out variable length motifs using our proposed positional inverted index [23]. Candidate motifs are analyzed for its discriminative features by adopting Information Gain (IG) measure. Finally, discriminative motif patterns are trained with machine learning based classification algorithms SVM and MNB.

4.1 How SAX was Adopted?

Real-valued time series data is transformed to SAX representation in pre-processing step. Overlapped subsequences are extracted by using sliding window of length *m* across the original time series data, because non-overlapping subsequences would result in too much loss in information [8]. Then, each subsequence whose length of *m* is discretized into a set of "symbols" by using two parameters: word size and symbol size to form a word. During the discretization step, the main feature of SAX, numerosity reduction [9], is applied.

Here, we explain how numerosity reduction is applied by using an example shown in Figure 3. Figure 3(a) represents real-valued time series of length 25 with sliding window of length 6, where offset value shows positional index of original time series data. After adopting sliding window, we have a set of subsequences whose length is 6. Then each subsequence is discretized into a set of "symbols" where word size is 3, which means each subsequence is discretized into three. Thus, every two real-valued data forms one symbol and 3 symbols forms a word. In this example, we set symbol size as 3 and represent real-values by using three symbols: "a", "b", and "c". Figure 3(b) shows the overlapped SAX representation in which neighboring words share some original real-valued time series data where an offset value represents each start position of SAX word.

The next step is numerosity reduction. As proposed in [8], neighboring subsequences, i.e., neighboring words, are likely to be similar to each other because they have some overlapping. Thus, instead of keeping all the words, we keep only the first occurrence of each word which continuously appears in the overlapped SAX representation as shown in Figure 3(c). By only keeping them, we can keep away from discovering trivial matches defined in Section 3.3. Then, we put the positional location value to each word from its first occurrence offset.

4.2 Motif Discovery with Positional Inverted Index

Our proposed motif discovery adopts a term-inverted-index. A term-inverted-index is mainly used in Information Retrieval (IR) and it was originally designed for discrete data. To apply it on real-valued time series data, transformation from real-value to discrete value is required as preprocessing step.

Let us find out variable lengths motifs using *positional inverted index*. At first, we scan the original word array shown in Figure 3(c) once for building 1-word inverted index shown in Figure 4(a). In the inverted index, each start word is kept as a key and its positional locations are sorted and recorded as values. If the number of positional occurrence of each start word is less than a pre-defined minimum occurrence frequency (*minfreq*=2 in this example), the key is filtered out from the inverted index. There is one word "acc" to be filtered out in this example.

For building 2-word inverted index, we have to check positional locations of every combination of two lists in 1-word inverted index. In the example of Figure 4(a), i.e., 1-word inverted index, all the permutation to check are: bac-cab, bac-abb, bac-bbc, cab-bac, cab-abb, cab-bbc, abb-bac, abb-cab, abb-bbc, bbc-bac, bbc-cab, bbc-abb. Let's think about the case where we compare two lists {0,4,10,13} for "bac" and {1,5,11} for "cab" to extract the candidates for 2-word inverted index. We prepare two pointers: p1 and p2, where p1 initially points the start positional location of "bac" list, i.e. {0}, and p2 initially points the start positional location of "cab" list, i.e. {1}. When the value of positional location pointed by p1 is just before the value

of positional location pointed by p2, we record the sequence as a key of 2-word inverted index. Then only the keys whose occurrences are larger than or equal to *minfreq* are kept. Thus, in Figure 4(b), “bacabb”, “cababb”, “abbbbc”, and “bbcbac” are kept for 2-word inverted index before the refinement explained in 4.3.

Now, we can generalize the scheme. For building (n+1)-word inverted index where n is larger than or equal to 1, we have to check positional locations of every combination of (n)-word inverted index and 1-word inverted index. We use two pointers p_n and p_1 where p_n initially points the start positional location of each list in (n)-word inverted index and p_1 initially points the start positional location of each list in 1-word inverted index. Here, when we compare two positional locations pointed by p_n and p_1, it is enough to find out the case where (the value pointed by p_1) minus (the value pointed by p_n) is equal to (n), because the key length of (n)-word inverted index is (n). Then, only the new keys of (n+1)-word inverted index whose occurrences are larger than or equal to *minfreq* are kept.

Table 2. Algorithm for positional index arrangement

Input: key_n, key_1, inList_n, inList_1, p_n, p_1, minFreq
//key_n, inList_n: key and sorted positional locations for (n)-word key-values
//key_1, inList_1: key and sorted positional locations for 1-word key-values
//p_n: pointer initially pointed to the start position of inList_n
//p_1: pointer initially pointed to the start position of inList_1
//minFreq: minimum occurrence frequency (2 in the example of Figure 4)
Output: answer //key-values pairs for (n+1) word	

1	answer = NIL
2	tKey= key_n + key_1 // (n+1)-word key tmpList = NIL
3	**while**(p_1 ≠ NIL **and** p_n ≠ NIL) **do**
4	**if**((inList_1(p_1) – inList_n(p_n)) == n)
5	**then add**(tmpList, inList_n(p_n)) p_n ← **next**(p_n) p_1 ← **next**(p_1)
6	**else if**(inList_n(p_n) > inList_1(p_l)) **then** p_1 ← **next**(p_1) **else** p_n ← **next**(p_n) **end if** **end if**
7	**end while**
8	**if (**|tmpList| ≥ minFreq) **then add**(answer, (tKey, tmpList)) **end if**
9	**return** answer

Table 2 describes how positional inverted index works. Figure 4 illustrates how our motif discovery scheme works with positional inverted indices.

4.3 Motif Refinement

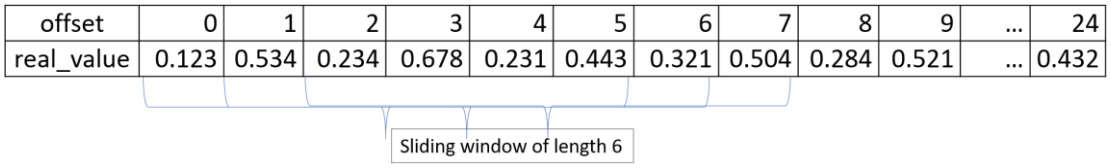
Motif refinement is required to prune off the overlapped positions both in (n)-word motifs and among different word-motifs. Two types of refinement are proposed in this work, called horizontal refinement and vertical refinement.

Horizontal refinement examines whether there are overlapped positions within a list of (n)-word key-values pair. It is an extended version of numerosity reduction to keep away from discovering trivial matches in the same (n)-word motif. The algorithm for horizontal refinement is shown in Table 3.

For clear understanding, an example is presented as follows. There are four 2-word motifs in Figure 4(b). The first motif starts with "bac" and its word length (the number of words) is two. Its occurrence frequency is three and their positional location starts at {0,4,10}. We then confirm there are enough gap between their offset values because we do not want to count the same motif twice in a same sliding window. According to the original word array shown in Figure 4(c), for the first motif "baccab", its positional location of first occurrence starts at "0"(offset value starts at "0") and ends at "1" (offset value ends at "2" because we had “cab” at offset 1 and 2.). The positional location of second occurrence starts at "4" (offset value starts at "6") and ends at "5" (offset value ends at "8"). Then the gap between these two same motifs becomes 4 (i.e., 6-2) which is less than the given window size 6 so that we can confirm that there exists a trivial match to prune off. If the gap (subtracting the end offset value of first occurrence from the start offset value of second occurrence) is above or equal to the length of sliding window, we keep it as independent occurrence of the same motif. Since the trivial match exists between the first occurrence and second occurrence of the first motif "baccab", we discard the second occurrence so that the occurrence frequency of the motif "baccab" becomes two which starts at {0, 10} after the horizontal refinement.

The next step is vertical refinement. Vertical refinement examines whether there exist overlapping positional locations or not in two different (n)-word key-values pairs. Overlapped locations are pruned off after the examination.

At first, we examine whether there exist (n)-word key pairs: key1 and key2, where the postfix word of key1 is



(a)

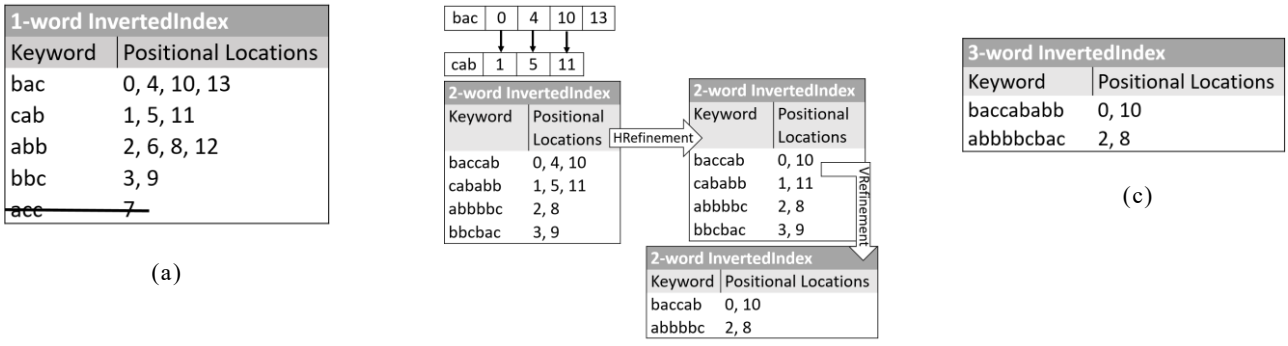
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
SAX Word	bac	cab	cab	abb	bbc	bbc	bac	bac	cab	abb	abb	acc	abb	bbc	bbc	bac	cab	abb	abb	bac

(b)

positional location	0	1	2	3	4	5	6	7	8	9	10	11	12	13
offset	0	1	3	4	6	8	9	11	12	13	15	16	17	19
numerosity reduced SAX	bac	cab	abb	bbc	bac	cab	abb	acc	abb	bbc	bac	cab	abb	bac

(c)

Figure 3. SAX representation scheme (a) Real-valued time series of length 25 with sliding window of length 6 (b) SAX representation for each subsequence with wordSize of 3 and symbolSize of 3 (c) Numerosity reduction of (b)



(a)

(b)

(c)

Figure 4. Motif discovery using positional inverted index (a) 1-word inverted index (b) 2-word inverted index with refinement scheme (c) 3-word inverted index

same as the prefix word of key2. If such key pairs exist, in the next step, we examine their positional locations to confirm the two motifs are concatenated or not. If concatenated, we can confirm that their positional locations are overlapped to prune off the positional location of the latter motif. It is enough to confirm the concatenation between postfix words and prefix words, because only one new word is added to the last position of (n-1)-word keys, which do not have any concatenations, to build (n)-word keys as described in Section 4.2.

For clear understanding, as an example in Figure 4(b) for 2-word motif, key pairs of "baccab" and "cababb" shares the word “cab” that means the postfix word of “baccab” is same as the prefix word of “cababb”. Then, their locations lists {0,10} and {1,11} are examined for overlapping. Because of 2-word motif, "bac" starts from positional location 0 and ends at 1 while "cab" starts from positional location 1 and ends at 2. Both words overlap at the position “1” and again overlap at position 10. Thus, the motif “cababb” is able to be totally pruned off because all the positional locations of “cababb” are overlapped with another motif “baccab”. If some of the positional locations were overlapped to be pruned off but not all, “cababb” might be still candidate motif. In that case, occurrence frequency of positional locations for “cababb” must be greater than or equal to pre-defined minimum occurrence frequency (in this example, it is set as 2). Then, “baccab” is kept as a candidate motif but “cababb” is discarded. The procedure for vertical refinement is described in Table 4.

Table 3: Algorithm for horizontal motif refinement

Input: inList_n, wordList, windowSize
//inList_n: sorted positional locations for (n)-word key-values, where inList_n(i) returns i-th positional location of (n)-word key.
//wordList: SAX word array with offset values, where wordList(i) returns the offset value of i-th positional location of (n)-word key.
//windowSize: sliding window length
Output: answer //set of positional locations	

1	answer = NIL **add** (answer, inList_n(0))
2	**for** i = 0 **to length**(inList_n) -1
3	end_idx=inList_n(i)+n //(end position+1)of current motif st_idx = inList_n(i+1) //start position of next motif

4	midx1= wordList(end_idx)−1 //last offset value of current motif
	midx2 = wordList(st_idx) //offset value of next motif

5	**if** ((midx2 – midx1) ≥ windowSize) **then add**(answer , inList(i+1)) **end if**
6	**end for**
7	**return** answer

4.4 Discriminative Motifs Selection

We discovered different length of candidate motifs from our motif discovery method. There would be a lot of candidate motifs based on data sets. Our idea is to find out the discriminative features from candidate motifs and trained the resultant motifs as feature vectors using Multinomial Naïve Bays (MNB) and Support Vector Machine (SVM).

In order to find out discriminative motif patterns, we adopted Information Gain (IG) which is mostly used in feature selection for text categorization [3,20]. It measures the number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document [20]. The information gain of a term *t* is defined as follows:

$$G(t) = -\sum_{i=1}^{m} P_r(c_i) log P_r(c_i) + P_r(t) \sum_{i=1}^{m} P_r(c_i|t) log P_r(c_i|t) + P_r(\bar{t}) \sum_{i=1}^{m} P_r(c_i|\bar{t}) log P_r(c_i|\bar{t}). \quad (1)$$

where the first term $P_r(c_i)$ is the probability of a category in the target data set. The second and third term are the conditional probability of a category given a term *t* with and without including in it.

We calculate the IG for each term in the target data set and sorted them in descending order. Information gain threshold (for example: top 70 out of 100) is set based on the dataset. We analyzed the effectiveness of information gain on different target data sets on training time and classifies the unlabeled time series data.

Table 4: Algorithm for vertical motif refinement

Input: key1_n, key2_n, inList1_n, inList2_n, p1, p2, minFreq
//(key1_n, inList1_n),(key2_n, inList2_n): two pairs of key and sorted positional locations from (n)-word key-values
//p1: pointer initially pointed to the start position of inList1_n
//p2: pointer initially pointed to the start position of inList2_n
//minFreq: minimum occurrence frequency (2 in the example of Figure 4)
Output: answer //a set of (key and sorted positional location list)	

1	answer = NIL
2	**if**(postfix_word(key1_n) == prefix_word(key2_n)) **then add** (answer, (key1_n, inList1_n))
3	**while** (p1< |pList1_n| **and** p2< |pList2_n|) **do**
4	**if**(inList2_n(p2) == (inList1_n(p1) +n-1)) **then delete** (inList2_n(p2)) p1 ←**next**(p1) p2 ←**next**(p2) **else if**(inList2_n(p2) > (inList1_n(p1)+n-1)) **then** p1← **next**(p1) **else** p2 ← **next**(p2) **end if** **end if**
5	**end while**
6	**if** (|inList2_n| ≥ minFreq) **then add** (answer, (key2_n, inList2_n)) **end if**
7	**end if**
8	**return** answer

5. Experimental Evaluation

In this section, we report experimental result of our motif discovery by using UCR [2] time series benchmarks. We compare the classification accuracy of our approach with 1NN classifier with Euclidean distance. Our method is implemented in Java and all experiments have been done on a machine with a 2.2GHz core i7 processor and 8GB RAM running the 64-bit Windows operating system.

5.1 Analysis of motif based classification accuracy

Variable length motifs are discovered on symbolic representation using our proposed motif discovery method. We analyzed the candidate motifs with and without using information gain for its discriminative features. Discriminative motif patterns are trained with MBN and SVM classifiers, and then unlabeled time series are classified with them. Table 5 shows the characteristic of datasets used in the evaluation. Table 6 shows the comparison of classification accuracy with 1NNED classifier and our proposed method with MNB and SVM classifier with and without IG.

As shown in Table 6, we have evaluated classification accuracy on 11 UCR datasets [2]. 1NNED based classifier performed on real-valued time series data. On the other hand, our motif based classifiers worked on symbolic representation with motif as feature vectors. We analyzed the classification accuracy of motif-based approach on two

ways: with and without discriminative measure called information gain (IG).

Here, there are 3-parameters called window size, word size and symbol size for our motif discovery method. Window size mostly depends on the time series length. In this experiment, we defined word size and symbol size between (3 to 8). The best parameters are shown in Table 6.

We calculated IG on candidate motifs and reduced the number of candidate motifs for its discriminative features. Among the 11 datasets, 3 datasets gained its classification accuracy nearly 100 percent with IG and reduced its training data size. ECG 200 datasets cannot be applied IG because its category distribution is skewed and spike noise signals are included in it. The other datasets can classify well with IG measures. In comparison with 3 methods, i.e., 1NNED, motif with IG and without IG, 1NNED wins on 5 datasets, motif without IG wins on 6 datasets and motif with IG wins on 7 datasets. By applying IG on candidate motifs, both classification accuracy and its training time boosts up.

Table 5 Characteristics of 11 UCR datasets [2]

No	Name	#of class	Train size	Test size	Time series length
1	Two Lead ECG	2	23	1139	82
2	ECG200	2	100	100	96
3	ECG Five Days	2	23	861	136
4	CinC ECG Troso	4	40	1380	1639
5	Coffee	2	28	28	286
6	CBF	3	30	900	128
7	Wafer	2	1000	6174	152
8	Yoga	2	300	3000	426
9	Trace	4	100	100	275
10	Face four	4	24	88	350
11	Olive Oil	4	30	30	570

6. Conclusion

In this work, we presented the motif discovery method with IG for its discriminative features and analyzed its classification accuracy with and without discriminative feature. As the experimental evaluation, discriminative motif pattern works well for most of the datasets and it reduces the size of training data and boosts up training time and classification accuracy. As the future work, we plan to analyze our approach on different datasets, massive amount of times series data and other discriminative features.

Table 6 Comparison of classification accuracy on 1NNED, motif based MNB and SVM with and without information gain (IG)

No	1NN ED	Without IG		With IG			Window Size	Word Size	Symbol Size
		Motif based MNB	Motif based SVM	Motif based MNB	Motif based SVM	Reduced training data by IG (%)			
1	0.747	0.840	**0.841**	0.840	**0.864**	50	20	5	3
2	**0.880**	0.710	0.780	0.710	0.780	0	20	5	3
3	0.797	0.730	**0.816**	0.720	**0.810**	10	20	5	3
4	**0.897**	0.860	0.867	0.814	0.847	50	40	5	4
5	**1.000**	0.964	0.893	**1.000**	**1.000**	20	48	5	3
6	0.852	**0.964**	0.942	**0.956**	0.931	20	32	4	5
7	**0.995**	0.976	0.990	0.956	0.977	20	25	5	4
8	**0.830**	0.700	0.719	0.696	0.729	10	80	6	5
9	0.760	0.830	**0.940**	0.800	**0.930**	30	48	4	8
10	0.784	**0.977**	0.863	**0.966**	0.852	20	60	5	4
11	0.867	**0.933**	0.967	0.933	**1.000**	20	50	5	6

References

[1] A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. In Proc. 17th DaWak, 2015.

[2] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The UCR time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time series data/.

[3] H.Cheng, X. Yan, J. Han and C. W. Hsu. Discriminative Frequent Pattern Analysis for effective Classification. In Proc. of ICDE, 2007.

[4] B. Y. chi Chiu, E. J. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In Proc. of 9th ACM SiGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pages 493–498, 2003.

[5] H. A. Dau and E. J. Keogh. Matrix profile v: A generic technique to incorproate domain knowledge into motif discovery. In Proc. of the 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, 2017.

[6] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets, In

Proc. 20th SIGKDD, 2014.

[7] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems, 3(3):263–286, August 2001.

[8] Y. Li, J. Lin, and T. Oates. Visualizing variable-length time series motifs. In Proc. of the 2012 SIAM Int'l Conf. on Data Mining, pages 895–906, 2012.

[9] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. Data Mining and Knowledge Discovery, 15(2):107–144, October 2007.

[10] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems, 39(2):287-315, 2012.

[11] A. McGovern, D. H. Rosendahl, K. K. Droegemeier, and R. A. Brown. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. Data Mining and Knowledge Discovery, 22(1-2):232–258, 2011.

[12] A. Mueen. Time series motif discovery: dimensions and applications: WIREs Data Mining Knowl Discov, 4(2): 152-159, 2014.

[13] A. Mueen, E. Keogh, and N.Young. Logical-shapelets: An expressive primitive for time series classification. In Proc. 17th SIGKDD, 2011.

[14] A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, and M. B. Westover. Exact discovery of time series motifs. In Proc of the 2009 SIAM Int'l Conf. on Data Mining, pages 473–484, 2009.

[15] T. Rakthanmanon and E. Keogh. Fast-shapelets: A fast algorithm for discovering robust time series shapelets. In Proc. 13th SDM, 2013.

[16] P. Schaffer. The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery, 29(6): 1505-1530, 2015.

[17] P. Senin and S. Malinchik. SAX-VSM: interpretable time series classification using sax and vector space model. In Proc. 13th IEEE ICDM, 2013.

[18] Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of time-series motif from multi-dimensional data based on mdl principle. Machine Learning, 58(2-3):269–300, 2005.

[19] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. Data Mining and Knowledge Disovery, 26(2): 275-309, 2013.

[20] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In Proc. of ICML, pages 412-420, 1997.

[21] L. Ye and E. Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Mining and Knowledge Discovery, 22(1-2): 149-182, 2011.

[22] C.C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. J. Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In Proc. of the IEEE 16th Int'l Conf. on Data Mining (ICDM), pages 1317–1322, December 2016.

[23] C. T. Zan and H. Yamana. A Variable-Length Motifs Discovery Method in Time Series using Hybrid Approach. In Proc. of 19th Int'l Conf. on Information Integration and Web-based Applications & Services, December 2017.