ELIS++: a shapelet learning approach for accurate and efficient time series classification

Hanbo Zhang¹ · Peng Wang¹ · Zicheng Fang¹ · Zeyu Wang¹ · Wei Wang¹

Received: 29 June 2020 / Revised: 23 November 2020 / Accepted: 25 November 2020 /

Published online: 9 January 2021

© The Author(s) 2021

Abstract

In recent years, time series classification with shapelets, due to the high accuracy and good interpretability, has attracted considerable interests. These approaches extract or learn shapelets from the training time series. Although they can achieve higher accuracy than other approaches, there still confront some challenges. First, they may suffer from low accuracy in the case of small training dataset. Second, they must manually set some parameters, like the number of shapelets and the length of each shapelet beforehand, and some hyper-parameters, like learning rate and regulation weight, which are difficult to set without prior knowledge. Third, extracting or learning shapelets incurs a huge computation cost, due to the huge search space. In this paper, we extend our previous shapelet learning approach ELIS to ELIS++. To improve the accuracy on the small training dataset, we propose a data augmentation approach. To learn the higher quality shapelets, based on the PAA shapelet candidates search technique proposed in ELIS, ELIS++ first propose a novel entropy-based approach shapelet candidate selection mechanism to discover shapelet candidates, and then applies the logistic regression model to adjust shapelets. To avoid setting other parameters manually, we propose a Bayesian Optimization based approach. Moreover, two techniques are proposed to improve the efficiency, coarse-grained shapelet adjustment and SIMD-based parallel computation. We conduct extensive experiments on 35 UCR datasets, and results verify the effectiveness and efficiency of ELIS++.

✉ Peng Wang
pengwang5@fudan.edu.cn

Hanbo Zhang
16110240029@fudan.edu.cn

Zicheng Fang
zcfang16@fudan.edu.cn

Zeyu Wang
wangzeyu17@fudan.edu.cn

Wei Wang
weiwang1@fudan.edu.cn

¹ School of Computer Science, Fudan University, Shanghai, China

Keywords Time series · Classification · Shapelet

1 Introduction

Time series data are pervasive across almost all human endeavors, including medicine, finance and science. In consequence, there is an enormous interest in querying and mining time series data. Time series classification is a significantly attractive research problem with various real-life applications, ranging from human activity detection [20] to disease detection [30].

1.1 State Of the art

In last two decades, many approaches of time series classification have been proposed, most of which can be categorized into two groups, distance-based approaches and structure-based approaches. The first category is distance-based approach, which uses the raw time series, or subsequences, to build the classification model. Various combinations of similarity metrics and classification models are designed in this category, such as Euclidean distance [29] or Dynamic Time Warping (DTW) [21] distance for KNN classifier, and information gain or F-Stats [15] for the decision tree.

The second category is the structure-based approaches [19, 22, 23]. They transform the raw numeric time series into approximate representations and build classification models based on them, like Symbolic Aggregate approximation (SAX) [14]. These approaches perform efficiently, but they suffer from inaccuracy due to the loss of detailed characteristics. Another limitation is that it is difficult to find the most appropriate approximation granularity.

Within the distance-based category, the shapelet-based approaches attract a lot of attention. As the pioneer work, NSD [30] firstly proposed the concept of shapelet, which is the discriminative subsequence, and builds the classifier based on shapelets. Later, some works have been proposed to improve the accuracy or the efficiency [15, 18]. The advantage of shapelet-based approaches lie in that 1) they are more accuracy than other types of approaches [1], 2) they achieve the high interpretability, since the shape of the subsequence is easy to visualize and understand. These works discover shapelets by directly searching the entire possible subsequence space of the training time series, and then select the most discriminative ones according to their target prediction qualities.

Grabocka et al proposed LTS [7], which *learns*, instead of *searching* for, the shapelets. LTS randomly initializes some shapelets and applies the logistic regression to adjust them. LTS can achieve higher accuracy than other shapelet-based approaches, as well as other types of approaches. The major advantage of learning shapelets is that it can find shapelets which are absent in the training time series. However, its performance depends on the initialized shapelets, which have great impact on the accuracy. Moreover, LTS needs a long training time, since it tunes every data point of each shapelet, which makes a huge parameter space.

1.2 ELIS and ELIS++

Our previous work ELIS (Efficient Learning Interpretable Shapelets) is also a shapelet learning approach, which focuses on the problem of initializing the shapelets [5]. ELIS consists of two phases. The first phase is the shapelet candidate generation phase. ELIS

enumerates subsequences in the training time series, and represents them by PAA [28]. For each class, we rank all PAA words by the TFIDF-style scores. The mechanism of false set is proposed to determine the number of shapelets automatically. The second phase is the shapelet adjustment phase, in which, ELIS utilizes the machine learning approach to tune the shapelets and build the classification model. ELIS achieves higher accuracy and efficiency than LTS. However, ELIS has the following drawbacks. First, it may not work well in the case of small training dataset. The reason is that the training time series will be covered by a small number of shapelet candidates, so some “true” shapelets will be missed. Second, although ELIS can determine the key parameter, the number of shapelets and the shapelet lengths, we still need to manually tune some parameters, like the learning rate and the regulation weight. Third, for the large training dataset, training the model in ELIS is time consuming. The reason is that just as LTS [7], ELIS needs to adjust every data point of each shapelet.

In this paper, we extend ELIS further to ELIS++, which is more accurate and efficient. In the experiments, on 35 datasets of UCR repository [3], ELIS++ is always more accurate than ELIS. The contributions we make in this paper can be summarized as follows.

- To improve the accuracy on the small training dataset, we propose a data augmentation approach (Section 5.1). Considering the high expense of obtaining the labelled data, in many applications, analysts can only get a small training dataset. So improving the effectiveness in the case of small training dataset is important. In ELIS++, we propose a two-step approach to augment the training dataset. A parameter, the augmentation amount, is used to control the augmentation scale. Besides improving the accuracy, our approach even improves the efficiency in most cases because less iteration rounds are needed to converge the model training.
- To address the problem of manually setting the parameters, we propose an approach to set the parameters automatically (Section 5.2). ELIS++ has several parameters, including data augmentation amount Σ , regularization λ , learning rate η and the iteration round I . Tuning them manually is a tedious task. We propose a Bayesian Optimization based approach to set the parameters automatically. Experimental results verify the effectiveness of the proposed approach.
- We propose two techniques to improve the efficiency. First, we propose the coarse-grained shapelet adjustment mechanism (Section 4.4). Instead of adjusting every data point in the shapelets, we adjust fixed number of continuous data points together, which is determined by the adjustment granularity. Except improving the efficiency, it can also improve the accuracy by reducing the overfitting phenomenon of fine-grained adjustment. Second, we utilize the SIMD-based parallel computation to accelerate the model training (Section 4.5).
- To improve the quality of the shapelet candidates, we propose a novel shapelet candidate selection mechanism (Section 3.5). In ELIS, the mechanism of false set was used to evaluate the discrimination ability of candidate sets and determine the number of shapelet number. In ELIS++, we replace it with an entropy-based approach, since the format of the new measurement is consistent with that in the shapelet adjustment stage.
- We expand the experimental evaluation greatly (Section 6). First, we add three baseline approaches, 1NN-cDTW [4], Proximity Forest [17] and the deep learning based approach mWDN [27]. Second, we extend the datasets from 15 to 35.

The rest of the paper is organized as follows. First of all, in Section 2 some essential definitions are presented. The ELIS++ approach are introduced in Sections 3, 4 and 5

respectively. In Section 6, we present the experiment setting and the results. Next, We introduce the related works in Section 7. Finally, we conclude for the paper in Section 8.

2 Preliminaries

In this section, we present the necessary definitions.

Definition 1 (Time Series and Subsequence) A time series T is a sequence of ordered values $T = (t_1, t_2, \dots, t_i, \dots, t_n)$, where n is the length of T . A subsequence of time series T , $(t_b, t_{b+1}, \dots, t_{e-1}, t_e)$, can be denoted as $T[b : e]$ or $T(b, e - b + 1)$, in which b and e are the beginning and ending points of the subsequence respectively.

Definition 2 (Dataset) A dataset D is a set of pairs of time series, T_i , and its class label c_i . Formally, $D = \{ \langle T_1, c_1 \rangle, \langle T_2, c_2 \rangle, \dots, \langle T_N, c_N \rangle \}$, N is the number of time series in D . $C = \{c_1, c_2, \dots, c_{|C|}\}$ is the set of class labels, and $|C|$ denotes the number of labels.

Definition 3 (PAA Sequence) A subsequence $T(b, m)$ can be represented by a sequence $V = (v_1, v_2, \dots, v_w)$ by the PAA. The i -th element of V , v_i , is calculated as follows:

$$v_i = \frac{w}{m} \sum_{j=\frac{m}{w}(i-1)+1}^{\frac{m}{w}i} t_{b+j-1} \quad (1)$$

The main idea of PAA is to replace original subsequence with the mean values of disjoint windows. PAA window number, w , is a parameter that determines the length of the PAA sequence. Figure 1 shows two PAA sequences as w equals 5 and 10 respectively.

For two equal length sequences, we can measure the distance between them by traditional euclidean distance. Moreover, we need to define the distance between two sequences with different lengths.

Definition 4 (Distance Between Two Sequences) The distance between sequences T_u of length L_u and T_v of length L_v ($L_v \geq L_u$), no matter they are time series or subsequences, is denoted as:

$$dist(T_u, T_v) = \min_{j=\{1, \dots, J\}} \frac{1}{L_u} \sqrt{\sum_{l=1}^{L_u} (tv_{j+l-1} - tu_l)^2} \quad (2)$$

where $J = L_v - L_u + 1$, tu_i and tv_i are the i -th value of T_u and T_v respectively.

The PAA word and the SAX word are two symbolic representations for a subsequence. The difference is how to split the range of Y-axis (all the values of time series in D).

Definition 5 (PAA Word) The PAA word is obtained by transforming a PAA sequence into an equal-length sequence of discrete integers. To simplify the expression of the PAA word, we represent all these discrete integers as alphabets, i.e., 0 as “A”, 1 as “B” and so on. Formally, let the minimal and maximal values in time series of D be MIN and MAX respectively. The range $[MIN, MAX]$ is split into d equal-width buckets, each of which is represented by an alphabet. A subsequence of length m is first transformed to a length- w

Figure 1 Example of PAA sequences with $w = 10$ and $w = 5$ respectively

PAA sequence V , and then represented by a PAA word $P_m^w = (p_1, p_2, \dots, p_w)$. The i -th element of P_m^w is calculated as follows:

$$p_i = \lfloor \frac{d \cdot (v_i - MIN)}{MAX - MIN + \delta} \rfloor \quad (3)$$

in which δ is a small enough number compared to $MAX - MIN$.

The transformation is reversible, that is, PAA word P_m^w can be reverted into a length- m numeric subsequence $\hat{T} = (\hat{t}_1, \hat{t}_2, \dots, \hat{t}_m)$. The i -th element of $\hat{T}$ is calculated by the following equation:

$$\hat{t}_i = \frac{MAX - MIN}{d} (0.5 + p_{\lfloor \frac{w(i-1)}{m} \rfloor + 1}) + MIN \quad (4)$$

Based on $\hat{T}$, we define the distance between a PAA word P_m^w and a length- m sequence T' as $dist(\hat{T}, T')$ in (2). Figure 2 shows an example of the PAA word “CDHGGHFIFA” with $w = 10$ and $d = 10$.

Definition 6 (PAA Word Occurrence & Cover) The PAA word P_m^w occurs in T or covers T , if P_m^w can be transformed from certain subsequence $T(b, m)$ of time series T .

Definition 7 (Shapelet) A shapelet S is a time series subsequence which is maximally representative for a class. It semantically represents intelligence on how to discriminate one class against others.

Interpretability is an important property of shapelets. That means a shapelet can be interpreted as a shape characteristic of time series in a class.

Model training module In this phase, we apply the logistic regression model to adjust the shapelets to improve the accuracy. The classifier is initialized based on the shapelet candidates, and then learns better shapelets on the raw numeric time series. More, to avoid the high expense of adjusting each point of the shapelets, as LTS and ELIS, we propose the mechanism of coarse-grained shapelet adjustment. We use the adjustment granularity to control the “fine” extent of shapelet representing the raw time series.

Moreover, we propose a Bayesian Optimization based approach to set the parameters automatically. It first learns a model to represent the relationship between parameters and the classifier accuracy, and then based on the model to find the best parameters.

The proposed ELIS++ approach is outlined in Figure 3. In next sections, we introduce the technical details in turn.

3 Shapelet candidates discovery

In this section, we introduce the shapelet candidate discovery module. Specifically, for each class, we generate the candidates from the PAA word space, and then revert them into the raw numeric space.

3.1 Overview

The rationale of candidates discovery is that if a subsequence appears in one class frequently and other classes occasionally, it is a high-quality shapelet candidate for this class. The challenge is to determine the occurrence for each subsequence. The brute-force approach is too expensive, $O(N^2n^4)$, to process all possible subsequences, where N is the number of time series and n is the length of time series [30]. So we transform subsequences to PAA words to discover candidates efficiently.

Figure 3 ELIS++ Overview

However, we need to solve three problems. First, the PAA representation may cause the problem of mismatching, that is, two similar subsequences may be represented as two different PAA words. Second, we need to keep the number of shapelets as small as possible, which can guarantee the high interpretability and high training efficiency, in the meanwhile, they are sufficiently discriminative. The third one is how to alleviate the problem of parameter tuning. We need to determine the number and length of the shapelets, as well as the granularity of PAA representation, which have a great influence on the accuracy.

In this paper, we propose a three-step approach to discover the shapelet candidates. In the first step, we transform all subsequences in D to PAA words. Unlike the previous structure-based approaches, in which the window number of PAA or SAX word, w , is fixed, we generate the PAA words with multiple granularity for each subsequence. This approach can solve the problem of improper granularity introduced in Section 1. In the second step, we compute the TFIDF score for each PAA word in each class and rank them accordingly. The PAA word with higher score is more discriminative. We propose a concept of fuzzy counting to avoid the mismatching issue. In the third step, for each class, from the sorted PAA words, we use a coverage-based approach to select some of them as the final set of candidates. We use cover gain to measure the contribution of each PAA word, and use entropy to determine the number of candidates to be selected. The selected candidates are different from each other, and the set of them can provide high classification accuracy.

Previous works of shapelets [7, 23] need to set two important parameters, the length of shapelets and the total number of shapelets. Unfortunately, these parameters are difficult to determine in the absence of prior knowledge. In our approach, we rank and select the shapelet candidates of variable lengths together to avoid setting the shapelet length manually. Also, we use entropy to determine the number of shapelets adaptively.

3.2 PAA word generation

First, we generate a set of PAA words from the raw time series. We only select subsequences with length $i \cdot \phi \cdot n$ ($i = 1, 2, \dots$) and transform them to PAA words, where ϕ is a fixed step ratio, such as 0.05. The rationale is that subsequence $T(b, m)$ will be very similar with $T(b, m')$, if their lengths, m and m' , are very similar. In most cases, these two subsequences will have similar discrimination ability.

For each time series T in D , we traverse it from left to right to generate PAA words. Formally, from each beginning point b ($1 \leq b \leq n$), we enumerate subsequences of length $i \cdot \phi \cdot n$ ($i = 1, 2, \dots$), that is, $T(b, \phi \cdot n)$, $T(b, 2 \cdot \phi \cdot n)$, $\dots$. We transform each of these subsequences to multiple PAA words with variable PAA window number w . We transform each subsequence to PAA words with variable window lengths, $w \in \{2, 4, 8, 16, 32, 64\}$, respectively if possible. In fact, the values of w can be set freely. The only thing that needs to be guaranteed is that the values of w can reflect the granularity from coarse to fine.

Moreover, we eliminate the PAA words whose shapes are significantly different from the subsequences they correspond to. In this case, PAA word can not represent the shape of subsequence. For example, assume the subsequence is $(3.5, -3.5, -2.5, 2.5)$ and $w = 2$, PAA sequence will be $(0, 0)$. This PAA word should be dismissed because it cannot represent the shape of the raw subsequence. We use the PAA bucket width, $\epsilon = \frac{MAX-MIN}{d}$,

to constrain the distance. Specifically, for subsequence $T(b, m)$ and the transformed PAA word P_m^w , if the distance between $T(b, m)$ and P_m^w exceeds ϵ , we discard this PAA word to avoid the deformation caused by PAA.

Finally, we insert the PAA words into an inverted table, denoted as H . Each entry in H takes the PAA word as key, and the set of time series indexes in which this PAA word occurs, called cover set, as value. Note that if a PAA word occurs in a time series many times, it should be counted only once.

3.3 TFIDF score

In the second step, we use a TFIDF style score to evaluate how discriminative a PAA word is for a class. The TFIDF score of PAA word P for class c is defined as the product of two factors: term frequency (TF) and inverse document frequency (IDF). The term frequency is defined as:

$$tf_{P,c} = \frac{F_{P,c}}{N_c} \quad (5)$$

where $F_{P,c}$ is the frequency that PAA word P occurs in the time series of class c and N_c is the total time series number of class c . The inverse document frequency is defined as:

$$idf_{P,c} = \log \frac{2}{I_{P,c} + O_{P,c}} \quad (6)$$

in which $I_{P,c}$ and $O_{P,c}$ are defined as:

$$I_{P,c} = \begin{cases} 1, & F_{P,c} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$O_{P,c} = \begin{cases} 1, & F_P - F_{P,c} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where F_P is the frequency that the PAA word P occurs in time series of D .

When computing all the frequencies above, we also add the occurrence times of the similar PAA words, which can solve the problem of mismatching [20]. For example, the occurrence of PAA word “CCDDDB” should also be counted when computing the frequency of the PAA word “CCDDC”. Formally, we define the similar PAA words as follows,

Definition 8 (Similar PAA words) Let $P = (p_1, p_2, \dots, p_w)$ and $P' = (p'_1, p'_2, \dots, p'_w)$ be two different PAA words with the same PAA window number w and raw subsequence length m . P and P' are similar if they satisfy

$$|p_i - p'_i| \leq 1, i = 1, 2, \dots, w \quad (9)$$

After computing all TFIDF scores, we sort the PAA words of each class according to their TFIDF scores in descending order, and maintain the sorted PAA words for each class c in a list, denoted as L_c .

Algorithm 1 PAA word generation.

Input: Training set D , Class set C
Output: Lists of PAA words with TFIDF score $L = \{L_1, L_2, \dots, L_{|C|}\}$

```

1 Function SortedPAAWords( $D, C$ ):
2    $\epsilon = (\max(D) - \min(D))/d$ ;
3    $L = \phi$ ;
4   for  $m = \{0.05 \cdot n, 0.1 \cdot n, \dots, n\}$  do
5     for  $w = \{2, 4, 8, 16, 32, 64\}$  do
6       {Generate PAA words}
7        $H = \phi$ ;
8       for  $i = \{1, \dots, N\}$  do
9         for  $j = \{1, \dots, n - m + 1\}$  do
10           $P = \text{PAA}(T_i(j, m), w, d)$ ;
11          if  $\text{dist}(P, T_i(j, m)) \leq \epsilon$  then
12             $H[P].\text{add}(i)$ ;
13          {Calculate TFIDF score for each PAA word}
14          forall  $P$  in  $H$  do
15            for  $c = \{1, \dots, |C|\}$  do
16               $F_{P,c} = \text{FUZZYCOUNT}(P, c, H)$ ;
17               $F_P = \bigcup_1^{|C|} F_{P,c}$ ;
18              for  $c = \{1, \dots, |C|\}$  do
19                 $\text{tfidf} = \text{TFIDF}(F_P, F_{P,c})$ ;
20                 $L_c.\text{add}(\text{tuple}(P, m, \text{tfidf}))$ ;
21          for  $c = \{1, \dots, |C|\}$  do
22             $L_c = \text{SORTBYTFIDF}(L[c])$ ;
23  return  $L$ ;

```

3.4 Candidate selection

In the third step, we generate the final set of shapelet candidates for each class. We intend to find sufficient candidates to cover class characteristics, and in the meanwhile, guarantee the diversity between candidates. We use two techniques, *cover gain* and *entropy*, to select shapelets for each class.

For class c , we traverse the PAA words from top to bottom in L_c and select some of them to form the candidate set, denoted as Ω^c . Note that for different classes, the number of candidates, denoted as $|\Omega^c|$, may be different. The challenge here is how to determine $|\Omega^c|$ for each class. The rationale behind our approach is to form multiple candidate sets for each class. By evaluating the discrimination ability of each generated candidate set with the *coverage*, we choose the best one as the optimal candidate set. First, we give some definitions.

Definition 9 (Coverage) The coverage, denoted as θ , is a threshold that determines whether the characteristics of a time series is covered by enough candidates. Specifically, if each time series of class c is covered at least θ times by candidates selected from L_c , we believe that the candidates already cover all the features of the time series in class c with threshold θ .

Definition 10 (Cover Gain) The cover gain of a PAA word P equals to the number of time series covered by P whose coverages are lower than θ yet.

For each class c and coverage threshold θ , we generate a candidate set, denoted as Ω_θ^c . We convert the problem of finding out the optimal set of shapelet candidates to the goal of confirming the optimal value for θ . That is, we define a score to measure the discrimination ability of each candidate set, $score(\Omega_\theta^c)$, and select the candidate set with the maximal $score(\Omega_\theta^c)$. We have two tasks, i) find Ω_θ^c for each θ ; 2) define $score(\Omega_\theta^c)$. Next, we introduce them in turn.

We generate Ω_θ^c for each θ as follows. Initially, Ω_θ^c is empty. Then we traverse the PAA words in L_c from top to bottom. For each PAA word P , we have three cases.

1. If P has no cover gain, we simply discard it. It means P is meaningless, since it only covers the time series that have been covered at least θ times by candidates with higher TFIDF scores.
2. P has cover gain, but it is similar with certain candidate, say P' , which is already in Ω_θ^c . In this case, we just add the time series covered by P to those covered by P' . P will not be added into Ω_θ^c . By eliminating overlapped PAA words, we keep the size of the candidate set smaller, which is good for interpretability.
3. If P has cover gain and isn't similar with any candidate in Ω_θ^c , we put it into Ω_θ^c .

This process terminates when all time series are covered at least θ times, or all PAA words in L_c are processed. We discuss the definition of $score(\Omega_\theta^c)$ in the next section.

3.5 The score of Shapelet candidate set $score(\Omega_\theta^c)$

Now we discuss the second task, defining $score(\Omega_\theta^c)$. Let $\Omega_\theta^c = \{S_1, S_2, \dots, S_p\}$, where $p = |\Omega_\theta^c|$. S_j is the j -th shapelet candidate, which is reverted from the j -th PAA word with Eq. 4. The rationale behind our approach is to use $score(\Omega_\theta^c)$ to mimic the classification process with entropy to evaluate the discrimination ability of this candidate set. We first introduce the entropy of one single shapelet, and then extend it to the shapelet set.

Given shapelet S_j , we compute the distances between it and other time series in D , sort the distances, and place them on an ordered line, as shown in Figure 4. The blue square and red circle objects correspond to the time series of different classes. The left objects mean that the corresponding time series has smaller distance with S .

We use entropy to measure the distinguishing power of shapelet S . Specifically, we split all objects on the ordered line into left part and right part by a split point, denoted as SP , as shown in Figure 4. That is, we classify all time series T 's, which satisfy $dist(S, T) \leq SP$ as a group, and the other time series as another group. For any SP , we can compute the entropy based on the time series grouping caused by SP . For each shapelet S_j ($1 \leq j \leq p$), we select a split point with the smallest entropy, which means that it can produce two groups as “pure” as possible. We call this split point as the optimal split point of S_j , denoted as SP_j , and denoted its entropy as ENT_j .

Figure 4 Ordered line example

To define the score of a shapelet candidate set, $score(\Omega_\theta^c)$, we still project all time series into an ordered line, denoted as OL , find an optimal split point SP , and use the entropy to measure the purity of two time series groups. The challenge here is how to project time series onto OL , since the distances of a time series with different shapelets are different.

The basic idea of our approach is to generate the ordered lines for all candidates in Ω_θ^c , and merge them to form the final OL . First, we generate the ordered line, OL_j , for each candidate S_j by computing and sorting the distances between it and all the time series. Then we find the optimal split point SP_j and compute the corresponding entropy ENT_j .

Then we align these ordered lines by aligning the split points SP_j 's. Moreover, we normalize them by extending or shrinking the ordered lines based on the entropy ENT_j 's. Formally, for ordered line OL_j of candidate S_j , the normalized distance between S_j and time series T_i ($1 \leq i \leq N$) is

$$\frac{dist(S_j, T_i) - SP_j}{ENT_j} \quad (10)$$

Algorithm 2 Candidates selection.

Input: L , Training set D , Class set C
Output: Best candidate sets Ω

```

1 Function CandidatesSelection( $C, L$ ):
2   for  $c = \{1, \dots, |C|\}$  do
3      $\Omega^c = \phi$ ;
4      $best = INF$ ;
5     for  $\theta = \{1, \dots, 5\}$  do
6        $\Omega' = \phi$ ;
7        $\Delta = \{0, \dots, 0\}$ ;
8       forall ( $P, m, tfidf$ ) in  $L_c$  do
9          $S = revert(P)$ ;
10        if  $COVERGAIN(S) > 0$  then
11           $diff = True$ ;
12          forall ( $P', m', tfidf'$ ) in  $\Omega'$  do
13            if  $SIMILARCHECK(P, P')$  then
14              {Similar PAA word}
15               $UNIONCOVER(P, P')$ ;
16               $diff = False$ ;
17              break;
18          if  $diff$  then
19            {Find a new candidate}
20             $\Omega_\theta^c \leftarrow (S, m, tfidf)$ ;
21             $d = \{dist(T_1, S), \dots, dist(T_N, S)\}$ ;
22          compute  $score(\Omega_\theta^c)$ ;
23          if  $score(\Omega_\theta^c) < best$  then
24             $\Omega^c = \Omega_\theta^c$ ;
25             $best = score(\Omega_\theta^c)$ ;
26  return  $\Omega$ ;
  
```

We merge all normalized ordered lines to form the final ordered line, OL , on which the position of time series T_i is

$$pos(T_i) = \sum_{j=1}^{|\Omega_\theta^c|} \frac{dist(S_j, T_i) - SP_j}{ENT_j} \quad (11)$$

After obtaining positions of all time series on OL , we find the optimal split point, and compute the entropy, denoted as ENT . We set $score(\Omega_\theta^c)$ as $\frac{1}{ENT}$. The larger $score(\Omega_\theta^c)$, the most discriminative the candidate set Ω_θ^c .

Speedup the computation of $score(\Omega_\theta^c)$ When the number of shapelet candidates and that of time series are large, computing ENT_j by finding the best optimal split point for each shapelet candidate S_j is time consuming. Therefore, we compute $score(\Omega_\theta^c)$ with a more efficient way. By simple transformation of (11), we have

$$pos(T_i) = \sum_{j=1}^{|\Omega_\theta^c|} \frac{dist(S_j, T_i)}{ENT_j} - \sum_{j=1}^{|\Omega_\theta^c|} \frac{SP_j}{ENT_j} \quad (12)$$

Note that the right term in (12) is a constant for all time series. So removing it will not influence the relative positions of all time series on OL . As for the denominator ENT_j , it represents the discrimination power of candidate S_j . To improve the efficiency, we replace it with the $TFIDF$ value of the corresponding PAA word. So, the position of time series T_i on OL is

$$pos(T_i) = \sum_{j=1}^{|\Omega_\theta^c|} \frac{dist(S_j, T_i)}{TFIDF_j} \quad (13)$$

where $TFIDF_j$ is the TFIDF value of the j -th PAA word, as introduced in Section 3.3. Based on the adapted ordered line, we find the optimal split point, compute the entropy and set $score(\Omega_\theta^c)$ as $\frac{1}{ENT}$.

We compute $score(\Omega_\theta^c)$ for variable values of θ . The candidate set with maximal $score(\Omega_\theta^c)$ can best discriminate time series of different classes, so we choose it as the final candidate set. Ties are solved by choosing the one with fewer candidates. Because the fewer candidates we have, the more efficient follow-up operations will be and the more interpretable the shapelets will be.

According to extensive experiments, the average and standard deviation of θ for the best candidate set are 2.021 and 1.266 respectively, therefore we limit the range of θ from 1 to 5.

Analysis Indeed, later in Section 5, we use the logical regression model to adjust the shapelet. So (13) is very close to the model (16). Moreover, when training the model, we initialize the weight of each shapelet candidate as $\frac{1}{TFIDF}$. In this sense, $score(\Omega_\theta^c)$ makes the shapelet selection and shapelet adjustment have the uniform criterion, which can improve the accuracy. The pseudocode of the candidate selection is shown in Algorithm 2.

We use the example in Figure 5 to illustrate the candidate discovery process with $\theta = 1$. All the subsequences are transformed to PAA words and organized by a hash table, as in Section 3.2. We rank the PAA words by their TFIDF scores and select candidates for each class, as in Sections 3.3 and 3.4 respectively. It can be seen that, a PAA word occurs in many classes should have the smaller TFIDF value, such as “CB”. The PAA word which has no cover gain, such as “CEDDE”, or is similar with a selected candidate, such as “DCCCC”, will be ignored when selecting candidates.

Figure 5 Example of the candidates discovery process with $\theta = 1$

4 Shapelet adjustment and model training

In this section, we introduce the shapelets adjustment phase. The main idea of this stage is that the true shapelets which have the better discrimination ability may not occur in the training time series. So we adjust the shape of shapelets by learning the logical regression model to achieve the high-quality classification.

4.1 Redefine distance between Shapelets and time series

According to the (2), the distance of two sequences is not differentiable because of the minimum function. So we replace the minimum function by soft-minimum function. And the distance between the i -th shapelet S_i of length l and the time series T is denoted as:

$$\hat{dist}(S_i, T) = \frac{\sum_{j=1}^{n-l+1} D_{i,l,j} e^{\alpha D_{i,l,j}}}{\sum_{j=1}^{n-l+1} e^{\alpha D_{i,l,j}}} \quad (14)$$

where

$$D_{i,l,j} = dist(S_i, T(j, l)) \quad (15)$$

When the parameter $\alpha \rightarrow -\infty$, the soft-minimum function is the same with the minimum function. We omit the correctness proof of the soft-minimum function. Nevertheless, considering the precision loss of float number, we can not assign a very small number to α to avoid the case that $e^{\alpha D_{i,l,j}}$ equals to 0. We have found out that $\alpha = -25$ is small enough and do not cause the precision loss of float number in our experiments. Therefore, α is kept fixed throughout all our experiments.

4.2 The learning model

We leverage a linear learning model to predict approximate target values $\hat{Y}$.

$$\hat{Y} = W_0 + \sum_{i=1}^{K_c} W_i \hat{dist}(S_i, T) \quad (16)$$

where T is the time series to be classified, W_0 and W_i are linear weights and K_c is the number of shapelets for class c ¹.

We use the sigmoid function for the prediction of target variables via a logistic regression loss.

$$\mathcal{S}(\hat{Y}) = \frac{1}{1 + e^{-\hat{Y}}} \quad (17)$$

We convert multi-class problem into $|C|$ number of one-vs-rest sub-problems. Each sub-problem is a binary classification problem that discriminates one class against others. We regard $\mathcal{S}(\hat{Y}_c)$ for class c as a confidence probability. Therefore, the class with the most confident probability will be chosen as the predicted target class for a test time series.

Algorithm 3 Shapelets adjustment.

Input: Training set D , Class set C , Candidates Ω , Regularization λ , Learning rate η , Iteration I

Output: Shapelets S , Linear weights W

```

1 Function ShapeletsAdjustment( $D, C, \Omega, \lambda, \eta, I$ ):
2   {Initialize S and W for each class}
3   forall  $c$  in  $C$  do
4      $S_c, W_c = \text{INITIALIZATION}(\Omega^c)$ ;
5   {Iterate to train logistic regression}
6   for  $iter = \{1, \dots, I\}$  do
7     for  $i = \{1, \dots, N\}$  do
8       forall  $c$  in  $C$  do
9         {Calculate soft-minimum distances}
10        for  $k = \{1, \dots, K_c\}$  do
11           $\zeta_k = \hat{dist}(S_c[k], T^{(i)})$ ;
12        {Predict target value}
13         $\hat{Y} = W_c[0]$ ;
14        for  $k = \{1, \dots, K_c\}$  do
15           $\hat{Y} += \zeta_k * W_c[k]$ ;
16         $\xi_c = \mathcal{S}(\hat{Y})$ ;
17        {Adjust S and W}
18        for  $k = \{1, \dots, K_c\}$  do
19           $S_c[k] -= \eta \frac{\partial \mathcal{F}}{\partial S_c[k]}$ ;
20           $W_c[k] -= \eta \frac{\partial \mathcal{F}}{\partial W_c[k]}$ ;
21         $W_c[0] -= \eta \frac{\partial \mathcal{F}}{\partial W_c[0]}$ ;
22  return  $S, W$ ;
  
```

The logistic loss is denoted as:

$$\mathcal{L}(\hat{Y}, Y) = -Y \ln \mathcal{S}(\hat{Y}) - (1 - Y) \ln(1 - \mathcal{S}(\hat{Y})) \quad (18)$$

where Y is true target value. Specifically, if the time series belongs to class c , $Y_c = 1$. Otherwise $Y_c = 0$.

¹To ease the description, we use K_c to replace $|\Omega_c^c|$.

The objective function for class c with regularization terms is denoted as:

$$\underset{S, W}{\operatorname{argmin}} \mathcal{F}(S, W) = \underset{S, W}{\operatorname{argmin}} \sum_{i=1}^N \mathcal{L}(\hat{Y}^{(i)}, Y^{(i)}) + \lambda \|W\|^2 \quad (19)$$

where S is the set of shapelets for class c , $W = (W_0, W_1, \dots, W_{K_c})$ is the weight vector, λ is the L2 regularization coefficient, $\hat{Y}^{(i)}$ and $Y^{(i)}$ is the predicted target value and the true target value of i -th time series for class c .

We adopt a stochastic gradient descent approach as optimization that remedies the classification error caused by each time series at one time. Thus the decomposed objective function $\mathcal{F}^{(i)}$ for i -th time series and class c is:

$$\mathcal{F}^{(i)} = \mathcal{L}(\hat{Y}^{(i)}, Y^{(i)}) + \frac{\lambda}{N} \sum_{k=1}^{K_c} W_k^2 \quad (20)$$

4.3 Initialization

Since the gradient descent technique can not theoretically guarantee the discovery of the globally optimal solution in non-linear functions, the different initialized shapelet candidates may greatly affect the classification accuracy. We initialize the shapelets based on the shapelet candidates discovered in Section 3 for each binary classifier which can achieve high accuracy.

We transform the i -th candidate of class c back to subsequence to initialize the i -th shapelet of class c . The i -th linear weight of class c is initialized by the opposite number of the candidate's TFIDF score. The reason why using the opposite number is that the greater the distance between the shapelet and the time series is, the higher probability of $\mathcal{S}(\hat{Y}) < 0.5$ will be.

4.4 Coarse-grained Shapelet adjustment

In our previous work ELIS [5], the shapelets are adjusted at the finest granularity, that is, each point of the shapelet can increase or decrease individually. Although the fine-granularity gives us the most flexibility to refine the shapelet, it has two drawbacks. First, it may cause the phenomenon of overfitting, which will influence the accuracy, especially in the case of small dataset. Second, it causes the long training time, since each point of each shapelet is the potential dimension to tune.

To solve these problems, in ELIS++, we propose the mechanism of coarse-grained shapelet adjustment. In Figure 6, we illustrate the basic idea with a toy example. Figure 6a shows a shapelet candidate S , which is reverted from a PAA word. We amplify and show its length-12 prefix, denoted as $S[1 : 12]$. In Figure 6b, we show the shapelet X , which is obtained by adjusting S with fine-grained granularity, and the comparison between $S[1 : 12]$ and $X[1 : 12]$. It can be seen that some points decrease while others increase. In Figure 6c, we show the shapelet Y after coarse-grained adjustment of S . Instead of adjusting each point of S , we group continuous points of S as a whole, and adjust them together. In this example, each 4 data points are grouped. Now, we present the concept of adjustment granularity.

Definition 11 (Adjustment Granularity) The adjustment granularity, denoted as g , is a parameter which represents how many continuous points in a shapelet are adjusted together.

Figure 6 Shapelet adjustment

Specifically, given shapelet $S = (s_1, s_2, \dots, s_l)$ and g , g number of continuous data points, $s_{(i-1)*g+1}, s_{(i-1)*g+2}, \dots, s_{i*g}$ ($1 \leq i \leq \lceil \frac{l}{g} \rceil$), will be adjusted together.

In the example of Figure 6, we set $g = 4$. Note that the example in Figure 6b can be seen as a special case of $g = 1$. Next, we introduce how the coarse-grained adjustment works.

Given shapelet $S = (s_1, s_2, \dots, s_l)$ and adjustment granularity g , we represent S by a length- p ($= \lceil \frac{l}{g} \rceil$) shapelet, denoted as $S' = (s'_1, s'_2, \dots, s'_p)$, where $s'_i = \frac{\sum_{j=1}^g s_{(i-1)*g+j}}{g}$.² For any length- l subsequence $T(i, l) = (t_i, t_{i+1}, \dots, t_{i+l-1})$, we define the distance between S' and $T(i, l)$ as follows.

$$\text{dist}(S', T(i, l)) = \sum_{j=1}^p \sum_{k=1}^g \sqrt{(s'_j - t_{(j-1)*g+k})^2} \quad (21)$$

It is worth noting that although shapelet S' has different length with S and $T(i, l)$, in fact, S' represents a length- l shapelet. So we can define the Euclidean distance between S' and $T(i, l)$. Obviously, the distance in (21) is differentiable. By replacing the distance function in (15) with (21), we can achieve the desired coarse-grained shapelet adjustment. In other words, g number of continuous data points will be adjusted together.

To make the learning rate η uniform for all shapelets, we apply the parameter g to all shapelets. The value of g will be tuned by the parameter auto-tuning approach introduced in Section 5.2.

4.5 Accelerate the learning by parallel computation

At the candidate discovery stage, the time complexity of PAA words generation and TFIDF score calculation is $O(|C|Nn^2 \log(Nn^2))$ and the time complexity of candidate selection is $O(|C|K N n^2)$, where $|C|$ is the number of classes, K is the size of candidate set, N is the number of time series and n is the length of time series. The shapelets adjustment stage has a total time complexity of $O(|C|I K N n^2)$, where I is the iteration rounds.

The shapelet adjustment phase is more time consuming. In detail, the most expensive part is the calculation of distances between subsequences and the gradients. Specifically,

²If $l \bmod g \neq 0$, the value of the last point in S' is $s'_p, s'_p = \frac{\sum_{j=1}^z s_{(p-1)*g+j}}{z}$, where $z = l \bmod g$.

we use the *for* loop many times to calculate the distance for each matching position of two sequences in (15). Also, when we train the logistic regression classifiers, the gradient of each position of each shapelet (line 19 in Algorithm 3) also needs to be calculated by the *for* loop.

In ELIS++, we accelerate these computations by parallel computation [26]. We utilize the ubiquitous Single Instruction Multiple Data (SIMD) architecture, which assigns multiple threads to process the same set of instructions on multiple data. All software packages or hardware supporting SIMD can be used to accelerate our approach. Concretely, for distance calculation, we compute the distances of the positions where the two sequences match at the same time. When we calculate the distance between $S = (s_1, s_2, \dots, s_l)$ and $T(j, l) = (t_j, t_{j+1}, \dots, t_{j+l-1})$ For gradient calculation, we can simultaneously compute the gradients of all positions for each shapelet. That means when we use gradient decent to train the logistic regression classifiers, we can calculate $\frac{\partial \mathcal{F}}{\partial S_{i[k]}}$ for each position of the shapelet at the same time. We implement our approach with Eigen [8], which makes our approach over 8 times faster on training stage and over 19 times faster on testing stage than the naive implementation. We show the experiment results in Section 6.

5 Data augmentation and parameter auto-tuning

In this section, we introduce our approaches of data augmentation and parameter auto-tuning.

5.1 Data augmentation

When the size of training set is small, ELIS may not work well. The reason is that the training time series are covered by a small number of shapelets, which can already achieve the perfect purity. So some “true” shapelets will be missed due to the TFIDF score tie. Therefore, we propose the data augmentation approach to improve the quality of the training data before the candidate discovery.

Since we only focus on the best match position of a shapelet, it is reasonable to rotate the suffix subsequence of any length to the front to generate new time series. The rotation operation tends to select shorter candidates. It benefits enhancing generalization. Moreover, we add some noise obeying a normal distribution to the generated time series to make them different from the previous one. Specifically, the data augmentation consists of two operations:

- Rotation. We choose a cut point τ by the uniform probability on time series T . Then we slice T into two subsequences $T[1 : \tau]$ and $T[\tau + 1 : n]$. Finally, they are cascaded to generate a new time series $T' = (t_{\tau+1}, t_{\tau+2}, \dots, t_n, t_1, t_2, \dots, t_\tau)$.
- Noise insertion. We generate a length- n series of noise whose values satisfy normal distribution and add it to T' .

Data augmentation helps us to overcome the overfitting problem. We use a parameter, data augmentation amount, denoted as Σ , to control how many time series are generated from each time series. That is, given dataset D of size N , after the data augmentation, the dataset has $\Sigma * N$ time series.

An unexpected surprise is that data generation replaces the role of regularization. That means a fixed small regularization, such as 0.001, is enough for shapelets adjustment stage in most instances. Since Σ is an integer and only searched in a small range while λ is a float

and searched in a large range, it makes the parameter tuning much easier. We compare our approach with and without data augmentation in Section 6 to show the effectiveness of data augmentation.

5.2 Bayesian optimization based parameter tuning

Our approach has several parameters, including data augmentation amount Σ , regularization λ , learning rate η , the iteration round I and adjustment granularity g . Tuning them manually is a tedious task. In order to solve this problem, we propose a Bayesian Optimization based approach to determine the values of these parameters automatically. The Bayesian Optimization is a popular technique to optimize the objective function which is costly to evaluate [2, 24]. To ease the description, we denote the set of parameters as a vector H , one dimension for a parameter. Initially, we define a value range for each parameter, which forms a $|H|$ -dimensional search space $\mathcal{D}_H$.

Algorithm 4 Parameters auto-tuning.

Input: Train set D , Parameter Range R_x , Sample number N_1 , Iteration number of Bayesian Optimization N_2

Output: Parameter H^* , classification model M^*

```

1 Function Bayesianoptimization( $D, N_1, N_2, \mathcal{D}_H$ ):
2    $\mathcal{R}_H = \emptyset$ ;
3   {Initialsamples}
4   for  $i = 1$  to  $N_1$  do
5      $H_i = \text{random}(\mathcal{D}_H)$ ;
6      $(M_i, y_i) = \text{train}(H_i, D)$ ;
7      $\mathcal{R}_H = \mathcal{R}_H \cup \{H_i, M_i, y_i\}$ ;
8   for  $i = 1$  to  $N_2$  do
9      $M_G = \text{fitmodel}(\mathcal{R}_H)$ ;
10     $H_i = \arg \max_{H \in \mathcal{D}_H} F_a(H, M_G)$ ;
11     $(M_i, y_i) = \text{train}(H_i, D)$ ;
12     $\mathcal{R}_H = \mathcal{R}_H \cup \{H_i, M_i, y_i\}$ ;
13  temp=0;
14  for  $i = 1$  to  $N_1 + N_2$  do
15    if  $y_i \geq \text{temp}$  then
16      temp= $y_i$ ;
17       $H^* = H_i$ ;
18       $M^* = M_i$ ;
19  return  $H^*, M^*$ ;
```

Before introducing the approach, we first describe the relationships among H , the classification model M and the accuracy y . For each parameter vector H , we can learn a model M , which is composed of a set of shapelets S and weights W . Then for model M , we can apply it to dataset D , and obtain the accuracy y . So, the optimization goal is to find an optimal vector H^* which can achieve the highest value of y .

The Bayesian Optimization approach considers the vector H as the input and the model accuracy y as the output, which can be modeled by certain regression model, such

as linear regression or Gaussian process regression. In this paper, we utilize the latter due to its popularity.

We learn the optimal H^* with an iterative process, which consists of two phases. The pseudocode is shown in Algorithm 4. In the first phase, we sample N_1 number of parameter vectors H randomly. for each H_i ($1 \leq i \leq N_1$), the model M_i and the corresponding accuracy y_i are computed (line 4-7). After that, we maintain a candidate set of size N_1 , denoted as $\mathcal{R}_H$, each of which is a triplet (H_i, M_i, y_i) .

In the second phase, we generate another N_2 number of candidates. Different from the first phase, we try to learn the model between the parameter vector and the accuracy based on $\mathcal{R}_H$, and use the model to find the best parameter. Specifically, in the i -th round, we first learn the Gaussian process regression model M_G based on the current $\mathcal{R}_H$ (line 9). Then based on M_G , we compute the optimal parameter, H_i , with the Acquisition function (line 10). Acquisition function F_a defines the balance between exploring new areas in the search space and exploiting areas that are already known to have favorable value. In this paper, we use Upper Confidence Bound (UCB) [25] as the acquisition function. After H_i is obtained, we compute the corresponding model M_i and accuracy y_i , and add the triplet (H_i, M_i, y_i) to $\mathcal{R}_H$.

As more and more triplets are added into $\mathcal{R}_H$, the learned M_G will get better which can generate the better model M . After the second phase, we have $N_1 + N_2$ triplets in $\mathcal{R}_H$ in total. We select the optimal H and M from it, as the final one, denoted as H^* and M^* . In this paper, we set N_1 as 20 and N_2 as 50 by default.

6 Experimental results

In this section, we verify the effectiveness and efficiency of our approach.

6.1 Datasets and baselines

We conduct experiments on the UCR archive [3], a popular benchmark for the most time series classification works. We randomly select 35 datasets from the benchmark for the comparative experiments, whose detail is shown in Table 1. We compare our approach with 7 baselines in 4 different types,

- **Distance-based approach.** Within 1NN approaches, we select the best one, 1NN-cDTW [4], which uses cDTW distance as similarity measurement. Moreover, we select Proximity Forest [17] (Pforest for short) which is a novel random forest based approach. It uses diversity distances between time series to construct each tree in the forest.
- **Shapelet based approach.** We select two traditional shapelet-based approaches, and one shapelet learning approach. The first is the shapelet decision tree (NSD) [30], which uses information gain to measure the shapelet quality of each decision tree node. The second is Fast Shapelet (FSH) [20], which exploits a fast random projection technique on the SAX representation instead of searching from all subsequences to improve the efficiency. The learning shapelet approach (LTS) [7] first initializes the fixed number of shapelets and then uses logistic regression model to optimize shapelets.
- **Structure-based approach.** In this type, we select the best structure-based approach, SAX-VFSEQL [19], which applies SAX technique to discriminate all subsequences and discovers the most discriminative subsequences by training SEQL [11, 12].

Table 1 Datasets and parameters

Dataset	Train/Test/Class/Length	I	Σ	λ	η	g	Shapelet Number
ArrowHead	36/175/3/251	237	1	0.001	1.000	1	[23,13,38]
Beef	30/30/5/470	180	3	0.007	0.467	1	[10,5,7,8,6]
BeetleFly	20/20/2/512	100	2	0.001	1.000	1	[6,6]
BirdChicken	20/20/2/512	30	2	0.721	0.001	1	[5,6]
Car	60/60/4/577	270	3	0.001	0.303	4	[25,8,11,34]
CBF	30/900/3/128	35	3	0.001	1.000	1	[14,8,36]
Coffee	28/28/2/286	34	2	0.148	0.093	1	[2,6]
DiatomSize	16/306/4/345	308	2	0.001	1.000	1	[1,6,8,7]
Reduction							
DistalPhalanx	400/139/3/80	40	2	0.001	1.000	1	[53,63,110]
OutlineAge							
Group							
DistalPhalanx	600/276/2/80	491	3	0.001	0.300	4	[169,87]
OutlineCorrect							
DistalPhalanx	400/139/6/80	178	1	0.001	1.000	2	[28,26,62,23,20,19]
TW							
Earthquakes	322/139/2/512	289	1	0.187	0.346	2	[20,8]
ECG200	100/100/2/96	309	3	0.005	0.790	1	[71,22]
ECGFiveDays	23/861/2/136	9	5	0.001	1.000	1	[9,8]
FaceFour	24/88/4/350	173	5	0.001	1.000	1	[10,4,12,9]
GunPoint	50/150/2/150	144	1	0.001	1.000	1	[6,16]
Ham	109/105/2/431	5	1	1.000	0.001	4	[22,23]
Herring	64/64/2/512	5	1	0.001	0.303	4	[30,56]
ItalyPower	67/1029/2/24	40	5	0.001	1.000	1	[111,29]
Demand							
MiddlePhalanx	400/154/3/80	5	3	0.001	0.303	4	[62,73,56]
OutlineAge							
Group							
MiddlePhalanx	600/291/2/80	395	2	0.187	0.346	2	[54,118]
OutlineCorrect							
MiddlePhalanx	399/154/6/80	91	3	0.001	0.303	4	[53,38,28,27,39,53]
TW							
MoteStrain	20/1252/2/84	15	5	0.001	1.000	1	[8,12]
Plane	105/105/7/144	43	3	0.001	0.303	4	[2,14,4,1,8,15,4]
ProximalPhalanx	400/205/3/80	118	3	0.001	0.303	4	[37,74,91]
Outline							
AgeGroup							
ProximalPhalanx	400/205/6/80	116	3	0.001	0.303	4	[51,81,61,28,28,84]
TW							
ShapeletSim	20/180/ 2/500	70	2	0.721	0.001	1	[1,4]
SonyAIBO	20/601/2/70	43	4	0.028	0.882	1	[14,27]
RobotSurface1							
SonyAIBO	27/953/2/65	202	2	0.148	0.093	1	[37,20]

Table 1 Datasets and parameters (continued)

Dataset	Train/Test/Class/Length	I	Σ	λ	η	g	Shapelet Number
RobotSurface2							
Symbols	25/995/6/398	107	5	0.001	1.000	1	[3,7,1,2,7,1]
SyntheticControl	300/300/6/60	343	3	0.001	0.303	4	[20,36,124,48,109,66]
Toe	40/228/2/277	150	5	0.001	1.000	1	[25,27]
Segmentation1							
Toe	36/130/2/343	8	4	0.025	0.900	1	[7,4]
Segmentation2							
TwoLeadECG	23/1139/2/82	55	3	0.001	1.000	1	[16,13]
Wine	57/54/2/234	48	5	0.502	0.001	1	[35,62]

- **Deep learning approach.** We select mWDN [27] as the baseline because it is the star-of-the-art deep learning approach for the time series classification, which uses the multi-level wavelet decomposition to get the features and uses deep residual networks [9] as the classification model. We use the authors' default wavelet default function and the parameter values.

We does not compare the ensemble-based approach, HIVE-COTE [16], which focuses on how to combine multiple classifiers to improve the accuracy. It is orthogonal to our approach.

6.2 Experiment setting

All experiments are conducted on a Ubuntu PC with Intel(R) Core(TM) i7-6700K CPU (8-core 4.00GHz CPU), 32GB memory and 2TB disk space. The implementation of our approach is written in C++. The codes of most baselines are publicly available, and we use the public codes. Since we cannot find the public code of LTS [7], we implement it with C++ by ourselves.

There are 5 parameters in our approach. Instead of tuning them manually one by one, we use the Bayesian Optimization based approach to learn the optimal values automatically. So we only need to give the range for each parameter. The learning rate is searched in a range of $\eta \in [0.001, 1]$ and the regularization is searched from $\lambda \in [0.001, 1]$. The iteration rounds vary from 5 to 500, that is, $I \in \{5, \dots, 500\}$. Σ is searched from 1 to 5, in which 1 represents no data augmentation and 5 represents that we generate 4 additional time series for each time series. The adjustment granularity, g , is integers within [1, 8]. We show the tuned parameter values for each dataset and the number of shapelets for each class in Table 1. The source code and the parameter values of our approach are made publicly available.³

6.3 Accuracy

In the first experiment, we compare the accuracy of all approaches on the 35 datasets, and results are shown in Table 2. The best approach for each dataset is highlighted in

³<https://github.com/zhbfy/elisplus>

Table 2 Accuracy comparison

	ELIS++	mWDN	Pforest	NSD	FSH	CDTW- INN	LTS	SAX- VFSEQL
ArrowHead	0.823	0.800	0.829	0.737	0.594	0.800	0.846	0.789
Beef	0.667	0.865	0.700	0.500	0.567	0.667	0.733	0.733
BeetleFly	1.000	0.850	0.850	0.750	0.700	0.700	0.750	0.900
BirdChicken	1.000	0.850	0.900	0.800	0.750	0.700	0.800	1.000
Car	0.650	0.896	0.833	0.917	0.750	0.767	0.767	0.867
CBF	0.952	0.853	0.980	0.974	0.940	0.996	0.991	0.957
Coffee	0.964	1.000	1.000	0.964	0.929	1.000	1.000	0.964
DiatomSize	0.905	0.987	0.964	0.722	0.866	0.935	0.925	0.866
Reduction								
DistalPhalanx	0.741	0.750	0.712	0.775	0.750	0.626	0.779	0.728
Outline								
AgeGroup								
DistalPhalanx	0.772	0.728	0.775	0.770	0.655	0.725	0.719	0.842
OutlineCorrect								
DistalPhalanx	0.698	0.672	0.626	0.662	0.626	0.633	0.626	0.604
TW								
Earthquakes	0.770	0.750	0.763	0.741	0.705	0.727	0.741	0.748
ECG200	0.930	0.885	0.900	0.830	0.810	0.880	0.880	0.850
ECGFiveDays	1.000	0.955	0.797	0.984	0.998	0.797	0.663	0.955
FaceFour	0.955	0.875	0.932	0.841	0.909	0.886	0.955	0.932
GunPoint	0.993	0.938	0.993	0.893	0.947	0.913	0.987	0.987
Ham	0.581	0.667	0.629	0.686	0.648	0.600	0.667	0.810
Herring	0.594	0.641	0.625	0.672	0.531	0.531	0.625	0.625
ItalyPower	0.976	0.966	0.966	0.948	0.917	0.955	0.960	0.816
Demand								
MiddlePhalanx	0.617	0.486	0.597	0.794	0.729	0.520	0.780	0.677
OutlineAgeGroup								
MiddlePhalanx	0.835	0.743	0.821	0.643	0.546	0.766	0.571	0.546
OutlineCorrect								
MiddlePhalanx	0.597	0.549	0.539	0.520	0.533	0.507	0.507	0.487
TW								
MoteStrain	0.898	0.859	0.885	0.825	0.777	0.866	0.866	0.827
Plane	1.000	0.969	1.000	1.000	1.000	1.000	1.000	0.991
ProximalPhalanx	0.844	0.828	0.839	0.883	0.804	0.785	0.849	0.828
OutlineAgeGroup								
ProximalPhalanx	0.805	0.899	0.790	0.805	0.702	0.756	0.776	0.610
TW								
ShapeletSim	1.000	0.511	0.761	0.956	1.000	0.700	0.950	0.717
SonyAIBORobot	0.915	0.885	0.822	0.844	0.686	0.696	0.810	0.644
SurfaceI								
SonyAIBORobot	0.894	0.857	0.875	0.934	0.790	0.859	0.875	0.816

Table 2 Accuracy comparison (continued)

	ELIS++	mWDN	Pforest	NSD	FSH	CDTW- INN	LTS	SAX- VFSEQL
Surface2								
Symbols	0.930	0.889	0.954	0.782	0.934	0.938	0.932	0.887
SyntheticControl	0.983	0.955	0.993	0.983	0.910	0.983	0.997	0.890
ToeSegmentation1	0.974	0.607	0.908	0.965	0.956	0.750	0.934	0.930
ToeSegmentation2	0.885	0.836	0.892	0.908	0.692	0.908	0.915	0.862
TwoLeadECG	0.997	0.923	0.971	0.850	0.925	0.868	0.996	0.948
Wine	0.630	0.917	0.519	0.796	0.759	0.611	0.500	0.963
avg acc	0.850	0.818	0.827	0.819	0.781	0.781	0.819	0.817
avg rank	2.886	4.229	3.371	4.200	5.800	5.142	3.600	4.971
win number	17	4	5	6	2	3	7	4

bold. Moreover, we also present the average accuracy and the number of winners for each approach.

The superiority of our algorithm is obvious. ELIS++ achieves the highest accuracy on 17 datasets, and the average accuracy is also the highest. The results verify that our approach obtains high-quality shapelets, which can distinguish the classes well.

Among all the baselines, LTS is the most competitive one, which wins on 7 datasets. So, the results demonstrate the advantage of the shapelet learning approach. mWDN and Pforest, have the highest accuracy on 4 and 5 datasets respectively.

In essential, mWDN uses the wavelet basis as the features, and builds the network model based on them. Considering the wavelet basis as the subsequence, mWDN can also be considered as a “shapelet” based approach. Although the network model has good learning performance in many applications, it has some limitations in the time series classification problem.

First, each wavelet basis can only simulate specific shape of subsequence, and is only suitable for the time series dataset in which the shapes of shapelets are similar with this wavelet basis. It may not work well if the wavelet basis is totally different with the discriminative shapelets.

Second, since the wavelet decomposition is lossless, so theoretically, if all wavelet coefficients are used as the features to build the deep learning network, the mWDN approach can model any time series perfectly. However, the large number of features will incur a large network which needs a large number of training time series to make the learning process converge, which is not true in many time series dataset. Moreover, it is difficult to determine the optimal network structure given an arbitrary training dataset.

In contrast, ELIS++ can learn the shapelets of arbitrary shape, and it has better capability to learn the shapelets based on a small number of training time series.

6.4 Efficiency

In this experiment, we compare the efficiency of shapelet learning approaches, LTS and ELIS++, because they are more accurate than other baselines. In order to verify the efficiency improvement of ELIS++, we also compare the efficiency of ELIS. In Figure 7, we

Figure 7 Efficiency

show the average training and testing time of 35 datasets for all three approaches. Moreover, we compare the parameter tuning efficiency of ELIS and ELIS++ (the “tune” bar). ELIS++ tunes parameters with the Bayesian Optimization based approach, while ELIS uses the brute force approach to find the best combination of parameters in the given range [27]. In Table 3, we show the training time of ELIS++ (with or without SIMD optimization) and ELIS on each dataset.

ELIS++ is 373 times faster on training phase and 15.7 times faster on testing phase than LTS. ELIS is 37 times faster on training phase and 1.2 times faster on testing phase than LTS. One reason why ELIS++ is more efficient is that initializing the shapelets by our candidates discovery approach not only can produce a smaller shapelet set than both LTS and ELIS. Moreover, Table 3 shows that ELIS++ with SIMD is about 10 times faster on training phase than the ELIS. It means that our approach has the capability to improve efficiency with SIMD, even when CPU is used.

To find the best parameters, ELIS++ trains 70 models, as introduced in Section 5.2. ELIS has two parameters to tune, regularization λ and learning rate η . For each parameter, ELIS tries 3 possible values, so that ELIS trains 9 models in total. Although ELIS++ trains much more models than ELIS, it is still slightly faster than ELIS, which verifies the effectiveness of SIMD-based parallel computation.

6.5 ELIS++ vs. ELIS

In this experiment, we compared ELIS++ with ELIS in terms of the accuracy and the training time, and the results are shown in Table 3. ELIS++ achieves higher accuracy on all 35 datasets, especially on the small training dataset, such as “BeetleFly”, “BirdChicken”, “DiatomSizeReduction” and so on. It can be seen that in Table 1, the data augmentation amount, Σ , is larger than 1, which verifies the effectiveness of the data augmentation mechanism.

Table 3 Effectiveness of ELIS++ comparing with ELIS

	ELIS++			ELIS	
	Accuracy	Runtime		Accuracy	Runtime
		SIMD	w/o SIMD		w/o SIMD
ArrowHead	0.823	328.90	3254.19	0.823	3256.27
Beef	0.667	187.13	1851.49	0.633	20604.89
BeetleFly	1.000	32.41	220.22	0.850	105.99
BirdChicken	1.000	27.59	313.75	0.900	89.99
Car	0.650	339.76	3031.12	0.617	3629.47
CBF	0.952	18.69	148.92	0.946	122.79
Coffee	0.964	6.63	63.50	0.964	47.73
DiatomSizeReduction	0.905	120.97	1108.35	0.899	643.04
DistalPhalanxOutline	0.741	362.83	3589.90	0.712	2809.94
AgeGroup					
DistalPhalanxOutline	0.772	2645.10	30007.64	0.768	23215.96
Correct					
DistalPhalanxTW	0.698	626.24	6824.28	0.676	1182.35
Earthquakes	0.770	1039.28	10282.84	0.748	1309.20
ECG200	0.930	353.90	3501.54	0.930	3499.56
ECGFiveDays	1.000	2.46	24.34	1.000	20.65
FaceFour	0.955	98.56	971.45	0.955	159.32
GunPoint	0.993	37.74	368.23	0.993	421.45
Ham	0.581	39.52	340.39	0.486	665.28
Herring	0.594	45.53	396.61	0.516	1457.01
ItalyPowerDemand	0.976	12.00	118.73	0.976	548.54
MiddlePhalanxOutline	0.617	29.04	197.17	0.565	432.77
AgeGroup					
MiddlePhalanxOutline	0.835	751.08	7313.57	0.814	6145.76
Correct					
MiddlePhalanxTW	0.597	192.82	1778.05	0.578	1355.90
MoteStrain	0.898	2.10	20.78	0.898	1291.53
Plane	1.000	24.50	242.41	0.981	168.20
ProximalPhalanxOutline	0.844	210.38	1899.25	0.561	6185.24
AgeGroup					
ProximalPhalanxTW	0.805	357.45	3154.38	0.722	816.37
ShapeletSim	1.000	7.07	67.93	1.000	69.95
SonyAIBORobotSurface1	0.915	5.20	51.45	0.878	232.48
SonyAIBORobotSurface2	0.894	45.99	455.03	0.854	327.69
Symbols	0.930	119.94	1186.71	0.783	1175.98
SyntheticControl	0.983	883.55	8111.39	0.867	12726.96
ToeSegmentation1	0.974	160.64	1589.40	0.920	1120.81
ToeSegmentation2	0.885	163.86	1898.76	0.838	4739.01
TwoLeadECG	0.998	8.88	87.86	0.998	297.29
Wine	0.630	113.61	1032.90	0.500	1087.54

We can still find some datasets which satisfy, 1) no data augmentation, $\Sigma = 1$, and 2) using the coarse-grained adjustment granularity ($g > 1$), like “Ham” and “Herring”, on which ELIS++ is more accurate than ELIS. These results demonstrate that the coarse-grained adjustment can alleviate the overfitting.

As for the efficiency, if SIMD-based parallel computation is utilized, ELIS++ is clearly faster than ELIS. One interesting phenomenon is that, in 16 out of 35 datasets, ELIS++ without SIMD is still has better efficiency, although more training time series are used due to the data augmentation. The major reason is that, due to the entropy-based shapelet candidate selection, ELIS++ can generate better shapelet candidates, which makes the shapelet adjustment phase converges faster.

7 Related work

The time series classification works can be divided into four major categories.

The first category is the distance-based approach. It computes similarity between time series based on the raw numeric data. The traditional approaches use Euclidean distance [29] or DTW [21] distance to build a 1NN classification model. They are easy to implement and understand, but the accuracy is lower than other types of approaches. To improve the accuracy, some novel distances have been proposed [4, 31]. These approaches still utilize the 1NN as the classification model, but they use different similarity definitions. Also, Benjamin et al proposes Proximity Forest [17], which uses the random forest model to combine multiple similarity measurements. Currently, distance based semi supervised learning has gradually attracted the attention of researchers [13].

The second category, structure-based approach, uses high-level time series representations as the features to describe the characteristics of classes. SAX-VFSEQL [19] and SAX-VSM [23] build classifiers based on SAX representation [14]. BOSS VS [22] transforms time series from time domain to frequency domain. Since the structure-based approaches may lose some detailed information of time series, they suffer from the lower accuracy.

The third category is the deep learning based approaches [6, 27], which construct the complex network model. They are usually end-to-end algorithms. At the same time, they usually use a large number of labeled data, select the appropriate objective function, effective network structure and good parameters to achieve extraordinary fitting effect for classification. Neural network model ability is highly related to the number of the train dataset. Some deep learning models, like CNN and Resnet[6], have been applied in the time series classification. However, the interpretation is the weak point of DL-based approach. Reasonable hyper-parameter tuning method has rarely been discussed in the former DL-based works. [27] is one of the state-of-the-art in time series classification. It constructs a model that can automatically extract time-frequency characteristics of time series from the perspective of digital signal processing, and gets a good classification effect through deep learning model. But its fundamental wave’s shape needs to be specified manually, which is its weakness.

The forth category is the shapelet-based approach [7, 18, 20, 30]. The concept of “shapelet” has been proposed in [30]. It first extracts some typical subsequences and then uses them to build the classification model. Other metrics are also utilized to evaluate the discrimination of a subsequence, such as F-Stats [15], Kruskal-Wallis and Mood’s median [10]. To find shapelet candidates efficiently, Fast Shapelet [20] transforms time series subsequences to SAX words to estimate the frequency of the subsequences.

Unlike other shapelet-based approaches, LTS [7] learns the discriminative shapelets, instead of searching them from the raw numeric data. So LTS can generate shapelets that do not appear in the training time series, which can improve the accuracy significantly. However, the training phase is computationally intensive and needs to manually tune lots of hyper-parameters, such as shapelet number and shapelet length. Moreover, LTS generates too many shapelets, which makes the interpretability weak. SAX-VFSEQL [19], a novel shapelet-learning based approach, generates a pool of SAX words as the shapelets, and iteratively optimizes the weights of the shapelets. However, this approach still cannot generate shapelets which are absent in training data. Also, the discovered shapelets are numerous and redundant which makes interpreting the classification decision hard.

Although, the ensemble-based approach [16] has been proposed, which focuses on how to combine multiple classifiers to improve the accuracy. It is orthogonal to our approach.

8 Conclusion

In this paper, we introduce a shapelet learning approach ELIS++. ELIS++ first discovers shapelet candidates from the PAA space in candidate generation phase, and then applies the logistic regression model to adjust shapelets in the model training phase, by which we utilizes the advantages of both structure-based approaches and shapelet-based approaches. To improve the accuracy on small training dataset, we propose a data augmentation approach which can avoid the missing of some “true” shapelets. Moreover, we propose a Bayesian Optimization based approach to automatically set the parameters. The extensive experimental results show that ELIS++ can achieve higher accuracy and efficiency. In future, we aim to improve ELIS++ in two directions. First, we will try different learning models on the model training phase, especially the deep learning based model. Second, we hope to extend ELIS++ to the multi-dimensional time series classification.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Bagnall, A., Bostrom, A., Large, J., Lines, J.: The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version. ArXiv e-prints (2016)
2. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: NIPS, pp. 2546–2554 (2011)
3. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The ucr time series classification archive. https://www.cs.ucr.edu/eamonn/time_series_data (2015)
4. Dau, H.A., Silva, D.F., Petitjean, F., Forestier, G., Bagnall, A.J., Mueen, A., Keogh, E.J.: Optimizing dynamic time warping’s window width for time series data mining applications. *DMKD* **32**(4), 1074–1120 (2018)
5. Fang, Z., Wang, P., Wang, W.: Efficient learning interpretable shapelets for accurate time series classification. In: ICDE 2018, pp. 497–508 (2018)

6. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.: Deep learning for time series classification: a review. *DMKD* **33**(4), 917–963 (2019)
7. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: *SIGKDD*, pp. 392–401 (2014)
8. Guennebaud, G., Jacob, B., et al.: Eigen v3. <http://eigen.tuxfamily.org> (2010)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *CVPR*, pp. 770–778 (2016)
10. Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. *DMKD*, 851–881 (2014)
11. Ifrim, G., Wiuf, C.: Berror-based and entropy-based discretization of continuous features. In: *SIGKDD*, pp. 708–716 (2011)
12. Ifrim, G., Bakir, G., Weikum, G.: Fast logistic regression for text categorization with variable-length n-grams. In: *SIGKDD*, pp. 354–362 (2008)
13. Liang, S., Zhang, Y., Ma, J.: Active model selection for positive unlabeled time series classification. In: *ICDE*, pp. 361–372. *IEEE* (2020)
14. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. *DMKD*, 107–144 (2007)
15. Lines, J., Davis, L.M., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In: *SIGKDD*, pp. 289–297 (2012)
16. Lines, J., Taylor, S., Bagnall, A.: Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In: *ICDM*, pp. 1041–1046 (2016)
17. Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., Petitjean, F., Webb, G.I.: Proximity forest: An effective and scalable distance-based classifier for time series. *DMKD* **33**, 607–635 (2019)
18. Mueen, A., Keogh, E., Young, N.: Logical-shapelets: An expressive primitive for time series classification. In: *SIGKDD*, pp. 1154–1162 (2011)
19. Nguyen, T.L., Gsponer, S., Ifrim, G.: Time series classification by sequence learning in all-subsequence space. In: *ICDE*, pp. 947–958 (2017)
20. Rakthanmanon, T., Keogh, E.: Fast shapelets: A scalable algorithm for discovering time series shapelets. pp. 668–676 (2013)
21. Sakoe, H., Chiba, S.: Fast time series classification using numerosity reduction for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.*, 43–49 (1978)
22. Schäfer, P.: Scalable time series classification. *DMKD*, 1273–1298 (2016)
23. Senin, P., Malinchik, S.: Sax-vsm: Interpretable time series classification using sax and vector space model. In: *ICDM*, pp. 1175–1180 (2013)
24. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: *NIPS*, pp. 2951–2959 (2012)
25. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.W.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: *ICML*, pp. 1015–1022 (2010)
26. Tchendji, V.K., Zeutouo, J.L.: An efficient cgm-based parallel algorithm for solving the optimal binary search tree problem through one-to-all shortest paths in a dynamic graph. *Data Sci. Eng.* **4**(2), 141–156 (2019)
27. Wang, J., Wang, Z., Li, J., Wu, J.: Multilevel wavelet decomposition network for interpretable time series analysis. In: *SIGKDD*, pp. 2437–2446 (2018)
28. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. *DMKD*, 275–309 (2013)
29. Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: *ICML*, pp. 1033–1040 (2007)
30. Ye, L., Keogh, E.: Time series shapelets: A new primitive for data mining. In: *SIGKDD*, pp. 947–956 (2009)
31. Yuan, J., Lin, Q., Zhang, W., Wang, Z.: Locally slope-based dynamic time warping for time series classification. In: *CIKM* (2019)