Information Sciences 570 (2021) 428–450

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Efficient class-specific shapelets learning for interpretable time series classification

Zhiyu Liang, Hongzhi Wang *

Harbin Institute of Technology, Harbin, China

ARTICLE INFO

Article history:
Received 29 July 2020
Received in revised form 17 March 2021
Accepted 29 March 2021
Available online 6 April 2021

Keywords:
Interpretable time series classification

ABSTRACT

In the last decade, time series classification approaches based on time-independent shapelets, have received considerable attention due to their high prediction accuracy and intuitive interpretability. However, most existing shapelets discovery approaches find shapelets by evaluating the discriminatory power of all subsequences of the series, which is computationally expensive even with certain speed-up techniques. Even though some shapelet learning approaches learn the near-to-optimal shapelets from the training series rather than searching from numerous segments, they still have significant drawbacks in their performance regarding the accuracy, efficiency, and interpretability due to the numerous class-shared shapelets with fixed lengths. Thus, we propose a new shapelet learning approach that can learn as few as possible class-specific and variable-length shapelets. Extensive experiments demonstrate that our proposed method is competitive about classification accuracy over 18 baselines on 25 datasets, outperforms 2 orders of magnitude about efficiency, and is more interpretable than existing classifiers.

1. Introduction

Time series classification is a critical problem in data mining, with many applications in diverse areas, including medical, biological, financial, engineering, and industrial [1]. As a result, hundreds of algorithms [2] are proposed to tackle this problem.

Among those methods, shapelet-based approaches [3–5] have attracted increasing attention in recent years due to the comprehensive interpretability of the classification results. Shapelet-based methods discover discriminatory subsequences from the raw time series and take the similarity between the time series and the shapelets as features to build the classifier. The shapelets represent distinguishing local patterns of some classes [3] thus a classification decision can be directly made by the presence or absence of particular shapelets.

Despite the significant advantages, most existing shapelet-based methods discover shapelets by checking all possible subsequences with $O\left(M^2N^4\right)$ time, where M is the number of time series in the training dataset and N is the length of the training series, which is extremely time-consuming, even with some speed-up techniques such as caching, pruning and early-abondoning [6,4]. Instead of restricting the shapelets to subsequences of the raw time series, a recent study [5] proposed to directly learn the true shapelets rather than searching from all the problem space, which can achieve

* Corresponding author.
E-mail addresses: zyliang@hit.edu.cn (Z. Liang), wangzh@hit.edu.cn (H. Wang).

https://doi.org/10.1016/j.ins.2021.03.063
0020-0255/

competitive classification accuracy by finding the near-to-optimal shapelets which do not exist in the raw time series. The learning-based approach requires $O\left(IN_uMN^2\right)$ time where I is the number of iterations, N_u is the number of shapelets to be updated per iteration, and M and N are the number and length of the training series, respectively. Given that usually $IN_u \ll MN^2$, the learning-based method is theoretically more efficient than searching-based approaches. Experimental results also confirm the correctness of the theoretical analysis [5].

However, the current learning shapelet algorithm is still computationally expensive mainly because it requires a high number of calculations of shapelet updating, i.e., the N_u of the time complexity discussed above. The reasons are analyzed as follows. First, the shapelets are designed as parameters shared by every submodel[1] for solving a generalized classification problem with the one-vs-rest strategy. Therefore, all shapelets need to be updated when training each submodel at one iteration, and the submodels have to be updated sequentially during each iteration. Suppose there are a total of K shapelets to be learned for a $|C|$-classes problem($|C| > 2$), then $N_u = K|C|$ calculations of the shapelet updating are required. Secondly, the length of all shapelets is fixed at some predefined values, such as 10% of the training series length, while the true shapelets could be in any length shorter than the training series. Without matching the true shapelet length, the true shapelet that is highly discriminatory cannot be learned. As a result, the current algorithm has to maintain numerous shapelets, i.e., a large K, to ensure the classification accuracy because of the low quality of these shapelets. The above two reasons cause a very large N_u of the time complexity, i.e., a large number of calculations of the even time-consuming shapelets updating process which takes $O\left(MN^2\right)$ time. Fig. 1(a) illustrates the process of learning shared shapelets at a single epoch. Each submodel is a binary classifier that distinguishes one category from the others with its shapelets. Conversely, the shapelets are parameters to be optimized when learning the corresponding submodels, as the dashed arrows represent. As shown in Fig. 1(a), in the fashion of learning shapelets shared by all classes, optimizing the parameters of the submodels at each epoch requires updating sequentially all shapelets for $|C|$ times. For example, on the *Lightning7* dataset of UCR Archive [7], which contains time series distributed in 7 classes, the algorithm of [5] requires to learn up to 195 shapelets shared by each category. Each of them needs to be sequentially optimized for every submodel of the 7 classes, causing 1,365 calculations for shapelet updating at each iteration. Additionally, too many shapelets also complicate the interpretability issue.

Another reason that could also make the current learning shapelet method inefficient is the way the shapelets are initialized. Although initializing the shapelets with the k-means centroids of the segments of the training series as the current approach can offer a good convergence with a high accuracy, the initial values of the shapelets could be far from a optimum. As a result, the algorithm requires a high number of iterations, i.e., a large I of the time complexity, to converge, which will also slow down the training.

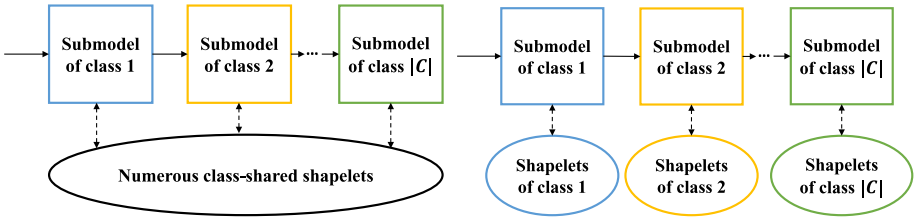
In this study, our motivation is to achieve a more efficient and interpretable time-series classification while guaranteeing accuracy by solving the above problems. To fulfill our goal, we propose a novel shapelets learning algorithm. Our algorithm adopts a similar shapelet updating strategy as the study of [5] which requires $O\left(MN^2\right)$ time, but learns the shapelets in a completely different way that can dramatically reduce the calculations of the shapelet updating at each iteration N_u and also the number of learned shapelets K, and effectively decreases the number of iterations I.

To reduce the number of calculations of shapelet updating N_u, we first propose to learn class-specific shapelets for each category within the class labels. The class-specific shapelets of one category can maximally distinguish that class from the others. Compared with learning the shapelets that are shared by all classes, learning class-specific shapelets is more efficient because the shapelets are only updated when learning the submodel of the same class value, as shown in Fig. 1(b). Thus, the number of shapelet updating calculations at each iteration N_u becomes identical to the number of shapelets K, rather than $K|C|$, which accelerates the training process. Moreover, since the classification decision of the shapelet-based classifier is interpreted as the presence or absence of the shapelets, it is intuitive that each class has the specific shapelets that can distinguish itself from others. Consequently, the classifier composed of class-specific shapelets is more understandable.

Since the number of shapelet updating calculations per iteration N_u is equal to the number of shapelets K by learning class-specific shapelets, to further reduce the number of N_u for acceleration, we need to reduce the number of shapelets. Unfortunately, the determination of the proper value of K without any prior knowledge is a big challenge. Apparently, an inadequate number of shapelets will limit the classification precision, while learning redundant shapelets performs badly on efficiency and will make the explanation of the prediction results complicated. As discussed above, we believe that the current shapelet learning approach requires numerous shapelets because of the non-matched shapelet length. However, how to determine the length of the shapelets without any experience is also challenging. In practice, the true shapelets can be in any unknown length, but it untenable to try out all candidates by brute force. Besides, we should consider how to initialize the shapelets with appropriate values for efficient and robust convergence of the learning model.

To tackle these challenges, we propose a method to automatically determine the length and number of the shapelets to be learned by firstly discovering the variable-length frequent subsequences from the training series of each class, and then selecting as few as possible representatives from these subsequences with the best discriminatory power. The basic idea

[1] For the sake of clarity, we call the one-vs-rest binary classifier that distinguishes one class against others as a "submodel" of that class, and the combination of the submodels of all classes is called a "model" of the classification problem.



(a) Learning shapelets shared by all classes (b) Learning class-specific shapelets

Fig. 1. The process of learning class-shared and class-specific shapelets at each epoch.

is that the shape of the true class-specific shapelets represents some distinctive conditions of the class they belong to, such as arrhythmia shown in a set of ECG signals of some heart disease patients. Therefore, the subsequences that frequently occur in the training series of a particular class but stay away from the others are very likely to be similar to the true shapelets specific to that class in shape, i.e., they have almost the same length and differ little in values. Thus, we discover such subsequences from the training data as shapelet candidates and set the number and length of the shapelets to be learned the same as the candidates in order to learn the true shapelets to ensure a high classification accuracy with as few as possible shapelets. Moreover, from the perspective of an optimization problem, the similarity between the candidates and the shapelets in shape means that they are probably located nearby. This motivates us to initialize the values of the shapelets with the corresponding candidates to pursue more efficient convergence due to a lower number of iterations, I. Fig. 2 is an example to illustrate the relation of the training series, the shapelet candidate, and the learned shapelet. The black line is an instance in a training data set of heartbeat measurements of patients with different medical conditions. The blue line is one of the representative subsequences of the training series in the same class(medical condition). Thus, it could be selected as a shapelet candidate specific to that class. The red series is the class-specific shapelet learned from the blue candidate. It can maximally distinguish the time series of its own class from the others since it represents a distinctive heartbeat message due to the particular medical condition.

We label our novel shapelet learning algorithm as CSSL (**C**lass-**S**pecific **S**hapelets **L**earning). As discussed above, our CSSL method can automatically learn as few as possible variable-length and class-specific shapelets for each category. For each class, CSSL discovers the possible lengths and values of the shapelets by frequent subsequence mining, and it determines the optimal number of shapelets for each class as well as their lengths by measuring the discriminatory power of the frequent sequences. Afterward, CSSL initializes the class-specific shapelets learning model with the determined number, length and initial values and learns the model by updating iteratively the shapelets and weights. Similar to learning shared shapelets, the learned model is the final classifier for predicting the category of the new arriving time-series with the learned shapelets and weights. Nevertheless, evaluated on the same dataset of *Lightning7*, CSSL learns only 15.7 shapelets on average for each class, and only conducts 110 shapelet updating calculations at each iteration, i.e., about 12 speedups theoretically over the learning shared shapelets discussed above, while achieving a 10.3% higher classification accuracy. Besides, beneficial from few and class-specific shapelets, our classifier is easier to interpret than existing methods.

Additionally, by discovering class-specific shapelet candidates and updating class-specific shapelets for each class in our CSSL, all calculations of the shapelet learning for different classes are decoupled, it means that the calculation of every class can be easily parallelized for further acceleration, which will provide an up to $|C|$ extra speedup theoretically for a classification problem with $|C|$ distinct class values.

The main contributions of this study are summarized as follows.

- We propose to learn class-specific shapelets rather than shared shapelets in order to reduce the number of calculations of shapelet updating N_u to achieve a more efficient and interpretable time-series classification.
- We propose a class-specific shapelet candidates discovery method to automatically determine the number, length, and initial values of the shapelets in the learning model. It can dramatically reduce the number of calculations of shapelet updating N_u by reducing the number of shapelets K, and effectively decrease the number of iterations I.
- The calculations for learning shapelets of different classes in our proposed CSSL method are totally decoupled, thus the calculation of each class can be easily parallelized for further acceleration.
- We conduct extensive experiments using the UCR dataset. The results show that our method promotes a 2 orders of magnitude greater efficiency and is more interpretable while guaranteeing an outstanding accuracy.

The remaining parts of this paper are organized as follows. Section 2 is the survey of related studies. In Section 3, we provide a preliminary outline of our proposed method, including the key terms and the required techniques. The proposed method, class-specific shapelet learning, is described in Section 4. The experimental results are presented and explained in Section 5. Finally, we conclude this paper in Section 6.

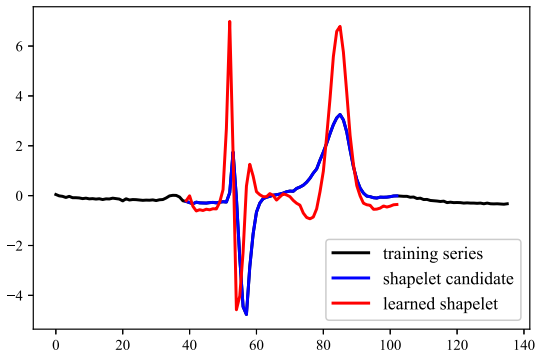


Fig. 2. An example of the training series, the shapelet candidate and the learned shapelet.

2. Related work

Time series is one of the most common data types in the real world, such as the stock price, the daily temperature, as well as the electrocardiogram. Classical time series analysis focuses on time series prediction or forecasting, which is to predict the future based on historical data. This topic has been studied for a long time, and many effective algorithms are continuously proposed, such as the works of [8–14]. The performance evaluation of state-of-the-art statistical and machine learning models for time series prediction is conducted by [15].

With the increase in available labeled time series data, more and more studies show interest in another important problem, i.e., time series classification, that is to identify the category of a new coming time series based on the labeled training data. In this paper, we work on the time series classification problem as well, so we discuss the related work in this area in detail as follows.

In the last decade, a large amount of time series classification algorithms have been proposed [2]. The most fundamental approach classifies the label of a series by measuring its whole-series similarity to the training data. The simplest one, the nearest neighbor classifier using the Euclidean distance (1NN-Euclidean), is shown to be robust for many datasets [2]. However, Euclidean distance performs badly when two semantically similar sequences differ in alignments [16,17]. As a result, many elastic distances [18–20] are introduced in this field. The most representative elastic measure for time series classification is dynamic time warping (DTW) [18]. Whole-series-based methods only care about the global similarity of the whole series, so the classification decision is difficult to understand. In addition, these methods have a high computational complexity [21] even with several acceleration techniques [22,23].

Inspired by the truth that the discriminatory features are usually located in some intervals of the whole series while the regions out of the intervals could be noise for classification, some studies [24–27] try to extract features from the intervals of the time series and train classifiers with these features. Since the number of possible intervals is very high, and it is impossible to evaluate all of them, current algorithms have to generate limited random intervals for feature extraction. As a result, the way that the intervals are used becomes the key factor of the classification performance.

Similar to documentation classification [28], some studies try to design a feature dictionary for the time series dataset and transform every instance into a high dimensional sparse vector based on the count of each feature. The Bag of Patterns (BOP) method [29] directly implements this idea. It converts raw time series into words using SAX [30] with a sliding window and counts the distribution of the words over a series to form a pattern histogram. All histograms of a time series are taken as the features for classification. SAX-VSM presented in [31] further improves the BOP. This method adopts a well-known information retrieval technique, called the vector space model, to describe the features of the SAX words. SAX-VSM generates a collection of bags for each class rather than an instance. Each collection represents the features of a class, thus can be used to interpret the classification. The recent works in [32,33] propose an alternative dictionary-based classifier. Instead of SAX, they adopt the Symbolic Fourier Approximation (SFA) to obtain a time series representation. Although these approaches show advantages over SAX-VSM regarding accuracy and efficiency, the SFA representation is much harder to interpret than the SAX words. A recent work of [34] proposes two new time series classifiers that combine the SAX representation and sequence learning. It can be seen as a special case of the dictionary-based method where each bag of SAX words records a classification weight learned by SEQL, rather than the word frequency or tf*idf weight.

Another large collection of works focused on several discriminatory subsequences of the raw time series, named shapelets [35]. In the early work, shapelets are searched from all possible candidates, and a decision tree model is constructed with the discovered shapelets for classification [35,3]. The brute-force search is computationally expensive. Speed-up techniques, such as entropy and search space pruning, early abandoning of minimum distance calculation, and reusing of computations [6,35], are proposed to accelerate the searching process. However, the search is still inefficient, especially for a long time series. The work of [36] proposes a novel shapelets discovery methodology by SAX representation and random projection

technology, which is more efficient than searching over subsequences. Nevertheless, the prediction accuracy of this approach is not so good. Instead of training a decision tree with shapelets, some works attempted to make use of shapelets in more effective ways. The shapelet transform (ST) [4,37] method transforms a time series to a vector by measuring the distance between the raw data to all discovered shapelets. Afterwards, any traditional classification model can be trained on these vectors. This approach improves the accuracy but is still limited in terms of efficiency due to the expensive shapelet searching. Instead of searching from the raw time series, the work of [5] proposes to directly learn discriminatory shapelets. This method is promisingly accurate in many cases and also improves in efficiency. Thus, it becomes the state-of-the-art non-ensemble interpretable time series classification algorithm [2].

With the great success of deep learning in computer vision and NLP areas, some works adapt or design deep neural networks for time series classification. Though performing well on accuracy, deep learning approaches not only are not significantly better than non-deep classifiers but sacrifice the interpretability due to their well-known black-box effect [38].

Additionally, some works of [39,27,40] attempted to combine several classifiers with different time series representations into an ensemble which achieve almost the best accuracy on the UCR/UEA benchmark [2]. However, these methods are limited in real applications due to their huge time complexity.

In this paper, we study the shapelet-based method for its intuitive interpretability and potential of progress in terms of accuracy and efficiency. Inspired by the success of learning shapelets rather than searching, we attempt to develop a novel shapelet learning methodology that can be more efficient and easier to interpret while guaranteeing a competitive accuracy. The proposed method will be presented later. Before that, we first introduce the preliminary definitions, notations, and techniques of this study in the next section.

3. Preliminaries

In order to concretely state our proposed method, we first define the key terms used and provide a brief description of the fundamental techniques incorporated in this study, including the generalized shapelet learning model and the approximate motif discovery approach.

3.1. Definitions and notations

We begin with the formal definition of the data type of our interest, the *time series*.

Definition 1 (*Time Series*). A time series T is a sequence of scalar observations $T = (t_1, t_2, \ldots, t_i, \ldots, t_N)$ ordered by time, where t_i is the value at timestamp i and N is the length of T.

Then, we define the *time-series dataset* as follows.

Definition 2 (*Time Series Dataset*). A time series dataset D is composed of M time series instances, i.e., $D = \{T_1, T_2, \ldots T_j, \ldots, T_M\}$, where $T_j = (t_{j_1}, t_{j_2}, \ldots, t_{j_i}, \ldots, t_{j_N})$, $j \in 1, 2, \ldots, M$ is the j-th instance of the dataset. The class value of T_j is denoted as $Y_j \in C$, where C is a set of distinct labels and $|C|$ is the size of C, i.e., the number of labels.

In this study, we focus on the local features of the time series, thus we define the subsection of the time series, *subsequence*.

Definition 3 (*Subsequence*). A subsequence of time series $T_j = (t_{j_1}, t_{j_2}, \ldots, t_{j_i}, \ldots, t_{j_N})$ is a continuous sampling of T_j starting at time p with length L, denoted as $T_j^{p,L} = \left(t_{j_p}, t_{j_{p+1}}, \ldots, t_{j_{p+L-1}}\right)$, where $1 \leqslant p \leqslant N - L + 1$. Particularly, the p-th point of T_j, i.e. $T_j^{p,1}$, is abbreviated as T_j^p.

To measure the distance of two sequences with an arbitrary length, we define the *distance between two sequences* as follows.

Definition 4 (*Distance Between Two Sequences*). Let T_m and T_n be two time sequences of length L_m and L_n where $L_m \geqslant L_n$, we denote the *point-wise distance* between a subsequence of T_m which starts at p and has length L_n, i.e., T_m^{p,L_n}, and the whole series T_n as $Dis^p(T_m, T_n)$, i.e.,

$$Dis^p(T_m, T_n) = \frac{1}{L_n} \sum_{l=1}^{L_n} \left(T_m^{p+l-1} - T_n^l\right)^2 \tag{1}$$

where $1 \leqslant p \leqslant L_m - L_n + 1$. Then, we define the *distance between T_m and T_n* as the minimum point-wise distance between the L_n-length subsequence of T_m and T_n, i.e.,

$$Dis(T_m, T_n) = \min_{p=1,2,\ldots J} \frac{1}{L_n} Dis^p(T_m, T_n) \tag{2}$$

where $J = L_m - L_n + 1$.

Finally, we introduce the definition of the feature to learn for classifying time series in this study, the *class-specific shapelet*. We first present a more general concept, the time series *pattern*.

Definition 5 (*Pattern*). We define a time series pattern P as a series that can reflect certain features of the time series. Note that the pattern could be, but is not restricted to the subsequence of the time series.

Based on the *pattern*, we define the *class-specific shapelet* as follows.

Definition 6 (*Class-Specific Shapelet(CSS)*). The pattern that can maximally reflect the distinguishing feature of a specific category is called the class-specific shapelet of that class. A CSS has a great discriminatory power to determine whether or not a time series belongs to a category.

Based on these definitions, we next introduce a generalized model for the shapelet learning.

3.2. Generalized shapelet learning model

Since the distance between a time series and a shapelet can indicate the category [35], a recent study [5] proposed a logistic regression model that takes the linear combination of the distance between a time series and a set of shapelets to predict the target of the time series. For a data set D of two categories $C = \{0, 1\}$, the target $Y_i \in C$ of instance $T_i \in D$ can be approximately predicted by the linear model shown in Eq. 3.

$$\widehat{Y}_i = W_0 + \sum_{k=1}^{K} W_k Dis(S_k, T_i) \tag{3}$$

where $Dis(S_k, T_i)$ is the distance between the k-th shapelet S_k and the i-th time series T_i, and $\boldsymbol{W}$ is the linear weights. W_0 is the bias. The model can be learned by minimizing the logistic loss between the true target and the predicted one [5].

A test instance T_i is classified by utilizing the logistic sigmoid function, as shown in Eq. 4.

$$\sigma\left(\widehat{Y}_i\right) = \left(1 + e^{-\widehat{Y}_i}\right)^{-1} \tag{4}$$

where $\widehat{Y}_i$ is the predicted output of the model in Eq. 3, and $\sigma(\cdot)$ is the sigmoid function.

The output can be interpreted as the probability the test instance belongs to the positive class, which contributes to the extension to the multi-class problem.

The multi-class problem with label set C is converted into $|C|$ many one-vs-rest subproblems. A binary submodel is established based on Eq. 3 for each class to distinguish that class from the others. The target of the instances in dataset D is transformed into a binary target where an instance is labeled as "1" if it belongs to the class c else "0" for each class c in the label set C, as represented in Eq. 5.

$$Y_{i,c} = \begin{cases} 1, & Y_i = c \\ 0, & Y_i \neq c \end{cases} \quad \forall T_i \in D, \quad \forall c \in C \tag{5}$$

In this study, we adopt this logistic regression architecture for shapelet learning, but we learn the shapelets in a novel way. We discuss it in detail in Section 4.

3.3. Approximate motif discovery

Finally, we present an approximate motif discovery technique that is helpful for us to find the shapelet candidates. The time series motif [41] is the frequently occurring sequence of the time series that can reflect significant features of the data. While mining motifs of unknown length is computationally expensive, a recent study [42] proposed an ultra-fast algorithm that can find the approximate motif of variablelength in linear time with regard to the length of the time series.

First, the input time series T of length N is discretized to a word sequence by Symbolic Aggregate Approximation (SAX) [30] with word size w, alphabet size a and sliding window size n, producing an ordered sequence of equal-length words. The word length is w and the sequence length is $N - n + 1$.

Next, a grammar induction algorithm, called *Sequitur* [43], is conducted on the transformed sequence to extract repeated series of words. *Sequitur* is a string compression algorithm that recursively replaces repeated substrings with grammatical rules in linear time and space. *Sequitur* sequentially traverses every symbol of the input string. When a new symbol appears, the symbol and its predecessor symbol compose a digram. The algorithm maintains a hash table to store all the digrams. If the new-formed digram has existed, i.e., it appears somewhere in the preceding subsequence, these diagrams are replaced by a non-terminal symbol. If the non-terminal symbol has not yet existed, a new rule is formed with the non-terminal symbol on the left-hand side and the diagram on the right-hand side. Otherwise, the count of the existing rule is increased by one. *Sequitur* expands any non-terminal symbol to its digram if the corresponding rule appears less than twice. This mechanism

Table 1
An example of sequitur.

Grammar rule	Expanded rule
$R_0 \rightarrow R_1\ cda\ R_1\ R_2$	$\underbrace{\underbrace{abb\ abc}_{R_2}\ abd}_{R_1}\ cda\ \underbrace{\underbrace{abb\ abc}_{R_2}\ abd}_{R_1}\ \underbrace{abb\ abc}_{R_2}$
$R_1 \rightarrow R_2\ abd$	$\underbrace{abb\ abc}_{R_2}\ abd$
$R_2 \rightarrow abb\ abc$	$abb\ abc$

ensures that every rule represents a specific repeated subsequence. By adapting *Sequitur* to treat each SAX word as a symbol, each expanded rule becomes a repeated word subsequence of the input. As an example, the input $R_0 = abb\ abc\ abd\ cda\ abb\ abc\ abd\ abb\ abc$ is converted to the rules shown in Table 1.

The grammar rules of input R_0 are R_1 and R_2. The expanded rules of them, *abb abc abd* and *abb abc* are repeated subsequences of R_0.

The repeated SAX sequences can be easily mapped to their original time series subsequences by recording the offsets of the SAX words while conducting SAX transformation and grammar induction. Those time series subsequences are discovered approximate motifs.

4. Class-specific shapelet learning

Based on the preliminaries introduced in Section 3, in this section, we begin to describe our proposed CSSL approach, which learns as few as possible class-specific shapelets with variable-length for each class in order to achieve an accurate, efficient and interpretable classification. First, we provide an overview of our method.

4.1. Overview

Unlike existing study [5] that learns the shapelets shared by all classes, we modify the logistic regression model described in Section 3.2 to treat the subproblem of distinguishing every class from the others with specific shapelets for more efficient model training since shapelets are only required to be updated for the submodel of the same class. Moreover, while some of the class-shared shapelets are useless for identifying certain classes, all class-specific shapelets are beneficial for distinguishing one from the others which contributes to a simpler model explanation. For a time series dataset D with targets C, the submodel of a class $c(c \in C)$ predicts the binary target of an instance $T_i \in D$ by the class-specific shapelets of class c, as shown in Eq. 6. For the sake of description, we label the model as the CSSL(class-specific shapelets learning) model.

$$\widehat{Y}_{i,c} = W_{c,0} + \sum_{k=1}^{K_c} W_{c,k} Dis(T_i, S_{c,k}) \tag{6}$$

where K_c is the number of shapelets of class c,$S_{c,k}$ identifies the k-th shapelet of class c, and $W_{c,k}$ is the corresponding weight. $Dis(\cdot,\cdot)$ is the distance between two sequences defined in *Definition 4*.

As the sigmoid function of the predicted target represents the probability confidence that an instance T_i belongs to class c, a test instance T_{test} is classified to the class whose classifier yields to the maximal confidence, as shown in Eq. 7.

$$\widehat{Y}_{test} = \arg\max_{c \in C} \sigma\left(\widehat{Y}_{test,c}\right) \tag{7}$$

where $\widehat{Y}_{test,c}$ is the output of the CSSL model of Eq. 6, $\sigma(\cdot)$ is the sigmoid function, and $\widehat{Y}_{test}$ is the final predicted label.

In summary, we demonstrate the architecture of the CSSL model in Fig. 3. Note that we've mapped the generalized label set C to $\{1, 2, \ldots, |C|\}$ for a unifying and simplified presentation.

As the model is established for classification, we formulate the optimization problem as minimizing the regularized classification loss $\mathcal{F}$, as shown in Eq. 8.

$$\arg\min_{S,W} \mathcal{F} = \sum_{i=1}^{M} \sum_{c \in C} \mathcal{L}\left(Y_{i,c}, \widehat{Y}_{i,c}\right) + \lambda_W \|W\|^2 \tag{8}$$

where $\mathcal{L}\left(Y_{i,c}, \widehat{Y}_{i,c}\right)$ is the loss between the true target and the predicted one, and λ_W is the regularization term. M is the number of training instances in the time-series dataset, and C is the label set. S and W represent the shapelets and weights to be learned, respectively.

Based on the CSSL model in Eq. 6 and the objective function in Eq. 8, we can learn the class-specific shapelets by minimizing the objective function $\mathcal{F}$. Considering that $\mathcal{F}$ is a non-convex function since both shapelets S and the linear weights W

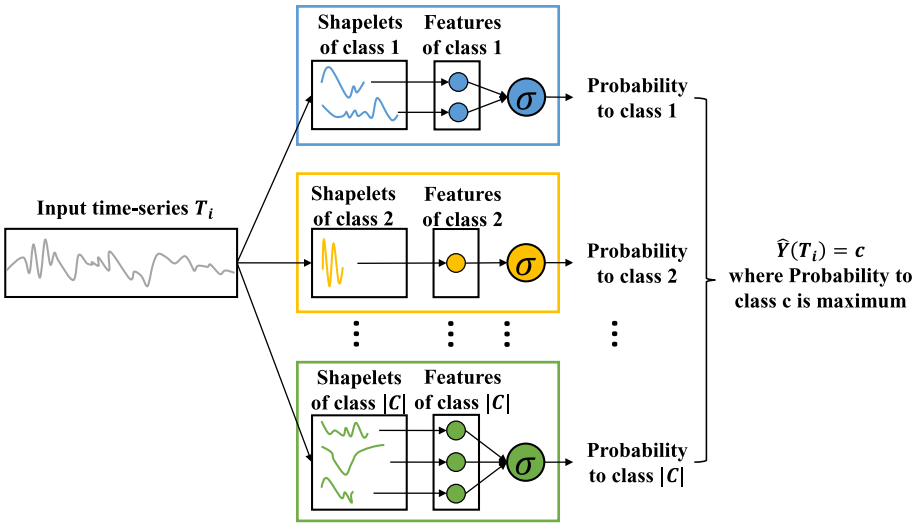


Fig. 3. The architecture of the CSSL model.

are required to be learned, we adopt a stochastic gradient descent to minimize $\mathcal{F}$ for the efficiency issue. Therefore, the time complexity of updating a shapelet at one iteration is still $O\left(MN^2\right)$ as the work of [5], while the number of calculations of shapelet updating is equal to the total number of all the class-specific shapelets, i.e., $N_u = K = \sum K_c$, which we will discuss in detail in Section 4.3.

To reduce the number of shapelets for efficiency and interpretability, we desire as few as possible shapelets while guaranteeing a competitive classification performance. Furthermore, we find that setting the proper length of the shapelets is a key to achieve our goal because if the set length can match the length of the true shapelets, we can finally learn the true shapelets when the algorithm converges. Also, by the definition of the class-specific shapelets, the true shapelets are exactly the least number of discriminatory patterns.

Therefore, we propose a method to automatically determine the length and number of the shapelets to be learned. It first discovers the variable-length frequent subsequences from the training series of each class, then it selects as few as possible representatives from these subsequences with the best discriminatory power. The basic idea is that the shape of the true class-specific shapelets represent some distinctive conditions of the class they belong to, such as arrhythmia shown in a set of ECG signals of some heart disease patients. Therefore, the subsequences that frequently occur in the training series of a particular class but removed from the others are very likely to be similar to the true shapelets specific to that class in shape, i.e., they have almost the same length and differ little in values. Thus, we discover such subsequences from the training data as shapelet candidates and set the number and length of the shapelets to be learned the same as the candidates. Moreover, from the perspective of an optimization problem, the similarity between the candidates and the shapelets in shape means that they are probably located nearby. This motivates us to initialize the values of the shapelets with the corresponding candidates to pursue more efficient convergence due to a lower number of iterations.

Based on this discussion, we can divide the overall training process of our CSSL method into two steps. The first step determines the number of shapelets, K_c, for each class $c(c \in C)$ and the length and initial values of each shapelet. The second iteratively optimizes the classification objective $\mathcal{F}$ by updating the shapelets and weights. Thus, our CSSL method is designed as a two-phase algorithm. In the first phase, we discover an optimal number of variable-length discriminatory motifs from the training series as the shapelet candidates to initialize the number, lengths, and values of the shapelets in the learning model. During the second phase, the CSSL model is initialized with the discovered shapelet candidates, and the classification objective is minimized in a gradient descent fashion to simultaneously learn the final shapelets and the weights. Once the learning is finished, the learned model, including the class-specific shapelets and the weights, are jointly used to predict the new arriving time-series. The block diagram of our proposed method is depicted in Fig. 4. We describe the two steps of the training phase, i.e., class-specific shapelet candidates discovery and classification objective optimization in detail in Section 4.2 and Section 4.3, respectively.

4.2. Class-specific shapelet candidates discovery

The purpose of this step is to automatically determine the number of class-specific shapelets of each class as well as the length and initial values of each shapelet in order to initialize the CSSL model. To guarantee a good classification performance, three requirements should be met.

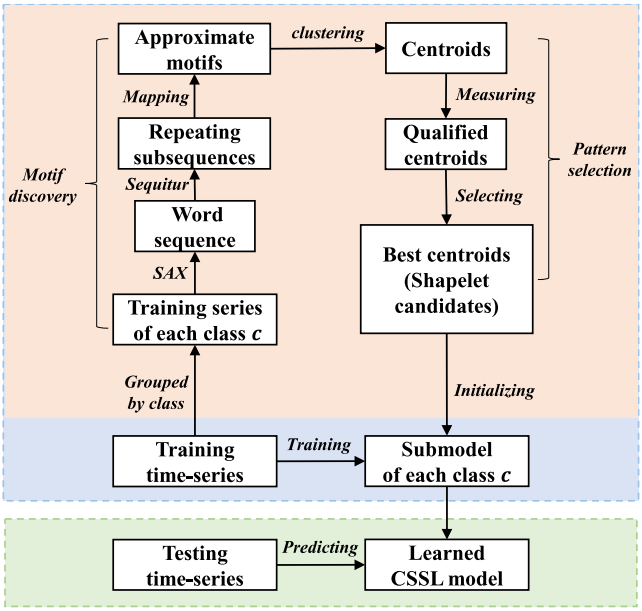


Fig. 4. Block diagram of our proposed CSSL method.

- The amount of class-specific shapelets of each class should be as few as possible without decreasing the accuracy to ensure the efficiency of model optimization and the interpretability of classification results.
- In the real-world problem, the true shapelet can be in any unknown length. Thus, the shapelet of any length should be considered so that to learn the optimum.
- The initial shapelets should be discriminatory to near the optima of the classification objective $\mathcal{F}$ in order to achieve a high prediction accuracy and contribute to the efficiency.

To satisfy these requirements, we discover the optimal number of variable-length discriminatory motifs from the training time series as the class-specific shapelets candidates to initialize the learning model. For each class, we first discover the variable-length repeated subsequences, i.e. the motifs, of the time series of the class. We then select as few as possible discriminatory sequences from the motifs as the shapelet candidates of the class. The details of the class-specific shapelet candidates discovery are discussed in Sections 4.2.1 and 4.2.2.

4.2.1. Motif discovery

Intuitively, class-specific shapelets are distinguishing features of the corresponding class, and thus should frequently occur in such a class. Consequently, we concatenate all the series of the training data from the same class into a whole long series. We then discover the motifs of the long series as the shapelet candidates. Note that the discovered motifs will be subsequently adjusted when optimizing the objective function to ensure their discriminatory power, so we prefer to mine the approximate motifs rather than the accurate ones on account of efficiency. Moreover, since the shapelets could have an arbitrary unknown length, it is desirable to automatically detect motifs of any possible lengths. Consequently, we adopt the approximate variable-length motif discovery algorithm described in Section 3.3 to initially find the class-specific shapelet candidates.

As discussed in Section 3.3, the motifs are mapped from the discovered repeated SAX strings. Because the sequences represented by the same SAX words with the same parameters have a similar discriminatory power, we always map the discovered word sequences to the time sequences first appearing on the whole series. This one-to-one mapping is conducive to our following processes. In addition, our algorithm ignores subsequences spanning time series junction points, because they are artifacts due to concatenation.

4.2.2. Pattern selection

The motifs discovered in Section 4.2.1 contains many similar patterns due to the feature of SAX and the Sequitur algorithm. These similar patterns have almost no difference in discriminatory power. Thus, we only select one representative from each group of similar motifs as a class-specific shapelet. This strategy can significantly improve the efficiency and interpretability of classification by reducing the number of shapelets, while guaranteeing accuracy. The pattern selection

is typically a clustering problem where each centroid is a representative. Thus, we develop a clustering algorithm to select the class-specific shapelet candidates based on k-means and design a heuristic to automatically determine the number of clusters k, i.e., the number of shapelets of each class. The pattern selection algorithm is discussed as follows.

First, the classical k-means algorithm calculates the average of the observations in a cluster as its center. However, the average of sequences with variant length is meaningless. Thus, we take the sequence of a cluster that is the global nearest to all the other sequences in the cluster as the centroid, as defined in *Definition 7*.

Definition 7 (*Centroid*). The centroid of a set of patterns A is defined as the pattern $P \in A$ that contains the minimum summary of distance to all other patterns within A, as formulated in Eq. 9.

$$centroid = \arg\min_{P \in A} \sum_{\forall P' \in A} \left(Dis(P', P)\right)^2 \tag{9}$$

where $Dis(\cdot,\cdot)$ is the distance between two sequences defined in *Definition 4*.

Next, we design a heuristic to initialize the k clustering centers to accelerate convergence. The basic idea is to locate each initial center far from the others in order to disperse these centers in different clusters. The first center is selected at random from the input instances. The instance having maximum distance or minimum similarity with the previously initialized centers is then selected. This step is repeated $k-1$ times to initialize the remaining centers one by one. The selected observations will not be checked in the following rounds to guarantee every center is unique.

Finally, but most importantly, we propose a method to automatically determine the number of clusters, i.e., the value of k. For each class of the training dataset, its k value determines the number of class-specific shapelets found by our method. However, we have no prior knowledge of how many discriminative patterns each class has. A straightforward solution is to search for the optimal k for each class of the training dataset by cross-validation, taking the classification accuracy as the metric. Nevertheless, this method needs to conduct not only motifs clustering but shapelet learning and classification model training(described in Section 4.3) hundreds, even thousands of times.

Although the efficiency of the latter process is significantly improved by pattern reduction, executing it so many times is still time-consuming. Instead, we choose to determine the value of k for each class without training classification models for saving time, while guaranteeing the final classification performance. Specially, we propose a metric called *closeness* in *Definition 8*, to evaluate the quality of the motifs clustering.

Definition 8 (*Closeness*). Given a time series dataset D with targets C, the closeness between a set of patterns A and a class c ($c \in C$) is defined as the opposite of the mean of the average distance from all patterns within A to class c, as Eq. 10 shows.

$$Closeness(S, c) = -\frac{1}{|A|} \sum_{\forall P \in A} \bar{Dis}(P, c) \tag{10}$$

where $\bar{Dis}(P, c)$ is the average distance between pattern P and a class c ($c \in C$), defined as follows.

Definition 9 (*Average Distance*). Let D be a time series dataset with label set C. The average distance between a pattern P and a class c ($c \in C$) is defined as the difference between the mean of the distance from the pattern P to every time-series within class c and the mean of the distance from p to every time-series out of class c, as formulated in Eq. 11.

$$\bar{Dis}(P, c) = \frac{\sum_{T_i \in D_c} \sqrt{Dis(T_i, P)}}{|D_c|} - \frac{\sum_{T_j \in \bar{D}_c} \sqrt{Dis(T_j, P)}}{|\bar{D}_c|} \tag{11}$$

where $Dis(\cdot,\cdot)$ is the distance between two sequences defined in *Definition 4*, $D_c = \{T_i \in D |\ Y_i = c\}$ and $\bar{D}_c = \{T_i \in D |\ Y_i \neq c\}$ denote the sets of instances in and out of class c, respectively.

Intuitively, the higher the *closeness* is, the more the selected motifs can differentiate the same category with others on average. Thus, for each class, we vary the number of clusters from 1 to the maximal number, i.e., $k = 1, 2, \ldots, k_{max}$, respectively. The clustering centroids having the highest *closeness* are selected as the final shapelet candidates.

The patterns can be clustered in two fashions due to different similarity measurement spaces, i.e., string similarity or time sequence similarity. In the first fashion, the repeated word sequences found by sequitur are clustered, and then the centroids are mapped to time sequences, while in the second fashion, clustering is conducted on time series transformed from the SAX sequences. Both of them make sense, so we set the similarity metric of clustering as a hyper-parameter and determine it by cross-validation.

Algorithm 1. Class-Specific Shapelet Candidates Discovery

Input: Training dataset D, Label set C, SAX parameters $para$, Maximal number of candidates k_{max}, Distance metric Dis

Output: Class-specific shapelet candidates $SC \in \mathbb{R}^{|C|\times **\times}$

1: **for** All$c \in C$ **do**
2: $T_{con} = Concatenate(c)$
3: $R_0 = SAX\_Converter(T_{con},\ para)$
4: $repeated\_seq = Modified\_Sequitur(R_0)$
5: $max\_closeness = 0$
6: $best\_centroids = \phi$
7: **for** $k = 1, 2, \ldots, k_{max}$ **do**
8: $centroids = Clustering(repeated\_seq,\ k,\ Dis)$
9: **if** $Closeness(centroids) \geqslant max\_closeness$
10: $best\_centroids = centroids$
11: $max\_closeness = Closeness(centroids)$
12: **end if**
13: **end for**
14: $SC_c = best\_centroids$
15: **end for**
16: **return** SC

The pseudo-code of the class-specific shapelet discovery algorithm is depicted in Algorithm 1. For each class c (Line 1), the algorithm concatenates the training time series belonging to c (Line 2) and detects the variable-length approximate motifs of the concatenated series (Lines 3–4). The discovered motifs are then clustered into k groups with k ranging from 1 to the maximal number k_{max} (Lines 7–8). The centroids of each clustering process compose a group of representative subsequences and the group with maximal *Closeness* (Lines 9–11) is determined as the class-specific shapelet candidates of class c. The SAX and Sequitur algorithm have a linear time complexity with respect to the sequence length. Suppose there are r repeated sequences with length L on average for each class. The complexity of clustering them into k clusters is $O\left(IL^2(kr + r^2)\right)$, where I is the number of iterations. The time for one calculation of *closeness* is $O(MNL)$, where M is the number of time series and N is the average series length of the training dataset. Thus, the time complexity of phase one is $O\left(MN^2 + |C|Lk_{max}(ILr + k_{max}MN)\right)$.

4.3. Classification objective optimization

After determining the number, length, and initial values of the class-specific shapelets in the learning model, in this section, we introduce the optimization algorithm to minimize the objective function $\mathcal{F}$ in Eq. 8 in order to jointly learn the class-specific shapelets and linear weights for classification. Since $\mathcal{F}$ is a non-convex function in terms of shapelets S and weights W, we adopt a stochastic gradient descent to optimize it for efficiency. At the beginning of the optimization, we initialize the shapelets with the candidates discovered in Section 4.2 and set the initial weights based on the discriminatory power of the corresponding shapelets in order to ensure a desirable classification accuracy and improve the learning efficiency. The shapelets and weights of the submodel of each category are then updated by taking steps towards minimal classification objective $\mathcal{F}$.

4.3.1. Shapelets initialization

We initialize the class-specific shapelets in the submodel of class $c(c \in C)$ with the corresponding shapelet candidates SC_c discovered in Section 4.2. Since the candidates have been discriminatory to some extent, they are very likely to be located in the regions around the true shapelets, i.e., the optima of the objective function, which contributes to a high classification precision and is also beneficial for fast convergence with fewer iterations.

4.3.2. Weights initialization

It can be analyzed from the learning model in Eq. 6 that the more discriminative a class-specific shapelet is, the smaller its weight should be, in order to predict a higher value for a time series close to the corresponding shapelet. As the discriminatory power of a shapelet can be initially evaluated with the average distance defined in Eq. 11, we initialize each category with the normalized average distance, which is a number around 0 and proportional to the average distance, as depicted in Eq. 12.

$$W_{c,k} = \frac{\bar{Dis}(S_{c,k}, c)}{\sum_{k'=1}^{K_c} \bar{Dis}(S_{c,k'}, c)} \tag{12}$$

where $\bar{Dis}(\cdot,\cdot)$ is the average distance between a shapelet and a class defined by *Definition 9*.

4.3.3. Per-instance objective function

As discussed above, we adopt a stochastic gradient descent fashion to optimize the classification objective on account of the feasibility of the running time. SGD remedies the prediction error caused by one instance at a time. Therefore, we decompose the objective function in Eq. 8 into a per-instance objective function for each class $c(c \in C)$, denoted as $\mathcal{F}_{i,c}$, which is a portion of the objective function that only contains the terms with respect to the instance T_i and the class c, as shown in Eq. 13.

$$\mathcal{F}_{i,c} = \mathcal{L}\left(Y_{i,c}, \widehat{Y}_{i,c}\right) + \frac{\lambda_W}{M} \sum_{k=1}^{K_c} W_{c,k}^2 \tag{13}$$

Subsequently, we declare the required terms for deriving the optimization algorithm, including the loss function and the differentiable distance function.

4.3.4. Loss function and differentiable distance function

The loss function of the logistic regression model is the logistic loss [44,5]. The logistic classification loss between true target $Y_{i,c}$ and the estimated one $\widehat{Y}_{i,c}$ is shown in Eq. 14.

$$\mathcal{L}\left(Y_{i,c}, \widehat{Y}_{i,c}\right) = -Y_{i,c} \ln \sigma\left(\widehat{Y}_{i,c}\right) - (1 - Y) \ln\left(1 - \sigma\left(\widehat{Y}_{i,c}\right)\right) \tag{14}$$

where $\sigma(\cdot)$ is the sigmoid function.

The stochastic gradient descent approach requires the involved functions of the learning model to be differentiable. However, the minimum function is not differentiable, and the partial derivative $\partial Dis/\partial S$ is undefined. Thus, a differentiable approximation of the minimum function, i.e., the *Soft Minimum* function is introduced to substitute the minimum function, and the distance between two sequences T_m and T_n of length L_m and L_n ($L_m \geqslant L_n$) is redefined as Eq. 15.

$$\hat{Dis}(T_m, T_n) = \frac{\sum_{j=1}^{J} Dis^j(T_m, T_n) e^{\alpha Dis^j(T_m, T_n)}}{\sum_{j'=1}^{J} e^{\alpha Dis^{j'}(T_m, T_n)}} \tag{15}$$

where $Dis^j(\cdot,\cdot)$ is the point-wise distance between a subsequence of a series and another series defined in *Definition 4*, and $J = L_m - L_n + 1$. The parameter α controls the precision of the function and Eq. 15 is equivalent to the distance $Dis(T_m, T_n)$ when α is negative infinity. Practically, $\alpha = -30$ is small enough to make the soft minimum distance yield to the real minimum distance. Therefore, we fix this value throughout this study.

We have declared the loss function and refined all functions within the learning model to be differentiable, thus, the gradients of the per-instance objective function $\mathcal{F}_{i,c}$ with respect to the class-specific shapelets and classification weights can be derived through the chain rule of derivation.

4.3.5. Gradient for shapelet

The derivative of $\mathcal{F}_{i,c}$ with respect to the k-th shapelet of class c at the l-th position is derived through the chain rule of derivation in Eq. 16-Eq. 19.

$$\frac{\partial \mathcal{F}_{i,c}}{\partial S_{c,k}^l} = -\left(Y_{i,c} - \sigma\left(\widehat{Y}_{i,c}\right)\right) \frac{\partial \hat{Dis}(T_i, S_{c,k})}{\partial S_{c,k}^l} W_{c,k} \tag{16}$$

where $S_{c,k}$ is the k-th shapelet of class c, $W_{c,k}$ is the corresponding weight and $S_{c,k}^l$ is the l-th point of the shapelet. The derivative $\frac{\partial \hat{Dis}(T_i, S_{c,k})}{\partial S_{c,k}^l}$ is

$$\frac{\partial \hat{Dis}(T_i, S_{c,k})}{\partial S_{c,k}^l} = \sum_{j=1}^{J_k} \frac{\partial \hat{Dis}(T_i, S_{c,k})}{\partial Dis^j(T_i, S_{c,k})} \frac{\partial Dis^j(T_i, S_{c,k})}{\partial S_{c,k}^l} \tag{17}$$

where the derivatives $\frac{\partial \hat{Dis}(T_i, S_{c,k})}{\partial Dis^j(T_i, S_{c,k})}$ and $\frac{\partial Dis^j(T_i, S_{c,k})}{\partial S_{c,k}^l}$ are derived in Eq. 18 and Eq. 19, respectively.

Algorithm 2. Classification Objective Optimization

Input: Training dataset D, Label set C, Shapelet Candidates SC, Learning Rate η, Regularization λ_W, Number of Iterations I

Output: Class-Specific Shapelets $S \in \mathbb{R}^{|C|\times\ *\times *}$, Linear Weights $W \in \mathbb{R}^{|C|\times *}$ and Bias items $W_0 \in \mathbb{R}^{|C|}$

Initialize S, W, W_0
for $iter = 1, \ldots, I$ **do**
for $i = 1, \ldots, M$ **do**
for $c \in C$ **do**
for $k = 1, \ldots, K_c$ **do**
$J_k = L_{T_i} - L_{Sc,k} + 1$
$\Psi_k = 0$
for $j = 1, \ldots, J_k$ **do**
Calculate $D^j(T_i, S_{c,k})$ with Eq. 1
$\xi_k^j = e^{\alpha D^j(T_i, S_{c,k})}$
$\Psi_k += \xi_k^j$
end for
Calculate $\widehat{D}(T_i, S_{c,k})$ with Eq. 15
end for
Calculate $\widehat{Y}_{i,c}$ with Eq. 6
$\vartheta = \widehat{Y}_{i,c} - \sigma\left(\widehat{Y}_{i,c}\right)$
for $k = 1, \ldots, K_c$ **do**
$W_{c,k} += \eta\left(\vartheta \widehat{D}(T_i, S_{c,k}) - \frac{2\lambda_W}{M} W_{c,k}\right)$
for $j = 1, \ldots J_k$ **do**
$\phi = \frac{2\xi_k^j\left(1+\alpha\left(D^j(T_i,S_{c,k}) - \widehat{D}(T_i,S_{c,k})\right)\right)}{L_{S_{c,k}}\Psi_k}$
for $l = 1, \ldots, L_{S_{c,k}}$ **do**
$S_{c,k}^l += \eta\vartheta\phi\left(S_{c,k}^l - T_i^{j+l-1}\right)W_{c,k}$
end for
end for
end for
$W_{c,0} += \eta\vartheta$
end for
end for
end for
return S, W, W_0

$$\frac{\partial \hat{Dis}(T_i, S_{c,k})}{\partial Dis^j(T_i, S_{c,k})} = \frac{\xi_k^j\left(1 + \alpha\left(Dis^j(T_i, S_{c,k}) - \hat{Dis}(T_i, S_{c,k})\right)\right)}{\sum_{j'=1}^{J_k}\xi_k^{j'}} \tag{18}$$

$$\frac{\partial Dis^j(T_i, S_{c,k})}{\partial S_{c,k}^l} = -\frac{2}{L_{S_{c,k}}}\left(T_i^{j+l-1} - S_{c,k}^l\right) \tag{19}$$

where $\xi_k^j = e^{\alpha Dis^j(T_i, S_{c,k})}$ and $J_k = L_{T_i} - L_{S_{c,k}} + 1$.

4.3.6. Gradient for weights

The gradients of $\mathcal{F}_{i,c}$ with respect to the k-th linear weight and the bias of class c are depicted in Eq. 20 and Eq. 21.

$$\frac{\partial \mathcal{F}_{i,c}}{\partial W_{c,k}} = -\left(Y_{i,c} - \sigma\left(\widehat{Y}_{i,c}\right)\right)\hat{Dis}(T_i, S_{c,k}) + \frac{2\lambda_W}{M} W_{c,k} \tag{20}$$

$$\frac{\partial \mathcal{F}_{i,c}}{\partial W_{c,0}} = -\left(Y_{i,c} - \sigma\left(\widehat{Y}_{i,c}\right)\right) \tag{21}$$

where $W_{c,k}$ and $W_{c,0}$ represent the weight of the distance between a time series T_i and the k-th shapelet, and the bias of class c, respectively.

Having derived the gradients, we introduce the classification objective optimization algorithm that iteratively updates the shapelets and weights in the negative direction of the gradients. As shown in Algorithm 2, in the beginning, the class-specific shapelets and weights are initialized with the candidates and corresponding normalized average distance (Line 1). The algorithm then iteratively updates the values of the class-specific shapelets and the weights of each class in the negative direction of the gradient versus each instance. The frequently used intermediate items of the gradients are pre-calculated (Lines 6–16, Line 20) to avoid repeatedly computing the same items to accelerate the optimization. These intermediated items are utilized to calculate the gradients and update the values of the shapelets and weights (Lines 18–26). The time complexity of the model training stage is $O\left(IKMN^2\right)$, where I is the number of iterations, M is the size of the training dataset, K is the total number of shapelets of all classes, and N is the maximum length of the training time series. As can be seen, the main time consumption of the optimization is to update each of the K shapelets for I iterations, i.e., the number of shapelet updating calculations at each iteration, N_u, is equivalent to the number of shapelets K, where updating a shapelet at one iteration takes $O\left(MN^2\right)$ time. The time complexity of predicting one instance of length N_t is $O\left(KN_t^2\right)$.

We conducted extensive experiments to evaluate our proposed CSSL method. The experimental results are presented and discussed in the following section.

5. Experimental results

5.1. Experiment setup

We evaluate the classification performance of our method on 25 datasets of the UCR archive [7] commonly used by shapelet-based methods [6,36,3–5]. For a fair comparison, we use the default train/test splits. The information about the datasets is presented in Table 2.

Six hyper-parameters are required to tune in our method, including the word size w and the alphabet size a of SAX, the similarity metric of sequence clustering *Dis*, the learning rate η, the regularization λ_W and the number of iterations I of the optimization algorithm. They are set through cross-validation over the training data. The word size and alphabet size are searched from $w \in \{4, 6, 8\}$ and $a \in \{4, 6\}$, respectively. The similarity metric can be *ED* (Edit Distance) or *LCS* (Longest Common Subsequence) in the SAX word space, or *DS* (distance between two sequences defined in Eq. 2) in the time sequence space. Both the learning rate η and the regularization λ_W are selected from $\{0.01, 0.1\}$, while the number of iterations is searched in the range of $I \in \{500, 1000, 1500, 2000, 2500, 3000\}$. The window size of SAX is fixed at $n = 0.1 * N$, and the

Table 2
Dataset information and hyper-parameter setting.

		Dataset Information					Hyper-Parameter Setting					
#	Dataset	Type	Classes	Length	Train Size	Test Size	w	a	*Dis*	η	λ_W	I
1	Beef	Spectro	5	470	30	30	4	4	*ED*	0.1	0.01	500
2	BeetleFly	Image	2	512	20	20	4	6	*DS*	0.01	0.1	500
3	BirdChicken	Image	2	512	20	20	6	4	*DS*	0.1	0.01	500
4	CBF	Simulated	3	128	30	900	4	6	*LCS*	0.1	0.1	500
5	ChlorineConcentration	Simulated	3	166	467	3840	6	6	*ED*	0.01	0.01	2500
6	Coffee	Spectro	2	286	28	28	4	6	*LCS*	0.01	0.01	500
7	ECG200	ECG	2	96	100	100	4	4	*LCS*	0.01	0.01	1000
8	ECGFiveDays	ECG	2	136	23	861	4	4	*DS*	0.1	0.1	500
9	FaceFour	Image	4	350	24	88	6	4	*DS*	0.1	0.01	500
10	FacesUCR	Image	14	131	200	2050	8	4	*DS*	0.1	0.01	500
11	Gun_Point	Motion	2	150	50	150	6	6	*DS*	0.01	0.01	1500
12	ItalyPowerDemand	Sensor	2	24	67	1029	4	4	*ED*	0.1	0.01	1000
13	Lighting2	Sensor	2	637	60	61	6	6	*ED*	0.01	0.01	3000
14	Lighting7	Sensor	7	319	70	73	8	4	*DS*	0.1	0.01	1500
15	MedicalImages	Image	10	99	381	760	4	4	*LCS*	0.01	0.1	2500
16	MoteStrain	Sensor	2	84	20	1252	4	6	*ED*	0.01	0.1	1000
17	OliveOil	Spectro	4	570	30	30	4	4	*LCS*	0.01	0.1	500
18	OSULeaf	Image	6	427	200	242	6	6	*DS*	0.1	0.01	500
19	SonyAIBORobotSurface	Sensor	2	70	20	601	4	6	*DS*	0.1	0.01	2000
20	SonyAIBORobotSurfaceII	Sensor	2	65	27	953	6	4	*DS*	0.01	0.1	500
21	Symbols	Image	6	398	25	995	6	4	*ED*	0.1	0.01	2000
22	synthetic_control	Simulated	6	60	300	300	8	4	*LCS*	0.01	0.1	1000
23	Trace	Sensor	4	275	100	100	8	4	*ED*	0.01	0.1	500
24	TwoLeadECG	ECG	2	82	23	1139	4	6	*ED*	0.01	0.01	2500
25	wafer	Sensor	2	152	1000	6164	6	4	*ED*	0.01	0.1	2000

maximal number of shapelets is $k_{max} = |C|log(N)$, where N is the length of the time series, and $|C|$ is the number of categories. The tuned hyper-parameters of each dataset are shown in Table 2. The algorithm is implemented in standard C++, compiled with GCC 4.8 and run on a CentOS server with Intel(R) Xeon(R) E7-8860 v3 CPU (16 cores, 2.20GHZ).

We first evaluate the acceleration of our CSSL over the existing shared shapelets learning approach, Learning Shapelet(LS). Since the process of shapelet candidates discovery and classification objective optimization of each class is independent with others, we can easily parallelize the training algorithm of our CSSL into $|C|$ branches for further acceleration, where $|C|$ is the number of categories of a classification problem. Thus, we also develop the parallel version of CSSL, denoted as CSSL-PL, and evaluate its performance on efficiency. We depict the pseudo-code of CSSL-PL in Algorithm 3. For each class in parallel, the algorithm discovers shapelet candidates specific to that class to initialize the submodel of the class (Lines 2–3) and then updates the class-specific shapelets and weights in SGD (Lines 6–7). We implement CSSL-PL with OpenMP [45].

Algorithm 3. CSSL-PL

Input: Training dataset D, Label set C, SAX parameters *para*, Maximum number of shapelets k_{max}, Distance metric *Dis*, Learning Rate η, Regularization λ_W, Number of Iterations I

Output: Class-Specific Shapelets $S \in \mathbb{R}^{|C|\times **\times}$, Linear Weights $W \in \mathbb{R}^{|C|\times *}$ and Bias items $W_0 \in \mathbb{R}^{|C|}$

1: **parallel for** $c \in C$ **do**
2: Discover shapelet candidates of class c
3: Initialize S_c, W_c, $W_{c,0}$
4: **for** $iter = 1, \ldots, I$ **do**
5: **for** $i = 1, \ldots M$ **do**
6: Pre-calculate frequently used intermediate terms
7: Update S_c, W_c, $W_{c,0}$
8: **end for**
9: **end for**
10: **end parallel for**
11: **return** S, W, W_0

We then evaluate the classification performance of our CSSL approach. We compare our method against 18 representative classifiers in terms of classification accuracy. These methods can be divided into two different types. The first type is the standard classifier that takes each time-series as a high dimension vector, including the well-known one nearest neighbor classifier with the Euclidean distance as the distance metric(1NN-ED), Naive Bayesian classifier(NB), C4.5 decision tree algorithm(C45), support vector machine with the linear kernel (SVML) and quadric kernel (SVMQ), Bayesian Network (BN), random forest classifier (RandF), rotation forest classifier(RotF) and multi-layer perceptron model (MLP). The second group is the state-of-the-art classifier specially designed for time-series classification problems by extracting representative time-series features. Based on the difference in the features, these classifiers can be further split into 5 types, as follows.

- Whole-series-based
  Including one nearest neighbor classifier using Dynamic Time Wrapping as distance metric with best window size (1NN-DTWB), which is the most representative baseline of the time-series classification problem [2].
- Interval-based
  1) TSF [24] that selects random intervals from the training series and extracts summary statistics of each interval as features to train a random forest classifier, and 2) RISE [27] that extracts Fourier, autocorrelation, and partial autocorrelation features of the random intervals to train each tree of the random forest classifier.
- Dictionary-based
  1) SAX-VSM [31] that builds the classifier based on the frequency of repeating subsequences and 2) SAX-VFSEQL [34] that transforms raw time series into SAX words and adapting a sequence learning model for classification, and 3) BOSS [21] that transforms raw time series into SFA words and conducts nearest neighbor classification by measuring a bespoke distance between the frequencies of the words.
- Shapelet-based
  1) Shapelet Transform (ST) [37] that first searches shapelets from the training data and then takes the distance between the training series and the shapelets as the feature vector to train a standard classifier like support vector machine or random forest, and 2) Learning Shapelets method (LS) [5] which learns the top-K shapelets by stochastic gradient descent.
- Hybrids
  HIVE-COTE [40] that combines four classifiers with different feature representations using a probabilistic hierarchy.

The results of SAX-VFSEQL are taken from the author's publication, while the others are reproduced by [2].
Finally, we discuss the advantage of our CSSL in terms of interpretability.
The experiment results are presented and discussed in detail in the following chapters.

5.2. Efficiency

We test the total training time of LS, CSSL, and CSSL-PL. The results are shown in Fig. 5. Each group of the histograms displays the total running time of LS, CSSL, and CSSL-PL on one dataset from left to right. For better visualization, the running time is depicted on logarithmic coordinates. Note that we present the histogram of CSSL with two stacked bars which represents the running time of the two phases of our CSSL method, i.e., shapelet candidates discovery(phase 1, depicted in the red bar) and classification objective optimization(phase 2, as orange histogram shows). The result shows that the running time of the shapelet candidates discovery is negligible compared with the time consumed by the classification objective optimization process, thus the efficiency of CSSL is mainly determined by the latter phase. This is reasonable because SAX and Sequitur are both linear algorithms with respect to the series number and length, and the clustering algorithm only processes a few small pieces of strings or sequences in stage one, while in the second phase, the shapelets need to be optimized for every training time series over hundreds of iterations. Compared to LS that learns numerous shapelets shared by all categories, our proposed method makes a breakthrough in efficiency. The non-parallel algorithm is up to 70 times faster, and the parallel version is up to 622 times faster. The main reason why our CSSL performs more efficiently than LS is that CSSL conducts much less shapelet updating at each iteration benefiting from the idea of learning class-specific and as few as possible shapelets. For example, on the *FacesUCR* dataset which contains 200 pieces of equal-length training time series in 14 classes, LS generates up to 420 shapelets shared by every category. Thus, each of them needs to be sequentially optimized for every submodel of the 14 classes, causing 5,880 calculations on shapelet updating in each iteration. On the contrary, with our CSSL approach, the average number of shapelets for each category is 14.3, thus there are only 200 shapelets updatings during one iteration. Table 3 shows the average number of shapelets of LS and CSSL and the number of calculations of them in terms of shapelet updating at each iteration for all the datasets we have evaluated. Besides, unlike LS that initialized the shapelets with the centroids of all subsequences, CSSL discovers discriminatory shapelet candidates by Algorithm 1 for shapelet initialization of the classification model. These candidates are more likely to be near the optimums thus causing fewer iterations. As Table 2 shows, our algorithm ends up with only 500 iterations on half of the datasets, while LS requires more than 5,000 iterations with a fixed learning rate and 1,000 iterations even using a variable learning rate strategy to speed up convergence [5]. Fewer iterations contribute to a faster training as well.

5.3. Classification accuracy

After testing the running time, we evaluate the classification accuracy of our CSSL method. The results are as follows.

The results of the classification accuracy rate of all the nine standard classifiers as well as our proposed CSSL on the UCR archive are shown in Table 4. The best performance of every dataset is highlighted in bold.

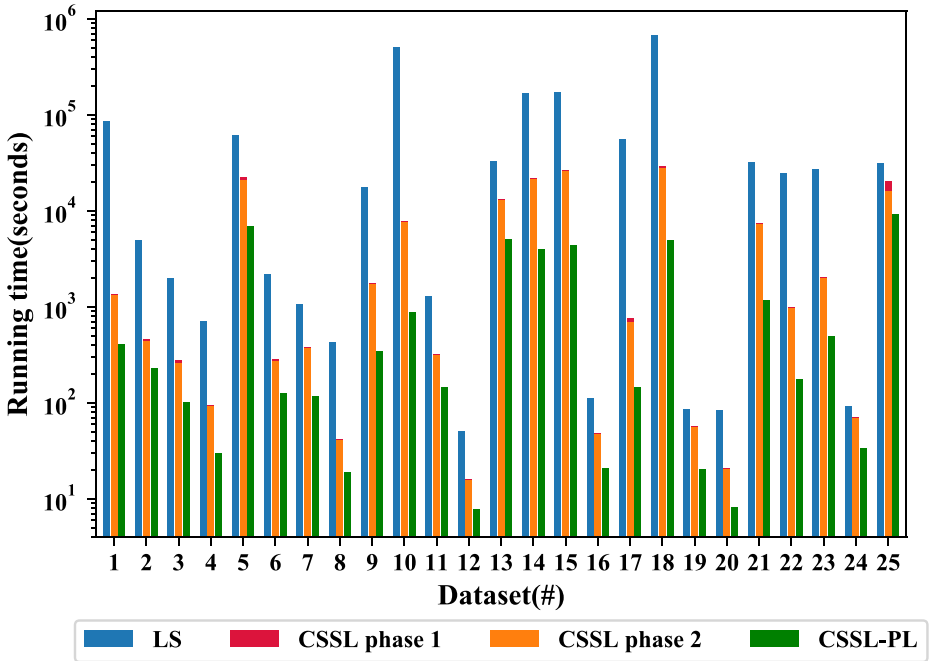


Fig. 5. The total training time of LS, CSSL, and CSSL-PL.

Table 3
The average number of shapelets for each class and the number of calculations of shapelet updating at each iteration.

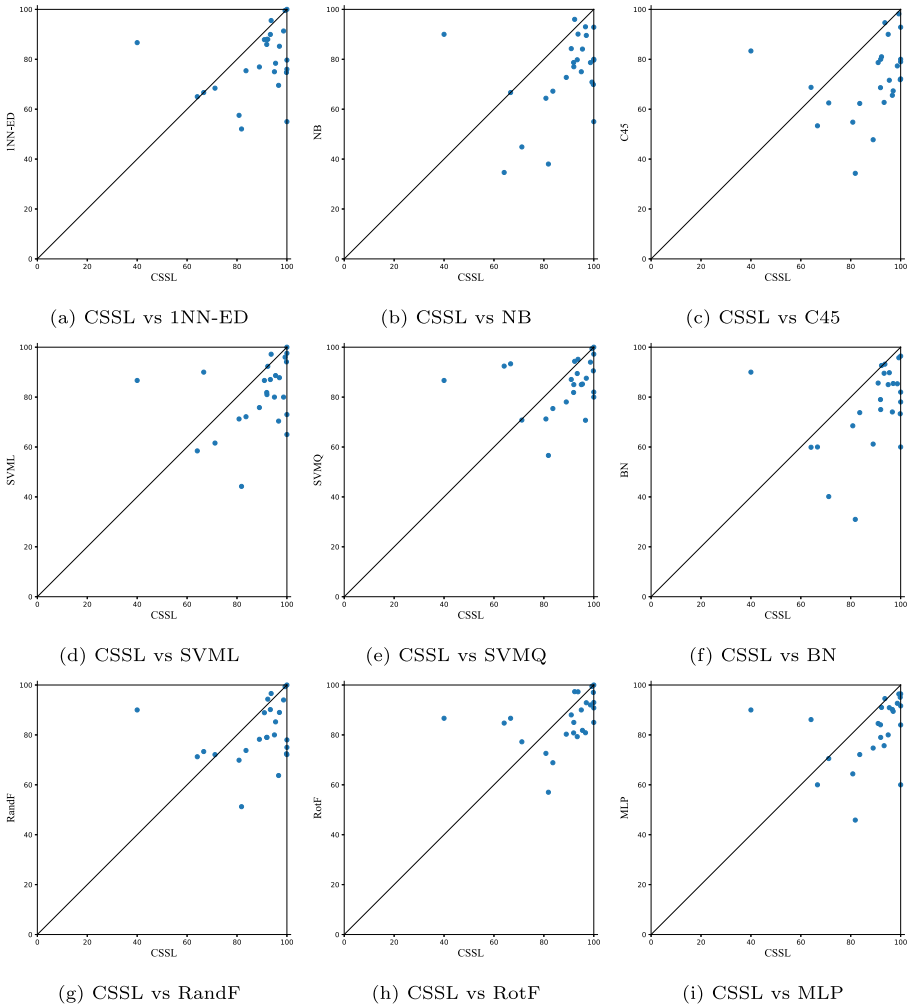
#	Dataset	$Num_{shapelet}$		$Num_{updating}$	
		LS	CSSL	LS	CSSL
1	Beef	120	4.2	600	21
2	BeetleFly	27	4	27	8
3	BirdChicken	18	2	18	4
4	CBF	34	6	102	18
5	ChlorineConcentration	69	9	207	27
6	Coffee	27	6	27	12
7	ECG200	27	8	27	16
8	ECGFiveDays	24	4	24	8
9	FaceFour	87	15	348	60
10	FacesUCR	420	14.3	5,880	200
11	Gun_Point	27	4	27	8
12	ItalyPowerDemand	24	4	24	8
13	Lighting2	33	5	33	10
14	Lighting7	195	15.7	1,365	110
15	MedicalImages	196	12	1,960	120
16	MoteStrain	24	6	24	12
17	OliveOil	93	1.75	372	7
18	OSULeaf	181	13	1,086	78
19	SonyAIBORobotSurface	24	5.5	24	11
20	SonyAIBORobotSurfaceII	24	6.5	24	13
21	Symbols	96	7.3	576	44
22	SyntheticControl	156	5.3	936	32
23	Trace	64	6.25	256	25
24	TwoLeadECG	14	3.5	14	7
25	Wafer	36	6.5	36	13

Table 4
Accuracy against standard classifiers.

#	Dataset	1NN-ED	NB	C45	SVML	SVMQ	BN	RandF	RotF	MLP	CSSL (Ours)
1	Beef	66.667	66.667	53.333	90.000	**93.333**	60.000	73.333	86.667	60.000	66.667
2	BeetleFly	75.000	75.000	90.000	80.000	85.000	85.000	80.000	90.000	80.000	**95.000**
3	BirdChicken	55.000	55.000	80.000	65.000	80.000	60.000	75.000	85.000	60.000	**100.000**
4	CBF	85.222	89.556	67.333	87.778	87.556	85.444	89.000	92.889	89.444	**97.000**
5	ChlorineConcentration	65.000	34.635	68.750	58.438	**92.422**	59.896	71.250	84.740	86.146	64.089
6	Coffee	**100.000**	92.857	92.857	**100.000**	**100.000**	96.429	**100.000**	**100.000**	96.429	**100.000**
7	ECG200	88.000	77.000	80.000	81.000	85.000	75.000	79.000	85.000	79.000	**92.000**
8	ECGFiveDays	79.675	79.675	72.125	97.561	97.213	78.049	72.125	90.825	91.638	**100.000**
9	FaceFour	78.409	84.091	71.591	88.636	85.227	89.773	85.227	81.818	90.909	**95.455**
10	FacesUCR	76.927	72.732	47.756	75.805	78.049	61.171	78.244	80.293	74.732	**88.976**
11	GunPoint	91.333	78.667	77.333	80.000	94.000	85.333	94.000	92.000	92.667	**98.667**
12	ItalyPowerDemand	95.530	90.087	94.655	97.182	95.141	93.197	96.599	**97.279**	94.558	93.683
13	Lightning2	75.410	67.213	62.295	72.131	75.410	73.770	73.770	68.852	72.131	**83.607**
14	Lightning7	57.534	64.384	54.795	71.233	71.233	68.493	69.863	72.603	64.384	**80.822**
15	MedicalImages	68.421	44.868	62.500	61.579	70.789	40.132	72.105	**77.237**	70.526	71.184
16	MoteStrain	87.859	84.265	78.674	86.661	87.061	85.623	88.898	88.019	84.585	**90.974**
17	OliveOil	86.667	**90.000**	83.333	86.667	86.667	**90.000**	**90.000**	86.667	**90.000**	40.000
18	OSULeaf	52.066	38.017	34.298	44.215	56.612	30.992	51.240	57.025	45.868	**81.818**
19	SonyAIBORobotSurface1	69.551	93.012	65.557	70.383	70.715	74.043	63.727	80.865	90.183	**96.672**
20	SonyAIBORobotSurface2	85.939	78.699	68.625	81.847	81.847	79.014	79.014	80.797	84.050	**91.920**
21	Symbols	89.950	79.799	62.714	87.035	89.447	89.548	90.151	79.296	75.678	**93.367**
22	SyntheticControl	88.000	96.000	81.000	92.333	94.333	92.667	94.333	**97.333**	91.000	92.333
23	Trace	76.000	80.000	79.000	73.000	82.000	82.000	78.000	93.000	84.000	**100.000**
24	TwoLeadECG	74.715	69.886	71.817	94.118	90.518	73.310	72.432	97.015	95.083	**99.824**
25	Wafer	**99.546**	70.847	98.199	95.993	99.416	95.766	99.351	99.448	96.398	99.270
	Performing Best	2	1	0	1	3	1	2	4	1	**18**
	Percentage (%)	6.061	3.030	0.000	3.030	9.091	3.030	6.061	12.121	3.030	**54.545**
	Average Rank	5.84	7.52	8.32	5.76	4.00	6.88	4.94	3.52	5.58	**2.64**
	Wilcoxon Test p-value	0.001	0.000	0.000	0.004	0.037	0.000	0.005	0.072	0.002	—

Based on the results, our method performs the best on 18 out of the 25 datasets, with the highest percentage of 54.545% among the approaches. The average rank of our CSSL is 2.64, which is also the best. Besides, we conduct the Wilcoxon test to compare the rankings of our method against the other classifiers. The results of the p-values show that CSSL is significantly better than all the others. CSSL accuracy against each of the other standard classifiers is presented in Fig. 6. Each point represents the pairwise accuracy comparison on a dataset. Points below the line mean that CSSL is more accurate than the others in the pairs. Obviously, CSSL outperforms any of the others in most cases. The result makes sense because the standard classifiers take each time series as a vector which could neglect some discriminatory time-dependent features. That's why we aim to find the features specific to the time-series data, such as the class-specific shapelets in this study, to achieve a more accurate classification.

The accuracy of our CSSL against the state-of-the-art time-series classification methods is shown in Table 5. Our CSSL performs better than all the other state-of-the-art methods in terms of the number of performing the best (9 out of 25, 17.647% among all approaches) and average rank (4.66) apart from HIVE-COTE, which performs the best on 14 out of the 25 datasets, taking 27.451% of all classifiers and is ranked 2.96 on average. It's not surprising because HIVE-COTE is an ensemble of four types of classifiers using different features, and the shapelet-based feature learned by our CSSL is only one of those features. This means that our CSSL can be a component of the ensemble of HIVE-COTE to further improve its performance. The result of the Wilcoxon test between CSSL and the other approaches shows that our CSSL is not significantly better than TSF, ST, LS, and BOSS and is not significantly worse than the best performing HIVE-COTE method at a level of 0.1, as the underlined p-value shows. The pairwise comparison in Fig. 7 gives a similar conclusion. To further evaluate the performance of our CSSL against the 9 state-of-the-art methods, we present their average rankings in a critical difference diagram, as depicted in Fig. 8. Each bold line represents a clique of classifiers ranked at the same statistical level and classifiers in different cliques are significantly different in the average ranking. As can be seen, HIVE-COTE is significantly better than SAXVSM, SAX-



(a) CSSL vs 1NN-ED (b) CSSL vs NB (c) CSSL vs C45

(d) CSSL vs SVML (e) CSSL vs SVMQ (f) CSSL vs BN

(g) CSSL vs RandF (h) CSSL vs RotF (i) CSSL vs MLP

Fig. 6. Pairwise accuracy comparison to the standard classifiers. Region below the diagonal represents the CSSL wins.

Table 5
Accuracy against state-of-the-art time-series classifiers.

#	Dataset	1NN-DTWB	TSF	RISE	SAX-VSM	SAX-VFSEQL	ST	LS	BOSS	HIVE-COTE	CSSL (Ours)
1	Beef	66.667	83.333	83.333	43.333	56.667	90.000	86.667	83.333	**93.333**	66.667
2	BeetleFly	65.000	80.000	75.000	90.000	**95.000**	90.000	80.000	90.000	**95.000**	**95.000**
3	BirdChicken	70.000	80.000	95.000	**100.000**	95.000	80.000	80.000	95.000	85.000	**100.000**
4	CBF	99.444	99.667	95.111	95.667	95.333	97.444	99.111	99.778	**99.889**	97.000
5	ChlorineConcentration	65.000	72.917	**76.849**	65.443	67.786	69.974	59.245	66.276	71.198	64.089
6	Coffee	**100.000**	**100.000**	**100.000**	92.857	92.857	96.429	**100.000**	**100.000**	**100.000**	**100.000**
7	ECG200	88.000	88.000	88.000	85.000	88.000	83.000	88.000	87.000	85.000	**92.000**
8	ECGFiveDays	79.675	97.793	99.884	95.470	99.187	98.374	**100.000**	**100.000**	**100.000**	**100.000**
9	FaceFour	89.773	98.864	89.773	93.182	94.318	85.227	96.591	**100.000**	95.455	95.455
10	FacesUCR	90.780	89.317	87.512	92.537	73.902	90.585	93.902	95.707	**96.293**	88.976
11	GunPoint	91.333	96.000	98.000	98.667	98.000	**100.000**	**100.000**	**100.000**	**100.000**	98.667
12	ItalyPowerDemand	95.530	**96.696**	95.335	81.633	73.761	94.752	96.016	90.865	96.307	93.683
13	Lightning2	**86.885**	80.328	70.492	85.246	75.410	73.770	81.967	85.246	81.967	83.607
14	Lightning7	71.233	73.973	69.863	57.534	68.493	72.603	79.452	68.493	73.973	**80.822**
15	MedicalImages	74.737	73.816	66.184	50.789	57.500	66.974	66.447	71.842	**77.763**	71.184
16	MoteStrain	86.581	86.581	87.220	79.393	89.137	89.696	88.339	82.428	**93.291**	90.974
17	OliveOil	86.667	**90.000**	**90.000**	**90.000**	70.000	**90.000**	16.667	86.667	**90.000**	40.000
18	OSULeaf	59.917	59.917	64.876	85.124	82.645	96.694	77.686	95.868	**97.934**	81.818
19	SonyAIBORobotSurface1	69.551	80.865	82.196	81.364	70.882	84.359	81.032	63.228	76.539	**96.672**
20	SonyAIBORobotSurface2	85.939	83.421	91.081	81.637	89.507	**93.389**	87.513	85.939	92.760	91.920
21	Symbols	93.769	89.246	93.266	62.211	92.161	88.241	93.166	96.683	**97.387**	93.367
22	SyntheticControl	98.333	99.000	66.667	89.000	75.667	98.333	**99.667**	96.667	**99.667**	92.333
23	Trace	99.000	98.000	96.000	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**
24	TwoLeadECG	86.831	80.246	88.762	89.728	98.156	99.737	99.649	98.068	99.649	**99.824**
25	Wafer	99.594	99.530	99.546	99.903	99.562	**100.000**	99.611	99.481	99.935	99.270
	Performing Best	2	3	3	3	2	5	5	5	**14**	9*
	Percentage (%)	3.922	5.882	5.882	5.882	3.922	9.804	9.804	9.804	**27.451**	17.647*
	Average Rank	6.54	5.88	6.44	6.78	6.74	5.04	4.88	5.08	**2.96**	4.66*
	Wilcoxon Test p-value	0.068	0.219	0.024	0.007	0.003	0.511	0.158	0.910	0.117	—

VFSEQL, 1NN-DTWB, RISE, and TSF because they are in different cliques. CSSL, LS, ST, and BOSS are in the overlap of the two cliques. Thus, we can conclude they all achieve state-of-the-art accuracy like HIVE-COTE.

Moreover, we find that though the performance of CSSL is at the same level with LS, our approach learns fewer shapelets, as shown in Table 3. By the fact that both LS and CSSL adopt the linear combination of the distance between the shapelets and the time series to classify the instance, we can draw the conclusion that the class-specific shapelets learned by our CSSL are more discriminatory. It demonstrates that our strategy of discovering representative variable-length subsequences as shapelet candidates to initialize the class-specific shapelet learning model is more effective in learning the *true shapelets*.

Fig. 9 shows the performance of the 10 algorithms against the problem, which gives us an indication of the suitable problems for our method. Each histogram in a cluster is the percentage of times that the algorithm performs the best on that type of problem. The results show that CSSL is the most accurate algorithm for ECG, sensor readings, and motion capture problems among the 25 datasets. It makes sense that CSSL performs the best on ECG and sensor readings problems in terms of the applications, since the profile of the ECG fluctuation and the change in the physical quantity will be a series in arbitrary length, occurring anywhere and different depending on the categories, which corresponds to the definition of the class-specific shapelet. We cannot draw any conclusion at this time on the motion capture problem because it has only one dataset sample.

5.4. Interpretability

Finally, we discuss the interpretability of our CSSL method. CSSL is highly interpretable because the class-specific shapelets represent the distinguishing shape characteristics of one class, such as a fluctuation within a sensor reading reflecting a specific physical process. Thus, the more the shape characteristics of a time series are similar to the class-specific shapelets of a class, the more likely the time series belongs to the class. We show the discriminatory power of our class-specific shapelets by a case study of an ECG problem, which is the type that CSSL can significantly outperform other state-of-the-art methods.

We study the ECGFiveDays dataset. This dataset includes the ECG of a male at two different dates. Fig. 10 represents the training data and the best shapelet of class 1. The instances of each class holding the minimum and maximum distance to the shapelet are depicted. It can be observed that instances from different classes differ slightly in fluctuation, while the shapelet better reflects the fluctuation of class 1. Note that the maximum distance between the class 1 instance and the shapelet (the 18-th instance of class 1, $D = 3.3079$) is smaller than the minimum distance from the class 2 instance to the same shapelet (the 9-th instance of class 2, $D = 3.7742$), which means that the time series from different categories can be fully separated based on the distance to this shapelet. As visualized in Fig. 11, we map the distance between all instances from the training dataset and the best shapelet of class 1 to a number axis. This step transforms the raw time series into a one-dimensional space where instances from different classes distribute in different regions. As can be seen, the instances of the two classes are completely separated in the transformed space, which means the discovered shapelet is discriminatory enough to distinguish the time series of different classes in this dataset.

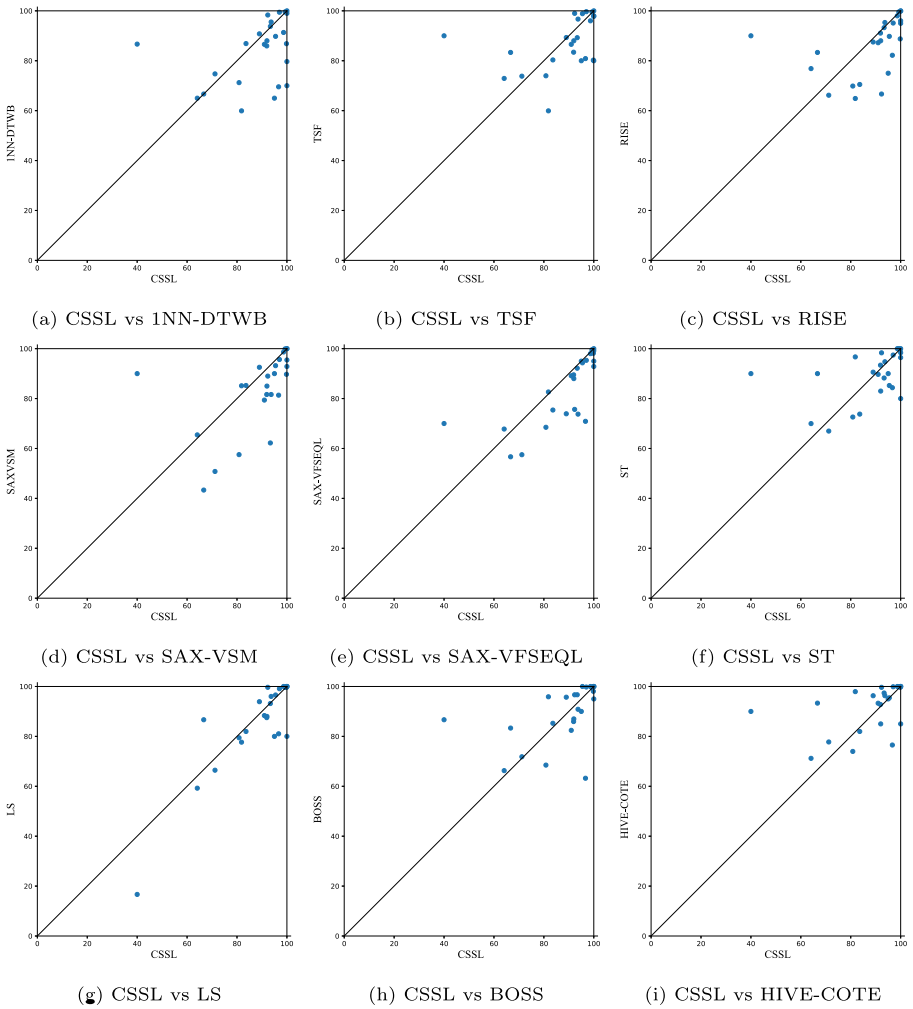


Fig. 7. Pairwise accuracy comparison to the state-of-the-art classifiers. Region below the diagonal represents the CSSL wins.

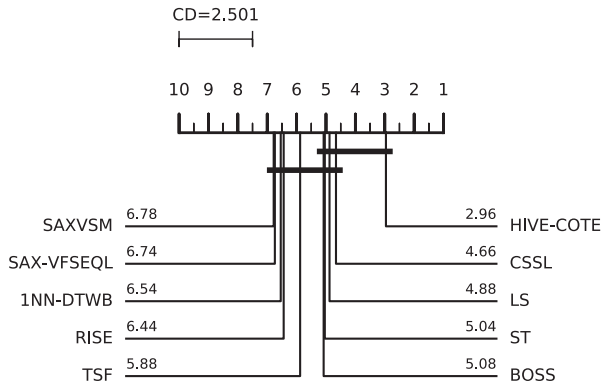


Fig. 8. Critical difference diagram of CSSL and the 9 state-of-the-art classifiers.

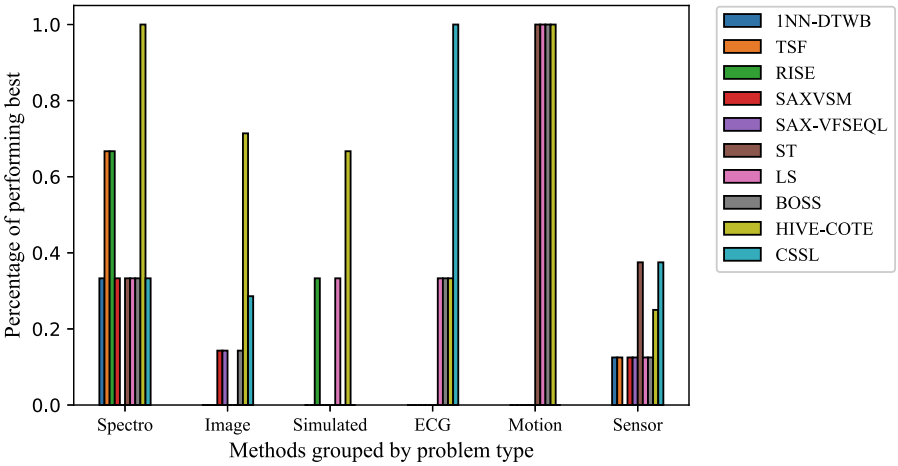


Fig. 9. Best performing methods split by problem type.

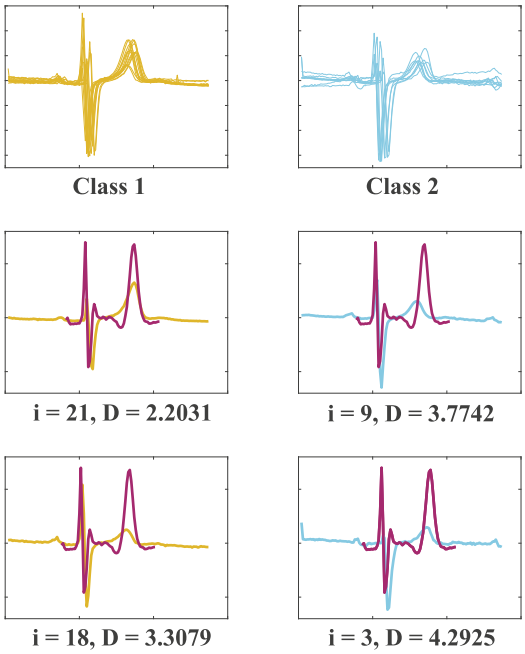


Fig. 10. ECGFiveDays time series and the best shapelet of class 1.

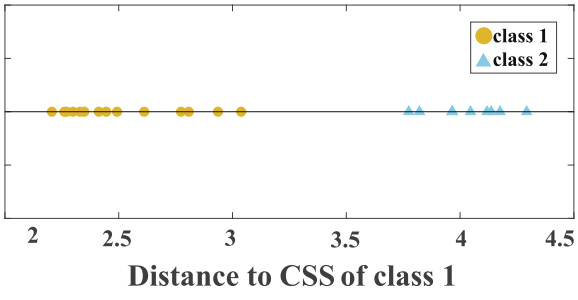


Fig. 11. Distance between each ECGFiveDays instance and the best shapelet of class 1.

In more general cases where one class cannot be simply distinguished against the others by one shapelet, a set of class-specific shapelets in one class are jointly used to transform the time series into a high dimensional space based on the distances from the time series to all these shapelets, where two classes can be maximally separated. The number of shapelets for each class is automatically determined by our algorithm, as discussed in Section 4.2.

Furthermore, we believe that the classification decision of our CSSL is easier to understand compared to other state-of-the-art classifiers. The whole-series-based method is difficult to interpret because it only measures the similarity of the whole time series. The interval-based method generates random intervals to train hundreds of tree classifiers for classification. Interpreting the classification rule is not easy because it needs to analyze all the trees in the forest. Dictionary-based classifiers extract a large amount of time series features in the SAX or SFA space. A classification result must be interpreted by transforming the SAX or SFA features into time series features, which is very complicated. The existing shapelet-based method requires numerous shapelets to guarantee accuracy. These shapelets are shared by all classes. As a result, all the shapelets require consideration to understand a classification decision, which is much more complex and exhaustive. Instead, our CSSL learns a few class-specific shapelets. Thus, when a time series is classified into one class, we can easily interpret this determination by the occurrence of the shapelets-like subsequences of that class. Finally, understanding the hybrid classifier is undoubtedly more difficult because all component classifiers in the ensemble need to be interpreted.

6. Conclusion

In this paper, we aim to achieve a more efficient and interpretable time-series classification while guaranteeing a competitive accuracy by proposing a novel shapelet learning algorithm called CSSL. To reduce the number of calculations of shapelet updating, we first propose to learn class-specific shapelets that can maximally distinguish a class against others. Afterward, we propose a class-specific shapelet candidates discovery method to automatically determine the number, length, and initial values of the shapelets in the learning model to further reduce the number of shapelets to be updated at each iteration and the number of iterations. Our CSSL method has two stages. During the first stage, it discovers as few as possible discriminatory subsequences from the training series of each class as shapelet candidates. A class-specific shapelets learning model that treats each class with specific shapelets is then initialized with the corresponding shapelet candidates to learn the final shapelets as well as the classifier. CSSL processes each category independently, which enables parallelization for further acceleration. We conduct extensive experiments on 25 datasets to evaluate the performance of our CSSL and compare it to 9 well-known standard classifiers and 9 state-of-the-art approaches specially designed for time-series classification. The experimental results show that CSSL produces a breakthrough on efficiency over existing shapelet learning methods, and achieves state-of-the-art performance on the accuracy, especially in ECG and sensor readings problems. Moreover, our method has an advantage in interpretability over existing classifiers.

In the future, we plan to extend our CSSL approach in the following three directions. First, the hyper-parameters tuning of CSSL could be more efficient and intelligent by adopting AutoML techniques. Secondly, although CSSL is designed for univariant time-series data, the basic idea of automatically learning as few as possible variable-length class-specific shapelets can be introduced into the multivariant time series classification problem. Finally, we expect to extend our CSSL algorithm to distributed architecture to deal with the scalability issue in a big data environment.

CRediT authorship contribution statement

Zhiyu Liang: Methodology, Software, Data curation, Validation, Formal analysis, Writing - original draft, Visualization. **Hongzhi Wang:** Conceptualization, Resources, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This paper was supported by NSFC grant U1866602, CCF-Huawei Database System Innovation Research Plan CCF-HuaweiDBIR2020007B.

References

[1] E. Keogh, S. Kasetty, On the need for time series data mining benchmarks: a survey and empirical demonstration, Data Min. Knowl. Disc. 7 (4) (2003) 349–371.
[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Disc. 31 (2017) 606–660.
[3] L. Ye, E. Keogh, Time series shapelets: a novel technique that allows accurate, interpretable and fast classification, Data Min. Knowl. Disc. 22 (1–2) (2011) 149–182.

[4] J. Lines, L.M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 289–297.
[5] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 392–401.
[6] A. Mueen, E. Keogh, N. Young, Logical-shapelets: an expressive primitive for time series classification, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 1154–1162.
[7] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The ucr time series classification archive, URL:www.cs.ucr.edu/eamonn/time_series_data/ (July 2015).
[8] J. Soto, P. Melin, O. Castillo, Time series prediction using ensembles of anfis models with genetic optimization of interval type-2 and type-1 fuzzy integrators, Int. J. Hybrid Intell. Syst. 11 (3) (2014) 211–226.
[9] J. Soto, P. Melin, O. Castillo, A new approach for time series prediction using ensembles of it2fnn models with optimization of fuzzy integrators, Int. J. Fuzzy Syst. 20 (3) (2018) 701–728.
[10] J. Soto, O. Castillo, P. Melin, W. Pedrycz, A new approach to multiple time series prediction using mimo fuzzy aggregation models with modular neural networks, Int. J. Fuzzy Syst. 21 (5) (2019) 1629–1648.
[11] T. Xiong, Y. Bao, Z. Hu, R. Chiong, Forecasting interval time series using a fully complex-valued rbf neural network with dpso and pso algorithms, Inf. Sci. 305 (2015) 77–92.
[12] S.-H. Cheng, S.-M. Chen, W.-S. Jian, Fuzzy time series forecasting based on fuzzy logical relationships and similarity measures, Inf. Sci. 327 (2016) 272–287.
[13] F. Gaxiola, P. Melin, F. Valdez, O. Castillo, Interval type-2 fuzzy weight adjustment for backpropagation neural networks with application in time series prediction, Inf. Sci. 260 (2014) 1–14.
[14] M. Pulido, P. Melin, O. Castillo, Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the mexican stock exchange, Inf. Sci. 280 (2014) 188–204.
[15] A.R.S. Parmezan, V.M. Souza, G.E. Batista, Evaluation of statistical and machine learning models for time series prediction: identifying the state-of-the-art and the best conditions for the use of each model, Inf. Sci. 484 (2019) 302–337.
[16] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, Proc. VLDB Endowment 1 (2) (2008) 1542–1552.
[17] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, E. Keogh, Experimental comparison of representation methods and distance measures for time series data, Data Min. Knowl. Disc. 26 (2) (2013) 275–309.
[18] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD Workshop, vol. 10, Seattle, WA, 1994, pp. 359–370.
[19] D.S. Hirschberg, Algorithms for the longest common subsequence problem, J. ACM (JACM) 24 (4) (1977) 664–675.
[20] L. Chen, R. Ng, On the marriage of lp-norms and edit distance, in: Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30, VLDB Endowment, 2004, pp. 792–803.
[21] P. Schäfer, Scalable time series classification, Data Min. Knowl. Disc. 30 (5) (2016) 1273–1298.
[22] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Searching and mining trillions of time series subsequences under dynamic time warping, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 262–270.
[23] F. Petitjean, G. Forestier, G.I. Webb, A.E. Nicholson, Y. Chen, E. Keogh, Dynamic time warping averaging of time series allows faster and more accurate classification, in: 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 470–479.
[24] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inf. Sci. 239 (2013) 142–153.
[25] M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2796–2802.
[26] M.G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Min. Knowl. Disc. 30 (2) (2016) 476–509.
[27] J. Lines, S. Taylor, A. Bagnall, Time series classification with hive-cote: the hierarchical vote collective of transformation-based ensembles, ACM Transa. Knowl. Discovery Data 12 (5).
[28] C.C. Aggarwal, C. Zhai, A survey of text classification algorithms, in: Mining Text Data, Springer, 2012, pp. 163–222.
[29] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of-patterns representation, J. Intell. Inf. Syst. 39 (2) (2012) 287–315.
[30] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing sax: a novel symbolic representation of time series, Data Min. Knowl. Disc. 15 (2) (2007) 107–144.
[31] P. Senin, S. Malinchik, Sax-vsm: Interpretable time series classification using sax and vector space model, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1175–1180.
[32] P. Schäfer, The boss is concerned with time series classification in the presence of noise, Data Min. Knowl. Disc. 29 (6) (2015) 1505–1530.
[33] P. Schäfer, U. Leser, Fast and accurate time series classification with weasel, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 637–646.
[34] T. Le Nguyen, S. Gsponer, G. Ifrim, Time series classification by sequence learning in all-subsequence space, in: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), IEEE, 2017, pp. 947–958.
[35] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 947–956.
[36] T. Rakthanmanon, E. Keogh, Fast shapelets: a scalable algorithm for discovering time series shapelets, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 668–676.
[37] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Min. Knowl. Disc. 28 (4) (2014) 851–881.
[38] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Disc. 33 (4) (2019) 917–963.
[39] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with cote: the collective of transformation-based ensembles, IEEE Trans. Knowl. Data Eng. 27 (9) (2015) 2522–2535.
[40] A. Bagnall, M. Flynn, J. Large, J. Lines, M. Middlehurst, A tale of two toolkits, report the third: on the usage and performance of hive-cote v1. 0, arXiv e-prints (2020) arXiv–2004.
[41] S. Torkamani, V. Lohweg, Survey on time series motif discovery, Wiley Interdisc. Rev. Data Min. Knowl. Disc. 7 (2) (2017) e1199.
[42] Y. Li, J. Lin, Approximate variable-length time series motif discovery using grammar inference, in: Proceedings of the Tenth International Workshop on Multimedia Data Mining, ACM, 2010, p. 10.
[43] C.G. Nevill-Manning, I.H. Witten, Identifying hierarchical structure in sequences: a linear-time algorithm, J. Artif. Intell. Res. 7 (1997) 67–82.
[44] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[45] L. Dagum, R. Menon, Openmp: an industry-standard api for shared-memory programming, Comput. Sci. Eng. (1) (1998) 46–55.