Neurocomputing 275 (2018) 2317–2328

Contents lists available at ScienceDirect

Neurocomputing

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

Discriminative extraction of features from time series

Zhenguo Zhang[a,b], Haiwei Zhang[a,*], Yanlong Wen[a], Ying Zhang[a], Xiaojie Yuan[a]

[a] College of Computer and Control Engineering, Nankai University, 38 Tongyan Road, Tianjin 300350, PR China
[b] Department of Computer Science and Technology, Yanbian University, 977 Gongyuan Road, Yanji 133002, PR China

ARTICLE INFO

Article history:
Received 9 May 2017
Revised 27 October 2017
Accepted 4 November 2017
Available online 10 November 2017

Communicated by Prof. Zidong Wang

Keywords:
Time series
LFDA
Sparse restriction
Shapelets

ABSTRACT

A primary challenge of time series classification is how to extract powerful features from training samples. Two kinds of classification methods, global-based and local-based methods, have been studied widely in recent years. The global-based methods, like 1-Nearest Neighbor(1-NN), take the entire series as features, which have the drawback that they are not able to indicate the intrinsic characters of a class. The local-based methods overcome this weakness by employing discriminative time series subsequences as features, called *shapelets*. However, most local-based methods are computationally expensive because of the massive number of shapelet candidates. In this paper, we propose a novel shapelets extraction method which takes each time series as a high-dimensional data and then finds the discriminative dimensions corresponding to the positions of shapelets. More specifically, the discriminative dimensions are determined by combining Local Fisher Discriminant Analysis (LFDA) method and two sparse restrictions which can encourage the continuous characteristic of time series. Extensive experimental results show that the proposed method achieves significant improvement compared to the existing shapelet-based methods in terms of classification accuracy and running time on the commonly used time series datasets. In addition, comparing with the accepted time series classification methods, NNDTW and COTE, our method still gets better results.

1. Introduction

Time series classification has attracted significant interest in data mining community because time series are present in a wide range of real-life domains such as finance [1], motion capture [2,3], medicine [4,5] and social systems [6], etc. A primary challenge for time series classification is to select discriminative features that can distinguish different classes. Unlike traditional classification problems which do not consider the correlation of variables, the order of the variables is a crucial factor for time series data. Thus many studies take the entire time series as one feature and focus on alternative similarity measures for 1-NN classifiers [7,8]. Although the experimental evidence suggests that 1-NN with Dynamic Time Warping (DTW) has high classification accuracy and is very difficult to beat [9], it has several disadvantages. First of all, it requires storing and searching the entire training set, which results in high time and space complexity when classifying large datasets. More importantly, it does not give a clear insight to reveal the most important patterns of different classes. To overcome these problems, a local-based feature, called *shapelet*, is proposed and has attracted lots of attention in time series domain in recent years [10–15].

Informally, shapelets are discriminative subsequences of time series with high prediction accuracy. More importantly, shapelets also offer good interpretability to domain experts for the classification results [11]. There are two commonly used methods to extract adequate shapelets. One is to discover shapelets by measuring a pool of candidates form all permissible time series subsequences, which can be categorized as *search-based* methods [10–12]. Due to the massive number of shapelet candidates, the process is time-consuming although several speed-up techniques, like intelligent caching, reuse of computations [10] and random projecting technique on a symbolic representation of time series [12], are used to reduce the discovery time. When classifying unknown time series, a decision tree constructed by the obtained shapelets is used to predict their labels. The other shapelets discovery method called Learning Time Series (LTS) is to learn shapelets jointly with the classifier by optimizing an objective function [14]. Instead of searching for shapelets from a candidate pool, LTS aims to learn shapelets from time series. Thus the obtained shapelets may differ from the time series subsequences. This technique can get the

* Corresponding author.
E-mail addresses: zhangzhenguo@dbis.nankai.edu.cn (Z. Zhang), zhanghaiwei@dbis.nankai.edu.cn (H. Zhang), wenyanlong@dbis.nankai.edu.cn (Y. Wen), zhangying@dbis.nankai.edu.cn (Y. Zhang), yuanxiaojie@dbis.nankai.edu.cn (X. Yuan).

https://doi.org/10.1016/j.neucom.2017.11.002
0925-2312/

near-to-optimal shapelets for classification, but it is space consuming and has high computational complexity.

Another study about shapelets is how to make full use of the characteristics of shapelets to improve prediction accuracy. While decision tree is highly interpretable, it is often outperformed by other classifiers and tends to overfit. What's more, the process of tree building is time-consuming. Instead of using shapelets to build a decision tree, Hill et al. [13] propose to transform the shapelets into a vector in which each item is the distance between the original time series and the discovered shapelets. This kind of shapelets-transformed vectors can be treated as features of time series and then we can utilize the off-the-shelf classifier, like SVM, to predict the label of test series. It has been proved that this technique can improve classification accuracy while still maintaining the explanatory ability of shapelets [13] [15].

No matter using the search-based or the learning-based shapelets extraction methods, the shapelets discovery is both time and space consuming even utilizing some speed-up techniques. In this paper, we consider each time series as a point of high-dimensional space and obtain the discriminative features (shapelets) by discarding the dimensions without class separability by a dimension reduction method. This is a novel perspective compared to other methods. To ensure that we get the shapelets, LFDA that fully considers the local structure of data is employed for finding the projection vector which can separate the time series in different classes well [16]. On the other hand, the sparse technique which is widely studied in machine learning and pattern recognition community [17] and [18] is used to make the obtained projection vector sparse. Meanwhile, to preserve the continuous characteristic of time series, the selected features are expected to be time-ordered and we utilize a blocky regularizer [19] to solve this problem. The nonzero parts of the projection vector are the most discriminative dimensions and the corresponding subsequences of time series are considered as shapelet candidates. To get the shapelets, we employ information gain ratio to prune the candidates with low discriminative power. The experimental results on 28 datasets show that our method not only can achieve significant improvement in accuracy but also is orders of magnitudes faster than most time series classification methods.

Our contributions can be summarized as follows:

- We propose a novel perspective to extract the time series shapelets efficiently by treating a time series as a point of high-dimensional space and reducing the undesired dimensions (locations).
- We use two sparse restrictions to ensure that we can find the most discriminative locations and preserve the continuous characteristic of time series.
- The information gain ratio is employed to measure the discriminative power of the extracted shapelet candidates and the candidates with high scores constitute the set of shapelets.
- To demonstrate the efficiency and effectiveness of the proposed method, we conduct extensive experiments on commonly used time series datasets and the results show that our method achieves significant improvement.

The rest of this paper is organized as follows. In the next section, we discuss the related works about shapelets extraction. Section 3 gives the preliminaries used in this paper. In Section 4, we describe the proposed method in detail, including the shapelet candidates generation and pruning. The experimental setup and results are presented in Section 5. Finally, we conclude the paper in Section 6.

2. Related work

In recent years, there are several approaches to extract the local features of time series based on the datasets' characteristics. For example, in EEG signal analysis, the approximately repeated subsequences (motifs) are taken as the local features for classification tasks [20] and [21]. In electricity consumption research, the similar subsequences are employed to do the forecasting and clustering work [22–25]. Although these methods are useful for some specific domains, they cannot be applied to other time series datasets. Note that the purpose of our work is precisely to extract the discriminative features from any time series datasets. As mentioned in Section 1, shapelets are an excellent choice and we mainly review the shapelets extraction methods in this section. There are two types of shapelets extraction methods in recent studies: search-based and learning-based shapelets extraction. We will discuss them in detail.

2.1. Search-based shapelets extraction

The first work using shapelets to classify time series is proposed by Ye and Keogh [11]. In their study, all subsequences of training time series with permissible length are considered as potential candidates and shapelets are selected by the measure of information gain. Due to the massive number of shapelet candidates, the brute-force shapelets discovery is time-consuming ($O(n^2m^4)$ for n length-m time series). Therefore, speed-up techniques are necessary for discovering effective shapelets. In Ye's work [11], they utilize the tricks of early abandoning of distance computations and entropy pruning of the information gain metric. However, it is still not feasible for large dataset. Mueen et al. [10] accelerate the process by reusing the sufficient statistics to reduce the repetitive computation of distance. It is a tradeoff between time-consuming and space-consuming. The worst-case running time is $O(n^2m^3)$ and it still requires memory space as large as $O(nm^2)$. Another speed-up work is to transform the raw time series into a discrete and low-dimensional representation by using Symbolic Aggregate approXimation (SAX) [26] and utilize the random projecting method to reduce the most ineffective candidates [12]. The time complexity can get down to $O(nm^2)$. Besides, Chang et al. address this issue on highly parallel Graphics Process Units (GPUs) [27]. Instead of using the information gain as the quality metrics for shapelet candidates, other metrics such as F-Stats, Kruskal–Wallis and Mood's median have been proposed for evaluating the prediction accuracy [13]. Although these metrics are easier to compute than information gain, the speed-up is minor and the prediction accuracy has no considerable difference.

All above search-based shapelets methods require the enumeration of all subsequences of training data. It is infeasible for large datasets and long time series. One common trick is to only consider the subsequences of specific lengths, e.g., within predefined ranges [10–12]. However, the optimal shapelet length is unknown and the predefined ranges are selected empirically. To get rid of this constraints, Hills et al. [13] introduce a heuristic-based algorithm for estimating the shapelets length.

Another idea about search-based shapelets extraction [28] is to use a number of random shapelets instead of measuring each shapelet candidate. The algorithm is based on the assumption that the discriminative shapelets should frequently appear in time series, so random sampling has a high opportunity to discover qualified shapelets. However, a large number of shapelets are required to ensure high accuracy, which slows the prediction. There is a drawback that the selected shapelets are hard to interpret due to the random sampling technique.

2.2. Learning-based shapelets extraction

Unlike the search-based shapelets methods, Grabocka et al. [14] present a new shapelets discovery method called Learning Time Series (LTS), where the shapelets are learned directly from the training time series instead of generating and verifying lots of candidates. In their work, they simultaneously minimize the training error using a logistic loss function and the distance using a soft minimum. The learned shapelets may differ from the time series subsequences, but they can gain the near-to-optimal features and get better results. It can be considered as the current state-of-the-art shapelets method regarding classification accuracy. This technique has also been used for time series cluster problem [29]. However, the objective function to be optimized is nonlinear, which makes the computational complexity high and more importantly, it is memory-consuming which is not suitable for large datasets.

3. Preliminaries

The problem studied in this paper is how to extract the effective shapelets from time series and then use these shapelets to predict the class labels of unseen time series accurately. For convenience to state our method, we first give the background definitions and notations.

Definition 1 (Time Series Dataset)**.** A set of time series is denoted as $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ with n time series of C different classes. Each time series $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{im})$ contains m ordered observations with a label y_i.

The variables of $\boldsymbol{x}$ are typically arranged by temporal order. Before we start our work, it is necessary to normalize the time series, which is a standard step of most data mining tasks. We use the z-normalization method to complete this process and the formulation is

$$norm(\boldsymbol{x}) = \frac{\boldsymbol{x} - mean(\boldsymbol{x})}{std(\boldsymbol{x})} \tag{1}$$

For convenience, we still use $\boldsymbol{x}$ to denote $norm(\boldsymbol{x})$ throughout this paper.

Generally, we are more interested in the local properties of a time series rather than the global properties. A local segment of time series is termed as a subsequence.

Definition 2 (Time Series Subsequence)**.** Given a time series $\boldsymbol{x}$ of length m, a subsequence ss is a sequence of l contiguous segments from $\boldsymbol{x}$, denoted as $\boldsymbol{ss} = x_p, ..., x_{p+l-1}$, for $1 \le p \le m - l + 1$.

The obtained subsequences can be considered as shapelet candidates and we use a set to represent them, $SC = \{\boldsymbol{sc}_1, \boldsymbol{sc}_2, ...\}$. All shapelets are selected from SC by a metric.

Definition 3 (Shapelet)**.** The shapelets, denoted as $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_N\}$, are the most discriminative subsequences of SC, which can distinguish the time series of different classes.

To evaluate the discriminative power of a shapelet candidate, we will employ the concept of information gain ratio. The related definitions are given next.

Definition 4 (Entropy)**.** Suppose a time series dataset X consists of two classes, C_1 and C_2 with n_{C1} and n_{C2} samples, respectively. Let $p(C_1) = n_{C1}/n$, $p(C_2) = n_{C2}/n$, thus the entropy of X is:

$$E(X) = -p(C_1)log(p(C_1)) - p(C_2)log(P(C_2)) \tag{2}$$

If a shapelet candidate $\boldsymbol{sc}$ divides X into two subsets, X_1 and X_2 with the number n_1 and n_2 by a given partition sp, the entropy after the dividing becomes

$$\hat{E}(X) = n_1/n \cdot E(X_1) + n_2/n \cdot E(X_2) \tag{3}$$

Definition 5 (Information Gain)**.** Given a shapelet candidate $\boldsymbol{sc}$ and a partition sp, the information gain is the difference between $E(X)$ and $\hat{E}(X)$:

$$IG(\boldsymbol{sc}, sp) = E(X) - \hat{E}(X) \tag{4}$$

For unbalance dataset, the information gain is incapable of different shapelet candidates because the value of IG is similar. To solve this problem, we normalize IG:

Definition 6 (Information Gain Ratio)**.** The formulation of information gain ratio is defined as

$$IG\_ratio(\boldsymbol{sc}, sp) = 1 - \hat{E}(X)/E(X) \tag{5}$$

If $\boldsymbol{sc}$ is a good shapelet, IG_ratio is close to 1 and vise versa.

4. Extraction of time series shapelets by dimension selection

We first give the whole process of our method to extract the shapelets and build a classifier, which consists of three distinct stages. First and foremost, we take each training time series as a high-dimensional data and employ the techniques of LFDA and sparse restrictions to find the most discriminative dimensions, i.e., locations of time series variables. Secondly, the subsequences are extracted based on the selected dimensions (nonzero locations) to form shapelet candidates. To speed up the process of the following classification, we use the information gain ratio to estimate the discriminative power of each candidate and the candidates with high scores constitute the set of shapelets. Finally, the training time series are transformed to feature vectors by calculating the distance between each time series and shapelets. These transformed vectors are then applied to train a linear SVM classifier. There are two reasons for us to choose linear SVM classifier: one is it is a mature classifier in machine learning and the other is the recent shaplets studies also take SVM as the classifier to evaluate the extracted shapelets, like IGSVM [13] and FLAG [15]. This process is described in Fig. 1. For new time series, we transform them to feature vectors as the training time series and predict their labels through the trained classifier.

4.1. Discriminative dimensions selection

The first stage of our method needs a supervised dimension reduction method to estimate a lower dimensional embedding space into which time series $\boldsymbol{x}_i$ may be transformed, although we do not really project the time series into this space. Recall that we aim to extract the discriminative subsequences from training time series. To be specific, we want to find out the dimensions (locations) where the subsequences are similar for one class and have significant differences for other classes. So the method which preserves the locality relations of the data is important for our purpose. In this paper, we employ LFDA to complete this task [16]. LFDA combines the separability-enhancing power of Linear Discriminant Analysis (LDA) [30,31] with Locality-Preserving Projections (LPP) [32], which can handle multi-modal non-Gaussian class distributions while preserving the local neighborhood structure of the class distributions. It has been widely used in many applications, like pedestrian re-identification [33]. We first consider the simple scenario of binary classification and then extend to the multi-class problem in Section 4.4.

4.1.1. Finding the projection vector by LFDA

When $C = 2$, we can embed each time series into a one-dimensional space by a projection vector. Let X: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\}$ be the training set of n samples, where $y_i \in \{\pm 1\}$. If the projection vector $\boldsymbol{v}$ is defined, we have:

$$z_i = \boldsymbol{v}'\boldsymbol{x}_i \tag{6}$$

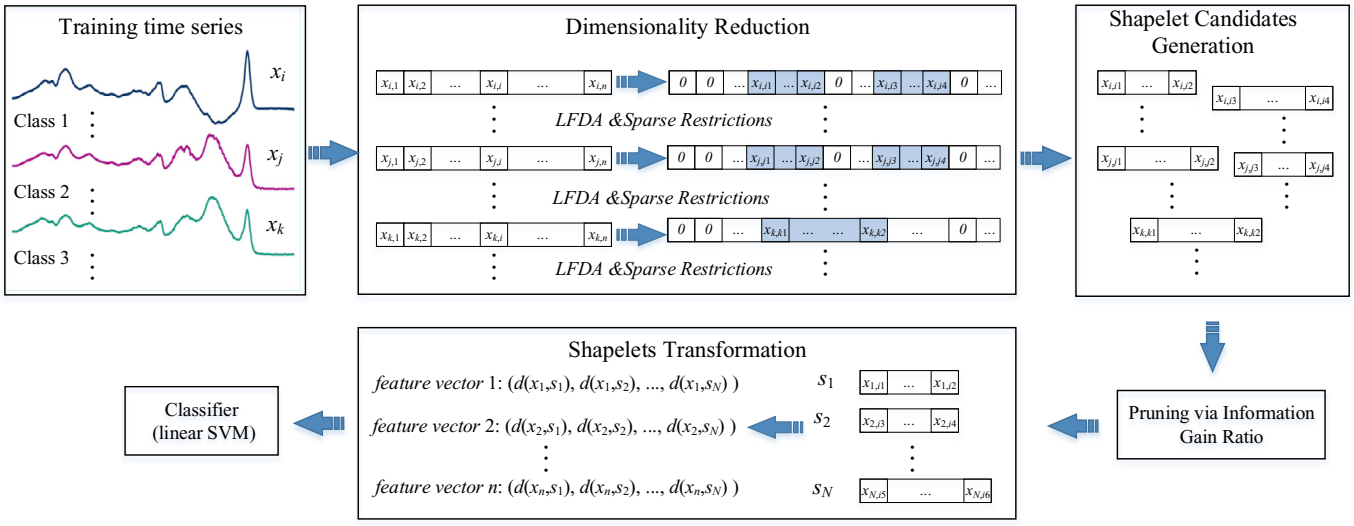


Fig. 1. Overview of our method. In this example, the dataset contains 3 classes (left-top). With the help of LFDA and sparse restrictions, the undiscriminating parts of time series are set to zeros (middle-top). The non-zero subsequences are extracted to form shapelet candidates (right-top), and then we employ information gain ratio to evaluate these candidates' power. After getting shapelets, each time series is transformed into a feature vector (middle-bottom) and the mature linear SVM classifier is used to get the results.

where z_i is the new representation of $\boldsymbol{x}_i$ in one-dimensional space. The purpose of dimension reduction method is to find the appropriate $\boldsymbol{v}$ which makes $z_i (i \in [1, n])$ with the different label more separate. LFDA evaluates the levels of within-class scatter and between-class scatter in a local manner to find the proper $\boldsymbol{v}$. The local within-class scatter matrix and the local between-class scatter matrix are defined as:

$$S_w = \frac{1}{2}\sum_{i,j=1}^{n} A_{ij}^{w}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)^T \tag{7}$$

$$S_b = \frac{1}{2}\sum_{i,j=1}^{n} A_{ij}^{b}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)^T \tag{8}$$

where A_{ij}^{w} and A_{ij}^{b} are $n \times n$ matrices defined as

$$A_{ij}^{w} = \begin{cases} A_{ij}/n_c\,, & if\ y_i = y_j = c \\ 0\,, & y_i \neq y_j \end{cases} \tag{9}$$

$$A_{ij}^{b} = \begin{cases} A_{ij}(1/n - 1/n_c)\,, & if\ y_i = y_j = c \\ 1/n\,, & if\ y_i \neq y_j \end{cases} \tag{10}$$

where n_c is the number of training time series for cth class and $\sum_{c=1}^{C} n_c = n$. Using S_w and S_b, the objective function $J(\boldsymbol{v})$ can be represented as

$$J(\boldsymbol{v}) = \frac{\boldsymbol{v}^T S_b \boldsymbol{v}}{\boldsymbol{v}^T S_w \boldsymbol{v}} \tag{11}$$

The projection vector can be obtained by maximizing $J(\boldsymbol{v})$, i.e.,

$$\boldsymbol{v} = \arg\max_{\boldsymbol{v}} J(\boldsymbol{v}) = \arg\max_{\boldsymbol{v}} \frac{\boldsymbol{v}^T S_b \boldsymbol{v}}{\boldsymbol{v}^T S_w \boldsymbol{v}} = \arg\min_{\boldsymbol{v}} \frac{\boldsymbol{v}^T S_w \boldsymbol{v}}{\boldsymbol{v}^T S_b \boldsymbol{v}} \tag{12}$$

In formula (9) and (10), A_{ij} is the affinity matrix which determines how to preserve the local structure. LFDA can easily deal with the multi-modal scenario existing in training data by using an appropriate A_{ij}.

Generally, the elements of A are in [0,1] and a large value indicates the corresponding data are closer. A simple definition is $A_{i,j} = 1$ if $\boldsymbol{x}_j$ is the K-nearest neighbor of $\boldsymbol{x}_i$; otherwise $A_{i,j} = 0$. However, this definition does not distinguish the distance of different K-nearest neighbor points. To overcome the drawback, Ng et al. [34] use a scale parameter σ and define $A_{i,j} = \exp\left(-\frac{d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\sigma^2}\right)$ for $i \neq j$ and $A_{i,i}=0$, where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a distance like Euclidean distance. This is a useful definition, but it still has some weakness such as σ is hard to choose. Instead of selecting a single scaling parameter, Zelnik-Manor et al. [35] suggest utilizing different local scaling parameter σ_i for data $\boldsymbol{x}_i$. Thus A is defined as

$$A_{i,j} = \exp\left(-\frac{d^2\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)}{\sigma_i \sigma_j}\right) \tag{13}$$

where $\sigma_i = d(\boldsymbol{x}_i, \boldsymbol{x}_K)$ and $\boldsymbol{x}_K$ is the Kth neighbor of $\boldsymbol{x}_i$.

By this definition of A, the values of local scatter matrix used in LFDA are mainly determined by the points that are close to each other, because the points with large distance have small contributions and near to 0 in A

4.1.2. Regularization

It must be concerned that S_b may not be full rank when we have finite training time series. In particular, if there are fewer than m examples in the training set, then S_b is guaranteed to be rank deficient. When such a matrix appears in the denominator of (12), the projection vector $\boldsymbol{v}$ may be unstable and overly sensitive. This drawback can be solved by adding a multiple of the identity to regularize S_b [36]. A typical way is to set a multiple of the average eigenvalue of S_b [37]. That is $S_b + \frac{\gamma}{m} trace(S_b) I$, thus

$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \frac{\boldsymbol{v}^T S_w \boldsymbol{v}}{\boldsymbol{v}^T (S_b + \frac{\gamma}{m} trace(S_b) I) \boldsymbol{v}} \tag{14}$$

where γ is a regularization parameter, m is the length of time series and the function *trace* is to calculate the trace of S_b. The equation (14) is equivalent to (12) with an additional upper-bound constraint on the norm of $\boldsymbol{v}$. For convenience, we use $\bar{S}_b$ to denote the regularized denominator in the following sections, i.e., $\bar{S}_b = S_b + \frac{\gamma}{m} trace(S_b) I$.

4.1.3. Sparse restriction

Recall that our purpose is to find the discriminative dimensions for shapelets extraction, so the sparsity and local continuity of projection vector calculated by Eq. (14) is crucial for us. Theoretically, l_1-norm is used to encourage sparsity and has been widely used

in fields like image classification [38] and signal processing [39]. In our method, we use l_1-norm as a constraint to ensure that the projection vector $\boldsymbol{v}$ has only several nonzero variables.

On the other hand, we want the non-zero locations of $\boldsymbol{v}$ is successive. Thus the relevant subsequences can be selected from training time series to form shapelet candidates. This is very important for time series because its variables are recorded in order. A universal solution for keeping the local continuity is to regularize the continuous variables of $\boldsymbol{v}$ [19]. So the sparse restriction technique can be described as

$$r(\boldsymbol{v}) = \alpha \sum_{i=2}^{m} \|v_i - v_{i-1}\|_1 + \beta \|\boldsymbol{v}\|_1 \tag{15}$$

where α, β are parameters. The first item of this equation can be rewritten in a more compactly way, i.e., $\alpha\|D\boldsymbol{v}\|_1$, where D is a matrix with $D_{i,i} = 1$, $D_{i,i-1} = -1$ and $D_{i,j} = 0 (i \in [2, m], i \neq j)$.

4.1.4. Objective function

By adding a sparse restriction on the regularized LFDA formula, the objective function becomes:

$$f(\boldsymbol{v}) = \frac{\boldsymbol{v}^T S_w \boldsymbol{v}}{\boldsymbol{v}^T \bar{S}_b \boldsymbol{v}} + r(\boldsymbol{v}) \tag{16}$$

When $f(\boldsymbol{v})$ is minimized, the corresponding $\boldsymbol{v}$ is our projection vector. Up to a scaling on $\boldsymbol{v}$, we can let $\boldsymbol{v}^T \bar{S}_b \boldsymbol{v} = 1$ and Eq. (16) can be rewritten as

$$f(\boldsymbol{v}) = \boldsymbol{v}^T S_w \boldsymbol{v} + r(\boldsymbol{v}) \quad s.t. \quad \boldsymbol{v}^T \bar{S}_b \boldsymbol{v} = 1 \tag{17}$$

Due to the restriction of $r(\boldsymbol{v})$, it is hard to get the minimum of Eq. (17) directly. We transform $r(\boldsymbol{v})$ to $r(\boldsymbol{u}, \boldsymbol{w})$ by setting $u = D\boldsymbol{v}$ and $w = \boldsymbol{v}$, i.e.,

$$r(\boldsymbol{u}, \boldsymbol{w}) = \alpha \|\boldsymbol{u}\|_1 + \beta \|\boldsymbol{w}\|_1 \tag{18}$$

Thus, we have three variables (i.e., $\boldsymbol{v}$, $\boldsymbol{u}$, $\boldsymbol{w}$) and three constraints for the objective function. This optimization problem can be efficiently solved using alternating direction method of multipliers (ADMM) [40]. It is an iterative process and we minimize the objective function based on the current solution in each iteration.

Based on the above transformation, the augmented Lagrangian of Eq. (17) is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = {} & \boldsymbol{v}^T S_w \boldsymbol{v} + r(\boldsymbol{u}, \boldsymbol{w}) + \lambda(\boldsymbol{v}^T \bar{S}_b \boldsymbol{v} - 1) \\ & + \boldsymbol{\mu}_1 (\boldsymbol{u} - D\boldsymbol{v}) + \boldsymbol{\mu}_2 (\boldsymbol{w} - \boldsymbol{v}) \\ & + \frac{\rho_1}{2} \| \boldsymbol{u} - D\boldsymbol{v} \|^2 + \frac{\rho_2}{2} \| \boldsymbol{w} - \boldsymbol{v} \|^2 \end{aligned} \tag{19}$$

where $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ are Lagrange multipliers and ρ_1, ρ_2 are penalty parameters. Suppose $\boldsymbol{v}^t$, $\boldsymbol{u}^t$, $\boldsymbol{w}^t$, ${\boldsymbol{\mu}_1}^t$, ${\boldsymbol{\mu}_2}^t$ are the solutions of the tth iteration, we calculate the value of these variables in $t+1$ iteration.

For $\boldsymbol{v}^{t+1}$,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = {} & (S_w + S_w^T)\boldsymbol{v} + \rho_1 \left(D^T D\boldsymbol{v} - D^T \left(\boldsymbol{u}^t - \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} \right) \right) \\ & + \rho_2 \left(\boldsymbol{v} - \left(\boldsymbol{w} - \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} \right) \right) - 2\lambda \bar{S}_b \boldsymbol{v} = 0 \end{aligned} \tag{20}$$

we can obtain

$$\begin{aligned} \boldsymbol{v}^{t+1} = {} & \frac{1}{2} L^{-T} \left(L^{-1} \left(\bar{S_w} + \frac{1}{2}\rho_1 D^T D + \frac{1}{2}\rho_2 I \right) L^{-T} - \lambda I \right)^{-1} L^{-1} \\ & \left(\rho_1 D^T \left(\boldsymbol{u}^t - \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} \right) + \rho_2 \left(\boldsymbol{w} - \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} \right) \right) \end{aligned} \tag{21}$$

where $\bar{S}_b = LL^T$ is the Cholesky decomposition and λ can be obtained by $\partial\mathcal{L}/\partial\lambda$.

For other variables, we also calculate the partial derivatives and set them to zero to get $\boldsymbol{u}^{t+1}$, $\boldsymbol{w}^{t+1}$, ${\boldsymbol{\mu}_1}^{t+1}$, ${\boldsymbol{\mu}_2}^{t+1}$ after the $\boldsymbol{v}^t$ is updated to $\boldsymbol{v}^{t+1}$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} = \frac{\partial}{\boldsymbol{u}} \left(\frac{\rho_1}{2} \| \boldsymbol{u} - D\boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} \|^2 + \alpha \| \boldsymbol{u} \|_1 \right) = 0 \tag{22}$$

This optimization problem can be efficiently solved by soft-thresholding algorithm [41,42] and the solution is

$$\boldsymbol{u}^{t+1} = \begin{cases} D\boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} + \frac{\alpha}{\rho_1}, & D\boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} < -\frac{\alpha}{\rho_1} \\ 0, & | D\boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} | < \frac{\alpha}{\rho_1} \\ D\boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} - \frac{\alpha}{\rho_1}, & D\boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_1}^t}{\rho_1} > \frac{\alpha}{\rho_1} \end{cases} \tag{23}$$

Similarly,

$$\boldsymbol{w}^{t+1} = \begin{cases} \boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} + \frac{\beta}{\rho_2}, & \boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} < -\frac{\beta}{\rho_2} \\ 0, & | \boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} | < \frac{\beta}{\rho_2} \\ \boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} - \frac{\beta}{\rho_2}, & \boldsymbol{v}^{t+1} + \frac{{\boldsymbol{\mu}_2}^t}{\rho_2} > \frac{\beta}{\rho_2} \end{cases} \tag{24}$$

$\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are updated by gradient descent method, i.e.,

$${\boldsymbol{\mu}_1}^{t+1} = {\boldsymbol{\mu}_1}^t - \eta_1 \nabla \mathcal{L}({\boldsymbol{\mu}_1}^t) \tag{25}$$

$${\boldsymbol{\mu}_2}^{t+1} = {\boldsymbol{\mu}_2}^t - \eta_2 \nabla \mathcal{L}({\boldsymbol{\mu}_2}^t) \tag{26}$$

where η_1, η_2 are preset step size parameters.

The optimization process can be described by algorithm 1

Algorithm 1: getVector: Getting the projection vector through iterative optimization.

Input: time series training set: X, parameters: α, β
Output: projection vector

```
random initialize v, Lagrange multipliers μ1 and μ2;
calculate the local within-class scatter matrix Sw and the
local between-class scatter matrix Sb based on Eqs.(7) and
(8);
regularize Sb to S̄b;
old_v = v;
while ||old_v − v|| > ε do
    update v, u, w, μ1, μ2 based on Eqs. (21–26);
end
return v;
```

4.2. Shapelet candidates generation and pruning

For each class, we get a projection vector which contains a lot of zero items due to the l_1-norm constraint. The non-zero items are the discriminative locations of this class. Generally, $\boldsymbol{v}$ looks like: $\boldsymbol{v} = [0, ..., 0, v_i, ...v_j, 0, ..., 0, v_k, ..., v_l, ...]$. The shapelet candidates are generated according to $\boldsymbol{v}$'s non-zero locations from all time series of this class. For two classes, one $\boldsymbol{v}$ is adequate for extracting candidates. Fig. 2 gives an example on *Gun-Point* dataset from UCR archives[1].

The subsequences corresponding to these non-zero locations are shapelet candidates. As we can see, the number is very large and not all candidates have high power for classification. We hope to discard the less powerful candidates before classifying new time series. We utilize information gain ratio to accomplish this process. When we get a candidate $\boldsymbol{sc}$ from a class, if it is a useful feature, it must distinguish this class and the other class by a distance comparison and have a high *IG_ratio* value. Suppose the distance between $\boldsymbol{sc}$ and all training time series is like in Fig. 3, we need to

[1] http://www.cs.ucr.edu/~eamonn/time_series_data/.

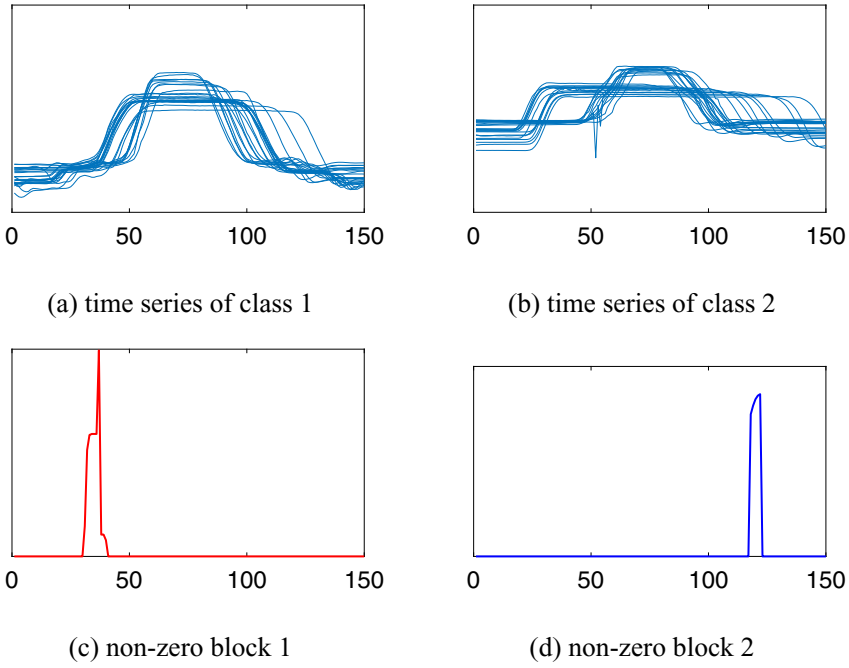


Fig. 2. An example of shapelet candidates generation from the *Gun-Point* dataset from the UCR archives. (a) and (b) are the training time series of each class. (c) and (d) are the non-zero locations of projection vector.

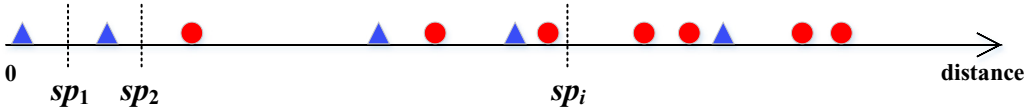


Fig. 3. The positions of time series represent their distances to $\boldsymbol{sc}$ and all distances are arranged in one-dimensional representation. The purpose is to find the best split point sp_i, which has the largest information gain ratio value.

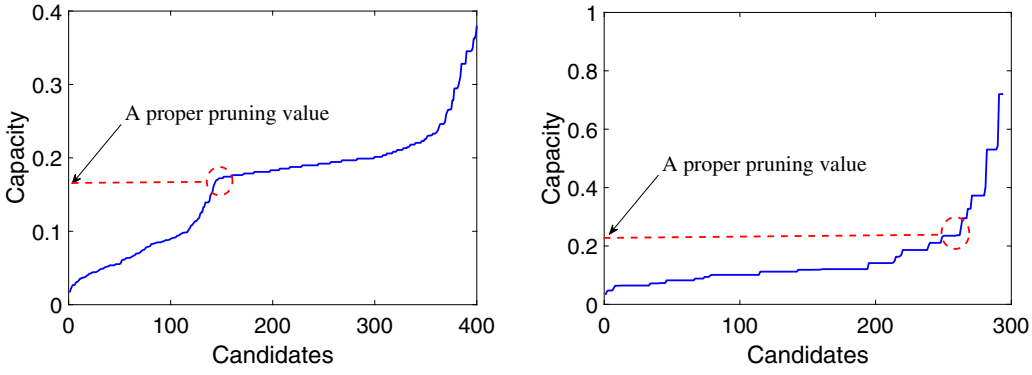


Fig. 4. The capacity of all shapelet candidates on *Beef* and *Synthetic* datasets. In most cases, there is an inflection point on the curve. Usually, the inflection point is a proper choice for the threshold *th*. The candidates who have lower capacity than *th* are pruned and the left candidates are considered as shapelets.

find out the best split point sp_i and calculate the capacity of $\boldsymbol{sc}$:

$$capacity(\boldsymbol{sc}) = \max(IG\_ratio(\boldsymbol{sc}, sp_i)) \quad i \in [1, n-1] \tag{27}$$

If this value is less than a preset threshold *th*, we discard $\boldsymbol{sc}$. Generally, *th* can be determined by experiments. Fig. 4 shows the capacity of shapelet candidates on *Beef* and *Synthetic* datasets from UCR archives[1] and we can easily find a proper threshold.

4.3. Shapelet transformation

When shapelets are obtained, the original time series can be represented as feature vectors by the process of shapelet transformation [13]. It is a helpful technique to preserve the character of time series while conveniently utilizing the mature classifiers. Fig. 1 (middle-bottom) explains the transformation process. For each time series $\boldsymbol{x}_i$, we first calculate the distance between $\boldsymbol{x}_i$ and each extracted shapelet and then arrange them according to the order of shapelets. Thus, the arranged distances can be considered as the feature vector of $\boldsymbol{x}_i$: ($d(\boldsymbol{s}_1, \boldsymbol{x}_i)$, $d(\boldsymbol{s}_2, \boldsymbol{x}_i)$, ..., $d(\boldsymbol{s}_N, \boldsymbol{x}_i)$). Function $d(\cdot)$ is a distance metric between each shapelet and $\boldsymbol{x}_i$, which can be calculated as

$$d(\boldsymbol{s}_j, \boldsymbol{x}_i) = \min_{p=1,\ldots,q} \sqrt{\frac{1}{l} \sum_{h=1}^{l} \left(\boldsymbol{s}_{j(h)} - \boldsymbol{x}_{i(p+h-1)}\right)^2} \tag{28}$$

where l is the length of $\boldsymbol{s}_j$, $q = m - l + 1$ denotes the number of subsequences with length l in $\boldsymbol{x}_i$.

4.4. Extension for multi-class situation

For $C(C > 2)$ classes, we find these locations with one-vs-rest strategy. When multi-classes are considered as one class, it is a typical multi-model problem and LFDA can deal with this situation well by local scaling. Therefore, this is also the problem of binary classification. The difference is that, when class i is treated as a dominant class while others are considered as one class, the non-zero locations of the obtained projection vector only stand for the difference between class i and other classes. So we extract the corresponding subsequences only from class i to form shapelet candidates for this projection vector. To find the discriminative parts of each class, we need to take each class as the dominant class recurrently.

In this case, the objective function has no difference with Eq. (17) in form, but when calculating the scatter matrix S_w and S_b, the original labels should be changed to $\{-1, +1\}$.

The algorithm 2 depicts the process:

Algorithm 2: Finding the projection vectors for the multi-class problem.

Input: time series training set: X, parameters: α, β
Output: projection vectors

```
for i = 1 : C do
    labels(X_i) = -1; /* X_i denotes the time series of class i*/;
    labels(X_others) = 1;
    v = getVector(X, α, β); /* the labels of time series in X
    have been changed*/;
    V(i) = v ;
end
return V;
```

5. Experiments

5.1. Datasets and evaluation criterion

For the sake of equivalent comparison, we use the UCR[1] and UEA[2] repository to evaluate the performance of the proposed method, which contain 28 datasets with various numbers of samples, lengths and classes. Table 1 shows the details. These datasets cover a range of domains, such as image outlines, motion capture, sensor readings and simulated data, which have been widely used in other literature [13–15]. We use the default train and test data splits, which is the same as the other methods.

We do the experiments on 28 datasets comparing with multi-baselines. Generally, an accepted statistic for comparing n_c classifiers over n_d datasets is a non-parametric Friedman test based on ranks [43]. The null hypothesis is that the average rank of n_c classifiers on n_d datasets is the same. Given an n_c by n_d matrix M of accuracy rates, the first step is to evaluate the rank matrix R, where $rank_{ij}$ is the rank of the jth classifier on the ith dataset. Concretely, the best performing classifier receives a rank of 1 (for a particular dataset), the second best receives rank 2, and so on, while ties are resolved by assigning the average rank. The average rank of the classifier j is: $rank_j = \sum_{i=1}^{n} rank_{ij}/n_d$. Under the null hypothesis of the ranks for all classifiers, the Friedman statistic χ_F^2

$$\chi_F^2 = \frac{12n_d}{n_c(n_c+1)}\left(\sum_{j=1}^{n_c} rank_j^2 - \frac{n_c(n_c+1)^2}{4}\right) \tag{29}$$

can be approximated by a Chi-squared distribution with $(n_c - 1)$ degree of freedom. On this basis, Demšar [43] proposes

$$F = \frac{(n_d - 1)\chi_F^2}{n_d(n_c - 1) - \chi_F^2} \tag{30}$$

to follow an F distribution with $(n_c - 1)$ and $(n_c - 1)(n_d - 1)$ degrees of freedom. It allows the average ranks and groups of no significantly different classifiers to be plotted on an order line referred to as a critical difference diagram. This test has been widely used to compare different time series classification methods [8,44].

Table 1
Datasets and parameters used in our experiments.

Name	#Train	#Test	Length	#Classes	α	β	th	K
Adiac	390	391	176	37	0.01	0.2	0	5
Beef	30	30	470	5	0.001	0.007	0.36	2
BeetleFly	20	20	512	2	0.001	0.005	0.19	3
BirdChicken	20	20	512	2	0.004	0.01	0.3	2
Chlorine.	467	3840	166	3	0.001	0.001	0.02	4
Coffee	28	28/28	286	2	0.01	0.2	0.5	3
Diatom.	16	306	345	4	0.04	0.08	0.5	3
DP_Little	400	645	250	3	0.07	0.03	0.03	9
DP_Middle	400	645	250	3	0.008	0.001	0.03	9
DP_Thumb	400	645	250	3	0.002	0.002	0.03	9
ECGFive.	23	861	136	2	0.01	0.001	0.11	3
FaceFour	24	88	350	4	0.002	0.02	0.25	3
Gun_Point	50	150	150	2	0.006	0.004	0.33	3
ItalyPower.	67	1029	24	2	0.003	0.003	0.18	7
Lighting7	70	73	319	7	0.03	0.18	0.13	5
Medical.	381	760	99	10	0.003	0.001	0.04	10
MoteStrain	20	1252	84	2	0.003	0.0005	0.07	6
MP_Little	400	645	250	3	0.001	0.001	0.02	9
MP_Middle	400	645	250	3	0.05	0.007	0.05	13
Herring	64	64	512	2	0.07	0.02	0.11	3
PP_Little	400	645	250	3	0.004	0.005	0.04	9
PP_Middle	400	645	250	3	0.07	0.005	0.03	9
PP_Thumb	400	645	250	3	0.005	0.007	0.03	11
SonyAIBO.	20	601	70	2	0.06	0.01	0.38	5
Symbols	25	995	398	6	0.01	0.05	0.11	2
Synthetic.	300	300	60	6	0.05	0.01	0.08	3
Trace	100	100	275	4	0.008	0.12	0.2	5
TwoLead.	23	1139	82	2	0.06	0.04	0.26	3

5.2. Parameters search

Our method (hereafter denoted as SEDS in experiments, for **S**hapelets **E**xtraction by **D**imensionality **S**election) requires the tuning of 4 parameters: regularization parameters α, β for sparse restriction, the local parameter K used by local scaling and the information gain ratio threshold th. In our experiments, th is readily determined in terms of shapelet candidates' capacity as mentioned in Section 4.2. For α and β, we find the proper value by cross-validation over the training data as the other literature [14,15]. For K, we will discuss it in detail in next paragraph. All these parameters used in our experiments are listed in Table 1.

The affinity matrix A is the key part for calculating the within-class and the between-class scatter matrix. Each item of A is determined by the local parameter K of Eq. (7). In other literatures like [16,33], they usually set a fixed value (e.g., $K = 7$) for their experiments. It is reasonable due to KNN classifier's experience. However for time series datasets, a fixed K is not appropriate because some training sets have only several training samples in one class. Therefore, we employ the cross-validation method to find the best K for

[2] http://www.uea.ac.uk/computing/machine-learning/shapelets/shapelet-data.

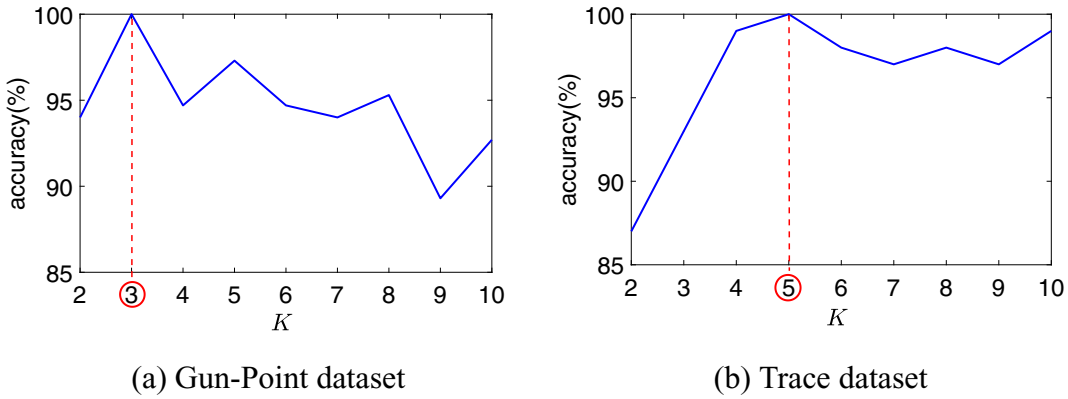


Fig. 5. The choice of parameter K for LFDA on *Gun-Point* and *Trace* datasets. The K value is critical to keep the multi-modal structure of data and different K will lead to different results (see the curves on these two datasets). The best K can be easily found from the curve.

each dataset. Fig. 5 gives the accuracy changes with different K on *Gun-Point* and *Trace* datasets from UCR archives [1].

The selected Ks in our experiments are given in the last column of Table 1. Generally, when a class has a little number of training samples, a small K (e.g., 3–5) is a suitable selection. But for a large number samples in a class like *DP_Little* dataset of UEA archives[2], a lager K is necessary, like 7–9, which conforms to [16]'s choice.

5.3. Performance evaluation compared with other shapelet-based methods

We first compare our method with the following shapelet-based methods: (1) Standard shapelet-based classifier with information gain (IG) [11], Kruskal–Wallis statistic (KW) and F-statistic (FS) [13]; (2) The Fast-shapelets algorithm with the help of SAX representation (FSH) [12]; (3) Learning Time Series which is the state-of-the-art shapelet-based algorithm for accuracy (LTS) [14]; (4) Shapelet-transform algorithm which transforms the shapelets into feature vectors for the first time (IGSVM) [13]; (5) FLAG which is known as the fastest algorithm in stand-alone mode [15]. All experiments are performed on a computer with intel i7 CPU and 16GB memory.

5.3.1. Classification accuracy

The classification results of different methods are shown in Table 2 and the best method of each dataset is highlighted in **bold**. The symbol "-" indicates that we cannot get these methods' results in a reasonable time (over 24 h). The last line gives the number of wins for each method and we can find that our method (SEDS) and LTS are best for these datasets (win 12 times), which demonstrates that our method is effective for shapelets discovery.

To show our method's superiority on accuracy clearly, Fig. 6 describes the critical difference diagram for different shapelet-based methods. We observe that our method is more accurate than the others, which has the best average rank (2.14) while other methods have larger ranks.

When making 1-to-1 comparison, our method still gets better accuracies than the others. From Table 3, we can see that our method obviously outperforms the standard shapelet-based methods (i.e., IG, KW and FS) in all datasets. When compared with FSH, IGSVM and FLAG, our method gets better results in most datasets and only loses several times. As for LTS, the result is a draw, which wins 12 times and losses 12 times. Although LTS has no significant difference with our method in classification accuracy, we will show our method is faster in next section.

Table 2
Classification accuracy(%) of our method and other 7 shapelet-based baselines on 28 datasets.

Dataset	IG	KW	FS	FSH	IGSVM	LTS	FLAG	SEDS
Adiac	29.9	26.2	15.6	57.5	23.5	51.9	75.2	**77.2**
Beef	50	33.3	56.7	50	**90**	76.7	83.3	86.7
BeetleFly	77.5	70	90	65	**97.5**	95	90	90
BirdChicken	85	87.5	90	90	95	**100**	85	**100**
Chlorine.	58.8	52	53.5	58.8	57.1	73	76	**77.9**
Coffee	96.4	85.7	**100**	92.9	**100**	**100**	**100**	**100**
Diatom.	76.5	62.1	76.5	87.3	93.1	94.2	96.4	**98.7**
DP_Little	–	–	–	60.6	66.6	**73.4**	68.3	64.5
DP_Middle	–	–	–	58.8	69.5	**74.1**	71.3	73.6
DP_Thumb	–	–	–	63.4	69.6	**75.2**	70.5	73.3
ECGFive.	77.5	87.2	99	99.8	99	**100**	92	**100**
FaceFour	84	44.3	75	92	**97.7**	94.3	90.9	96.6
Gun_Point	89.3	94	95.3	94	**100**	99.6	96.7	**100**
ItalyPower.	89.2	91	93.1	91	93.7	95.8	94.6	**96.1**
Lighting7	49.3	48	41.1	65.2	63	79	76.7	**79.5**
Medical.	48.8	47.1	50.8	64.7	52.2	71.3	71.4	71.1
MoteStrain	82.5	84	84	83.8	88.7	**90**	88.8	89.5
MP_Little	–	–	–	56.9	70.7	**74.3**	69.3	73.2
MP_Middle	–	–	–	60.3	76.9	**77.5**	75	74.4
Otoliths	67.2	60.9	57.8	60.9	64.1	59.4	64.1	**75**
PP_Little	–	–	–	57.6	**72.1**	71	67.1	69.3
PP_Middle	–	–	–	61.6	**75.9**	74.9	73.8	75.2
PP_Thumb	–	–	–	55.8	**75.5**	70.5	67.4	69.5
SonyAIBO.	85.7	72.7	95.3	68.6	92.7	91	92.9	**96.2**
Symbols	78.4	55.7	90.1	92.4	84.6	**94.5**	87.5	91.3
Synthetic.	94.3	90	95.7	94.7	87.3	97.3	**99.7**	97.7
Trace	98	94	**100**	**100**	98	**100**	99	**100**
TwoLead.	85.1	76.4	97	92.5	**100**	**100**	99	99.8
# Best	0	0	2	1	9	12	2	12

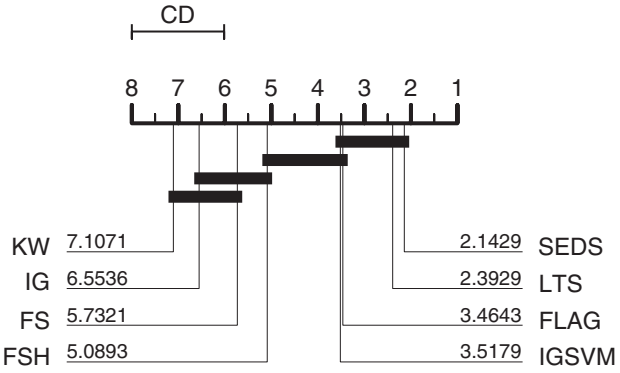


Fig. 6. Critical difference diagram of accuracy for our method against other shapelet-based methods on 28 datasets.

Table 3
1-to-1 comparison between our method and other baselines.

	IG	KW	FS	FSH	IGSVM	LTS	FLAG
# Datasets	20	20	20	28	28	28	28
SEDS Wins	20	20	17	26	17	12	22
SEDS Draws	0	0	3	1	2	4	2
SEDS Losses	0	0	0	1	9	12	4

5.3.2. Running time comparison

Besides accuracy, efficiency is also an important factor for evaluating time series classification methods. In this section, we first give the time complexity of our method and then offer the real running time of different methods on our computer.

For C classes in a dataset, when calculating the within-class and between-class scatter matrix, we need $O(nm^2)$ time. The solution procedure of our objective function by ADMM solver take $O(C(n^3 + tm^2))$ time, where t is the number of iterations [15]. Generally, C is less than 5 and $nm^2 \ll C(n^3 + tm^2)$, so the time complexity of our method is $O(C(n^3 + tm^2))$. Table 4 shows the total running time of different methods including shapelets discovery and classification stages.

As we can see from Table 2 and Table 4, although LTS has a similar classification accuracy with our method, it is too slow and will become infeasible for large datasets. For FLAG, it is faster in almost all datasets but has a lower accuracy, which only wins 4 times while our method wins 22 times in all 28 datasets (see Table 3). When compared to other baselines, our method is also about $2 \sim 4$ orders of magnitude faster. To more visually exhibit the difference of running time, the last line of Table 4 gives the average rank of each shapelet-based methods. It shows our method is fast (average rank 1.82) and has little difference with FLAG, while the standard shapelet-based methods (IG, KW and FS) have the worst time efficiency.

5.4. Compared with other time series classification methods

Among the studies of time series classification, 1-NN classifier with Dynamic Time Warping (DTW) distance measure (called NNDTW in this section) has been demonstrated that it is hard to beat [8,9], because DTW can mitigate against distortions in the time axis. Fig. 7 shows the accuracy comparison of our method and NNDTW . The coordinate axis denotes the accuracy of each method. The diagonal represents two methods have the same classification accuracy and the points away from this line indicates the corresponding method has a better result on these datasets. As can be seen, our method is more accurate than NNDTW in these datasets, which wins 25 times (25 of 28) and gets remarkable improvement in most datasets.

Another efficient method is proposed by Bagnall et al. [44] called COTE, which is a collective of ensembles of classifiers on different time series transformations. They combine 35 classifiers proportionate to improve the classification accuracy. The classifiers in the COTE collective are constructed in the time, frequency, change and shapelet transformation domains, which takes almost all time series classification algorithms of recent years into account. Fig. 8 shows the comparison of our method and COTE. As we can see, our method still has a slight advantage than COTE and only loses 5 times in all 20 datasets. Note that we cannot get the results of other 8 datasets because COTE cannot finish in a reasonable time. As [44] referred, COTE is no doubt slower because it needs to run all collected classifiers and weights the vote of each classifier by its cross-validation accuracy on the training data. It is hard for us to get the results of these datasets on our computer.

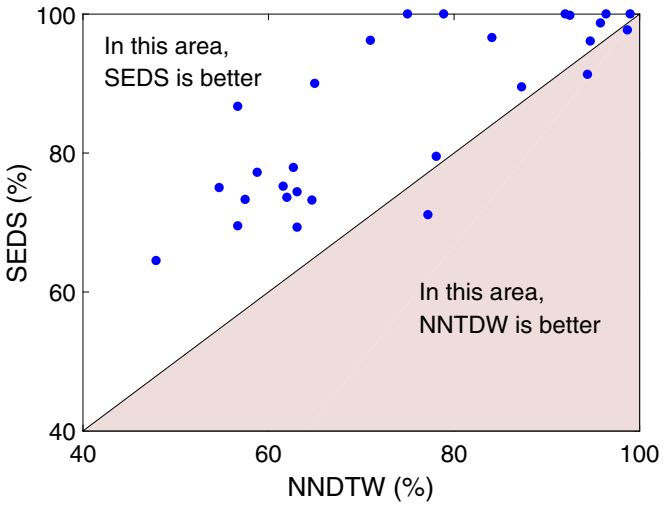


Fig. 7. Scatter plot of test accuracy of our method against 1-NN classifier with DTW on 28 datasets. The coordinates of each point are the accuracy of two methods. The points in the left upper triangle area denote our method is better than NNDTW. In this comparison, our method is better on 25 datasets and NNDTW is only better on 3 datasets.

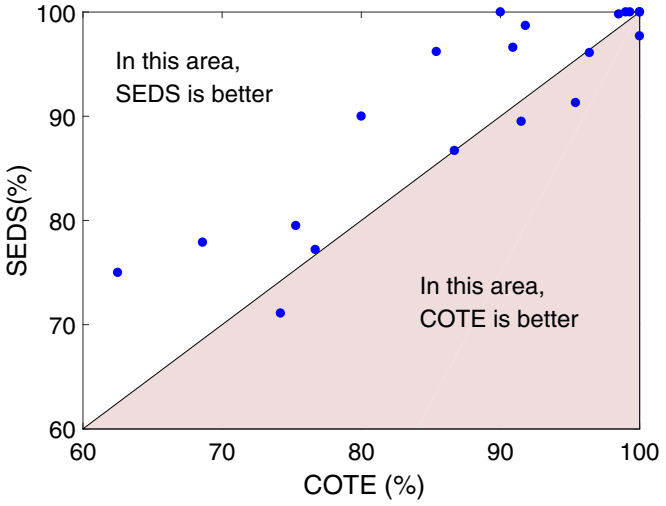


Fig. 8. Scatter plot of test accuracy of our method against FOTE on 20 datasets. In this comparison, our method is better on 13 datasets, FOTE is only better on 5 datasets, and they tie on 2 datasets.

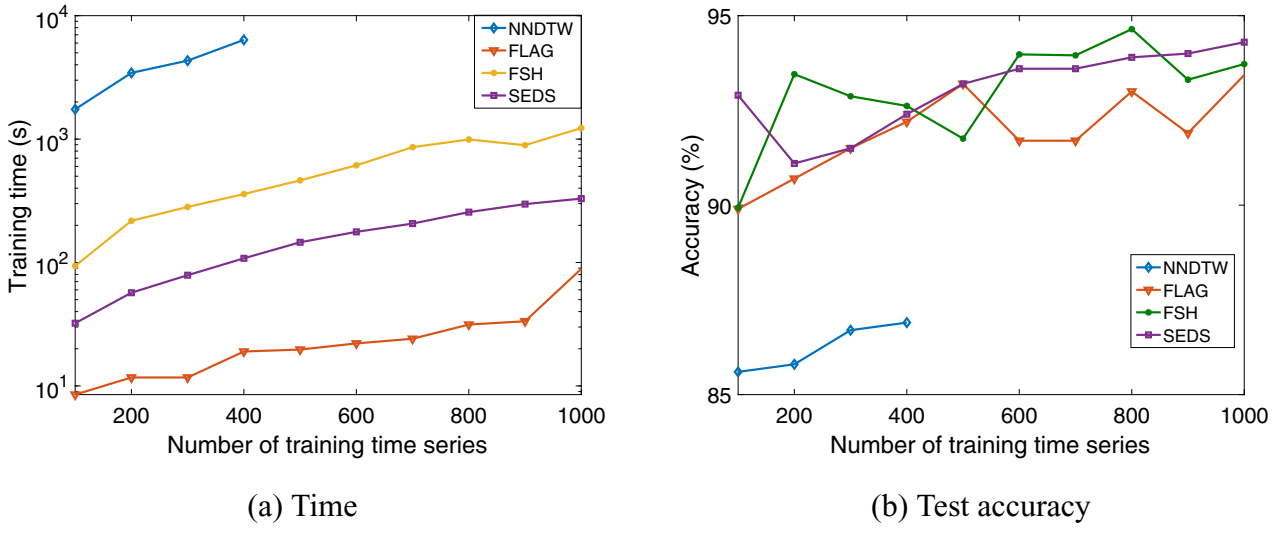
5.5. Scalability

In this section, we use the largest time series dataset *StraLightCurves* in the UCR archives[1] to test the scalability of our method. The dataset contains 9236 starlight time series of length 1024. There are 3 types of star objects: *Eclipsed Binaries* (2580 objects), *Cepheids* (1329 objects) and *RR Lyrae Variables* (5236 objects). The training set has 1000 time series in default partition. Two key factors for testing the scalability of time series classification methods are the number of training samples and the length of each sample.

First, we vary the number of training samples from 100 to 1000 while the length is fixed to 1024. In this case, IG, KW, FS, LTS, IGSVM and COTE are too slow and we cannot get the results in a reasonable time. So we do not compare them with other methods. Meanwhile, we get only 4 results for NNDTW because of the same reason. The total run time of different methods and the cor-

Table 4
Running time(in seconds). Number in brackets is the method's ranking (1 is the best). The best method for each dataset is highlighted in bold. The rankings of the method that cannot finish in a reasonable time are taken as 7.

Dataset	IG	KW	FS	FSH	IGSVM	LTS	FLAG	SEDS
Adiac	3287(7)	1349(5)	1513(6)	288(3)	706(4)	80596(8)	**2.78(1)**	6.25(2)
Beef	471(6)	484(7)	576(8)	154(3)	435(5)	414(4)	**1.15(1)**	4.43(2)
BeetleFly	22014(8)	21265(6)	21497(7)	9.4(3)	2700(5)	183.6(4)	1.54(2)	**0.5(1)**
BirdChicken	20612(8)	20369(6)	20466(7)	6.9(3)	3193(5)	181.1(4)	1.19(2)	**1.1(1)**
Chlorine.	9751(8)	3213(7)	3050(6)	617(4)	1181(5)	556(3)	**6.88(1)**	26.5(2)
Coffee	22.3(7)	21.8(5)	21.9(6)	16.5(4)	15.9(3)	76(8)	**0.13(1)**	0.17(2)
Diatom.	9.3(5)	8.99(3)	9.2(4)	13.7(7)	9.3(5)	88(8)	**0.41(1)**	0.84(2)
DP_Little	–(7)	–(7)	–(7)	1131(4)	9516(5)	803(3)	**1.81(1)**	5.3(2)
DP_Middle	–(7)	–(7)	–(7)	1286(3)	7041(5)	6082(4)	**1.78(1)**	6.55(2)
DP_Thumb	–(7)	–(7)	–(7)	1105(4)	11353(5)	1006(3)	**3.14(1)**	9.71(2)
ECGFive.	19(8)	18.4(6)	18.6(7)	4.6(3)	11.7(5)	9.1(4)	**0.08(1)**	0.13(2)
FaceFour	1021(8)	1012(7)	1010(6)	75(3)	410(5)	116(4)	**0.26(1)**	0.78(2)
Gun_Point	116.4(7)	112(6)	148.9(8)	7.6(3)	74.8(5)	14.1(4)	**0.07(1)**	0.12(2)
ItalyPower.	0.38(6)	0.22(3)	0.22(3)	0.5(7)	0.22(3)	7.3(8)	**0.03(1)**	0.09(2)
Lighting7	3442(7)	3438(6)	3584(8)	307(3)	1473(5)	965(4)	**0.31(1)**	1.56(2)
Medical.	4347(8)	2625(7)	2616(6)	164(3)	1547(4)	2199(5)	**1.69(1)**	8.43(2)
MoteStrain	1.41(7)	1.29(4)	1.29(4)	1.4(6)	0.84(3)	6(8)	**0.04(1)**	0.09(2)
MP_Little	–(7)	–(7)	–(7)	1297(3)	7394(5)	5233(4)	**2.95(1)**	9.53(2)
MP_Middle	–(7)	–(7)	–(7)	1186(4)	12102(5)	724(3)	**2.23(1)**	5.9(2)
Otoliths	17864(7)	18166(8)	17789(6)	183(3)	8536(5)	264(4)	1.25(2)	**0.48(1)**
PP_Little	–(7)	–(7)	–(7)	1107(3)	8142(5)	4805(4)	**3.91(1)**	10(2)
PP_Middle	–(7)	–(7)	–(7)	1135(3)	4753(5)	3406(4)	**2.19(1)**	6.98(2)
PP_Thumb	–(7)	–(7)	–(7)	1172(3)	8209(5)	6638(4)	**4.2(1)**	9.32(2)
SonyAIBO.	1.17(7)	1.02(4)	1.02(4)	1.1(6)	0.75(3)	25.4(8)	**0.04(1)**	**0.04(1)**
Symbols	3325(7)	3318(6)	3622(8)	59.4(3)	1263(5)	528(4)	**0.85(1)**	4.19(2)
Synthetic.	291(6)	164(4)	161(4)	39.4(3)	922(8)	293(7)	**0.26(1)**	0.87(2)
Trace	11542(7)	11622(8)	11411(6)	98.7(3)	4838(5)	394(4)	**0.35(1)**	1.21(2)
TwoLead.	0.48(6)	0.42(4)	0.42(4)	1.1(7)	0.26(3)	5.2(8)	0.08(2)	**0.03(1)**
Average rank	**7**	**6**	**6.21**	**3.82**	**4.71**	**4.96**	**1.14**	**1.82**

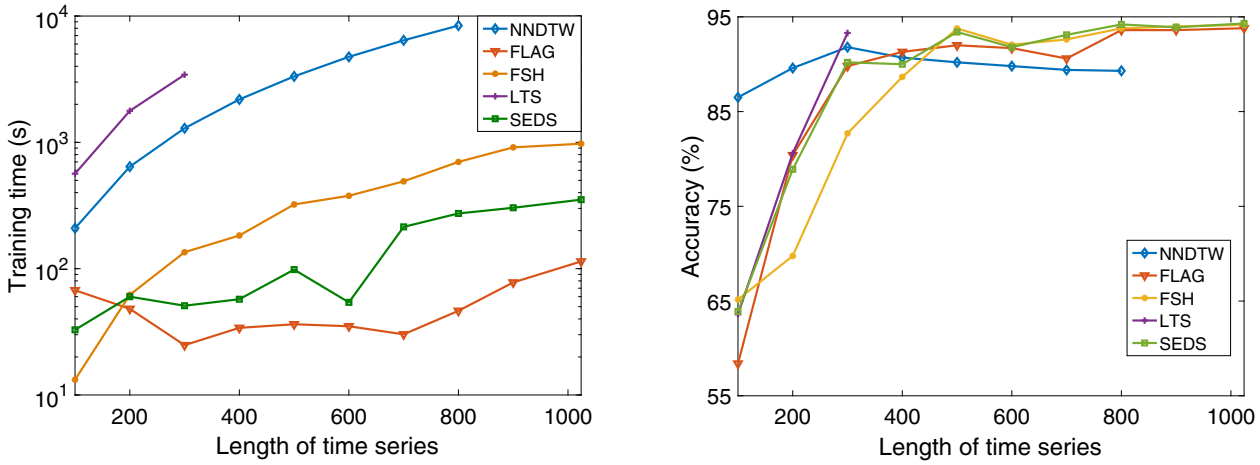


(a) Time (b) Test accuracy

Fig. 9. Training time and test accuracy when varying the number of training time series on *StarLightCurves* dataset. (a) The running time comparison with the number of training time series increasing. (b) Test accuracy comparison. There are only four results for NNDTW due to the time-consuming.

responding classification accuracies are shown in Fig. 9. It has the same trends as Table 4, FLAG and our method SEDS are more scalable when training data size increases, while NNDTW and FSH take much time to get the results even for a small data number. This is because we extract the shapelet candidates only from the discriminative locations by utilizing local scatter matrix and ignore other subsequences. Compared to verify all subsequences' discriminative power, the computation of the local scatter matrix is cheap. Due to the random selection of training samples, the accuracy has a little fluctuate at some points for these methods in Fig. 9(b). But basically, the accuracy is improved with the number of training samples increasing.

Then, we fix the number of training samples to 1000 and vary the length from 100 to 1024. Again we do not compare IG, KW, FS, COTE and IGSVM with other methods due to the unbearable running time. LTS can only run for length 100, 200 and 300. Fig. 10 shows the results. In this case, we also get the same trends as Table 4 in running time. Our method is only slower than FLAG but faster than others when the length is increasing. For accuracy, the length has little influence with NNDTW while other methods get



(a) Time (b) Test accuracy

Fig. 10. Training time and test accuracy when varying the length of time series on *StarLightCurves* dataset. (a) The running time comparison with the length of time series increasing. (b) Test accuracy comparison. We only get 3 results for LTS and 8 results for NNDTW in a reasonable time.

poor accuracy when the length is less than 300. However, with the length increasing, our method gets the best results.

6. Conclusion

In this paper, we propose a novel perspective to extract discriminative features (shapelets) from time series. Unlike search-based and learning-based shapelets algorithm, we utilize a dimension selection method. By considering each time series as a high-dimensional data, we reduce the dimensions without discriminative power by the well-known local Fisher discriminant analysis method and sparse restrictions. The retained dimensions correspond to the discriminative locations of time series. After getting the shapelet candidates, a capable criterion, information gain ratio, is utilized to prune these candidates and the retained candidates form the shapelets. Then the original time series are transformed into feature vectors by the extracted shapelets. When predicting the label of test time series, the powerful SVM classifier is applied. We have conclusively shown that the proposed method achieves significant improvement than other shapelet-based methods and typical time series classification methods through extensive experiments.

Acknowledgments

The authors like to thank Lu Hou for valuable discussions. This work is supported by the National 863 Program of China [grant number 2015AA015401] and National Natural Science Foundation of China [grant numbers 61772289, 61702285 and 61402243].

References

[1] A.J. Hussain, D. Al-Jumeily, H. Al-Askar, N. Radi, Regularized dynamic self-organized neural network inspired by the immune algorithm for financial time series prediction, Neurocomputing 188 (2016) 23–30, doi:10.1016/j.neucom.2015.01.109.
[2] S. Kodba, M. Perc, M. Marhl, Detecting chaos from a time series, Eur. J. Phys. 26 (2005) 205–215, doi:10.1088/0143-0807/26/1/021.
[3] M. Perc, The dynamics of human gait, Eur. J. Phys. 26 (2005a) 525–534, doi:10.1088/0143-0807/26/3/017.
[4] M. Perc, Nonlinear time series analysis of the human electrocardiogram, Eur. J. Phys. 26 (2005b) 757–768, doi:10.1088/0143-0807/26/5/008.
[5] G. RodrÃguez-BermÃºdez, P.J. GarcÃa-Laencina, J. Roca-GonzÃ!'lez, J. Roca-Dorda, Efficient feature selection and linear discrimination of eeg signals, Neurocomputing 115 (2013) 161–165, doi:10.1016/j.neucom.2013.01.001.
[6] D. Helbing, D. Brockmann, T. Chadefaux, K. Donnay, U. Blanke, O. Woolley-Meza, M. Moussaid, A. Johansson, J. Krause, S. Schutte, M. Perc, Saving human lives: what complexity science and information systems can contribute, J. Stat. Phys. 158 (3) (2015) 735–781, doi:10.1007/s10955-014-1024-9.
[7] 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 Endow. 1 (2) (2008) 1542–1552, doi:10.14778/1454159.1454226.
[8] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Min. Knowl. Discov. 29 (3) (2015) 565–592, doi:10.1007/s10618-014-0361-2.
[9] G.E. Batista, X. Wang, E.J. Keogh, A complexity-invariant distance measure for time series, in: Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM, 2011, pp. 699–710, doi:10.1137/1.9781611972818.60.
[10] 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, 2011, pp. 1154–1162, doi:10.1145/2020408.2020587.
[11] 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, 2009, pp. 947–956, doi:10.1145/1557019.1557122.
[12] T. Rakthanmanon, E. Keogh, Fast shapelets: a scalable algorithm for discovering time series shapelets, in: Proceedings of the 13th SIAM International Conference on Data Mining, SIAM, 2013, pp. 668–676, doi:10.1137/1.9781611972832.74.
[13] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Min. Knowl. Discov. 28 (4) (2014) 851–881, doi:10.1007/s10618-013-0322-1.
[14] 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, 2014, pp. 392–401, doi:10.1145/2623330.2623613.
[15] L. Hou, J.T. Kwok, J.M. Zurada, Efficient learning of timeseries shapelets, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 1209–1215.
[16] M. Sugiyama, Local fisher discriminant analysis for supervised dimensionality reduction, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 905–912, doi:10.1145/1143844.1143958.
[17] Z. Zhang, Y. Xu, J. Yang, X. Li, D. Zhang, A survey of sparse representation: algorithms and applications, IEEE Access 3 (2015) 490–530, doi:10.1109/ACCESS.2015.2430359.
[18] S. Cai, L. Zhang, W. Zuo, X. Feng, A probabilistic collaborative representation based approach for pattern classification, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2950–2959, doi:10.1109/CVPR.2016.322.
[19] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, K. Knight, Sparsity and smoothness via the fused lasso, J. R. Stat. Soc.: Ser. B (Stat. Methodol.) 67 (1) (2005) 91–108, doi:10.1111/j.1467-9868.2005.00490.x.
[20] K. Buza, L. Schmidt-Thieme, Motif-based classification of time series with bayesian networks and svms, in: Advances in Data Analysis, Data Handling and Business Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 105–114, doi:10.1007/978-3-642-01044-6_9.
[21] K. Buza, J. Koller, K. Marussy, Process: projection-based classification of electroencephalograph signals, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2015, pp. 91–100, doi:10.1007/978-3-319-19369-4_9.

[22] A. Galicia, J.F. Torres, F. Martnez-lvarez, A. Troncoso, Scalable forecasting techniques applied to big electricity time series, in: Proceedings of the 14th International Work-Conference on Artificial Neural Networks, 2017, pp. 165–175, doi:10.1007/978-3-319-59147-6_15.
[23] R. Perez-Chacon, R.L. Talavera-Llames, F. Martinez-Alvarez, A. Troncoso, Finding electric energy consumption patterns in big time series data, in: Proceedings of the 13th International Conference on Distributed Computing and Artificial Intelligence, 2016, pp. 231–238, doi:10.1007/978-3-319-40162-1_25.
[24] R.L. Talavera-Llames, R. Prez-Chacn, M. Martnez-Ballesteros, A. Troncoso, F. Martnez-lvarez, A nearest neighbours-based algorithm for big time series data forecasting, in: Proceedings of the 11th International Conference on Hybrid Artificial Intelligence Systems, 2016, pp. 174–185, doi:10.1007/978-3-319-32034-2_15.
[25] J.F. Torres, A.M. Fernndez, A. Troncoso, F. Martnez-lvarez, Deep learning-based approach for time series forecasting with application to electricity load, in: Proceedings of the 2017 International Work-Conference on the Interplay Between Natural and Artificial Computation, 2017, pp. 203–212.
[26] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing sax: a novel symbolic representation of time series, Data Min. Knowl. Discov. 15 (2) (2007) 107–144, doi:10.1007/s10618-007-0064-z.
[27] K.-W. Chang, B. Deka, W.-M.W. Hwu, D. Roth, Efficient pattern-based time series classification on gpu, in: Proceedings of the 2012 IEEE 12th International Conference on Data Mining, 2012, pp. 131–140, doi:10.1109/ICDM.2012.132.
[28] M. Wistuba, J. Grabocka, L. Schmidt-Thieme, Ultra-Fast Shapelets for Time Series Classification, ArXiv e-prints (2015).
[29] Q. Zhang, J. Wu, H. Yang, Y. Tian, C. Zhang, Unsupervised feature learning from time series, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 2322–2328.
[30] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (2) (1936) 179–188, doi:10.1111/j.1469-1809.1936.tb02137.x.
[31] K. Fukunaga, Introduction to Statistical Pattern Recognition, second, Academic press, Boston, 1990, doi:10.1016/B978-0-08-047865-4.50002-8.
[32] X. He, P. Niyogi, Locality preserving projections, Adv. Neural Inf. Process. Syst. 16 (1) (2003) 186–197. 10.1.1.19.9400.
[33] S. Pedagadi, J. Orwell, S. Velastin, B. Boghossian, Local fisher discriminant analysis for pedestrian re-identification, in: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 3318–3325, doi:10.1109/CVPR.2013.426.
[34] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: Proceedings of the 14th International Conference on Neural Information Processing Systems, 2001, pp. 849–856.
[35] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: Proceedings of the 17th International Conference on Neural Information Processing Systems, 2004, pp. 1601–1608.
[36] J.C. Platt, K. Toutanova, W.-t. Yih, Translingual document representations from discriminative projections, in: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, 2010, pp. 251–261.
[37] J.H. Friedman, Regularized discriminant analysis, J. Am. Stat. Assoc. 84 (405) (1989) 165–175, doi:10.1080/01621459.1989.10478752.
[38] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227, doi:10.1109/TPAMI.2008.79.
[39] M. Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, first, Springer Publishing Company, Incorporated, 2010.
[40] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2011) 1–122, doi:10.1561/2200000016.
[41] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202, doi:10.1137/080716542.
[42] M. Elad, M.A.T. Figueiredo, Y. Ma, On the role of sparse and redundant representations in image processing, Proc. IEEE 98 (6) (2010) 972–982, doi:10.1109/JPROC.2009.2037655.
[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30, doi:10.1016/j.jecp.2010.03.005.
[44] 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, doi:10.1109/TKDE.2015.2416723.

Zhenguo Zhang a Ph.D. student in Computer Science and Engineering at the College of Computer and Control Engineering, Nankai University (China). Prior to undertaking a Ph.D., he obtained a Master Degree in Computer Science in 2009 and a Bachelor Degree in Computer Science and Technology in 2006, Yanbian University. His research interests include machine learning, data mining and time series analysis.

Haiwei Zhang received the Ph.D. degree in Computer Science and Technology from Nankai University in 2008. Currently, he is working as a assistant professor in College of Computer and Control Engineering, Nankai University (China). His main research interests include database, data mining, information retrieval and graph data management.

Yanlong Wen received the Ph.D. degree in Computer Science and Technology from Nankai University in 2012. Currently, he is working as a senior engineer in College of Computer and Control Engineering, Nankai University (China). His main research interests include database, data mining and information retrieval.

Ying Zhang received the Ph.D. degree in Computer Science and Technology from Nankai University in 2013. Currently, she is working as an associate professor in College of Computer and Control Engineering, Nankai University (China). Her main research interests include sentiment analysis, data mining and information retrieval.

Xiaojie Yuan received the Ph.D. degree in Control Theory and Engineering from Nankai University in 2000. Currently, she is working as a professor in College of Computer and Control Engineering, Nankai University (China). Her main research interests include database, data mining and information retrieval.