Time Series Classification by Modeling the Principal Shapes

Zhenguo Zhang[1,2], Yanlong Wen[1], Ying Zhang[1], and Xiaojie Yuan[1(✉)]

[1] College of Computer and Control Engineering, Nankai University, 38 Tongyan Road, Tianjin 300350, People's Republic of China
yuanxj@nankai.edu.cn

[2] Department of Computer Science and Technology, Yanbian University, 977 Gongyuan Road, Yanji 133002, People's Republic of China

Abstract. Time series classification has been attracting significant interests with many challenging applications in the research community. In this work, we present a novel time series classification method based on the statistical information of each time series class, called Principal Shape Model (PSM), which can quickly and effectively classify the time series even if they are very long and the dataset is very large. In PSM, the time series with the same class label in the training set are gathered to extract the principal shapes which will be used to generate the classification model. For each test sample, by comparing the minimum distance between this sample and each generated model, we can predict its label. Meanwhile, through the principal shapes, we can get the intrinsic shape variation of time series of the same class. Extensive experimental results show that PSM is orders of magnitudes faster than the state-of-art time series classification methods while achieving comparable or even better classification accuracy over common used and large datasets.

Keywords: Principal shapes · Time series · Fitting · Classification

1 Introduction

Time series research has attracted significant interests in the data mining community, due to the fact that series data are presented in a wide range of our daily life. As a fundamental research, time series classification has been studied extensively and many algorithms have been proposed during the last decade. Recent empirical evidence has strongly suggested that the simple nearest neighbor classifier is very difficult to beat [5]. Thus, the vast majority of time series classification research has focused on alternative distance measures for 1-NN classifiers based on raw data, compression data or smoothing data [20]. In these kinds of methods, the similarity of time series is a key point for the classification task. *Ding et al.* [5] present a good survey of similarity calculation methods for time series. While the nearest neighbor classifier has the advantages of simplicity and not requiring extensive parameter tuning, it does have several disadvantages.

A. Bouguettaya et al. (Eds.): WISE 2017, Part I, LNCS 10569, pp. 406–421, 2017.
DOI: 10.1007/978-3-319-68783-4_28

One is its time and space consumption for large datasets, the other is it does not give us a reasonable explanation about the relationship between a time series and a class. Unlike the NN-based methods taking all series points into consideration, another kinds of methods treat the subsequences of time series as a basis for classification. Recent years, the shapelets-based methods [6,14,17,21] have been studied most, which attempt to find the subsequences with high discriminative power as features of one class. The idea is that the time series in different classes can be distinguished by their local subsequences instead of the whole ones. These kinds of methods provide interpretable results, which can help researchers understand the data.

Due to the influences of many factors, the time series of a class are slightly different at some time point. It is common for data acquisition. The above two kinds of methods, i.e., 1-NN classifier and shapelets-based methods, describe these differences by distance between time series or between subsequences. It is a reasonable assumption that two time series in the same class have a small distance. In 1-NN classifier, the distances between the test sample and all training time series should be calculated first and the one with the minimum value should have the same label with the test sample. For shapelets-based methods, all subsequences with any length are shapelet candidates and the most important step to find out shapelets is to calculating the distances between a candidate and all training time series [21]. Even though there are some pruning strategy to accelerate this process [17], the process of finding the most discriminative subsequences is computationally expensive.

Unlike existing methods, we utilize an entirely different method and build a model to describe these differences of the time series of a class. The label of a test sample can be obtained by comparing the similarity of the instance and the models. By this way, the large amount of distance calculation is unnecessary. Moreover, the variation characteristics of time series of a class can also be represented by the model. In this paper, the variation characteristics of time series is described by a linear model. Firstly, We extract the intrinsic variation characteristics of each class to generate the principal shapes based on its statistical information. Then each class corresponds to a model that constructed by its principal shapes. All time series of a class are considered as a linear combination of principal shapes. The similarity of a test time series and a class models is represented by an objective function. By optimizing these objective functions and comparing the similarity values, we can predict the class label of this test time series. The experimental results on a large number of time series datasets demonstrate that our method is orders of magnitudes faster than other methods, while achieving comparable or even better classification accuracies.

In summary, our contributions include the following:

- We utilize the principal shapes to describe the variation characteristics of time series in the same class, which gives a good understand for time series.
- The distance between the model constructed by principle shapes and a test sample is employed to predict the test sample's label. Compared with 1-NN based method, this technique can reduce the amount of distance calculation.

- We empirically validate the performance of our proposed method on commonly used time series datasets and some large datasets. The results show promising results compared to other methods.

The remainder of this paper is organized as follows: We provide some background into time series classification and review the common used algorithms in Sect. 2. Section 3 describes the principal shape model in detail and illustrates the principal shapes by a toy example. The effectiveness and efficiency of the proposed method are also discussed. Section 4 demonstrates the PSM is effective by adequate experiments. Conclusions and future work are drawn in Sect. 5.

2 Background and Related Work

2.1 Time Series Classification

A time series is an ordered set of real-valued variables. Usually, the time series are recorded in temporal order at fixed intervals of time. Time series classification is defined as the problem of building a classifier from a labelled training time series set to predict the label of new time series. A time series $\boldsymbol{x}_i$ with m values is represented as $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})^T$ and an associated class label l_i. The training set is a set of n labelled pairs $(\boldsymbol{x}_i, l_i)$ with C classes: $X = \{(\boldsymbol{x}_1, l_1), (\boldsymbol{x}_2, l_2), \ldots, (\boldsymbol{x}_n, l_n)\}$, where $l_i \in C$. For classification algorithm, the normalization of time series is necessary and the commonly used approach is z-normalization: $norm(\boldsymbol{x}) = \frac{\boldsymbol{x} - mean(X)}{std(X)}$. For ease of explanation, we still use $\boldsymbol{x}$ to denote $norm(\boldsymbol{x})$. Given an unlabelled time series dataset, the objective is to classify each sample of this dataset to one of the predefined classes.

The main difference between time series classification problems and the general classification task is the order of observations is very important. Therefore, the algorithm for time series classification requires specific techniques to meet this characteristic. Two kinds of methods are studied in recent years. One is global-based method by taking the whole time series into account, while the other considers the usage of time series subsequences, called local-based method.

2.2 Global-Based Methods

In global-based methods, the classifier takes the whole time series as features and predicts the label of a new time series based on its similarity metrics where distance is measured directly between time series points. Two kinds of distances, Euclidean distance (ED) and Dynamic Time Warping (DTW) distance, have been widely and successfully used as the similarity metrics [10,11,15,18]. ED is usually time and space efficient, but it often gets a poor classification accuracy [18]. DTW is considered as a better solution in time series research community because it allows time series to match even if they are out of phase in the time axis [12,19]. It has been proved that the 1-NN classifier with DTW is the best approach for small datasets. However, as the number of time series increases, the classification accuracy of DTW will converge to ED [5]. As we mentioned in

Sect. 1, these methods have two drawbacks: high time and space complexity for large datasets and long time series and results uninterpretable.

2.3 Local-Based Methods

To address the limitations of 1-NN classifier, a new shapelets-based classification algorithm is proposed by *Ye* in 2009 [21]. Informally, shapelets are time series subsequences which can maximally represent a class in some sense. The algorithm completes the classification task by constructing a decision tree classifier. The important point is that shapelet offers interpretable features to domain experts. The utility of shapelets has been confirmed and extended by many researchers [7]. Nevertheless, since all subsequences of time series could be a shapelet, finding a good shapelet is a time-consuming task. Although there are several speedup strategy, e.g., early abandon pruning, entropy pruning [21], intelligent caching and reuse of intermediate results (called Logical Shapelets) [14], etc., it still takes a long running time, especially in case of large datasets and long time series. To speed up the process of shapelets discovery, *chang et al.* [3] propose to parallelize the distance computations using GPUs. *Rakthanmanon et al.* [17] propose a heuristic strategy, called Fast-Shapelets (FSH), to speed up the searching process by exploiting a random projection method on the SAX representation of time series to find the potential shapelet candidates [13]. It is up to orders of magnitudes faster than Logical Shapelets, which reduces the time complexity from $O(n^2m^3)$ to $O(nm^2)$ (n is the number of time series in the dataset, and m is the length of the longest time series). Except for decision tree classifier, some off-the-shelf classifiers like SVM are also used on shapelets data [8,9], which are transformed by measuring distances between the original time series and discovered shapelets. It can improve prediction accuracy while still maintaining the explanatory power of shapelets.

Another way to find shapelets called Learning Time Series (LTS) is to learn (not search for) shapelets by optimizing an objective function [6]. The LTS algorithm enables learning near-to-optimal shapelets directly without the need to try out lots of candidates and can learn true top-k shapelets. It is an entirely new perspective on time series shapelets. The algorithm can also gain the higher accuracy in some time series dataset than other methods, but it is space and time-consuming.

3 Principal Shape Model

3.1 Principal Shapes Extraction from Time Series

If we take a time series as a vector, the vectors of a training set can form a distribution in m dimensional space. If we model this distribution, we can generate new series which are similar to those in the original training set, and also can examine the test time series to decide whether they are plausible examples. To simplify the problem, we wish to reduce the dimensionality of the training data from m to a reasonable dimension k. An effective approach is to apply Principal Component Analysis (PCA) method. Given a training set of n_i time series with the same class label, the mean time series $\overline{\boldsymbol{x}_i}$ is calculated as: $\overline{\boldsymbol{x}_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} (\boldsymbol{x}_j)$.

The principal shapes of variation in each class, i.e. the ways in which some points of time series tend to move together, can be found by using PCA. In order to remove the meaningless distortions, we first calculate the deviations $d\boldsymbol{x}_j = \boldsymbol{x}_j - \overline{\boldsymbol{x}_i}$ of each class. From these deviations, we can get the matrix M of the i-th class in training set. And then the eigenvectors (also called principal components), $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{n_i}$, and the corresponding eigenvalues, $\lambda_1, \ldots, \lambda_{n_i}$, can be obtained by singular value decomposition method. The variations of time series points can be described by the principal components and eigenvalues. Each principal component gives a pattern of variation of time series points. The first principal component, which is associated with the largest eigenvalue, λ_1, describes the largest part of the time series variation. The proportion of total variance described by the j-th principal component is equal to the λ_j.

Generally, most of the time series variation can be represented by a small number k of principal components. The parameter k can be assigned a certain number, but a more common practice is to choose the first k principal components from a sufficiently large proportion of the total variance of the training set. The proportion can be decided by the cross validation method for each dataset. The total variance S is defined as: $S = \sum_{j=1}^{n_i} \lambda_j$.

By this way, all time series of a class can be approximated by taking the mean time series and a weighted sum of the first k principal components:

$$\boldsymbol{x} \approx \overline{\boldsymbol{x}} + P\boldsymbol{b} \tag{1}$$

where $P = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_k)$ is the matrix that made up by the first k eigenvectors; $\boldsymbol{b} = (b_1, b_2, \ldots, b_k)^T$ is a weight vector.

By only choosing the first k principal components instead of the whole eigenvectors, we can get the principal shapes contained in each class. Besides, the distortions occasionally occur in one or several time series can be eliminated, because they are usually related to small principal components.

These above equations allow us to generate new instance of time series by varying the parameter $\boldsymbol{b}$ within a suitable limit, which should be related with the training set. Note that the variance of b_j on the training set can be given by λ_j, so a reasonable limit [16] is

$$\|b_j\| \leq 3\sqrt{\lambda_j} \tag{2}$$

The coefficient 3 is selected because most of the variations lie in 3 standard deviation of the mean. By applying limits of $\pm 3\sqrt{\lambda_j}$ to the parameter b_j, we can ensure that the generated time series is similar to those in the training set.

3.2 An Example of Principal Shapes on Toy Dataset

We use Car dataset as our toy dataset, which is one of time series datasets from UCR archive[1], to exhibit the extracted principal shapes and their variations. It is composed of four classes, which contains 60 training time series and 60 test time series. Each time series contains 577 real observations. We extract the principle shapes of each class from training set and the first four principal shapes of each class are shown in Fig. 1.

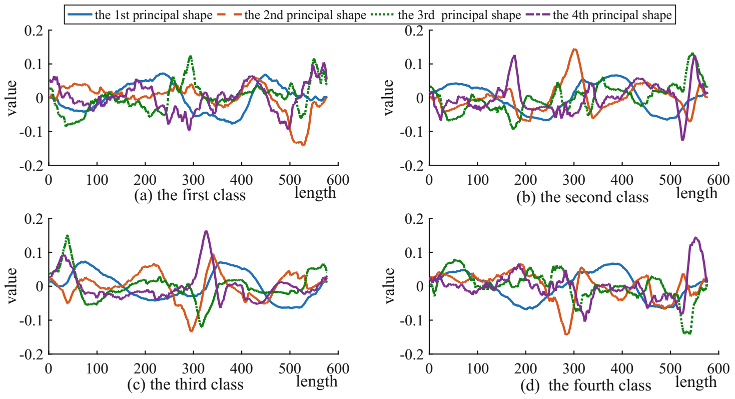

Fig. 1. The principal shapes of 4 classes in toy dataset. The 4 principal shapes of each class are presented in (a), (b), (c) and (d), respectively.

From this figure we can see, the principal shapes of each class are different, so we can build a model based on these principal shapes for time series classification. As mentioned above, the vector $\boldsymbol{b}$ defines a set of parameters of a deformable model. By varying the elements of $\boldsymbol{b}$, we can vary the principal shapes. All extracted principal shapes of each class are added up after varying by an admissible parameter, and then we get the overall shape variation of each class. Figure 2 shows the range of variation of each class. The shape is a superposition of all admissible principal shapes. The dotted arrow lines indicate that all time series of this class can only vary in these scope.

It is clearly that time series of different classes have different shape variations. The class information of a new time series that we want to predict can be obtained by fitting these shape variations. The one with the minimum distance is the best match with the test time series.

3.3 Classifying New Time Series

The principal shapes of class i and the corresponding parameter $\boldsymbol{b}^i$ form a linear model which we call it principal shape model[2]. After getting the models of all

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

[2] The superscript i denotes that we are dealing with the i-th class.

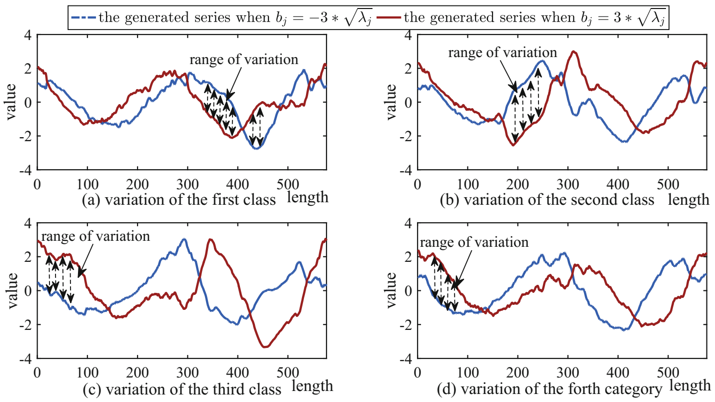

Fig. 2. The overall shape variation of each class in toy dataset.

classes, the training time series can be expressed by these models. When doing the classification task, for a new time series, our purpose is to fit these models to this time series. More specifically, if a time series belongs to a specific class i, the corresponding model $(P^i, \boldsymbol{b}^i)$ must fit it well, otherwise, the model has a large distance to this time series. Assume $\boldsymbol{y}$ is the time series to be classified, an objective function to measure the fitting degree of class i is defined by

$$f(\boldsymbol{b}^i) = \frac{1}{2}\|\boldsymbol{y} - (\overline{\boldsymbol{x}_i} + P^i\boldsymbol{b}^i)\|^2 \qquad s.t. \quad \|b^i_j\| \leq 3\sqrt{\lambda^i_j} \tag{3}$$

where $i \in C$ and $\overline{\boldsymbol{x}_i}$ is the mean of i-th class time series, P^i is a matrix of the first k eigenvectors generated by the i-th class, $\boldsymbol{b}^i$ is the weight vector.

Formula (4) is a least squares problem with an inequality constraint. It has been proved that f (without constraint) is a strictly downward convex function and has the only extremum [2]. The best $\boldsymbol{b}^{i*}$ is:

$$\boldsymbol{b}^{i*} = (P^{iT}P^i)^{-1}P^{iT}(\boldsymbol{y} - \bar{\boldsymbol{x}}_i) \tag{4}$$

As we can find that if we want to get the minimum value of objective function, an inverse operation is necessary. Due to the complexity of matrix inverse operation and the inequality constraint, we cannot easily get the best parameter vector $\boldsymbol{b}^i$. An alternative method to solve this problem is the Gradient Descent (GD). With the help of GD, we can gain the minimum value between a new time series $\boldsymbol{y}$ and the current model ($\overline{\boldsymbol{x}_i}$, P^i, $\boldsymbol{b}^i$). Algorithm 1 describes this process.

For gradient descent algorithm, the initial value of $\boldsymbol{b}^i$ is usually set to a random number and the algorithm should run many times to obtain the best value. But in this algorithm, due to the characteristic of f: $\frac{\partial^2 f(\boldsymbol{b}_i)}{\partial {\boldsymbol{b}^i}^2} = P^{iT}P^i$, there's no need to repeat the experiment many times to find the minimum value by setting different initial values of $\boldsymbol{b}^i$. In Algorithm 1, the lines 1 gives a default value of $\boldsymbol{b}^i$ and the distance between $\boldsymbol{y}$ and the model by using the default $\boldsymbol{b}^i$ is obtained in line 3. The constants A and B in line 7 are used to calculate the

Algorithm 1. Fitting the model to a new time series

```
Input: $\boldsymbol{y}$: the time series to be classified; $P^i$: the matrix of eigenvectors of class $i$
    $\overline{\boldsymbol{x}}_{\boldsymbol{i}}$: the mean of time series of class $i$; $\boldsymbol{\lambda}^i$: eigenvalues of class $i$
Output: $dis$: the minimum distance between $\boldsymbol{y}$ and the current model
 1: initialize $\boldsymbol{b}^i$=$\boldsymbol{0}$;
 2: $\boldsymbol{r} = 3\sqrt{\boldsymbol{\lambda}^i}$;
 3: $dis = \frac{1}{2}\|\boldsymbol{y} - (\overline{\boldsymbol{x}_i} + P^i\boldsymbol{b}^i)\|^2$;
 4: if $dis < \varepsilon$ then
 5:    return $dis$;
 6: end if
 7: $A = P^{iT}(\boldsymbol{y} - \bar{\boldsymbol{x}}_i)$; $B = P^{iT}P^i$;
 8: while true do
 9:    $\Delta = A + B\boldsymbol{b}^i$;
10:    $\boldsymbol{b}^i = \boldsymbol{b}^i - \alpha\Delta$;
11:    if $\|b^i_j\| > \boldsymbol{r}^i_j$ then
12:       $b^i_j = \boldsymbol{r}^i_j$;
13:    end if
14:    $dis2 = \frac{1}{2}\|\boldsymbol{y} - (\overline{\boldsymbol{x}_i} + P^i\boldsymbol{b}^i)\|^2$;
15:    if $dis2 < \varepsilon$ or $\|dis - dis2\| < \varepsilon$ then
16:       return $dis2$;
17:    end if
18: end while
```

gradient which is completed in line 9 for current $\boldsymbol{b}^i$. The iterative process for finding the best $\boldsymbol{b}^{i*}$ is from line 8 to line 18. Lines 11–13 limit the range of $\boldsymbol{b}^i$, i.e. $\|b^i_j\| \leq 3\sqrt{\lambda^i_j}$. Parameter α in line 10 is the step size. If its value is very small, the algorithm will converge slowly. Our objective function is a strictly downward convex function, so we do not need a small step size. In our experiments, α is set to 0.5 and only after a few iterations, we can get the minimum distance.

For each class, we can get a distance from the objective function. If $\boldsymbol{y}$ is an instance of this class, the function value must be smaller than other class. After we find the best parameter vector $\boldsymbol{b}^{i*}$ to f for each class and the label of $\boldsymbol{y}$ is obtained by:

$$label(\boldsymbol{y}) = \arg\min_i f(\boldsymbol{b}^{i*}) \qquad i \in C \tag{5}$$

3.4 Effectiveness and Efficiency

In PSM, the model is generated by the principal shapes extracted from the training set. So the ability of these principal shapes is very important for our model. If the training data is skewed, the principal shapes gained by this training set can not express the variation of the test time series. In this case, the accuracy of PSM decreases like other methods.

For the first step of our method, we get the eigenvalues and eigenvectors of each class by using SVD. It takes $O(min\{mn_i{}^2,\, m^2n_i\})$ time, where m is the length of the time series, n_i is the number of time series in i-th class of training set. In Algorithm 1, the parameter matrix A requires $O(km)$ computation time where k is the number of the selected principal shapes and $k \leq min(m, n_i)$. And B needs a matrix multiplication, which requires $O(k^2m)$ computation time. In each iterative process of fitting the model, the most time-consuming step is the distance calculation for a new $\boldsymbol{b}$. It take $O(km)$ time. So the Algorithm 1 needs $O(max\{k^2m, kmt\})$ time in total where t is the the number of iterations. Usually, we only use a small number of principal shapes to build our model, so k is very small. As we mentioned above, t is also very small because the objective function f is strictly convex. Thus, the total time complexity is $O(min\{mn_i{}^2, m^2n_i\}+ max\{k^2m, kmt\})$.

4 Experiments and Results

4.1 Datasets and Baselines

In order to test the classification performance of our PSM method, we first perform the experiments on the commonly used UCR (See footnote 1) and UEA[3] datasets (called *common datasets* in this paper) as other literatures [6,8,9,17]. They provide diverse characteristics with different lengths and number of the classes and different numbers of time series instances and Table 1 gives the details. We use the default train and test data splits, which is the same as the baselines.

Table 1. *Common datasets* used in the experiments

Name	#Train/test	Length	#Classes	Name	#Train/test	Length	#Classes
Adiac	390/391	176	37	Lighting7	70/73	319	7
Beef	30/30	470	5	MedicalImages	381/760	99	10
Chlorine.	467/3840	166	3	MoteStrain	20/1252	84	2
Coffee	28/28	286	2	MP_Little	400/645	250	3
Diatom.	16/306	345	4	MP_Middle	400/645	250	3
DP_Little	400/645	250	3	Herring	64/64	512	2
DP_Middle	400/645	250	3	PP_Little	400/645	250	3
DP_Thumb	400/645	250	3	PP_Middle	400/645	250	3
ECGFive.	23/861	136	2	PP_Thumb	400/645	250	3
FaceFour	24/88	350	4	SonyAIBO.	20/601	70	2
Gun_Point	50/150	150	2	Symbols	25/995	398	6
ItalyPower.	67/1029	24	2	Synthetic.	300/300	60	6

To verify the performance of our method on large and long time series datasets, we use extra 8 datasets from UCR archives (see footnote 1) (called *large datasets*) to do our experiments. The details are shown in Table 2.

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

Table 2. *Large datasets* for experiments

Name	#train/test	Length	#classes	Name	#train/test	Length	#classes
CinC_ECG_torso	40/1380	1639	4	MALLAT	55/2345	1024	8
ECG5000	500/4500	140	5	NonInvas.1	1800/1965	750	42
FordB	810/3636	500	2	NonInvas.2	1800/1965	750	42
HandOutlines	370/1000	2709	2	wafer	1000/6174	152	2

We compare our method with other 8 baselines. One is 1-NN classifier with DTW which has been proved it is very difficult to beat [1]. The other methods are: standard shapelet-based classifier with information gain (IG) [21], Kruskal-Wallis statistic (KW) and F-statistic (FS) [8]; The Fast-shapelets algorithm with the help of SAX representation (FSH) [17]; Learning Time Series which is the state-of-the-art algorithm for accuracy (LTS) [6]; Shapelet-transform algorithm which transforms the shapelets into feature vectors for the first time (IGSVM) [8] and FLAG which is known as the fastest shapelet-based algorithm [9]. All experiments are performed on a computer with intel i7 CPU and 16GB memory.

4.2 Accuracy and Running Time on *Common Datasets*

We first compare our method against the selected baselines in terms of classification accuracy and running time on *common datasets* of Table 1. The results are shown in Tables 3 and 4 respectively and the best method of each dataset is highlighted in **bold**. The symbol "-" denotes that we cannot get the method's result in a reasonable time (over 24 h).

To tell the significant difference in accuracy of different methods of Table 3, a non-parametric Friedman test based on ranks [4] described as a critical difference diagram is employed. Figure 3 shows the results on *common datasets*. The horizontal line, called *cliques*, denotes that the related methods has no significant difference [4]. As can be seen, PSM has a better classification accuracy than most baselines. Although PSM has a slight low accuracy than LTS and FLAG, there is no significant difference in rank with these two methods based on the *cliques* of critical difference diagram recommend.

Time-consuming is another evaluation criterion for different methods. Based on the running time shown in Table 4, PSM is fastest on most datasets and it is 3–4 orders of magnitude faster than most baselines. Recall that LTS has the best accuracy, but it is much slower than PSM in all datasets.

In general, our method PSM is fastest and outperforms almost all of baselines according to the running time while it has comparable accuracy. More precisely, PSM has a clear higher accuracy and less time consuming than NNDTW, IG, KW, FS and FSH methods in terms of the Friedman test and running time. When compared with IGSVM, PSM also has a slight accuracy advantage while it is 3–4 orders of magnitude faster on most datasets. LTS and FLAG is better than PSM on classification accuracy, but they are slow, especially that LTS takes much more running time for slightly larger datasets.

Table 3. Classification accuracy (%) on *common datasets*. The best method for each dataset is highlighted in **bold**.

Name	NNDTW	IG	KW	FS	FSH	IGSVM	LTS	FLAG	PSM
Adiac	58.8	29.9	26.2	15.6	57.5	23.5	51.9	**75.2**	72.4
Beef	56.7	50	33.3	56.7	50	**90**	76.7	83.3	83.3
Chlorine.	62.7	58.8	52	53.5	58.8	57.1	73	76	**86.1**
Coffee	96.4	96.4	85.7	**100**	92.9	**100**	**100**	**100**	**100**
Diatom.	95.8	76.5	62.1	76.5	87.3	93.1	94.2	**96.4**	93.5
DP_Little	47.9	-	-	-	60.6	66.6	**73.4**	68.3	61.1
DP_Middle	62	-	-	-	58.8	69.5	**74.1**	71.3	71
DP_Thumb	57.5	-	-	-	63.4	69.6	**75.2**	70.5	74.1
ECGFive.	78.9	77.5	87.2	99	99.8	99	**100**	92	95.7
FaceFour	84.1	84	44.3	75	92	**97.7**	94.3	90.9	83
Gun_Point	92	89.3	94	95.3	94	**100**	99.6	96.7	90.7
ItalyPower.	94.7	89.2	91	93.1	91	93.7	95.8	94.6	**97.3**
Lighting7	78.1	49.3	48	41.1	65.2	63	**79**	76.7	71.2
MedicalImages	**77.2**	48.8	47.1	50.8	64.7	52.2	71.3	71.4	64.9
MoteStrain	87.3	82.5	84	84	83.8	88.7	**90**	88.8	87.2
MP_Little	64.7	-	-	-	56.9	70.7	**74.3**	69.3	71
MP_Middle	63.1	-	-	-	60.3	76.9	**77.5**	75	74.6
Herring	54.7	**67.2**	60.9	57.8	60.9	64.1	59.4	64.1	65.6
PP_Little	63.1	-	-	-	57.6	**72.1**	71	67.1	71.2
PP_Middle	61.6	-	-	-	61.6	**75.9**	74.9	73.8	74
PP_Thumb	56.7	-	-	-	55.8	**75.5**	70.5	67.4	71.6
SonyAIBO.	71	85.7	72.7	**95.3**	68.6	92.7	91	92.9	87.9
Symbols	94.4	78.4	55.7	90.1	92.4	84.6	**94.5**	87.5	92.9
Synthetic.	98.7	94.3	90	95.7	94.7	87.3	97.3	**99.7**	97.3

4.3 Accuracy and Running Time on *Large Datasets*

We now test the performance of our method PSM on *large datasets*. In this experiments, we discard the comparison with IG, KW, FS and IGSVM methods because these methods are too slow to get the results in a reasonable time on these datasets. The results of PSM and other baselines are shown in Table 5 and Figure 4 gives the critical difference diagram.

Unlike the results on *common datasets*, PSM not only has the least time-consuming but also gets the best classification accuracy than other baselines on these large datasets. The reason is that the model built by PSM greatly depends on the training set. If the training set gives a good variation representation of the whole dataset, PSM can get a higher classification accuracy. For these large

Table 4. Running time (in seconds) on *common datasets*. The best method for each dataset is highlighted in **bold**

Name	NNDTW	IG	KW	FS	FSH	IGSVM	LTS	FLAG	PSM
Adiac	11.5	3287	1349	1513	288	706	80596	2.78	**1.43**
Beef	0.4	471	484	576	154	435	414	1.15	**0.03**
Chlorine.	148	9751	3213	3050	617	1181	556	**6.88**	10.51
Coffee	0.1	22.3	21.8	21.9	16.5	15.9	76	0.13	**0.01**
Diatom.	1.2	9.3	8.99	9.2	13.7	9.3	88	0.41	**0.28**
DP_Little	42.1	-	-	-	1131	9516	803	1.81	**0.6**
DP_Middle	43.2	-	-	-	1286	7041	6082	1.78	**0.45**
DP_Thumb	41.8	-	-	-	1105	11353	1006	3.14	**0.42**
ECGFive	1.1	19	18.4	18.6	4.6	11353	9.1	**0.08**	0.25
FaceFour	0.6	1021	1012	1010	75	410	116	0.26	**0.05**
Gun_Point	0.4	116.4	112	148.9	7.6	74.8	14.1	0.07	**0.04**
ItalyPower.	0.4	0.38	0.22	0.22	0.5	0.22	7.3	**0.03**	0.23
Lighting7	1.2	3442	3438	3584	307	1473	965	0.31	**0.17**
MedicalImages	9.2	4347	2625	2616	164	1547	2199	1.69	**1.01**
MoteStrain	0.6	1.41	1.29	1.29	1.4	0.84	6	**0.04**	0.29
MP_Little	42.1	-	-	-	1297	7394	5233	2.95	**0.38**
MP_Middle	42.9	-	-	-	1186	12102	724	2.23	**0.28**
Herring	2.1	17864	18166	17789	183	8536	264	1.25	**0.03**
PP_Little	42.2	-	-	-	1107	8142	4805	3.91	**0.45**
PP_Middle	43	-	-	-	1135	4753	3406	2.19	**0.34**
PP_Thumb	41.9	-	-	-	1172	8209	6638	4.2	**0.42**
SonyAIBO.	0.2	1.17	1.02	1.02	1.1	0.75	25.4	**0.04**	0.13
Symbols	7.5	3325	3318	3622	59.4	1263	528	0.85	**0.56**
Synthetic.	1.6	291	164	161	39.4	922	293	0.26	**0.25**

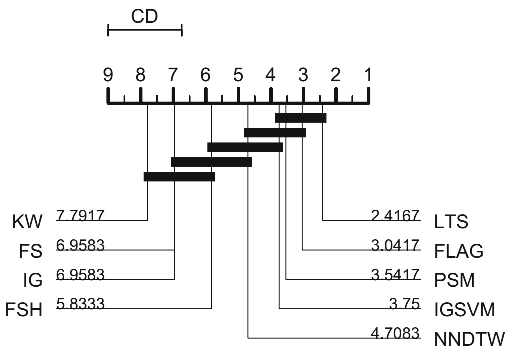

Fig. 3. Critical difference diagram for different methods on *common datasets*.

Table 5. Classification accuracy (%) and running time (s) on *large datasets*. The best method for each dataset is highlighted in **bold**

	Accuracy (%)					Time (s)				
Name	NNDTW	FSH	LTS	FLAG	PSM	NNDTW	FSH	LTS	FLAG	PSM
CinC_ECG.	75.2	56.5	69.9	91.1	**93.4**	483	1995	4911	34.69	**2.41**
ECG5000	92.8	92	93.7	91.9	**94.1**	1581	1681	13571	6.72	**3.22**
FordB	59	77.4	**89.8**	77.6	83.4	2121	8953	4757	95.68	**5.44**
HandOutlines	79.4	86.5	-	83.1	**86.7**	10382	11652	-	56.27	**0.75**
MALLAT	91.4	93.2	-	**96.2**	94.3	351	1042	-	11.99	**3.04**
NonInvasive.1	77.4	73.9	-	**93.6**	90.3	38602	47045	-	83.15	**33.57**
NonInvasive.2	84.8	75.9	-	**94.2**	93.1	3826	41836	-	79.62	**25.52**
wafer	98.6	**99.7**	**99.7**	99.2	99.5	371	214	396	11.31	**3.13**

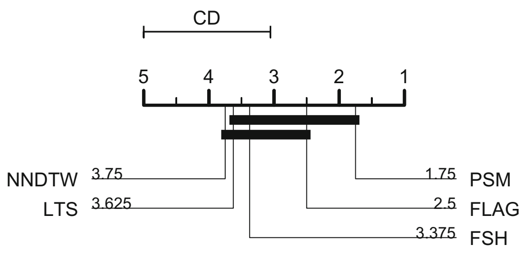

Fig. 4. Critical difference diagram for different methods on *large datasets*.

datasets, the abundant training samples basically contain the main variation of time series of datasets, so we get good results.

4.4 Scalability

In this section, we use the time series dataset *StraLightCurves* in the UCR archives (see footnote 1) to test the scalability of PSM. This dataset contains 9,236 starlight time series of length 1,024 and 3 types of star objects. The training set has 1,000 time series in default partition. Two key factors for testing the scalability are the number of training sample and the length of each sample.

We first fix the length to 1,024 and vary the number of training set from 100 to 1,000. In this case, IG, KW, FS and IGSVM methods are too slow to be run and LTS also cannot be run because of the large memory requirement. So we drop them from the comparison. We have only 4 results for NNDTW because it is also to slow when the number over 400. The running time and classification accuracy of different methods are shown in Fig. 5. As the number of training samples increasing, our method has a distinct advantage while other methods take much more running time, which has the same trends as Tables 4 and 5. One thing we should note that the time consuming of PSM is almost the same when the number of training samples increasing. The reason is that the principal shapes are extracted from the same size of covariance matrix when the training time series have the same length. This suggests that the time consumption of PSM is mainly related to the number of test samples.

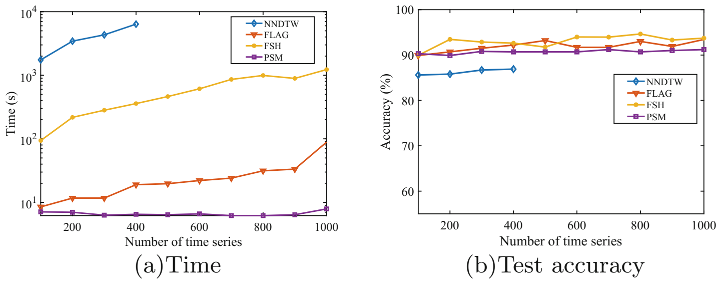

Fig. 5. Time and test accuracy when varying the number of training time series.

Now, we test the effect of the length of time series for different methods. In this experiment, the number is fixed to 1,000 and the length is varied from 100 to 1,024. When length over 300, LTS cannot be run in our computer for the memory constraint and we only get 3 results. For NNDTW, it is too slow when length over 800. Figure 6 gives the results. PSM is still fastest and it is almost linear increase with the length of time series. It's worth noting that PSM can still get a good accuracy when the length is very short while LTS, FSH and FLAG have low accuracy.

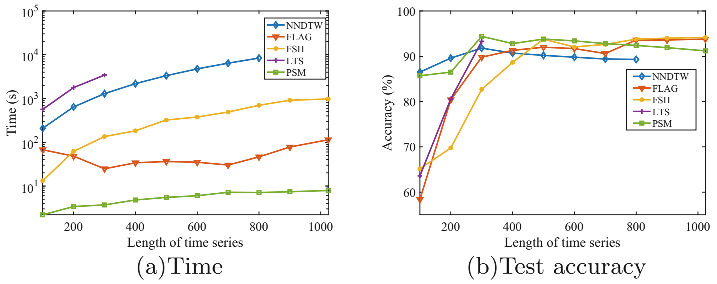

Fig. 6. Time and test accuracy when varying the length of time series.

5 Conclusion and Future Work

In this work, we propose a statistical method, PSM, to find the principal shapes of a time series dataset. With the principal shapes of each class, we can predict the class information of unlabeled time series quickly. Experimental results show that PSM is faster than other time series classification methods while it gains comparable classification accuracy compared with the state-of-the-art method on the commonly used time series datasets. Moreover, our method gets highest accuracy on large time series datasets, which fully demonstrates that the extracted principal shapes represent the intrinsic shape variation effectively when we have sufficient training data. Note that we still employ ED as the metric and do not consider the phase shift in time dimension. We propose to consider this situation in the future.

Acknowledgements. This work was supported by the National 863 Program of China [grant numbers 2015AA015401]; Research Foundation of Ministry of Education and China Mobile [grant number MCM20150507].

References

1. Batista, G.E., Wang, X., Keogh, E.J.: A complexity-invariant distance measure for time series. In: SDM, vol. 11, pp. 699–710. SIAM (2011)
2. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
3. Chang, K.W., Deka, B., Hwu, W.M.W., Roth, D.: Efficient pattern-based time series classification on GPU. In: 2012 IEEE 12th International Conference on Data Mining (ICDM), pp. 131–140. IEEE (2012)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(Jan), 1–30 (2006)
5. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB Endowment **1**(2), 1542–1552 (2008)
6. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 392–401. ACM (2014)
7. He, Q., Dong, Z., Zhuang, F., Shang, T., Shi, Z.: Fast time series classification based on infrequent shapelets. In: 2012 11th International Conference on Machine Learning and Applications (ICMLA), vol. 1, pp. 215–219. IEEE (2012)
8. Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. Data Min. Knowl. Disc. **28**(4), 851–881 (2014)
9. Hou, L., Kwok, J.T., Zurada, J.M.: Efficient learning of timeseries shapelets. In: Thirtieth AAAI Conference on Artificial Intelligence, AAAI, pp. 1209–1215 (2016)
10. Jeong, Y.S., Jeong, M.K., Omitaomu, O.A.: Weighted dynamic time warping for time series classification. Pattern Recogn. **44**(9), 2231–2240 (2011)
11. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Min. Knowl. Disc. **7**(4), 349–371 (2003)
12. Keogh, E., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowl. Inf. Syst. **7**(3), 358–386 (2005)
13. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing Sax: a novel symbolic representation of time series. Data Min. Knowl. Disc. **15**(2), 107–144 (2007)
14. Mueen, A., Keogh, E., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1154–1162. ACM (2011)

15. Petitjean, F., Forestier, G., Webb, G.I., Nicholson, A.E., Chen, Y., Keogh, E.: Dynamic time warping averaging of time series allows faster and more accurate classification. In: 2014 IEEE International Conference on Data Mining (ICDM), pp. 470–479. IEEE (2014)
16. Baldock, R., Tim, C.: Model-Based Methods in Analysis of Biomedical Images. Oxford University Press, Oxford (1999)
17. Rakthanmanon, T., Keogh, E.: Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the Thirteenth SIAM Conference on Data Mining (SDM), pp. 668–676. SIAM (2013)
18. Ratanamahatana, C.A., Keogh, E.: Making time-series classification more accurate using learned constraints. In: Proceedings of the 2004 SIAM International Conference on Data Mining, pp. 11–22. SIAM (2004)
19. Ratanamahatana, C.A., Keogh, E.: Three myths about dynamic time warping data mining. In: Proceedings of SIAM International Conference on Data Mining (SDM05), pp. 506–510. SIAM (2005)
20. Ueno, K., Xi, X., Keogh, E., Lee, D.J.: Anytime classification using the nearest neighbor algorithm with applications to stream mining. In: Sixth International Conference on Data Mining, ICDM 2006, pp. 623–632. IEEE (2006)
21. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 947–956. ACM (2009)