Knowledge and Information Systems (2022) 64:143–174
https://doi.org/10.1007/s10115-021-01630-z

REGULAR PAPER

Random pairwise shapelets forest: an effective classifier for time series

Jidong Yuan[1] · Mohan Shi[2] · Zhihai Wang[1] · Haiyang Liu[1] · Jinyang Li[3]

Received: 15 April 2019 / Revised: 15 November 2021 / Accepted: 20 November 2021 /
Published online: 10 January 2022

Abstract
Shapelet is a discriminative subsequence of time series. An advanced shapelet-based method is to embed shapelet into the accurate and fast random forest. However, there are several limitations. First, random shapelet forest requires a large training cost for split threshold searching. Second, a single shapelet provides limited information for only one branch of the decision tree, resulting in insufficient accuracy. Third, the randomized ensemble decreases comprehensibility. For that, this paper presents Random Pairwise Shapelets Forest (RPSF). RPSF combines a pair of shapelets from different classes to construct random forest. It omits threshold searching to be more efficient, includes more information about each node of the forest to be more effective. Moreover, a discriminability measure, Decomposed Mean Decrease Impurity, is proposed to identify the influential region for each class. Extensive experiments show that RPSF is competitive compared with other methods, while it improves the training speed of shapelet-based forest.

Keywords Time series classification · Pairwise shapelets · Random forest · Decomposed mean decrease impurity

1 Introduction

Time series is being produced every day and everywhere in real world, such as ECG recordings, financial data, industrial observations, etc. Time series classification (TSC) is an important subject in the field of data mining. Unlike general classification tasks, it takes attribute order into account. Recent studies have shown that the 1NN with Dynamic Time Warping (DTW) remains among the most competitive classification approaches [11,47,48].

✉ Jidong Yuan
yuanjd@bjtu.edu.cn

1 School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

2 Beijing Jingdong 360 Degree E-Commerce Co., Ltd., Beijing, China

3 Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong

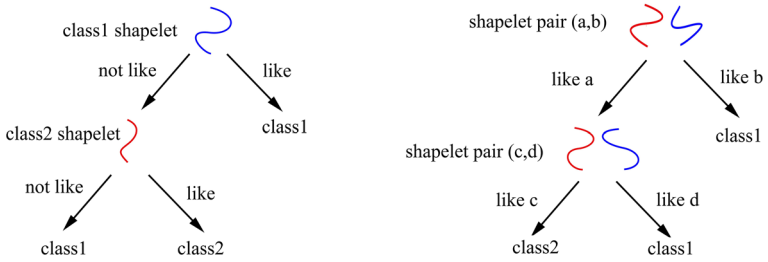

Fig. 1 *Left* Classic shapelet tree; *right* Pairwise shapelets tree

However, this method has drawbacks of high classification time complexity and lacks of interpretability.

Shapelet is the most discriminant, phase independent subsequence in time series [45]. It is interpretable and fast in the classification stage. Since random forest can achieve good performance through integrating a series of modest classifiers [7], the shapelet-based random forest has attracted significant attention and research effort recently. For example, Karlsson et al. [23] introduced a random shapelet forest. It selects both training instances and shapelet candidates randomly. In order to identify important features, a contribution metric Mean Decrease Impurity (MDI) was proposed [21]. Random shapelet forest has also been extended to multivariate time series forest, applied successfully to ECG classification [20] and early classification problem [22]. In addition, Deng et al. [10] introduced a combination of entropy and distance measure to evaluate the node split in the forest. Cetin et al. [8] proposed a shapelet discovery technique that allows efficient candidate evaluation in multivariate time series forest.

However, some limitations exist in the shapelet forest. First, a single shapelet often cannot provide enough information to distinguish different classes. Second, a time-consuming split threshold searching is needed to evaluate candidate shapelet. Third, randomization and ensemble lead to interpretation declining easily.

For these challenges above, this paper proposes a comparison-based ensemble algorithm that combines a pair of shapelets in decision tree node, called **R**andom **P**airwise **S**hapelets **F**orest (RPSF).[1] Figure 1 compares classic shapelet tree structure with the proposed base model in a simple binary classification task.[2] In the left sub-figure, 'like' or not is measured by the distance between instance D and shapelet S. However, the pairwise shapelets tree (as shown in the right sub-figure) asks whether instance D is closer to shapelet S_1 or shapelet S_2. It is much easier for human beings to give feedback when we do not have enough domain knowledge about the representation of time series data or the distances between data points [14].

RPSF provides more information by combining a pair of shapelets from different classes, which enhances the diversity of ensemble model. In addition, we present an effective metric for identifying influential data series regions (or subsequences) for a specific class. Our main contributions are as follows.

– The proposed pairwise shapelets tree extracts discriminant features from two different classes, and each tree node is split according to subsequence distances between instances and the two shapelets. This idea sharpens the contrast between the different two classes.

[1] An earlier version of the RPSF was presented with a limited empirical evaluation in [41].

[2] They could be applied on multi-class time series directly.

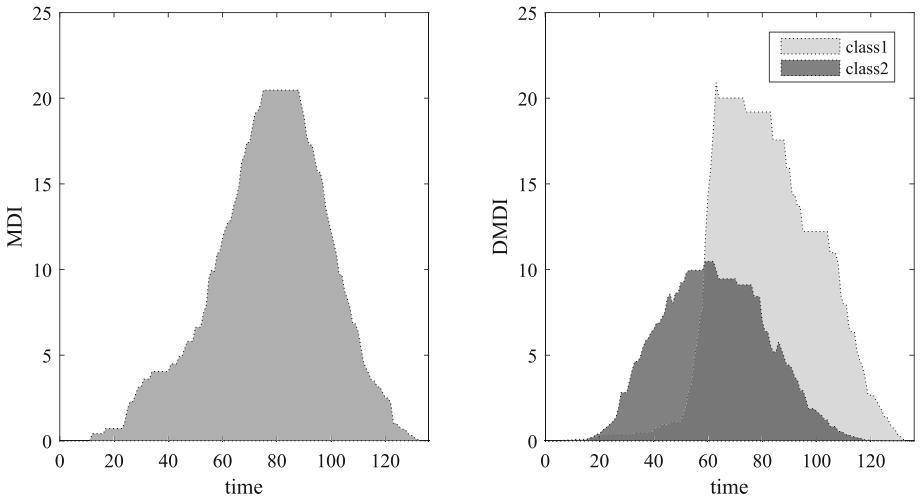

Fig. 2 *Left* Mean Decrease Impurity (MDI); *right* Decomposed Mean Decrease Impurity (DMDI). The right one indicates important features for both classes of ECGFiveDays dataset

Recent studies also explain the effectiveness and efficiency of comparison-based random forest framework theoretically and practically [14,36].

– RPSF no longer needs to find the split threshold, which saves computing resources. This is especially true when introducing entropy early pruning to speed up [45] (constantly evaluating shapelet under limited information and abandoning apparent inadequate candidates in advance).
– The proposed importance measure, Decomposed Mean Decrease Impurity (DMDI), could highlight which parts of the time series contributed the most to a certain class. Figure 2 shows results of MDI and DMDI on ECGFiveDays dataset, both of them point to similar subsequences (the left sub-figure and the class1 of the right sub-figure), while the DMDI indicates additional discriminant interval (the class2 of the right sub-figure) for the other class. This result seems to be in line with the medical conclusion in [33].

The remainder of our paper is organized by the following. In the next section, we will discuss about related works. Required preliminaries are described in Sect. 3. Section 4 presents RPSF algorithm in detail. The DMDI method is explained in Sect. 5. In Sect. 6, the experimental setup and results from the empirical investigation are discussed. Finally, main contributions are summarized and future work is discussed in Sect. 7.

2 Related work

According to the different characteristics of TSC algorithms, we group them as shapelet-based, interval-based, dictionary-based, similarity-based and deep learning methods.

Shapelet-based TSC methods focus on local features that are interpretable and discriminative. They could be divided into two large categories. The first category integrates shapelet selection within the process of constructing decision tree(s). Except for the shapelet-based random forest discussed above, researchers also proposed several single tree-based algorithms, such as the logical-shapelets [32], Symbolic Aggregate approXimation (SAX)

representation for fast shapelet discovery [33] and random-shapelet that extracts shapelet to build decision tree randomly [34]. In order to reduce the high time complexity of shapelet discovery, methods based on heuristic gradient decent [13], numerical optimization [16], and local fisher discriminant analysis [50] were introduced to learn discriminative features efficiently.

The second category disconnects the process of discovering shapelets from the classifier by adopting shapelets transformation. For example, Lines et al. [30] proposed the first method that maps the original time series into another data space, where discriminatory features can be detected easily. Then an alternative was presented to reduce similarities of selected shapelets [46], and a speed-up method was introduced to deal with multi-class TSC problems [6]. This idea has also been applied to ensemble classifiers successfully [2].

Interval-based TSC algorithms build classifiers by selecting one or more contiguous subsets of time series data points, or employing the summary measures of intervals. For instance, Time Series Forest (TSF) [10] summaries statistics of each interval as features, then builds random forests based on them. Both Time Series Bag of Features (TSBF) [5] and Learned Pattern Similarity (LPS) [4] extended the original TSF by introducing bag of features or changing the previous decision trees to regression ones.

Dictionary-based methods involve frequency counts of recurring patterns for classification. Bag-of-patterns (BOP) representation [28] is a dictionary classifier that employs SAX to form count histograms. SAX and Vector Space Model (SAXVSM) [39] is an extension of BOP that includes the vector space model commonly used in information retrieval. Classic Fourier coefficients that combined with the BOP model are considered as the state-of-the-art algorithm for TSC [37,38].

Different from previous introduced three types of methods, similarity-based methods deal with the whole series instead. DTW is a primary sequence alignment method, and there exist several ways to improve it. Traditionally, we can add constraints to prevent the matching path from being too distorted [17,35], try to increase the quality of alignment by including global/local weights [19,47], first-order derivatives [24], local shape features [52], and correction factors [3]. Apart from these methods for sequence alignment, researchers also proposed other measures like Time Warping Edit (TWE) [31], Move-Split-Merge (MSM) [42], and so on.

Convolutional neural networks (CNN) that play an important role in the computer vision community has also been widely considered for TSC problems recently. For example, Time LeNet (t-LeNet) was presented first to apply CNN on time series [26]. Then, a multi-scale CNN (MCNN) is proposed to train CNN based on extracted subsequences [9]. Moreover, classic methods like multi-layer perceptron (MLP), fully convolutional neural network (FCN), and residual network (ResNet) are successfully applied on time series [44]. Zhao et al. [51] proposed the Time-CNN that employs mean squared error for adapting the characteristics of time series. Other neural networks include the time warping invariant Echo State Network [43] that directly uses the original time series for classification, and an encoder algorithm combining attention layers [40]. A more detailed review could be found in [12].

3 Definition and notation

In this section, some relevant definitions of this study will be introduced. Table 1 summarizes the notations of this paper, and we expand on the definitions below.

Table 1 Symbol table

Symbol	Explanation
$\mathbf{D}$	Time series database
$T,\ D,\ X,\ Y$	Time series
$n, N, \lvert\mathbf{D}\rvert$	Number of time series
$m, \lvert X\rvert, q$	Length of time series
t, k	Time series attribute
c	Class items
$\lvert c\rvert$	Number of class items
$\mathbf{S}$	Candidate Shapelets pairs
(S_i, S_j)	Shapelets pair
ST	Shapelets tree
$\mathbf{R}$	Shapelets forest
l	Lower bound length of shapelet
u	Upper bound length of shapelet
p	Number of shapelet trees
r	Number of shapelets pairs
$dist, subdist$	Normalized distance between time series/subsequences
E	Entropy
I	Information gain
G	Separation gap
CV	Contribution value of shapelet
ER	Entropy reduction caused by shapelet

Suppose we have a set of n time series, $\mathbf{D} = \{T_1, T_2, \ldots, T_n\}$, where each time series has m ordered real-valued observations and a class value c, $T_i = \{t_{i1}, t_{i2}, \ldots, t_{im}, c_i\}$. The objective is to construct a function that maps the observations of a time series to a possible class value. Note that, we assume that all time series have the same length (and with fixed time intervals) for simplicity and computational consideration.

Definition 1 Time series subsequence and shapelet.

A time series subsequence $S_{i,q} = \{t_i, t_{i+1}, \ldots, t_{i+q-1}\}$ is a continuous subsequence of T starting at position i with length q, where $1 \leq i \leq m - q + 1$. Notice that data points $t_1, t_2, \ldots, t_m$ are arranged by temporal order with fixed time intervals. Time series subsequence could be considered as the local properties of a time series, while shapelet is one of the most discriminative subsequences.

Definition 2 Distance and subsequence distance.

For two given time series X and Y of the same length m, normally their differences are measured by the length-normalized Euclidean distance (As shown in Eq. 1). Note that, both X and Y must be z-normalized to make meaningful comparisons, and to avoid scale and offset variance.

$$dist(X, Y) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - y_i)^2}. \tag{1}$$

For the pairwise shapelets tree *ST*, it is required to calculate the similarity between a short subsequence and a much longer time series. For that, the shorter one must slide against the longer one to achieve the best possible alignment. The subsequence distance is defined as:

$$subdist(X, Y) = min(dist(X, Y_{|X|})), \tag{2}$$

where $Y_{|X|}$ represents subsequence of Y with length $|X|$, and what the function *subdist*() achieved is the minimum distance between two time series.

Definition 3 Shapelets pair.

A pair of shapelets is a tuple (S_1, S_2), where both S_1 and S_2 are shapelets. A shapelets pair divides the dataset $\mathbf{D}$ into two disjoint subsets $\mathbf{D}_1 = \{X : X \in \mathbf{D}, subdist(S_1, X) \leq subdist(S_2, X)\}$, and $\mathbf{D}_2 = \{X : X \in \mathbf{D}, subdist(S_1, X) > subdist(S_2, X)\}$. Shapelets pairs are used as tree nodes to construct decision trees and the random forest in this paper.

Definition 4 Entropy.

Assume that we have a dataset $\mathbf{D}$ of N time series from $|c|$ different classes, every class c_i has n_i ($n_1 + n_2 + \cdots + n_{|c|} = N$) labeled instances in the dataset. The entropy of $\mathbf{D}$ is

$$E(\mathbf{D}) = -\sum_{i=1}^{|c|} \frac{n_i}{N} \log(\frac{n_i}{N}). \tag{3}$$

Definition 5 Information gain.

The information gain of a shapelets pair (S_1, S_2) is

$$I_{(S_1,S_2)}(\mathbf{D}) = E(\mathbf{D}) - \frac{|\mathbf{D}_1|}{|\mathbf{D}|} E(\mathbf{D}_1) - \frac{|\mathbf{D}_2|}{|\mathbf{D}|} E(\mathbf{D}_2). \tag{4}$$

Although other quality measures are available, information gain is preferred to evaluate the discriminative power of each shapelets pair.

Definition 6 Separation gap.

The separation gap of a shapelets pair (S_1, S_2) is

$$G(S_1, S_2) = \frac{\frac{1}{|\mathbf{D}|} \sum_i |subdist(S_1, D_i) - subdist(S_2, D_i)|}{\sqrt{|S_1| + |S_2|}}. \tag{5}$$

It represents the average of differences between the subsequence distance of the left side of a tree node and that of the right side.

Note that, when we compare two pairs of shapelets, the one has greater information gain is selected. If their information gains are equal, the one has the maximum separation gap is chosen. If their separation gaps are also the same, we choose the relatively shorter one. However, if their lengths are equal, the one that appeared earlier is the answer. In other words, we break ties by maximizing the separation gap and minimizing the length of the shapelets pairs.

4 Random pairwise shapelets forest

RPSF is an ensemble model built by multiple pairwise shapelet-based trees, we will provide a detailed elaboration to the proposed algorithm in this section.

4.1 Intuition for pairwise shapelets

The shapelet-based decision tree makes prediction according to the subsequence distance between shapelet S and instance D. If D is similar to S (the subsequence distance is less than the split threshold) from a specific class, it is assigned to that class, or to other classes. However, similarity with features (shapelet) from only one class cannot accurately characterize discriminative information in the data. Just as each branch of a traditional decision tree indicates the corresponding split attribute value, it is meaningful to identify the intuition of each branch of the shapelet-based tree. Under this notion, we propose the idea of combining a pair of shapelets from different classes to construct a decision tree, so that 'like and not like' becomes 'like S_1 or like S_2' (as shown in Fig. 1). A decision can be made according to distances to the two shapelets in the pair rather than distance to a single one and a threshold. It improves the accuracy of the classifier due to the additional information. Besides, the diversity of the ensemble model can be enhanced since a pairwise combination has more possible candidates than single ones. In addition, the pairwise-based method has an advantage in terms of training time. The split threshold searching, which is time-consuming, is omitted because of the combination. This is especially true with the introduce of entropy early pruning, where the previously calculated distance information can be reused, but the split threshold, which is avoided by RPSF, must be recalculated each time.

4.2 Proposed algorithm

RPSF trains a set of trees by considering several parameters: the shapelet length interval $[l, u]$, the number of decision trees p, and the number of candidate shapelets pairs in tree node r. For the construction of each pairwise shapelets decision tree, $|\mathbf{D}|$ time series are sampled using the traditional Bootstrap approach. The main steps to learn the RPSF can be summarized as follows.

- Initially, $|\mathbf{D}|$ sampled instances are extracted from the training set $\mathbf{D}$ randomly.
- Shapelets are extracted randomly from two random different classes r times to form the candidate shapelets pairs $\mathbf{S}$. The lower bound length is l, while the upper is u.
- Estimate the best shapelets pair (S_1, S_2) as a split node according to information gain and separation gap, and it divides the original set into two parts, $\mathbf{D}_1$ and $\mathbf{D}_2$ separately.
- Recursive steps 2 and 3 until a random pairwise shapelets tree is built.
- Iterate steps 1 to 4 until p pairwise shapelets trees are built to construct a random forest.

Algorithm 1 RandomPairwiseShapeletsForest ($\mathbf{D}$, l, u, p, r)

Input: The training set $\mathbf{D}$, the lower shapelet length l, the upper shapelet length u, the number of shapelet trees p, and the number of inspected shapelets pair r
Output: Random Pairwise shapelets forest $\mathbf{R} = \{ST_1, ST_2, \ldots, ST_p\}$
1: **for** $i = 1$ to p **do**
2: $\quad \mathbf{D}_i \leftarrow$ BootstrapInstances($\mathbf{D}$)
3: $\quad ST_i \leftarrow$ RandomPairwiseShapeletsTree ($\mathbf{D}_i, l, u, r$)
4: $\quad \mathbf{R} \leftarrow \mathbf{R} \cup ST_i$
5: **end for**
6: **return** $\mathbf{R}$

Algorithms 1 - 4 describe the constructing process in detail. Algorithm 1 provides a brief overview of creating a random pairwise shapelets forest $\mathbf{R}$. First, bootstrap sample is

employed to pick instances from the training set **D** (line 2). Then, a random pairwise shapelets tree ST_i is built based on sampled $\mathbf{D}_i$ and the corresponding parameters (line 3, as shown in Algorithm 2). This process iterates until the number of needed trees is satisfied (lines 1–5).

Algorithm 2 RandomPairwiseShapeletsTree (**D**, l, u, r)

Input: The training set **D**, the lower shapelet length l, the upper shapelet length u and the number of inspected shapelets pair r
Output: Pairwise shapelets tree ST
1: **if** IsPure(**D**) **then**
2: **return** MakeLeaf(**D**)
3: **end if**
4: **S** $\leftarrow \varnothing$
5: **for** $i = 1$ to r **do**
6: **S** $\leftarrow$ **S**$\cup$ SampleShapeletsPair(**D**, l, u)
7: **end for**
8: $(S_1, S_2) \leftarrow$ BestShapeletsPair(**D**, **S**)
9: $(\mathbf{D}_1, \mathbf{D}_2) \leftarrow$Split(**D**, S_1, S_2)
10: $ST_L \leftarrow$ RandomPairwiseShapeletsTree ($\mathbf{D}_1, l, u, r$)
11: $ST_R \leftarrow$ RandomPairwiseShapeletsTree ($\mathbf{D}_2, l, u, r$)
12: **return** $ST \leftarrow ST_L \bigcup ST_R$

Algorithm 2 shows the process of building each pairwise shapelets tree. At the beginning, the creation of leaf nodes is determined according to the purity of data (if entropy < 0.1 or not) (lines 1–3). Then subsequences are extracted randomly for r times from two random classes to form a candidate set. To be more clear, we first select two random different classes, then two lengths and starting points are randomly chosen to form a pair of shapelets (lines 4–7). After that, we assess candidates and find the best pair (line 8, details in Algorithm 3 and 4). The best pair (S_1, S_2) splits the dataset **D** to two subsets $\mathbf{D}_1$ and $\mathbf{D}_2$. The distances between a training instance D_i and (S_1, S_2) are named as d_1, d_2. If d_1 is less than or equal to d_2, it means that D_i is closer to S_1, and it is added to $\mathbf{D}_1$, otherwise, added to $\mathbf{D}_2$ (line 9). Finally, the algorithm recursively calls itself on $\mathbf{D}_1$ and $\mathbf{D}_2$ to construct subtrees (lines 10–11).

The number of candidate shapelets in a dataset consisting of n instances of length m and possible shapelet candidate lengths interval $[l, u]$ is equal to

$$r = \left(\sum_{i=l}^{u} m - i + 1 \right) * n * percentage. \tag{6}$$

where $percentage$ means that a part of the candidates are sampled to build trees. For our method, the two shapelets must come from different classes, and the number of class labels $|c|$ and its corresponding instances vary a lot for different dataset. Here we assume that the number of instance for each class is the same, which will be $n/|c|$. Indeed, we have $C^2_{|c|}$ combinations. The number of possible candidate pairwise shapelets for each node should be

$$r = C^2_{|c|} \left[\frac{n}{|c|} \sum_{i=l}^{u} m - i + 1 \right]^2 * percentage. \tag{7}$$

It is clear that Eq. 7 cannot provide a stable value for experiment. When compared with gRSF, Eq. 6 is adopt for simplicity and fairness. Moreover, ignoring the subsequence distance consumption, the time complexity for each pairwise shapelets tree is $O(nrlogn)$, where $logn$

is the average depth of the decision tree. When p trees are built, the time complexity of RPSF will be $O(pnrlogn)$.

Algorithm 3 BestShapeletsPair (**D**, **S**)

Input: The training set **D**, candidate shapelets pairs **S**
Output: Best shapelets pair (S_1, S_2)
1: $Candidates \leftarrow \varnothing$
2: **for** $i = 1$ to $|\mathbf{S}|$ **do**
3: $(S_1^i, S_2^i) \leftarrow \text{Norm}(S_1^i, S_2^i)$
4: $(I_{(S_1^i, S_2^i)}(\mathbf{D}), G(S_1^i, S_2^i)) \leftarrow$ AssessCandidatePair (**D**, S_1^i, S_2^i)
5: $Candidates \leftarrow Candidates \bigcup (S_1^i, S_2^i, I_{(S_1^i, S_2^i)}(\mathbf{D}), G(S_1^i, S_2^i))$
6: **end for**
7: $Candidates \leftarrow$ sortByQuality($Candidates$)
8: $(S_1, S_2) \leftarrow$ Top($Candidates$)
9: **return** (S_1, S_2)

Algorithm 4 AssessCandidatePair (**D**, S_1, , S_2)

Input: Dataset **D**, candidate shapelets pair (S_1, S_2)
Output: Information gain $I_{(S_1,S_2)}(\mathbf{D})$, separation gap $G(S_1, S_2)$
1: $I_{(S_1,S_2)}(\mathbf{D}) \leftarrow 0$, $G(S_1, S_2) \leftarrow 0$, $\mathbf{D}_1 \leftarrow \varnothing$, $\mathbf{D}_2 \leftarrow \varnothing$
2: **for** i = 1 to $|\mathbf{D}|$ **do**
3: $d_1 \leftarrow subdist(S_1, D_i)$
4: $d_2 \leftarrow subdist(S_2, D_i)$
5: **if** $(d_1 \leq d_2)$ **then**
6: $\mathbf{D}_1 \leftarrow \mathbf{D}_1 \cup D_i$
7: **else**
8: $\mathbf{D}_2 \leftarrow \mathbf{D}_2 \cup D_i$
9: **end if**
10: **end for**
11: $I_{(S_1,S_2)}(\mathbf{D}) \leftarrow \text{InfoGain}(\mathbf{D}_1, \mathbf{D}_2)$
12: $G(S_1, S_2) \leftarrow \text{SepGap}(\mathbf{D}_1, \mathbf{D}_2)$
13: **return** $(I_{(S_1,S_2)}(\mathbf{D}), G(S_1, S_2))$

As shown in Algorithms 3 and 4, information gain and separation gap are typically used when shapelet assessment is needed. For a normalized shapelets pair (S_1, S_2) (Algorithm 3, line 3), we try to split training data by calculating the subsequence distance between training instance and (S_1, S_2) separately, and assigning each instance to its closer side (Algorithm 4, lines 2–10, similar with splitting process in the previous paragraph). When this process completed, information gain and separation gap can then be calculated to measure its quality (Algorithm 4, lines 11–12). A pair with greater information gain and separation gap is considered preferentially (Algorithm 3, lines 7–8). Entropy early pruning is also introduced in this process to abandon apparently inadequate candidates [45].

After training, each internal tree node consists of a shapelets pair (S_1, S_2) and the left, right subtree. The leaf node records the class value. To classify a test instance T, we begin from the root node. If the distance between T and S_1 is less than that of T and S_2, the left subtree is recursively used. Otherwise, the right one is traversed. The process repeatedly runs until it reaches a leaf node and gets a prediction. The final result is obtained by the majority

voting of p trees. Indeed, the time complexity of the test process is O($plogn$), where p is the ensemble size, $logn$ represents the average depth of the decision trees.

5 Decomposed mean decrease impurity

This method evaluates the importance of each time series attribute for each class and considers those with higher scores to contribute more for classification. Existing MDI provides only a global score, which is determined by its tree structure [8]. On the contrary, the proposed pairwise format allows us to identify discriminative regions of different classes.

We decompose the information gain of each node according to shapelet's contribution (Eq. 9), then add it to attributes that form the shapelet. For shapelets pair (S_1, S_2), if the dataset attracted by shapelet S_1 causes greater entropy reduction (Eq. 10), it is assumed that S_1 contributes more than S_2. The contribution of shapelet for different classes should be accumulated, respectively. Based on the above idea, we define the **D**ecomposed **M**ean **D**ecrease **I**mpurity (DMDI).

Given a pairwise shapelet forest $\mathbf{R} = \{ST_1, ST_2, \ldots, ST_p\}$, where ST is a pairwise shapelet tree. Each node of the tree corresponds to a shapelets pair (S_1, S_2). Given a training set $\mathbf{D}$ with series length m, for time series attribute k and class c, DMDI(k, c) is defined as follows.

$$\begin{aligned} DMDI(k,c) = \sum_{p}\Big(\sum_{node}{}_{(k\in S_1 \wedge class(S_1)=c)} CV(node, S_1) \\ + \sum_{node}{}_{(k\in S_2 \wedge class(S_2)=c)} CV(node, S_2)\Big) \end{aligned} \tag{8}$$

where CV is the contribution value of one shapelet. It is obtained from the decomposition of the total information gain of the node. Let the input dataset to a node be $\mathbf{D}_0$, the datasets obtained by dividing the $\mathbf{D}_0$ are $\mathbf{D}_1$, $\mathbf{D}_2$, then

$$CV(node, S_i) = \frac{ER(node, S_i)}{ER(node, S_1) + ER(node, S_2)} * I_{(S_1,S_2)}(\mathbf{D}_0) \tag{9}$$

where $I_{(S_1,S_2)}(\mathbf{D}_0)$ is the information gain of the tree node, and

$$ER(node, S_i) = \begin{cases} E(\mathbf{D}_0) - E(\mathbf{D}_i), & if E(\mathbf{D}_0) > E(\mathbf{D}_i); \\ 0, & otherwise. \end{cases} \tag{10}$$

where $E(\mathbf{D})$ is the entropy of $\mathbf{D}$, i equals to 1 or 2, $ER(node, S_i)$ is the entropy reduction caused by a shapelet. We cannot guarantee that ER is positive. It is set to zero in a negative case. If the two terms in the denominator are both zero, the node is discarded.

For every class, DMDI searches all nodes that embed a shapelet from it, decomposes the information gain of these nodes, and adds the contribution of shapelet to attributes forming it. Eventually, we recognize which attributes in the sequence contribute more for a particular class.

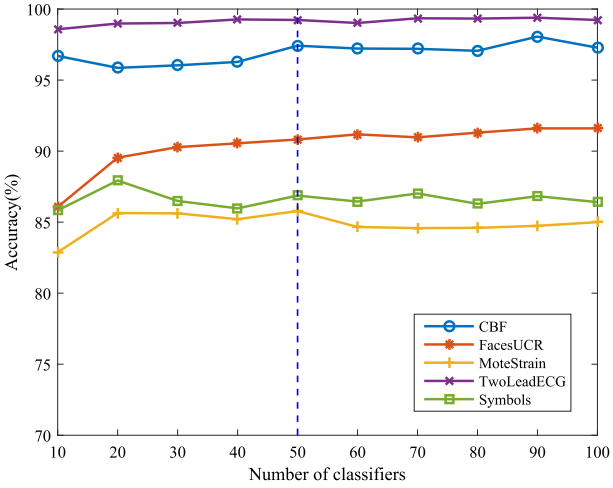

Fig. 3 Accuracy of RPSF when the number of classifiers p varies from 10 to 100

6 Experiment and evaluation

In this part, we first describe the datasets used to conduct our experiments, and then outline the parameter setting, before evaluating the performance of the proposed RPSF algorithm in terms of accuracy and time consumption.[3]

6.1 Datasets and parameter settings

Experiments are conducted on UCR/UEA datasets[4] that are widely used in studies on a 3.40 GHz Intel Core i7-6700 PC machine with 16 Gigabytes main memory. In order to build the forest and due to the computational reason, we discard datasets that cannot finish 10 times running in 120 hours. In addition, the predefined splits are employed for experiments. The RPSF model is built on bootstrapped training set, and its accuracy is recorded on the test set. For the proposed RPSF, there are three factors that strongly determine its efficiency and effectiveness, the number of pairwise shapelets trees p, the minimum and maximum length of shapelets (represented by l and u separately), and the number of random candidate shapelets pairs r for each split node. Note that, for the parameter validation, five datasets with comparatively larger test instances are chosen. We tried to select parameters based on the out-of-bag (OOB) error rate in the beginning. However, the OOB error rates we got are mostly equal to 0, no matter how we vary the ensemble size, the length of shapelets or the percentage of shapelets pairs. To illustrate the effect of parameters better, we evaluate them on the sampled training set, and report the error/accuracy on the test set directly.

We initially studied the impact of the number of integrated models p on the classification accuracy, increasing from 10 to 100. As shown in Fig. 3, the minimum and maximum length ratio (compared with the whole length m) is set to 0.25 and 0.67 independently, and r is set to 1% of all candidate shapelets.

[3] The source code of RPSF is available on our website https://github.com/nephashi/RandomPairwiseShapeletsForest.

[4] http://www.timeseriesclassification.com.

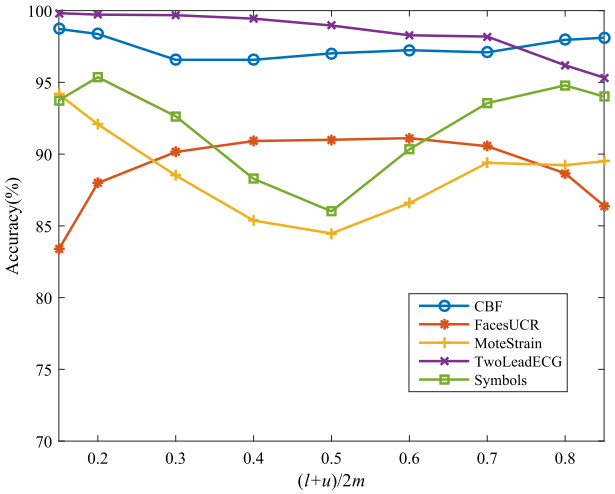

Fig. 4 Accuracy of RPSF when the shapelet length interval varies

It is widely accepted that the ensemble averaging reduces the variance of base models and improves the generalization behavior. Figure 3 is in accordance with this theory, and all the accuracy curves show an upward trend or fluctuate in a small range with the rise of p. However, since the time consumed increases linearly, we need to find a trade-off between accuracy and computation. It is visible that the improvement of accuracy after $p = 50$ is not obvious (e.g., datasets TwoLeadECG and Symbols), $p = 50$ is set as the real experimental parameter for the following experimentations thus accordingly.

For the shapelet lengths l and u, we fix the interval to 30% of the time series length, that is, $(u - l)/m = 0.3$, and the average of these lengths (e.g., $(l + u)/2m$) keep changing from 0.15 to 0.85 when sliding the window. Note that the 30% length interval is relatively small, which makes the fluctuation of accuracy more sensitive to the sliding window. r is set to 1% of all candidate shapelets, and $p = 50$ for this analysis.

The experimental results are shown in Fig. 4. The curves present several different trends. For datasets that are similar globally, and small local dissimilarities exist among different classes (e.g., CBF, Symbols, and MoteStrain), their accuracies decrease at first, and then climb up. The performance of TwoLeadECG keeps decreasing with the rise of average length, indicating that its two classes could be easily distinguished by local features. For dataset FacesUCR, it is lower at both ends and higher in the middle, meaning that relative long or short searching space cannot guarantee acceptable accuracy. Indeed, getting the optimal parameters of length depends on different personalities of datasets.

For the following experiment, we increase the number of candidate shapelets from 0.2% to 3% of the total extracted ones, observing the classification accuracy under different conditions. The ratio of shapelet interval is set as [0.25, 0.67], and the number of pairwise shapelets trees is set to 50.

The number of pairwise shapelets r represents the randomness of the ensemble model. The smaller the r, the greater the randomness of the model RPSF. If only one pair of shapelets is extracted, the algorithm is completely random. If all possible shapelet pairs are considered, the model is totally deterministic. Figure 5 depicts the experimental results. Accuracy curves show that our model is robust to the number of shapelets. The reason may lie in that the random strategy avoids over-fitting, and the ensemble improves generalization.

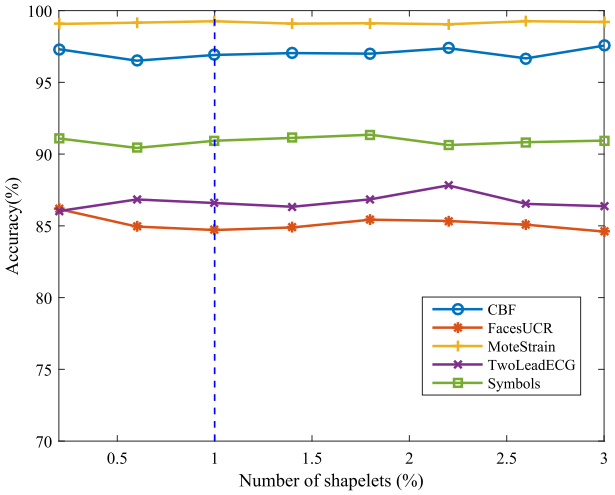

Fig. 5 Accuracy of RPSF when the number of selected shapelets for splitting varies from 0.2 to 3 (in percentage)

In a nutshell and for simplicity, shapelet length interval of gRSF and RPSF are set to 25% to 67% of the total length of corresponding time series, which covers a larger range and is a relatively safe value. The number of decision trees is set to 50. r is set to 1% of the possible candidate shapelets. Besides, the results are achieved on an average of ten runs.

6.2 Predictive performance

In this section, we demonstrate that RPSF is competitive in terms of classification accuracy contrast to state-of-the-art algorithms. First, the proposed method is compared with other local feature-based methods. Second, comparisons with other distance-based (global similarity) approaches are presented. Last but not least, RPSF is compared with other ensemble and deep learning alternatives.

6.2.1 Compared with shapelet-based/dictionary-based/interval-based benchmark classifiers

Several benchmark algorithms, from shapelet-based or dictionary-based to interval-based ones, are used for comparisons in this section. For example, FastShapelet (FS) refers to the decision tree algorithm proposed in [33] where getting the approximate shapelet quickly through SAX representation [27]. Learning time-series shapelets (LTS) is an algorithm proposed in [13] that searches shapelet by using optimization approach. BOP [28] and SAXVSM [39] are considered as the dictionary-based benchmarks. For forest-based classifiers, gRSF is the state-of-the-art shapelet-based random forest [23]. Interval-based classifiers, like TSF [10], TSBF [5] and LPS [4], also ensemble decision or regression trees for classification.

Figure 6 depicts the accuracy comparison peer-to-peer visually and areas below the diagonal line indicate that RPSF is better. W/D/L and the numerical numbers below represent the number of Win/Draw/Loss between the two compared methods. More details are shown

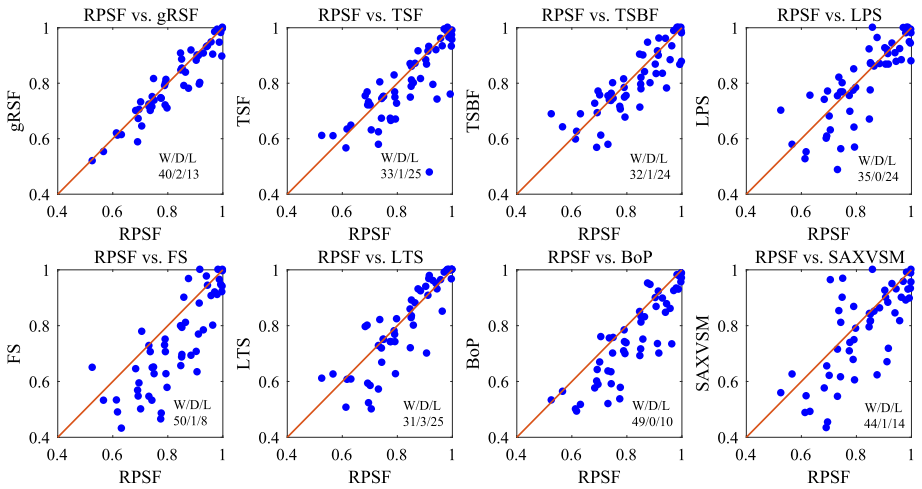

Fig. 6 Accuracy comparison between RPSF and other shapelet/dictionary/interval-based classifiers

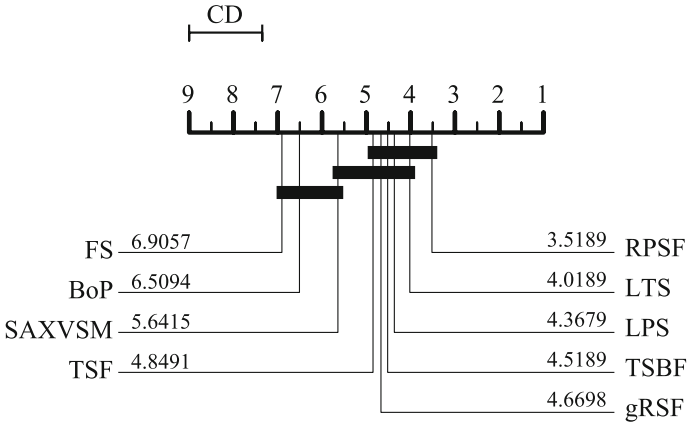

Fig. 7 Critical difference diagram for the average ranking comparison among RPSF and other shapelet/feature-based single/ensemble classifiers

in Table 2.[5] It is clear that our method shows outstanding performance. For the single shapelet/dictionary-based methods, RPSF beats FS, LTS, BOP and SAXVSM on 50, 31, 49, 44 out of 59 datasets, respectively. As we can see from the lower triangular area of Fig. 6, points far away from the diagonal line indicate that RPSF shows overwhelming superiority on these datasets. For example, it is 20.8% higher than the LTS on the OliveOil dataset, and 20.7% higher on the Wine dataset. When compared our method with shapelet/interval-based ensemble classifiers, it beats the gRSF on 40 out of 55 datasets and outperforms other interval-based forests on more than one half of the tested datasets.

Figure 7 gives the critical differences diagram for the accuracy of individual algorithms ($\alpha = 0.05$). RPSF is definitely the one with the lowest (best) ranking. It is also significantly better than FS, BoP, and SAXVSM. The data are not sufficient to conclude whether LTS,

[5] Some results of gRSF are not provided due to the huge required time consumption, while the reason for TSBF is the bug of the source code.

Table 2 Classification accuracy of RPSF compared to shapelet-based methods and other random forests

Dataset	RPSF	gRSF	TSF	TSBF	LPS	FS	LTS
Adiac	0.697 ± 0.014	0.704 ± 0.010	0.731	**0.770**	**0.770**	0.593	0.522
ArrowHead	0.799 ± 0.020	0.709 ± 0.019	0.726	0.754	0.783	0.577	**0.823**
Beef	0.710 ± 0.063	0.587 ± 0.083	0.767	0.567	0.600	0.567	**0.800**
BeetleFly	0.750 ± 0.058	**0.815±0.044**	0.750	0.800	0.800	0.650	0.750
BirdChicken	0.860 ± 0.046	0.790 ± 0.021	0.800	0.900	**1.000**	0.900	0.800
BME	0.847 ± 0.021	0.796 ± 0.012	0.940	–	0.787	0.680	**0.960**
Car	0.792 ± 0.026	0.720 ± 0.034	0.767	0.783	**0.850**	0.750	0.767
CBF	0.972 ± 0.009	0.984 ± 0.009	0.994	0.988	**0.999**	0.919	0.990
ChlorineConcentration	0.694 ± 0.007	0.671 ± 0.043	**0.720**	0.692	0.608	0.546	0.592
Coffee	0.989 ± 0.017	0.946 ± 0.023	0.964	**1.000**	**1.000**	0.964	**1.000**
Computers	0.703 ± 0.017	0.731 ± 0.018	0.720	**0.756**	0.680	0.500	0.584
CricketX	**0.778±0.009**	0.744 ± 0.002	0.664	0.705	0.697	0.485	0.741
CricketY	0.744 ± 0.007	0.714 ± 0.020	0.672	0.736	**0.767**	0.531	0.718
CricketZ	**0.776±0.010**	0.746 ± 0.018	0.672	0.715	0.754	0.464	0.741
DiatomSizeReduction	0.933 ± 0.019	0.795 ± 0.038	0.931	0.899	0.905	0.879	**0.967**
DistalPhalanxOutlineAgeGroup	**0.849±0.008**	0.846 ± 0.007	0.748	0.712	0.669	0.655	0.719
DistalPhalanxOutlineCorrect	0.795 ± 0.009	**0.812±0.009**	0.669	0.676	0.568	0.626	0.626
DistalPhalanxTW	0.793 ± 0.008	**0.800±0.008**	0.748	0.748	0.640	0.705	0.741
ECG200	0.866 ± 0.008	0.838 ± 0.014	0.870	0.840	0.860	0.810	**0.880**
ECGFiveDays	**1.000±0.000**	0.997 ± 0.002	0.956	0.877	0.879	0.995	**1.000**
FaceAll	0.752 ± 0.005	0.737 ± 0.006	0.751	0.744	**0.767**	0.626	0.749
FaceFour	0.998 ± 0.005	0.896 ± 0.057	0.932	**1.000**	0.943	0.920	0.966
FacesUCR	0.905 ± 0.011	0.881 ± 0.008	0.883	0.867	0.926	0.706	**0.939**
Fish	0.931 ± 0.010	0.907 ± 0.014	0.794	0.834	0.943	0.783	**0.960**

Table 2 continued

Dataset	RPSF	gRSF	TSF	TSBF	LPS	FS	LTS
GunPoint	0.999 ± 0.002	0.995 ± 0.002	0.973	0.987	0.993	0.940	**1.000**
Ham	0.745 ± 0.033	0.750 ± 0.032	0.743	**0.762**	0.562	0.648	0.667
Herring	0.568 ± 0.043	0.552 ± 0.036	0.609	**0.641**	0.578	0.531	0.625
InsectWingbeatSound	0.619 ± 0.005	0.611 ± 0.013	**0.633**	0.625	0.551	0.489	0.606
ItalyPowerDemand	0.959 ± 0.002	0.947 ± 0.003	0.960	0.883	0.923	0.906	**0.963**
LargeKitchenAppliances	**0.731±0.002**	–	0.571	0.528	0.717	0.560	0.701
Lightning2	0.738 ± 0.016	0.700 ± 0.068	0.803	0.738	**0.820**	0.705	**0.820**
Lightning7	0.685 ± 0.027	0.700 ± 0.014	0.753	0.726	0.740	0.644	**0.795**
Mallat	0.942 ± 0.022	0.930 ± 0.003	0.919	0.960	0.908	**0.976**	0.950
Meat	**0.965±0.005**	0.903 ± 0.018	0.933	0.933	0.883	0.800	0.850
MedicalImages	0.669 ± 0.008	–	**0.755**	0.705	0.746	0.624	0.664
MiddlePhalanxOutlineAgeGroup	0.790 ± 0.009	0.790 ± 0.014	**0.828**	0.814	0.773	0.729	0.780
MiddlePhalanxOutlineCorrect	**0.733±0.008**	0.723 ± 0.005	0.578	0.578	0.487	0.545	0.571
MiddlePhalanxTW	0.615 ± 0.006	**0.619 ± 0.030**	0.565	0.597	0.526	0.532	0.506
MoteStrain	0.848 ± 0.015	0.907 ± 0.013	0.869	0.903	**0.922**	0.798	0.858
OliveOil	**0.908±0.035**	0.900 ± 0.000	0.867	0.833	0.867	0.633	0.700
Plane	0.984 ± 0.006	0.991 ± 0.009	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

Table 2 continued

Dataset	RPSF	gRSF	TSF	TSBF	LPS	FS	LTS
ShapeletSim	0.918 ± 0.010	0.801 ± 0.049	0.478	0.961	0.867	**1.000**	0.978
SmallKitchenAppliances	0.800 ± 0.002	–	**0.811**	0.672	0.712	0.333	0.664
SonyAIBORobotSurface1	0.852 ± 0.027	**0.885±0.040**	0.787	0.795	0.774	0.686	0.827
SonyAIBORobotSurface2	0.853 ± 0.014	0.852 ± 0.012	0.810	0.778	0.872	0.792	**0.890**
SwedishLeaf	0.913 ± 0.006	0.898 ± 0.003	0.914	0.915	**0.920**	0.768	0.907
Symbols	0.877 ± 0.015	0.780 ± 0.020	0.915	0.946	0.963	**0.967**	0.930
SyntheticControl	0.981 ± 0.005	0.966 ± 0.005	0.987	0.993	0.980	0.910	**0.997**
ToeSegmentation1	**0.946±0.007**	0.933 ± 0.008	0.741	0.781	0.877	0.943	0.930
ToeSegmentation2	0.886 ± 0.028	0.918 ± 0.019	0.815	0.800	0.869	0.692	**0.923**
Trace	**1.000±0.000**	**1.000±0.000**	0.990	0.980	0.980	**1.000**	**1.000**
TwoLeadECG	0.994 ± 0.004	0.981 ± 0.007	0.759	0.866	0.948	0.946	**0.997**
UMD	**0.946±0.013**	0.940 ± 0.022	0.924	–	0.903	0.861	0.944
Wafer	0.992 ± 0.001	–	0.996	0.995	**0.997**	**0.997**	0.996
Wine	0.707 ± 0.038	0.644 ± 0.045	0.630	0.611	0.630	**0.778**	0.500
WordSynonyms	0.633 ± 0.007	0.613 ± 0.007	0.647	0.688	**0.755**	0.431	0.607
Worms	0.527 ± 0.022	0.519 ± 0.015	0.610	0.688	**0.701**	0.649	0.610
WormsTwoClass	0.733 ± 0.021	0.715 ± 0.013	0.623	**0.753**	**0.753**	0.727	0.727
Yoga	0.853 ± 0.007	0.852 ± 0.005	0.859	0.819	**0.869**	0.695	0.834
Win/Draw/Loss		40/2/13	33/1/25	32/1/24	35/0/24	50/1/8	31/3/25

For each dataset, the bold text indicates the best value.

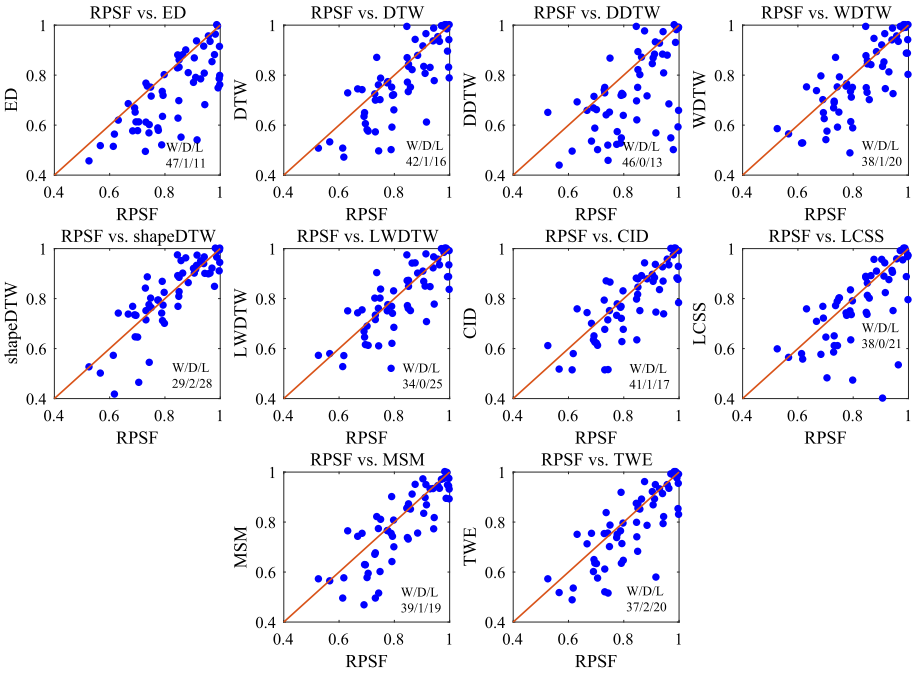

Fig. 8 Accuracy comparison between RPSF and other distance-based classifiers

LPS, TSBF, gRSF, and TSF perform the same as RPSF or SAXVSM, and similarly, whether SAXVSM is equivalent to FS and BoP or to the better five methods.

6.2.2 Compared with distance-based benchmark classifiers

The nearest neighbor classifier based on Euclidean distance (ED) is a widely-used benchmark whose performance can be improved by DTW. In order to improve the similarity measure, lots of advanced approaches are proposed. For example, Derivative Dynamic Time Warping (DDTW) [25], globally Weighted Dynamic Time Warping (WDTW) [19], Complexity-invariant DTW (CID) [3], Locally Weighted DTW (LWDTW) [47], shape Dynamic Time Warping (shapeDTW) [52], etc. Moreover, researchers also try to import other elastic distance measures to time series, the successful ones include the Time Warp Edit distance (TWE) [31], Longest Common Subsequence (LCSS) [18], and the Move-Split-Merge metric (MSM) [42].

For these comparisons, the results of LWDTW and shapeDTW are obtained from the original paper, others are picked up from [1] for simplicity. As shown in Fig. 8, our method is obviously superior to the nearest neighbor classifier on the vast majority of datasets. Especially for the sub-figure of RPSF vs. shapeDTW, there are three points apparently far away from the diagonal line, meaning that RPSF is 24.4%, 20.3%, and 20.2% higher than the alternative on datasets Ham, InsectWingbeatSound and Wine, respectively.

For the critical difference of the average ranking (as shown in Fig. 9), the proposed approach again achieved the first place among 11 classifiers. Figure 9 also indicates that RPSF performs significantly better than ED, DDTW, and DTW. The main advantage of distance-based methods lies in its effectiveness and efficiency. However, our RPSF is not

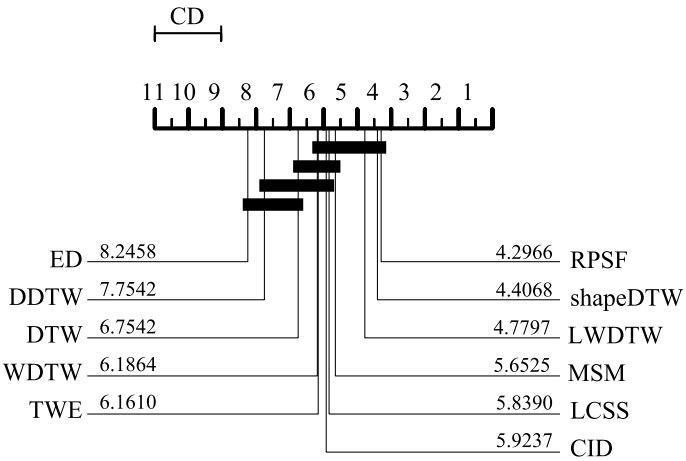

Fig. 9 Critical difference diagram for the average ranking comparison among RPSF and other distance-based classifiers

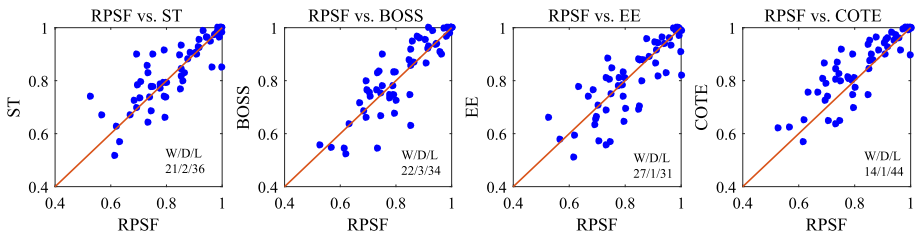

Fig. 10 Accuracy comparison between RPSF and other ensemble classifiers

only accurate, it can also be sped up easily using parallel/multi-core computing, details will be discussed in Sect. 6.3.

6.2.3 Compared with other ensemble benchmark classifiers

Ensemble classifiers combine multiple classification algorithms to obtain better predictive performance. For time series, apart from the forest-based methods introduced previously, other benchmark ones include the Collective of Transformation-based Ensembles (COTE) [2], Bag of Symbolic Fourier Approximation Symbols (BOSS) [37], Elastic Ensemble (EE) [29], and the Shapelet Transformation (ST) [15] combined with random forest (with 500 trees), rotation forest (with 50 trees) and other basic classifiers like SVM, Naive bayes etc.

As shown in Figs. 10 and 11, although our method is worse than COTE, it is competitive with ST, BOSS, and EE. In addition, RPSF just consists of 50 pairwise shapelets trees, it is far less than the number of classifiers contained in ST (more than 500 ones) and BOSS (normally equals to $m - 10$). COTE involves classifiers constructed in the time, frequency, change, and shapelet transformation domains, while our method only involves time domain. EE combines multiple advanced distance measures discussed in last section (such as LCSS, TWE, MSM, WDTW, and so on).

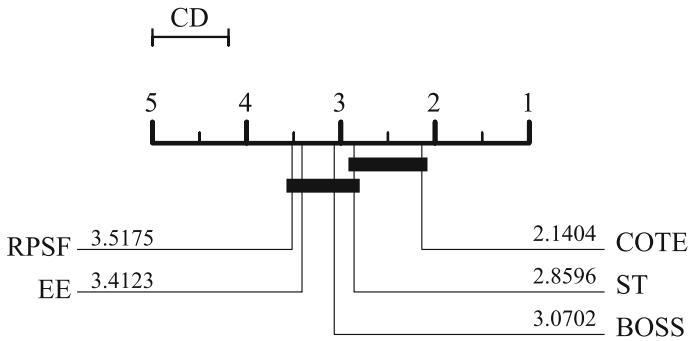

Fig. 11 Critical difference diagram for the average ranking comparison among RPSF and other ensemble classifiers

6.2.4 Compared with deep learning benchmark classifiers

Deep neural networks are considered widely in recent years to perform time series classification. We tried to compare RPSF with the nine deep learning benchmark classifiers mentioned in [12]. They are MLP, FCN, ResNet [44], Encoder [40], MCNN [9], t-LeNet [26], Multi Channel Deep Convolutional Neural Network (MCDCNN) [53] , Time-CNN [51], and Time Warping Invariant Echo State Network (TWIESN) [43]. As shown in Fig. 12, it is clear that, except FCN and ResNet, our method beats other deep learning classifiers on at least three-quarters of the total datasets.

The analysis in Fig. 13 reveals that RPSF performs significantly better than Time-CNN, MLP, TWIESN, MCDCNN, MCNN, and t-LeNet. The data are not sufficient to conclude whether RPSF performs the same as Encoder or to the better two methods FCN and ResNet.

6.3 Computational performance

In this section, the time scalability of RPSF is analyzed at first, then we will describe the significant increase in time performance due to the omission of split searching. Note that RPSF approach is easy to parallelize, several times of further acceleration can be achieved using parallel computing.

6.3.1 Time scalability

There are two factors that affect the time scalability of RPSF, the number of instances in the training set N, and the length of time series m. First, datasets Wafer and StarLightCurves that consist of 1000 training instances are employed. Second, datasets Mallat and BeetleFly with relatively larger number of attributes are selected. N and m are sampled randomly, and the sampling rate varies from 30% to 100%. Figures 14 and 15 show their wall clock time separately. It is easy to find that RPSF increases approximately linearly with the inflation of N and quadratically with m. These help us to make sure that the proposed algorithm has strong practicability.

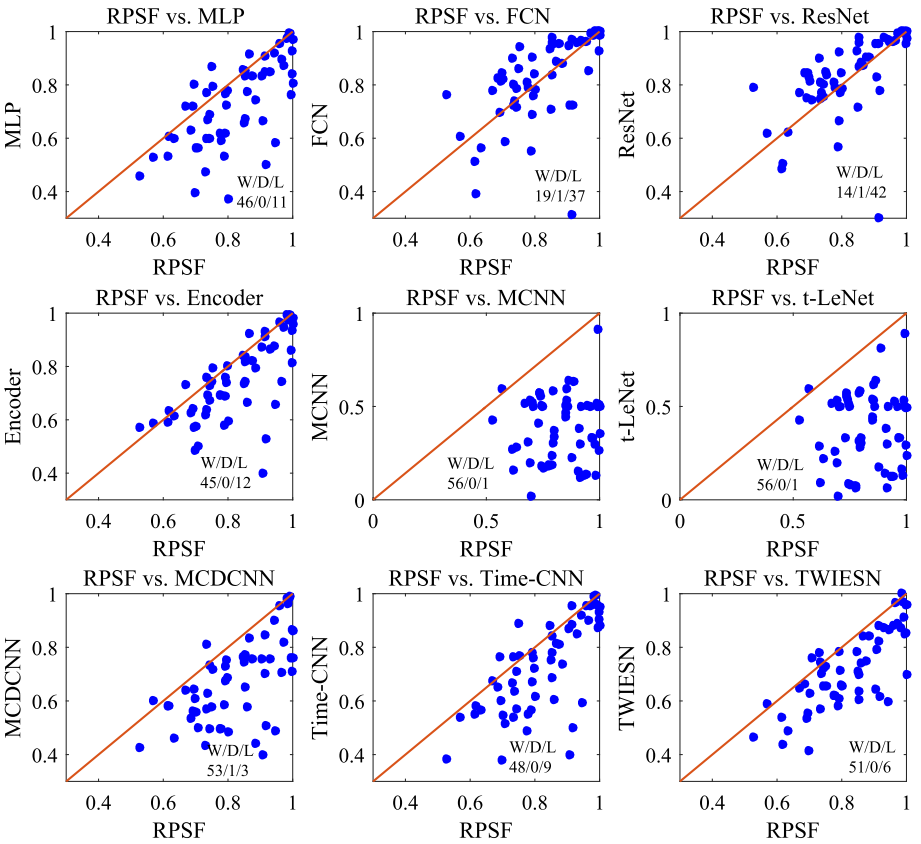

Fig. 12 Accuracy comparison between RPSF and deep learning classifiers

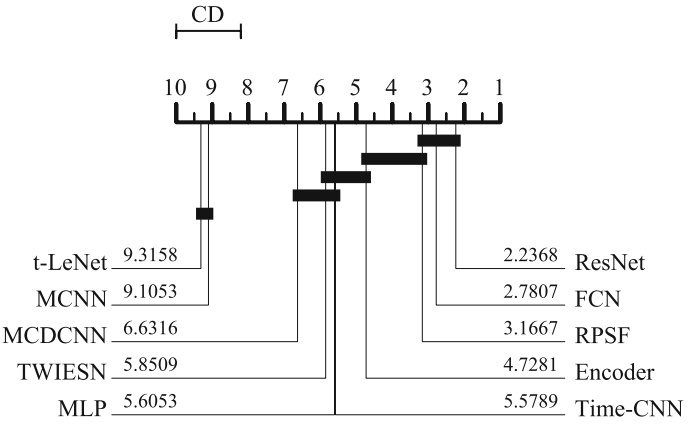

Fig. 13 Critical difference diagram for the average ranking comparison among RPSF and deep learning classifiers

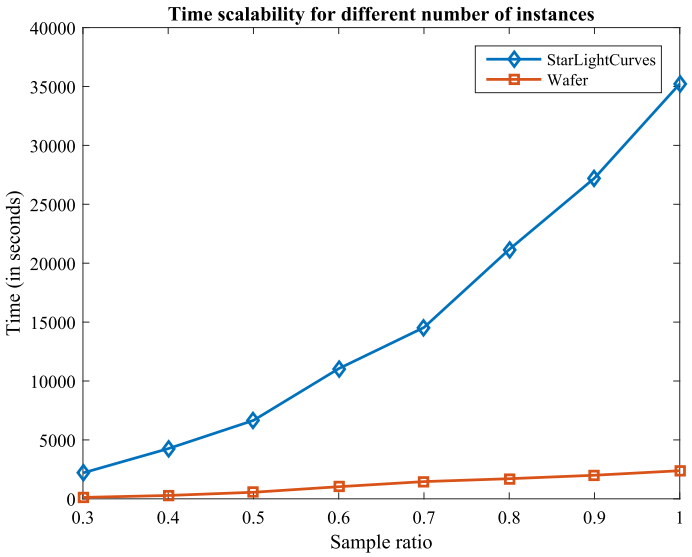

Fig. 14 Time scalability with respect to the training size

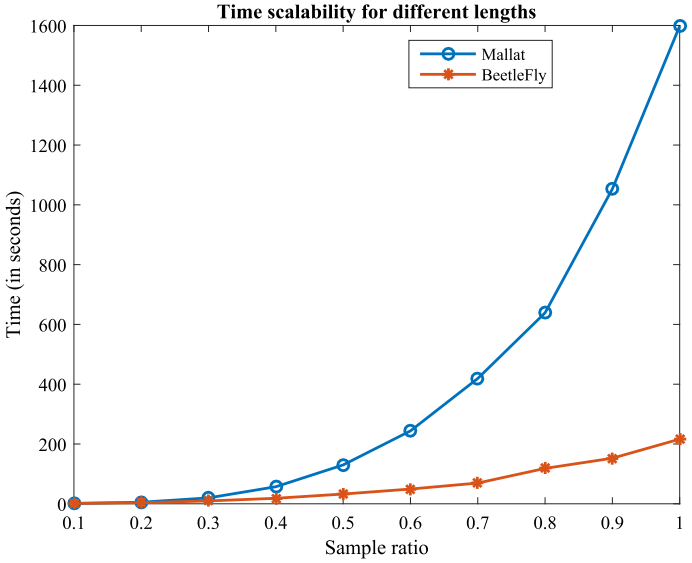

Fig. 15 Time scalability with respect to the length of time series

6.3.2 Versus other algorithms

We compare RPSF with other shapelet-based time series classification algorithms in terms of training time. Figure 16 shows boxplots of the relative time consumption using RPSF as benchmark to make it more intuitive. The dashed bold line indicates the time consumption of RPSF. FS is faster than other methods, which is the major advantage of this approach. RPSF is significantly better than LTS on the vast majority of datasets. It even appears tens of

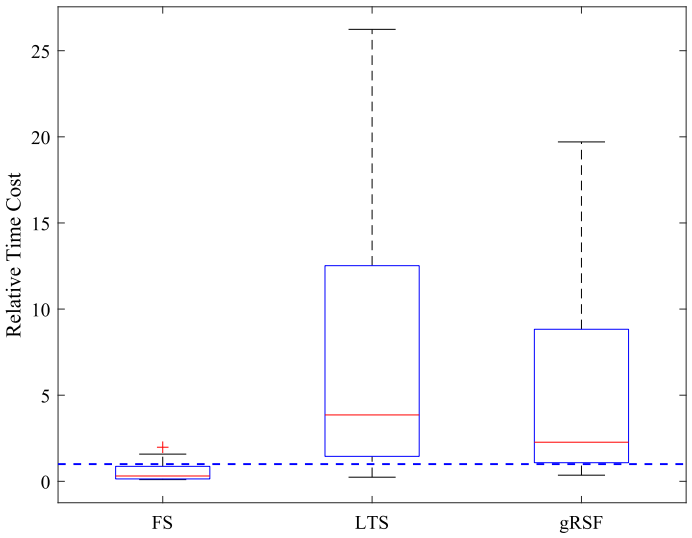

Fig. 16 Relative time consumption of FS, LTS, and gRSF compared with RPSF

times faster on some datasets. It is also noticeable that gRSF is slower than RPSF on almost all datasets. This result verifies our idea of omitting the calculation of the split threshold for time saving. In the next part, we will discuss it further.

6.3.3 Stage analysis

This part divides RPSF and gRSF into two main stages and analyzes the time consumption of them. We will show that eliminating split threshold indeed saves computation resources.

The decision tree of RPSF combines a pair of shapelets while gRSF is based on a single one. In terms of time, the main difference is that while assessing candidates, on the one hand, RPSF needs to calculate subsequence distances between two shapelets and all training instances, which is twice as much as that of single shapelet-based method; on the other hand, in the process of evaluating a single shapelet, a split threshold needs to be found. This is avoided by RPSF. Note that with entropy early pruning [45], threshold searching will be executed multiple times during candidate assessment, while subsequence distance will only be calculated once since its information can be reused. This expands the advantage of RPSF. A toy example is shown in Fig. 17.

As shown in Fig. 17a, for each candidate shapelet, subsequence distance between each instance and S is calculated and mapped on an orderline. Then every possible split point between each two objects is tested to find the optimal one (as indicated by the dashed vertical lines). After testing all the candidates, we pick the one achieving the best utility as the split node of decision tree.

For the pairwise shapelets (S_1, S_2), each instance should be compared with each shapelet in the pair. However, all the instances are divided into two clusters naturally (as shown in Fig. 17b). If it is closer to S_1, it belongs to the left part, otherwise the right. Indeed, we do not have to find the optimal split point for the pairwise shapelets.

Table 3 depicts the average running time in different stages and the average depth on several datasets. Although the depth of RPSF is either equal to or slightly greater than that

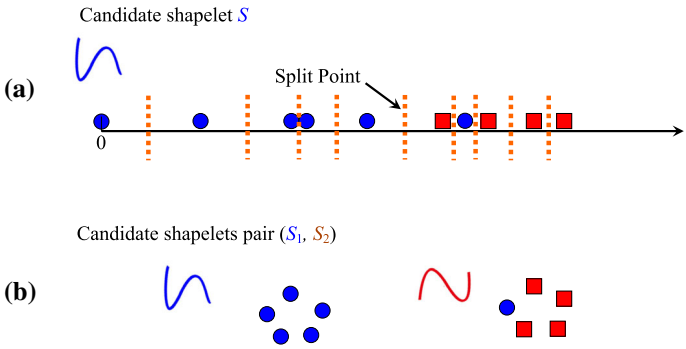

Fig. 17 Split point searching for single candidate shapelet and pairwise shapelets (a toy example)

of gRSF, the total training time is significantly smaller. On relatively small datasets, such as Coffee and FaceFour, the distance computation of RPSF consumes approximately twice as much as that of the gRSF. However, the time consumption of information gain on RPSF is obviously less than gRSF, and there is no doubt that RPSF is generally more efficient than gRSF, especially on relatively larger datasets (such as InsectWingbeatSound, Beef, SyntheticControl and Wine).

6.4 Case study

As discussed in Sect. 5, the DMDI provides an importance score for identifying influential time series regions. We briefly show the profit of DMDI on the ECGFiveDays dataset previously. More detailed synthetic and real-world examples will be included to demonstrate the usefulness of our proposed method in this section.

6.4.1 BME

The first row of Fig.18 shows the profile of some time series of *Begin*, *Middle*, and *End* classes of the synthetic dataset BME. For the time series of *Begin* class, the small bells arising at the initial period. For the time series of *End* class, the small bells arising at the final period, while only large up bells exist for the *Middle* class. The second row of Fig. 18 presents the DMDI values for each class. We can see that the learned scores succeed to localize the discriminative periods of the ground truth by achieving relatively higher scores during the beginning (ending) parts of the *Begin* (*End*) class. For the *Middle* class, the highest DMDI score appears in the middle parts, indicating the difference that large up or down bells arising in other classes.

6.4.2 GunPoint

GunPoint is a dataset that has been studied extensively in literature. The 150-length dataset describes the action curves of actors with or without a gun when making a motion (as shown in Fig. 19). The DMDI measure applied on this dataset is shown in the second part of Fig. 19. It is clear that the importance scores of the two classes arrive at their highest value near time stamps 100–120, and the score of *Gun* reaches a local peak among the indexes 40–60. These results are similar to the outcome of [45,49].

Table 3 Average depths and running time (in seconds) in different stages of gRSF and RPSF

Dataset	gRSF				RPSF			
	Total	Distance	Info_Gain	Depth	Total	Distance	Info_Gain	Depth
ArrowHead	253.63	95.21	154.86	2.17	136.49	122.43	6.38	2.29
Beef	1908.96	1337.20	550.70	3.77	1479.49	1408.37	34.48	3.80
CBF	6.42	1.54	4.45	2.00	3.97	2.90	0.55	2.00
Coffee	15.83	9.76	4.47	1.00	21.43	18.58	1.05	1.00
ECGFiveDays	3.12	0.97	1.91	1.00	1.64	1.23	0.18	1.00
FaceFour	62.06	42.33	15.51	2.53	95.39	86.26	3.84	2.76
GunPoint	105.76	21.33	83.31	1.17	43.44	36.45	3.85	1.95
InsectWing.	25467.07	2003.70	23391.57	7.33	5142.09	3213.02	1281.60	7.92
SyntheticControl	1315.86	17.87	1294.41	4.59	118.57	25.61	58.64	5.37
Wine	1502.65	401.75	1090.88	2.82	665.85	609.37	31.19	3.15

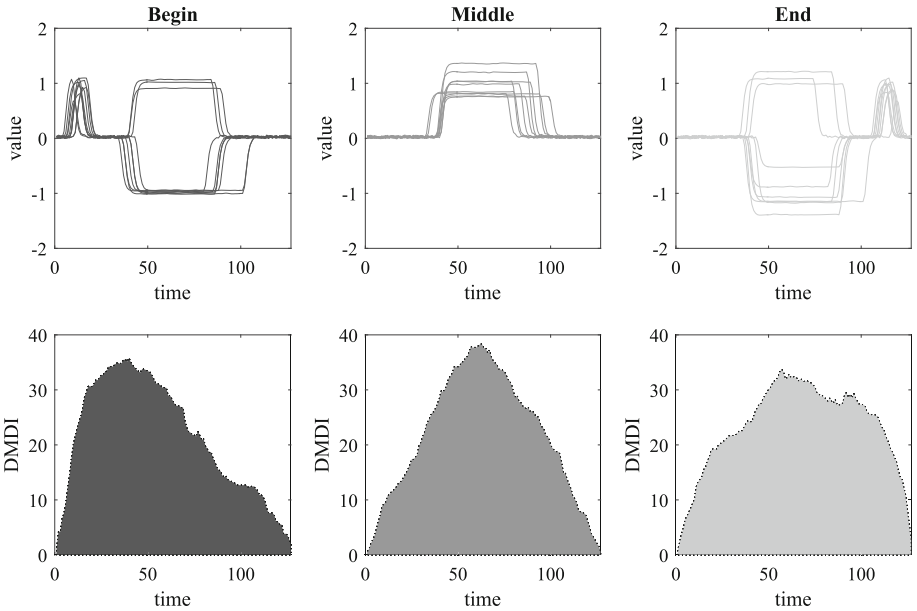

Fig. 18 DMDI for BME dataset

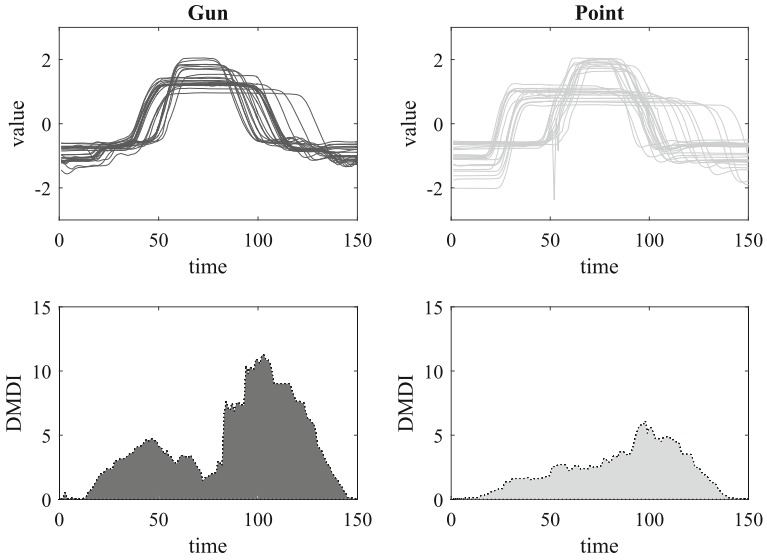

Fig. 19 DMDI for GunPoint dataset

6.4.3 ArrowHead

ArrowHead is a multi-class dataset with 251 attributes. It can be divided into three classes according to their place, age, and the race belonged: *Avonlea*, *Clovis*, and *Mix*. Figure 20 briefly depicts these outline series on the top three sub-figures, while our DMDI scores on

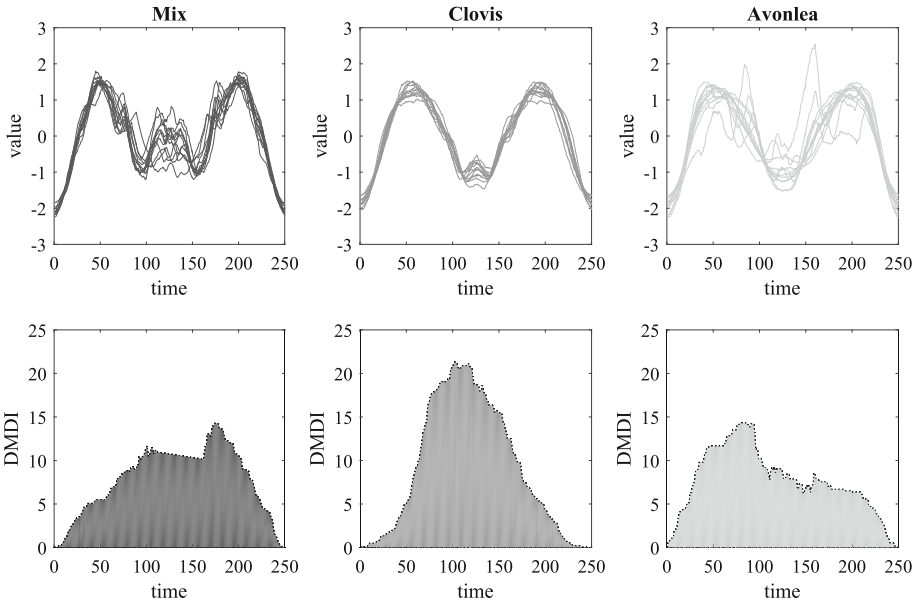

Fig. 20 DMDI for ArrowHead dataset

the bottom three. According to the importance score, the most significant area to distinguish *Clovis* with others lies near time stamp 125, and the other two locate at 160 and 90 separately.

7 Conclusion and future work

In this paper, we present an effective and efficient random forest combining shapelets from different classes randomly. The model diversity and classification accuracy are enhanced by including more information in a node. Due to the fact that pairwise shapelets do not have to search the split threshold, the time consumption is optimized. In addition, a novel importance measure DMDI is proposed to evaluate the contribution of each attribute to identify a certain class. Extensive experiments and case studies show that our method outperforms state-of-the-art random shapelet forest. Although the random combination presented is powerful, it is still relatively time-consuming. Trying to include a more sophisticated combination to further enhance the performance of random forest will be the focus of future work.

Acknowledgements The authors thank all the data donors of time series datasets, and the anonymous reviewers for their insightful comments and suggestions. This work is supported by Beijing Municipal Natural Science Foundation (No. 4214067) and National Natural Science Foundation of China (No. 61771058, 61702030).

Appendix

The time complexity of RPSF is mostly determined by the number of instances in the training set N, and the number of candidate shapelets r. In order to achieve the accuracy of relatively

Table 4 Accuracy of RPSF on relatively large datasets

Dataset	RPSF	Dataset	RPSF
CinC_ECG_torso	0.775±0.005	Phoneme	0.239 ± 0.004
Earthquakes	0.748 ± 0.000	ProximalPhalanxOutlineAgeGroup	0.841 ± 0.006
ECG5000	0.940 ± 0.001	ProximalPhalanxOutlineCorrect	0.863 ± 0.003
ElectricDevices	0.683 ± 0.021	ProximalPhalanxTW	0.782 ± 0.007
FiftyWords	0.677 ± 0.012	RefrigerationDevices	0.508 ± 0.029
FordA	0.899 ± 0.001	ScreenType	0.418 ± 0.011
FordB	0.752 ± 0.027	ShapesAll	0.804 ± 0.005
HandOutlines	0.895 ± 0.029	StarLightCurves	0.972 ± 0.006
Haptics	0.452 ± 0.025	Strawberry	0.934 ± 0.010
InlineSkate	0.385 ± 0.015	TwoPatterns	0.979 ± 0.002
NonInvasiveFetalECGThorax1	0.884 ± 0.004	uWaveGestureLibraryAll	0.928 ± 0.018
NonInvasiveFetalECGThorax2	0.909 ± 0.001	uWaveGestureLibraryX	0.780 ± 0.004
OSULeaf	0.625 ± 0.035	uWaveGestureLibraryY	0.702 ± 0.002
PhalangesOutlinesCorrect	0.839 ± 0.034	uWaveGestureLibraryZ	0.732 ± 0.002

large datasets, we decrease the percentage of candidate shapelets to 0.001, the sampling size to $N/2$. The corresponding results are shown in Table 4.

References

1. Bagnall AJ, Lines J, Bostrom A, Large J, Keogh EJ (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Discov 31(3):606–660. https://doi.org/10.1007/s10618-016-0483-9
2. Bagnall A, Lines J, Hills J, Bostrom A (2015) Time-series classification with cote: the collective of transformation-based ensembles. IEEE Trans Knowl Data Eng 27(9):2522–2535
3. Batista GEAPA, Keogh EJ, Tataw OM, de Souza VMA (2014) CID: an efficient complexity-invariant distance for time series. Data Min Knowl Discov 28(3):634–669. https://doi.org/10.1007/s10618-013-0312-3
4. Baydogan MG, Runger GC (2016) Time series representation and similarity based on local autopatterns. Data Min Knowl Discov 30(2):476–509. https://doi.org/10.1007/s10618-015-0425-y
5. Baydogan MG, Runger GC, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Trans Pattern Anal Mach Intell 35(11):2796–2802. https://doi.org/10.1109/TPAMI.2013.72
6. Bostrom A, Bagnall A (2015) Binary shapelet transform for multiclass time series classification. In: International conference on big data analytics and knowledge discovery. Springer, pp 257–269
7. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
8. Cetin MS, Mueen A, Calhoun VD (2015) Shapelet ensemble for multi-dimensional time series. In: Proceedings of the 2015 SIAM international conference on data mining. SIAM, pp 307–315
9. Cui Z, Chen W, Chen Y (2016) Multi-scale convolutional neural networks for time series classification. arXiv preprint. arXiv:1603.06995
10. Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153
11. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. Proc VLDB Endow 1(2):1542–1552
12. Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller P-A (2019) Deep learning for time series classification: a review. Data Min Knowl Discov 33(4):917–963
13. Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2014) Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 392–401

14. Haghiri S, Garreau D, von Luxburg U (2018) Comparison-based random forests. In: Proceedings of the 35th international conference on machine learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018, pp 1866–1875
15. Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2014) Classification of time series by shapelet transformation. Data Min Knowl Discov 28(4):851–881
16. Hou L, Kwok JT, Zurada JM (2016) Efficient learning of timeseries shapelets. In: Thirtieth AAAI conference on artificial intelligence
17. Itakura F (1975) Minimum prediction residual principle applied to speech recognition. IEEE Trans Acoust Speech Signal Process 23(1):67–72
18. Jansen T, Weyland D (2010) Analysis of evolutionary algorithms for the longest common subsequence problem. Algorithmica 57(1):170–186. https://doi.org/10.1007/s00453-008-9243-6
19. Jeong Y, Jeong M, Omitaomu O (2011) Weighted dynamic time warping for time series classification. Pattern Recognit 44:2231–2240
20. Karlsson I, Papapetrou P, Asker L (2015) Multi-channel ecg classification using forests of randomized shapelet trees. In: Proceedings of the 8th ACM international conference on pervasive technologies related to assistive environments. ACM, p 43
21. Karlsson, I, Papapetrou P, Boström H (2015) Forests of randomized shapelet trees. In: International symposium on statistical learning and data sciences. Springer, pp 126–136
22. Karlsson I, Papapetrou P, Boström H (2016) Early random shapelet forest. In: International conference on discovery science. Springer, pp 261–276
23. Karlsson I, Papapetrou P, Boström H (2016b) Generalized random shapelet forests. Data Min Knowl Discov 30(5):1053–1085
24. Keogh EJ, Pazzani MJ (2001a) Derivative dynamic time warping. In: SDM, Vol. 1. SIAM, pp 5–7
25. Keogh E, Pazzani M (2001) Derivative dynamic time warping. In: First international conference on data mining
26. Le Guennec A, Malinowski S, Tavenard R (2016) Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD workshop on advanced analytics and learning on temporal data
27. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144
28. Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. J Intell Inf Syst 39(2):287–315. https://doi.org/10.1007/s10844-012-0196-5
29. Lines J, Bagnall AJ (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Discov 29(3):565–592. https://doi.org/10.1007/s10618-014-0361-2
30. Lines J, Davis LM, Hills J, Bagnall A (2012) A shapelet transform for time series classification. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 289–297
31. Marteau P (2009) Time warp edit distance with stiffness adjustment for time series matching. IEEE Trans Pattern Anal Mach Intell 31(2):306–318. https://doi.org/10.1109/TPAMI.2008.76
32. Mueen A, Keogh E, Young N (2011) Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 1154–1162
33. Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the 2013 SIAM international conference on data mining. SIAM, pp 668–676
34. Renard X, Rifqi M, Erray W, Detyniecki M (2015) Random-shapelet: an algorithm for fast shapelet discovery. In: 2015 IEEE international conference on data science and advanced analytics (DSAA), 2015, 36678. IEEE, pp 1–10
35. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49
36. Sathe S, Aggarwal CC (2017) Similarity forests. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 395–403
37. Schäfer P (2015) The BOSS is concerned with time series classification in the presence of noise. Data Min Knowl Discov 29(6):1505–1530. https://doi.org/10.1007/s10618-014-0377-7
38. Schäfer P, Leser U (2017) Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on conference on information and knowledge management. ACM, pp 637–646
39. Senin P, Malinchik S (2013) SAX-VSM: interpretable time series classification using SAX and vector space model. In: 2013 IEEE 13th international conference on data mining, Dallas, TX, USA, December 7–10, 2013, pp 1175–1180. https://doi.org/10.1109/ICDM.2013.52
40. Serrà J, Pascual S, Karatzoglou A (2018) Towards a universal neural network encoder for time series. In: CCIA, pp 120–129

41. Shi M, Wang Z, Yuan J, Liu H (2018) Random pairwise shapelets forest. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 68–80
42. Stefan A, Athitsos V, Das G (2013) The move-split-merge metric for time series. IEEE Trans Knowl Data Eng 25(6):1425–1438. https://doi.org/10.1109/TKDE.2012.88
43. Tanisaro P, Heidemann G (2016) Time series classification using time warping invariant echo state networks. In: 2016 15th IEEE international conference on machine learning and applications (ICMLA). IEEE, pp 831–836
44. Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: a strong baseline. In: International joint conference on neural networks
45. Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 947–956
46. Yuan J-D, Wang Z-H, Han M (2014) A discriminative shapelets transformation for time series classification. Int J Pattern Recognit Artif Intell 28(06):1450014
47. Yuan J, Douzal-Chouakria A, Yazdi SV, Wang Z (2018) A large margin time series nearest neighbour classification under locally weighted time warps. Knowl Inf Syst, pp 1–19
48. Yuan J, Lin Q, Zhang W, Wang Z (2019) Locally slope-based dynamic time warping for time series classification. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 1713–1722
49. Yuan J, Wang Z, Han M, Sun Y (2015) A lazy associative classifier for time series. Intell Data Anal 19(5):983–1002
50. Zhang Z, Zhang H, Wen Y, Zhang Y, Yuan X (2018) Discriminative extraction of features from time series. Neurocomputing 275:2317–2328
51. Zhao B, Lu H, Chen S, Liu J, Wu D (2017) Convolutional neural networks for time series classification. J Syst Eng Electron 28(1):162–169
52. Zhao J, Itti L (2018) shapedtw: shape dynamic time warping. Pattern Recognit 74:171–184. https://doi.org/10.1016/j.patcog.2017.09.020
53. Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL (2014) Time series classification using multi-channels deep convolutional neural networks. In: International conference on web-age information management. Springer, pp 298–310

Jidong Yuan received the B.S. degree in Computer Science and Technology from Northeastern University, Shenyang, China, in 2010. He received the M.S. degree and Ph.D. degree in Computer Science and Technology from Beijing Jiaotong University, Beijing, China, in 2012 and 2016, respectively. Jidong Yuan is currently an associate professor in the School of Computer and Information Technology, Beijing Jiaotong University. He is conducting his research mainly on data mining and machine learning on time series data. He has published over twenty papers in international journals and conferences on his major research topics: time series classification, data stream mining and representation learning for time series.

Mohan Shi received the Bachelor Degree of Engineering in Computer Science and Technology from Capital Normal University, Beijing, China, in 2016. He received the Master Degree of Engineering in Computer Science and Technology from Beijing Jiaotong University, Beijing, China, in 2019. Mohan Shi is currently a software engineer in Beijing Jingdong 360 Degree E-Commerce Company. His research interests include machine learning and data minning.

Zhihai Wang received the Ph.D. degree in Computer Application from Hefei University of Technology in 1998. He is currently a professor at the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. He has published dozens of papers in international conferences and journals. His research interests include data mining and artificial intelligence.

Haiyang Liu received his Ph.D. degree in Computer Science from Beijing Jiaotong University in 2019. He is currently a lecturer at the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. He has published several papers in peer-reviewed international conferences and journals. His research interests include data mining and artificial intelligence.

Jinyang Li is a Junior Ph.D. student at Department of Computer Science in The University of Hong Kong, advised by Prof. Reynold C.K. Cheng. His research interests include Natural Language Processing, Knowledge Graph, and Semantic Parsing, and their application in Dialogue, Recommendation, etc. Prior to joining HKU, he received his Master of Science degree in Department of Electrical Engineering from Columbia University in the City of New York. In 2017, he acquired his bachelor degree in Harbin Institute of Technology with honour.