Data Min Knowl Disc (2016) 30:1053–1085
DOI 10.1007/s10618-016-0473-y

Generalized random shapelet forests

Isak Karlsson[1] · **Panagiotis Papapetrou**[1] · **Henrik Boström**[1]

Received: 10 December 2015 / Accepted: 28 June 2016 / Published online: 12 July 2016

Abstract Shapelets are discriminative subsequences of time series, usually embedded in shapelet-based decision trees. The enumeration of time series shapelets is, however, computationally costly, which in addition to the inherent difficulty of the decision tree learning algorithm to effectively handle high-dimensional data, severely limits the applicability of shapelet-based decision tree learning from large (multivariate) time series databases. This paper introduces a novel tree-based ensemble method for univariate and multivariate time series classification using shapelets, called the generalized random shapelet forest algorithm. The algorithm generates a set of shapelet-based decision trees, where both the choice of instances used for building a tree and the choice of shapelets are randomized. For univariate time series, it is demonstrated through an extensive empirical investigation that the proposed algorithm yields predictive performance comparable to the current state-of-the-art and significantly outperforms several alternative algorithms, while being at least an order of magnitude faster. Similarly for multivariate time series, it is shown that the algorithm is significantly less computationally costly and more accurate than the current state-of-the-art.

Keywords Multivariate time series · Time series classification · Time series shapelets · Decision trees · Ensemble methods

Responsible editor: Thomas Gärtner, Mirco Nanni, Andrea Passerini, and Celine Robardet.

✉ Isak Karlsson
isak-kar@dsv.su.se

Panagiotis Papapetrou
panagiotis@dsv.su.se

Henrik Boström
henrik.bostrom@dsv.su.se

[1] Stockholm University, Stockholm, Sweden

1 Introduction

In many domains, repeated measurements are collected in order to obtain characteristics of objects or situations that evolve over time. Examples of such domains include shape outline recognition, e.g., for classifying historical documents or projectile points (Ye and Keogh 2009), classification of electrocardiograms (ECGs) (Kampouraki et al. 2009), and anomaly detection in streaming data (Rebbapragada et al. 2009). These measurements are typically collected at a fixed rate and such collections are commonly referred to as *data series*. In the case of measurements over time, such series are referred to as *time series*, and they can be either univariate (with a single variable evolving over time) or multivariate (with d time-evolving variables).

Our main focus in this paper is *time series classification*. In other words, given a collection of time series, we would like to infer a model that can predict the value of a categorical output variable, based on the observed time series describing an instance.

Example Let us consider an example from the medical domain. More specifically, given an electrocardiogram (ECG) of a patient, we would like to infer the cardiovascular condition of that patient by exploiting the information contained in a collection of past observations of ECGs, labeled with a categorical outcome variable, taken from different patients. Hence, in this case, the outcome variable corresponds to the heart condition of the patient. An example of an ECG is illustrated in Fig. 1. We can observe that an ECG in this case can be modeled as a multivariate time series consisting of 15 concurrently evolving variables, each corresponding to the voltage of a heart signal over time.

Achieving fast and accurate time series classification has attracted significant interest of the data mining community over the past decade. One line of research has been focusing on representing time series as ordinary feature vectors. Such representations allow for direct application of standard supervised learning algorithms, such as decision trees (Rodríguez and Alonso 2004), support vector machines (SVMs) (Wu and Chang 2004), neural networks (Nanopoulos et al. 2001), and nearest neighbor classifiers (Batista et al. 2011). For univariate time series (UTS) classification, experimental evidence has repeatedly demonstrated (Xi et al. 2006; Wang et al. 2013) that

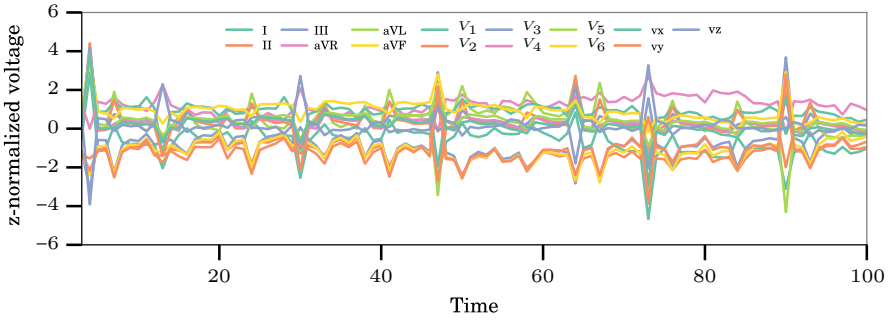

Fig. 1 An example of an electrocardiogram (ECG) with 15 variables. Each variable is a heart signal, measured by a separate voltage meter, that evolves over time. This ECG can be seen as a 15-dimensional time series

similarity-based approaches with elastic measures, such as dynamic time warping (DTW) (Berndt and Clifford 1994) provide state-of-the-art predictive performance. Multivariate time series (MTS), however, are characterized not only by similarities between individual attributes, but also by relations between different attributes. The latter is not captured by the traditional univariate approaches (Bankó 2012), which has directed the attention to feature based approaches such as learned pattern similarity (Baydogan and Runger 2015).

A recent approach for time series classification is to identify and extract time series subsequences, called *shapelets*, that can be used as discriminatory features for classification (Ye and Keogh 2009). Their main characteristic is that they are class-discriminant, phase-independent, i.e., location invariant, subsequences of a longer time series. Several shapelet-based approaches have been proposed for univariate time series classification, such as shapelet-based decision trees (Ye and Keogh 2009) and pre-processing approaches using shapelet transformations (Hills et al. 2014). Similar approaches have been proposed also for multivariate time series. One approach is to build a shapelet-based decision tree for each dimension and then combine the individual classifiers using majority voting (Cetin et al. 2015), while another approach is to perform a shapelet transformation with weighted voting (Patri et al. 2014). It has been demonstrated that while shapelet-based decision trees can provide interpretable rules and, hence, insights to practitioners (Ye and Keogh 2009), their classification accuracy and training costs are often prohibitive (Rakthanmanon and Keogh 2013), limiting their applicability when dealing with large and multivariate time series. Therefore, a natural extension to shapelet-based decision trees is to consider forests of such trees, where each individual tree is built by considering only a subset of all available shapelets. The latter can lead to a substantial speedup, especially for large multivariate time series, leading to that entire forest can be generated faster and obtain higher predictive performance than a single tree (generated by enumerating all shapelets), similar to how random forests (Breiman 2001) improve upon single decision trees.

The main contributions of the paper can be summarized as follows:

Novelty We propose a generalized method for efficient and effective time series classification. The key novelty of our approach is that it extends the random shapelet forest algorithm[1] to support both uni- and multivariate time series classification.

Efficiency and effectiveness Through an extensive experimental evaluation on both *univariate* and *multivariate* time series datasets, we demonstrate that the proposed algorithm can achieve *competitive predictive performance* compared to state-of-the-art methods, while reducing the *training cost* by up to *an order of magnitude* on average. In addition, we evaluate the robustness of the method with respect to different parameter settings and provide insights on how to tune them for different situations, exploring their impact on the evaluation measures through a decomposition of the mean square error into the bias and the variance of class probabilities assigned by the individual trees in the forest.

[1] An earlier version of the algorithm, restricted to univariate time series, was presented together with a limited empirical evaluation in Karlsson et al. (2015).

Applicability We demonstrate the applicability of the novel method on a *large* and *diverse* collection of datasets spanning various application domains, including 56 *univariate* and 14 *multivariate* time series datasets. Finally, the presented algorithm is: *simple to implement*, *accurate*, *fast* and *embarrassingly parallel*, i.e., applicable to a wide range of tasks.

The remainder of the paper is organized as follows. In the next section, we discuss related work on time series classification, focusing on shapelet-based classifiers and multivariate time series. In Sect. 3, the notation and problem setting are defined. In Sect. 4, we present the algorithm for generating forests of randomized shapelet trees and discuss various implementation choices. In Sect. 5, the experimental setup and results from the empirical investigation are presented. Finally, we summarize the main findings and point out directions for future work in Sect. 6.

2 Related work

Most approaches for univariate time series classification generally rely on instance-based classification algorithms, e.g., k-nearest neighbor, using different similarity (or distance) measures, of which the most common and simplest is the Euclidean norm. To improve accuracy, elastic distance measures have been proposed, such as dynamic time warping (DTW) or longest common subsequence (Maier 1978) and variants, e.g., cDTW (Sakoe and Chiba 1978), EDR (Chen and Özsu 2005), ERP (Chen and Ng 2004), that are robust to misalignment and time warps. These distance measures allow for localized distortion, e.g., DTW finds the optimal match between two sequences by allowing non-linearity in the distance calculation. By regularization using, e.g., a band (Ratanamahatana and Keogh 2004), the search performance and generalization behavior of k-nn can be greatly improved (Ding et al. 2008). For a more complete overview of instance based univariate time series classifiers, the reader is referred to, e.g., Ding et al. (2008). Although the extension of the DTW algorithm for the multivariate case is non-trivial (Shokoohi-Yekta et al. 2015), several alternatives have been proposed for multivariate time series classification, where the simplest and most commonly employed is the cumulative distance of all univariate distances. Another alternative is introduced by Bankó (2012), where a correlation-based version of DTW (CBDTW) is proposed that combines DTW and principal component analysis to preserve the correlation structure between time series.

To supplement instance-based classifiers and improve the interpretability of the generated models, the concept of shapelets has been introduced (Ye and Keogh 2009). Shapelets are usually described as subsequences whose distance to other time series provides discriminative features used for classification and interpretation (Ye and Keogh 2011). For shapelet-based classifiers, the idea is to consider all subsequences of the training data recursively in a divide-and-conquer manner, while assessing the quality of the shapelets using a *scoring* function to estimate their discriminative power, constructing an interpretable *shapelet tree* classifier (Ye and Keogh 2009, 2011).

The most common scoring function is information gain (Shannon 1948), which is also commonly employed when constructing traditional decision trees, (see e.g., Quinlan 1993). To prune non-informative shapelet candidates early and, hence, speed-up

the exhaustive search, Ye and Keogh (2009) employ early-abandoning and lower-bounding on the information gain. Due to the combinatorial explosion, however, lower-bounding the information gain does not scale and becomes infeasible when the number of classes increases. To overcome this limitation, other measures based on the analysis of variance have been considered, with the rationale being that these measures are computationally less costly to compute (Lines et al. 2012; Hills et al. 2014). Hills et al. (2014) showed that although there is no significant difference in predictive performance between using information gain and the measures based on variance, the computational cost is significantly reduced. In addition to these techniques for speeding up the exhaustive search for shapelets, several other approaches have been proposed in the literature. For example, methods for trading time complexity for memory consumption, while finding the optimal match have been explored (Mueen et al. 2011).

Decision trees are interpretable, something which is useful in many domains. However, when it comes to predictive performance, they are in almost all cases outperformed by other classifiers, such as support vector machines (Cortes and Vapnik 1995), deep neural networks, (Schmidhuber 2014) and random forests (Breiman 2001). To overcome this limitation, Hills et al. (2014) proposes a single-scan algorithm for finding the best k shapelets in a collection of time series. Subsequently, the produced set of k shapelets is used to generate a new transformed k-feature dataset, with each attribute being the (minimum) distance from the i:th shapelet to the j:th original time series. By disconnecting the shapelet search and the model generation, time series classification is reduced to a feature selection (or generation) problem, enabling the use of a wide range of efficient learning algorithms (Hills et al. 2014).

Shapelet transformation is one instance of a more general concept of feature generation, which has been thoroughly investigated for time series classification. For example, generated features include interval-based features (Rodríguez et al. 2005; Rodríguez and Alonso 2004), statistical features (Nanopoulos et al. 2001; Deng et al. 2013), and other interpretable features, such as correlation structure, distribution or entropy (Fulcher and Jones 2014). Typically the features produced by these transformations can be grouped into three main categories: correlation-based, auto-correlation-based, and shape-based, each denoting similarity in time, change, and shape, respectively. For example, a time series forest based on interval features, such as averages, standard deviations and slope has been proposed by Deng et al. (2013) and a transformation based on time series bag-of-words by Baydogan et al. (2013).

Since exhaustive shapelet discovery requires the enumeration of every subsequence in the training data, it is not feasible for large datasets with large, multivariate, time series. To improve speed, the search space is usually reduced by only considering subsequences of specific lengths, e.g., within predefined ranges. The optimal shapelet length is generally unknown, requiring brute-force searches over multiple classifiers using cross-validation. Due to computational constraints, this is however often infeasible. To improve the performance, Hills et al. (2014) introduce a heuristic-based algorithm for estimating the shapelet length.

For univariate time series classification, an alternative to exhaustively enumerating and searching among all shapelets has been introduced by Grabocka et al. (2014) through the notion of *shapelet learning* (LTS). In the domain of univariate time series

classification, this can be considered the current state-of-the-art in terms of classification accuracy. In this framework, shapelets are learned from the training data, while simultaneously minimizing both the training error using the logistic loss function and the minimum distance using a soft minimum (Grabocka et al. 2014). The learned shapelets can improve predictive performance significantly compared to other shapelet-based classifiers that are primarily based on exhaustive search (Grabocka et al. 2014). Apart from the high computational cost, the primary disadvantage of LTS is the large number of hyper-parameters that require tuning and the difficulty of choosing the initial shapelet prototypes, which typically has a large impact on the resulting accuracy (Grabocka et al. 2014).

As far as multivariate time series classification is concerned, two main approaches using shapelets have been introduced in the literature. Patri et al. (2014) proposes a shapelet forest for classifying heterogeneous multivariate sensor data. The algorithm uses the Fast Shapelet approach (Rakthanmanon and Keogh 2013) to extract the most informative shapelets from each dimension. Using these shapelets, a distance matrix is computed and weights are learned for each shapelet using different feature selection methods. The classification of a new multivariate time series is performed by weighting the votes of the individual shapelet trees built for each dimension by summing the learned weights of the used shapelets. A similar approach, in which one shapelet tree is built from each time series dimension, is presented by Cetin et al. (2015). For the latter, several techniques for improving the search speed are proposed, which makes building several trees tractable. The techniques improve the performance of tree generation significantly over existing methods by using multi-length indexing and dynamic stepping (Cetin et al. 2015). Different voting approaches are evaluated, showing that building one shapelet tree per dimension outperforms shapelets defined over multiple dimensions (Cetin et al. 2015). The main drawback of these approaches is however that, by building an individual tree per dimension, the trees do not take into consideration potential dependencies between the dimensions. In addition, the weighted voting technique that is employed for determining the final class does not consider the varying importance of different dimensions.

Similar to shapelet transformation, Wistuba et al. (2015) proposes a method for multivariate time series, Ultra-Fast Shapelets (UFS), where random shapelets are used as features. The authors show that such transformations are both fast and accurate. In particular, the method outperforms related algorithms for multivariate time series classification (Wistuba et al. 2015). The most prominent advantage of UFS, which is rather similar to the method proposed in this work, is the ability to model relationships between different dimensions at a low computational cost. A drawback, however, is the fact that UFS considers shapelets globally from a restricted pool of preselected shapelets, as opposed to the method proposed in this study, which, at each node in each tree, locally considers a set of shapelets, improving diversity and directing the search to more important regions. Finally, another line of research for multivariate time series classification includes feature based approaches, such as learned pattern similarity (LPS) (Baydogan and Runger 2015), symbolic representation for multivariate time series (SMTS) (Baydogan and Runger 2014), and multivariate extensions of time series bag-of-words (Baydogan et al. 2013).

3 Problem setting

The problem studied in this paper is uni- and multivariate time series classification, i.e., the task at hand is, given one or several set-valued variables, each with values ordered by time and sampled at regular and fixed intervals, we want to infer a model that is able to accurately predict the class labels of unseen examples based on the observed variables.

Next, we proceed by providing some background definitions along with the problem formulation.

Definition 1 (*d-dimensional time series*) A *d-dimensional time series* $\mathcal{T} = \{\mathbf{T_1}, \ldots, \mathbf{T_d}\}$ is a sequence of d variables, such that $\mathbf{T}_k \in \mathbb{R}^m$, $\forall k \in \{1, \ldots, d\}$, where $\mathbf{T_k} = \{T_{k,1}, \ldots, T_{k,m}\}$, with $T_{k,j} \in \mathbb{R}$, $\forall j \in \{1, \ldots, m\}$. For $d = 1$, $\mathcal{T}$ defines a *univariate* time series, denoted simply as $\mathcal{T} = \{T_1, \ldots, T_m\}$ consisting of a sequence of m ordered elements $T_j \in \mathbb{R}$. For $d > 1$, $\mathcal{T}$ defines a *multivariate* time series.

A local segment of a time series is called a *time series subsequence*. A more formal definition is given next.

Definition 2 (*time series subsequence*) A *time series subsequence* of the kth dimension of a time series $\mathcal{T}$ is a sequence of l contiguous elements of $\mathbf{T}_k$, denoted as $\mathbf{T}_k^{s:s+l-1} = \{T_{k,s}, \ldots, T_{k,s+l-1}\}$, where s is the starting position and l is its length. Time series subsequence and shapelet is used interchangeably.

As stated earlier, time series classification predominantly relies on the chosen distance (similarity) measure to compare and discriminate between instance pairs. Depending on the application domain and the nature of the time series, various distance measures can be used. For the case of a single time series dimension, k, we will consider the following two distance functions.

Definition 3 (*time series distance*) Given two time series $\mathcal{T}$ and $\mathcal{T}'$ of equal length l, the *time series distance* between their corresponding kth dimensions is the length-normalized Euclidean distance between $\mathbf{T}_k$ and $\mathbf{T}'_k$, i.e.:

$$Fdist(\mathbf{T}_k, \mathbf{T}'_k) = ED(\mathbf{T}_k, \mathbf{T}'_k) = \sqrt{\frac{1}{l}\sum_{i=1}^{l}(T_{k,i} - T'_{k,i})^2}\,. \tag{1}$$

Finally, we define the distance between a time series subsequence and a time series.

Definition 4 (*time series subsequence distance*) Given a 1-dimensional time series $\mathcal{S}$ and a d-dimensional time series $\mathcal{T}$ of lengths l and m, respectively, such that $l \leq m$, the *time series subsequence distance* between $\mathcal{S}$ and the kth dimension of $\mathcal{T}$, is the minimum distance between $\mathcal{S}$ and any subsequence of $\mathbf{T}_k$ of length l, i.e.:

$$Sdist(\mathcal{S}, \mathbf{T}_k) = \min_{s=1}^{m-l+1}\{Fdist(\mathcal{S}, \mathbf{T}_k^{s:s+l-1})\}\,. \tag{2}$$

Note that since a $\mathcal{S}$ is 1-dimensional time series of length equal to the length of each subsequence $\mathbf{T}_k^{i,s:s+l-1}$, Fdist can be applied directly. Also note that the distance function must be length invariant in order to avoid penalizing longer sequences (Ye and Keogh 2009).

It is important to mention that in the two previous definitions, alternative distance measures can be used, instead of the Euclidean distance, without loss of generality. We have chosen Euclidean distance since it is almost exclusively used in the literature for building shapelet-based time series classifiers.

A collection of n time series $\mathcal{D} = \{\mathcal{T}^1, \ldots, \mathcal{T}^n\}$ defines a *time series dataset*. For notational simplicity, we assume that every time series in $\mathcal{D}$ has the same number of dimensions d and that each dimension is of the same length m. Although this is one of the most common settings for time series classification, the equal length assumption is not a requirement in general (Hu et al. 2013) and the presented method can in fact handle time series of varying lengths.

Supervised learning involves a set of *training* instances, usually denoted as the *learning set*, labeled with one of a finite set of possible values, which we denote as $\mathcal{C} = \{c_1, c_2, \ldots, c_q\}$. For time series classification, the learning set consists of a collection of labeled time series. Next, we formally define a *time series learning set*.

Definition 5 (*time series learning set*) Given a time series dataset $\mathcal{D}$ of size n and a set of class labels $\mathcal{C}$, a time series learning set $\mathcal{Z}_{\{\mathcal{D},\mathcal{Y}\}}$ is defined as a vector of instance tuples $z^i = (\mathcal{T}^i, y^i)$, where each $\mathcal{T}^i \in \mathcal{D}$ and each $y^i \in \mathcal{C}$, $\forall i \in \{1, \ldots n\}$. We use $\mathcal{Y} = \{y^1, \ldots, y^n\}$ to denote the vector of assigned labels for the time series.

Hence, time series classification can be seen as the task of learning a *classification function* from a dataset of d-dimensional time series $\mathcal{D}$ to a set of labels $\mathcal{C}$, such that the predicted class labels are as close as possible to the true time series class labels.

Definition 6 (*time series classification function*) Given a dataset $\mathcal{D}$ of n time series and a finite set of labels $\mathcal{C}$, a classification function is a mapping $f : \mathcal{D} \to \mathcal{C}$, such that for each $\mathcal{T}^i \in \mathcal{D}$

$$f(\mathcal{T}^i) = \hat{y}_i \in \mathcal{C}\ , \forall i \in \{1, \ldots, n\}\ .$$

Based on the above, the problem studied in this paper can be formulated as follows:

Problem 1 (*time series classification*) Given a time series learning set $\mathcal{Z}_{\{\mathcal{D},\mathcal{Y}\}}$ of size n, a finite set of labels $\mathcal{C}$, and a loss function $\mathcal{L}$, we want to learn a classification function $f : \mathcal{D} \to \mathcal{C}$ that minimizes $E_{(x,y)\in\mathcal{U}}[\mathcal{L}(y, f(x))]$, i.e., the expected loss on samples drawn from the (unknown) target distribution $\mathcal{U}$.

In this work, we will consider the 0/1 loss function:

$$\mathcal{L}(y, y') = \begin{cases} 0, & \text{if } y = y' \\ 1, & \text{otherwise.} \end{cases}$$

Our goal in this paper is to solve Problem 1 using this loss function for the case of multivariate time series, using the distance measures described above. The proposed approach is, however, generic enough for any type of data series, given a well-defined distance or similarity function.

4 Forests of generalized random shapelet trees

4.1 Background

Given a learning set, the standard decision tree learning algorithms (Quinlan 1993; Breiman et al. 1984) deterministically produce a classifier. For decision trees, however, it has been noted that minor changes of the learning set or parameter settings can have a fairly large impact on the resulting model. This variability can be remedied, or in fact benefited from, by generating sets of, rather than single, trees, as done by *ensemble methods*, among which bagging is one of the simplest (Breiman 1996). Other examples, such as random forests, apply randomization both to the learning set and the learning algorithm (Breiman 2001). In random forests, the first is done by employing bagging when building each tree and the latter by only selecting a (small) fixed number of random features to evaluate at each node.

In the traditional decision tree framework (Breiman et al. 1984), a shapelet tree, denoted as ST, is in essence generated by a combination of a decision tree learning algorithm and a feature extractor. In the case of multivariate time series, a shapelet $\mathcal{S}$ is extracted from a selected dimension, say k, of a selected time series, say $\mathcal{T}^i$; hence, it corresponds to a time series subsequence, i.e., $\mathbf{T}_k$. Since it is highly essential to keep track of the dimension k as well as the time series index i, from which a shapelet is extracted, a shapelet is denoted as $\mathcal{S}_{k,i}$. In the algorithm, each split consists of an shapelet and a distance threshold τ. The distance threshold is found by computing the distance between $\mathcal{S}_{k,i}$ and all n time series of the k^{th} dimension in the dataset $\mathcal{D}$, resulting in a list of n distances, i.e., $\{Sdist(\mathcal{S}_{k,i}, \mathbf{T}_k) : \mathbf{T}_k \in \mathcal{T} \in \mathcal{D}\}$ (Ye and Keogh 2009, 2011).

The distance list is discretized, similar to the way numerical features are handled in decision trees (Quinlan 1993), i.e., by sorting the distances and at each possible split point τ evaluating the impurity measure for a binary split. This results in partitioning the instances into two groups: one with minimum distance $\leq\tau$ and one with minimum distance $>\tau$. Let $\mathcal{D}^t$ denote the set of time series appearing in node t of the shapelet tree and let $\mathcal{S}_{k,i}$ be a candidate shapelet. Then, $\mathcal{D}^t$ is partitioned into two sets: $\mathcal{D}^t_L = \{\mathcal{T} : \mathbf{T}_k \in \mathcal{T} \in \mathcal{D}^t, Sdist(\mathcal{S}_{k,i}, \mathbf{T}_k) \leq \tau\}$ and $\mathcal{D}^t_R = \{\mathcal{T} : \mathbf{T}_k \in \mathcal{T} \in \mathcal{D}^t, Sdist(\mathcal{S}_{k,i}, \mathbf{T}_k) > \tau\}$. A valid split point is the mean of two consecutive distances for which the associated instances are labeled differently. To evaluate the goodness of a split at a particular node t, an impurity measure is used. The impurity is defined as

$$Im(\mathcal{S}_{k,i}, t) = I(t) - \frac{n_t^L}{n_t} I\left(t^L\right) - \frac{n_t^R}{n_t} I\left(t^R\right), \tag{3}$$

where $n_t^L = |\mathcal{D}_L^t|$ and $n_t^R = |\mathcal{D}_R^t|$ denote the sizes of the two partitions emerging from node t, n_t the total number of time series instances in node t, and $I(t)$ evaluates the goodness of a partitioning of node t. In this work, entropy is employed as goodness measure, i.e., $I_{entropy} = -\sum_{c \in \mathcal{C}} p(c|t) \log_2 (p(c|t))$ since it has been shown to work well for both single decision trees, random forests (Breiman 2001), shapelet trees (Ye and Keogh 2009), and logical shapelets (Mueen et al. 2011). The selected shapelet is, hence, the time series subsequence that achieves the highest decrease in impurity when splitting the learning set at node t into two disjoint partitions using a time series shapelet from the kth dimension as a split attribute, i.e., $\max_{\mathcal{S}_{k,i}} Im(\mathcal{S}_{k,i}, t)$.

The most important factor limiting practical use of shapelets as discriminative patterns is the computational cost of evaluating shapelet candidates, especially in the case of multivariate time series. To overcome this limitation, the number of evaluated candidates needs to be reduced. One way of improving the computational (and predictive) performance of decision tree ensembles is to only evaluate a small number of attributes at each node (Ho 1998). In a similar way, the computational cost of building shapelet trees can be reduced by only evaluating a constant (small) number of, e.g., randomly selected, candidates, r, at each internal node. To reduce the variance of the generalized random shapelet trees, we adopt an ensemble approach for generating a set $\mathcal{R} = \{ST_1, \ldots, ST_p\}$ of p random shapelet trees and combining their predictions using majority voting to determine the final class label. The proposed learning algorithm, as outlined in Algorithm 1, is elaborated in detail in the next section.

4.2 Generalized random shapelet forests

The generalized random shapelet forest (gRSF) algorithm (Algorithm 1) is a randomized ensemble method, which generates p generalized trees (using Algorithm 2), each built using a random selection of instances and a random selection of shapelets. In Algorithm 1, `Sample` draws a random sample of indices from the learning set $\mathcal{Z}$, which we denote as $\mathcal{Z}_{I_j}$ and defines the in-bag instances for the jth tree. Although, the function `Sample` can be implemented in a number of ways, the traditional bootstrap approach (Breiman 1996) is chosen here since, for each tree, several instances will be left out during training, i.e., the out-of-bag instances. The out-of-bag instances can subsequently be used to compute the performance of the forest using a subset of trees for which an instance is left out, enabling estimates of the running performance. Hence, `Sample` returns a vector of n indices drawn with replacement from the range $[1, n]$. Using this sample, the algorithm continues by generating the jth tree (sequentially or in parallel) using the function `RandomShapeletTree` and the instances included in the sample $\mathcal{Z}_{I_j}$.

Each random shapelet tree is constructed using Algorithm 2. In the algorithm, `TerminalNode` returns *true* if certain conditions, which may vary between implementations, are met. Common conditions for traditional random forests, which are also employed here, include that all instances are labeled with the same class label, i.e., the node is *pure*, or that the number of instance is below a certain threshold (here we adopt $|\mathcal{Z}| \leq 2$). If the termination condition is not met, Algorithm 2 continues by sampling

Algorithm 1: RANDOM SHAPELET FOREST($\mathcal{Z}$, p, l, u, r)

Input : $\mathcal{Z}$: a learning set, p: number of trees, l: lower shapelet length, u: upper shapelet length, r: number of shapelets.
Output : An ensemble of random shapelet trees $\mathcal{R} = \{ST_1, \ldots, ST_p\}$.

```
for j ← 1 to p do
    I_j ← Sample(Z);
    ST_j ← RandomShapeletTree(Z_{I_j}, l, u, r);
    R ← R ∪ {ST_j};
return R
```

r shapelets randomly from the learning set using the function `SampleShapelet`. This function, which introduces the second type of randomization mentioned above, can, again, be implemented in a number of ways of which we here adopt a straightforward approach. `SampleShapelet` is implemented to randomly and uniformly select a single time series from a randomly selected dimension k, $T_k^i \in \mathcal{Z}$ and extract a shapelet $S_{k,i}^{j,e}$ from T_k^i by uniformly selecting a length l in the range $e = rand([l, u])$ and a start position in the range $j = rand([1, m-l])$, where l and u denotes minimum and maximum shapelet sizes. Employing this function r times, where r is a parameter of the algorithm, results in a subset $\mathbf{S}$ of candidates, where $|\mathbf{S}| = r$.

Algorithm 2: RANDOM SHAPELET TREE($\mathcal{Z}$, l, u, r)

Input : $\mathcal{Z}$: a learning set, l: lower shapelet length, u: upper shapelet length, r: number of shapelets.
Output : A randomized shapelet tree ST.

```
if IsTerminal(Z) then
    return MakeLeaf(Z)
for 1 to r do
    S ← S ∪ {SampleShapelet(Z, l, u, rand(1, d))};
(τ, S, k) ← BestSplit(Z, Y, S);
(Z_L, Z_R) ← Distribute(Z, S, τ, k);
ST_L ← RandomShapeletTree(Z_L, l, u, r);
ST_R ← RandomShapeletTree(Z_R, l, u, r);
return {(τ, S, k, ST_L), (τ, S, k, ST_R)};
```

The `BestSplit` function determines which shapelet, $\mathcal{S} \in \mathbf{S}$, from the k:th dimension and distance threshold, τ, should be selected as the test condition at node t. The chosen test condition is subsequently used to separate the instances into two groups, those with a distance $Sdist(\mathcal{S}, \mathbf{T}_k) \leq \tau$ and those with a distance $Sdist(\mathcal{S}, \mathbf{T}_k) > \tau$. The utility of a split is determined by the information gain, i.e., Eq. 3, using entropy. To resolve conflicts in information gain, i.e., two or more thresholds with the same gain, we choose the threshold that maximizes the separation gap (Rakthanmanon and Keogh 2013):

$$\frac{1}{n_t^L} \sum_{\mathbf{T}_k \in \mathcal{D}_L} Sdist(\mathcal{S}, \mathbf{T}_k) - \frac{1}{n_t^R} \sum_{\mathbf{T}_k \in \mathcal{D}_R} Sdist(\mathcal{S}, \mathbf{T}_k). \tag{4}$$

The `SplitDataset` function partitions the instances according to the chosen split point, i.e., $\mathcal{Z}_L$ contains the instances with a distance less than or equal to τ and vice versa for $\mathcal{Z}_R$. The partitions are subsequently used to recursively build sub-trees. Finally, `MakeLeaf` returns a representation of a leaf in the generated tree by simply assigning the class label that occurs most frequently among the instances reaching the node, dealing with ties by selecting a label at random according to a uniform distribution.

4.2.1 Parameters and computational cost

From a computational point of view, the rational behind the random shapelet tree is that the cost of generating such trees is significantly lower than that of generating a traditional shapelet tree (Ye and Keogh 2009) and also lower than the cost of a Fast Shapelet Tree (Rakthanmanon and Keogh 2013) and a single shapelet ensemble tree (Cetin et al. 2015). Furthermore, from a bias-variance point of view, the rationale behind the random shapelet forest is that the randomization of the selected shapelets combined with the ensemble averaging is able to reduce the variance of variable base models and hence improve the generalization behaviour of the ensemble.

The total number of shapelet candidates in a 1-dimensional time series dataset with n instances of length m is $\mathcal{O}(nm^2)$. Exhaustively comparing all pairs of candidates of equal length thus requires a runtime of $\mathcal{O}(n^2m^4)$, which hence results in a computational complexity of $\mathcal{O}(n^2m^3 \log nm^3)$ for the original shapelet tree algorithm (Ye and Keogh 2009). For the fast shapelet algorithm, an exhaustive search is only performed for a subset of r shapelets, found using a SAX approximation, reducing the complexity to $\mathcal{O}(rnm^2)$ for a single node and $\mathcal{O}(n^2rm^2 \log nrm^2)$ for a full tree. Similarly, the computational cost for a generalized random shapelet tree is $\mathcal{O}(n^2rm^2 \log nrm^2)$. For both generalized random shapelet trees and fast shapelet trees the amortized computational cost is $\mathcal{O}(n^2m^2 \log nm^2)$, since r is constant in both cases. In practice, however, generating a random shapelet tree is faster then building a fast shapelet tree, since the former randomly samples shapelets in constant time, whereas the latter performs a more costly SAX approximation (Rakthanmanon and Keogh 2013). Finally, the shapelet ensemble tree approach (Cetin et al. 2015) utilizes several speed-up techniques that, in practice, significantly improves the computational cost compared to the fast shapelet algorithm and could also be used to improve the performance of random shapelet trees. The worst case run-time, however, remains unchanged compared to the original shapelet tree algorithm (Cetin et al. 2015).

As summarized in Table 1, the random shapelet forest algorithm has three parameters that require tuning: r (the number of shapelets randomly selected at each node) and $[l, u]$ (the range of allowed shapelet sizes defined as fractions of m). It also has a final parameter, p (the number of constructed trees in the ensemble) that for most practical purposes does not require tuning (larger is better). The parameters r, $[l, u]$, and p have different effects: r determines the strength of the shapelet selection procedure, where a low number of selected shapelets results in a relatively high variance of the base models; while p determines the strength of the variance reduction of the ensemble model averaging. These parameters could be selected to suite the application in a number of manual or automatic ways using, for example, cross-validation, or

Table 1 The parameters of the defined random shapelet forest

Parameter name	Abbr.	Values
No. of trees	p	$[1, \infty]$
No. of inspected shapelets	r	$[1, \sum_{l=1}^{m} nd(m-l+1)]$
Lower shapelet length	l	$(0, 1]$
Upper shapelet length	u	$(l, 1]$

employing the less costly out-of-bag error estimate. In Sect. 5.4, we empirically investigate the effect of different parameter configurations. Similar to what is commonly chosen for the traditional random forest algorithm, the default value for r is defined as the square-root of the total number of possible shapelets in a single time series, i.e., $\sqrt{md(md+1)/2}$, and the default values for l and u are set to include shapelets of all possible lengths, i.e., as limited by the training set.

5 Experimental evaluation

In this section, the empirical methodology is outlined together with the experimental design for evaluating the predictive performance of the gRSF algorithm compared to state-of-the-art univariate and multivariate time series classifiers. We also present experiments for exploring the effect of different parameter configurations for gRSF in terms of the bias-variance decomposition.

5.1 Experimental setup

For time series classification, the most common metric for evaluating the predictive performance of classifiers is by measuring the accuracy, i.e., the fraction of correctly classified instances (Hills et al. 2014; Ye and Keogh 2009; Gordon et al. 2012). We note, however, that other measures, such as the area under ROC curve (Bradley 1997) can be more suitable in some cases, e.g., when the class distribution differ between training and testing.

To explore and analyze how different parameter configurations affect the predictive performance, internal estimates for computing the average strength and the level of correlation in predictions by the individual classifier (based on the out-of-bag performance) are employed. For completeness, the definitions of these metrics are given in Appendix 1. Briefly, the strength measures the accuracy of each classifier in the ensemble based on the margin, i.e., the average number of votes for the right class that exceeds the average number of votes for any other class; and the correlation measures the dependence between classifiers. To further explore the gRSF, we also consider the bias/variance decomposition and compare how different parameter configurations impact the predicted probabilities. To make the paper self-contained, Appendix 1 provides a decomposition of the bias and variance in terms of the posterior probabilities assigned by the forest.

5.1.1 Hypothesis testing

The generally accepted procedure for inferring significant differences in terms of, e.g., classification accuracy, when comparing multiple classifiers over many datasets is the non-parametric Friedman test based on ranks (Demšar 2006). Using this test, the best performing algorithm receives a rank of 1 (for a particular dataset), the second best received rank 2, and so on, while ties are resolved by assigning the average rank. If the Friedman test allows for rejecting the null hypothesis (stating that there are no differences in performance between the methods), then a post-hoc test, e.g., a Nemenyi test, may be employed for identifying pairs of methods for which the difference is significant. Following Demšar (2006), the latter can be visualized by depicting the ranks of classifiers as points on a horizontal line, connecting points for which the difference is not significant, i.e., where the difference between the algorithms is less than a critical distance, which is dependent on the chosen level of significance. For a more complete description of the employed statistical tests, the reader may consult Demšar (2006).

5.1.2 Datasets

To evaluate the classification accuracy and run-time of the gRSF algorithm compared to the state-of-the-art *univariate* time series classifiers, we have selected 56 diverse datasets (see Tables 2 and 3). The majority of the datasets are from the UCR repository (Keogh et al. 2015) and the rest are from Lines et al. (2012).[2] The datasets cover a range of domains, commonly grouped into four categories: image outline classification, motion classification, sensor reading classification and simulated classification problems (Lines and Bagnall 2014). Similarly, to evaluate the performance of gRSF against the state-of-the-art *multivariate* time series classifiers, we have selected 14 datasets covering domains such as image outline classification, motion classification and sensor reading classification (Baydogan and Runger 2015).

5.1.3 Parameter optimization protocol

To avoid selection of sub-optimal parameter values for the baseline approaches, the results of gRSF are compared to results previously reported in the literature using the already provided training and test sets, for which typically the hyper-parameters have been optimized. For the presented method and the UTS datasets, we perform a grid-search over some of the parameters enumerated in Table 1 using the out-of-bag error rate of the gRSF, i.e., using only the training data, to identify the best performing configuration. To resolve conflicts between configurations with the same out-of-bag error rate, the correlation and strength square ratio, i.e., $\bar{p}/s^2$, as defined in Appendix 1, is employed. To simplify replication of the experiment, the finally selected parameter configurations are listed in Tables 2 and 3. Since the generalization error of an ensemble is expected to decrease with an increasing number of members

[2] All currently available datasets in both repositories have been included.

Table 2 The predictive performance, as measured by accuracy, of gRSF compared to LTS, shapelet transformations and cDTW

Dataset	Parameters			Accuracy				
	l	u	r	cDTW	FST	SVM	LTS	gRSF
Adiac	0.900	1.000	500	0.609	0.159	0.283	0.542	**0.742**
Beef	0.025	0.100	500	0.533	0.567	**0.867**	0.800	0.800
Beetle/fly	0.025	0.200	10	0.650	0.900	**0.975**	0.950	**0.975**
Bird/chicken	0.025	0.400	10	0.725	0.900	0.950	**1.000**	0.975
CholorineC	0.200	0.500	500	0.650	0.535	0.562	**0.743**	0.673
Coffee	0.200	0.500	50	0.821	**1.000**	**1.000**	**1.000**	**1.000**
DiatomSize	0.900	1.000	50	0.935	0.765	0.922	0.952	**0.964**
DP_Little	0.025	0.400	500	0.439	0.603	**0.752**	0.727	**0.752**
DP_Middle	0.025	0.400	500	0.546	0.619	**0.796**	0.752	0.767
DP_Thumb	0.025	0.400	500	0.530	0.560	0.698	0.740	**0.755**
ECGFiveDays	0.025	0.300	100	0.797	0.990	0.990	**1.000**	**1.000**
FaceFour	0.025	0.400	10	0.886	0.750	0.977	**1.000**	**1.000**
GunPoint	0.025	0.400	10	0.913	0.953	**1.000**	**1.000**	**1.000**
ItalyPower	0.900	1.000	500	0.955	0.931	0.921	**0.962**	0.940
Lightning7	0.700	1.000	100	0.712	0.411	0.699	**0.877**	0.699
MedicalImages	0.800	1.000	def.	**0.747**	0.508	0.525	0.734	0.733
MoteStrain	0.025	0.100	def.	0.866	0.840	0.887	0.913	**0.921**
MP_Little	0.025	0.500	500	0.558	0.578	0.750	0.758	**0.761**
MP_Middle	0.025	0.400	500	0.470	0.609	0.769	0.770	**0.786**
Otoliths	0.025	0.300	100	0.594	0.578	0.641	**0.760**	0.594
PP_Little	0.025	0.300	500	0.495	0.586	0.721	0.710	**0.727**
PP_Middle	0.025	0.400	500	0.499	0.581	0.759	0.767	**0.781**
PP_Thumb	0.025	0.400	500	0.526	0.591	**0.755**	0.715	0.699
SonyAIBO	0.300	0.600	100	0.695	0.953	0.867	**0.952**	0.925
Symbols	0.025	0.400	1	0.938	0.801	0.846	0.959	**0.968**
SyntheticControl	0.300	0.600	500	0.983	0.957	0.873	**1.000**	**1.000**
Trace	0.025	0.300	100	0.990	0.980	0.980	**1.000**	**1.000**
TwoLeadECG	0.025	0.200	10	0.868	0.970	0.993	**1.000**	**1.000**
Average	–	–	–	0.712	0.721	0.813	0.860	0.855
Rank	–	–	–	3.969	4.234	3.000	1.922	1.875

The parameters (*def.* denotes default value) for gRSF was optimized using the out-of-bag accuracy and the $\bar{d}/s^2$-ratio (see Sect. 5.1)
For each dataset, the highest accuracy is highlighted in bold

(cf. Fig. 6), the forest size is kept static at $p = 500$, which has been shown to be a suitable forest size for traditional random forests (Boström 2011). For univariate time series, the following configurations of upper and lower shapelet length, i.e., l and u, are used in the optimization: $l = 0.025, u = \{0.1, 0.2, 0.3, 0.4\}$; $l = 0.2, u = 0.5$; $l = 0.3, u = 0.6$ and $l = \{0.6, 0.7, 0.8, 0.9\}, u = 1$, i.e. 10 possible configurations.

Table 3 The predictive performance, as measured by accuracy, of gRFS compared to LTS, cDTW and SAX

Dataset	Parameters			Accuracy			
	l	u	r	cDTW	SAX	LTS	gRSF
50words	0.700	1.000	def.	0.758	0.443	**0.768**	0.738
Adiac	0.900	1.000	500	0.609	0.486	0.542	**0.742**
Beef	0.025	0.100	500	0.533	0.553	**0.800**	**0.800**
CBF	0.025	0.400	100	0.996	0.947	0.994	**0.999**
ChlorineC	0.200	0.500	500	0.650	0.583	**0.743**	0.673
CinCECG-torso	0.600	1.000	500	**0.930**	0.826	0.833	0.841
Coffee	0.200	0.500	50	0.821	0.932	**1.000**	**1.000**
CricketJX	0.200	0.500	50	0.764	0.473	**0.791**	0.759
Cricket.Y	0.300	0.600	500	**0.803**	0.495	0.751	0.764
Cricket_Z	0.300	0.600	500	**0.820**	0.453	0.799	0.769
DiatomSize	0.900	1.000	50	0.935	0.883	0.952	**0.964**
ECG200	0.600	1.000	def.	**0.880**	0.773	0.874	0.830
ECGFiveDays	0.025	0.300	500	0.797	0.996	**1.000**	**1.000**
FaceAll	0.200	0.500	1	**0.808**	0.589	0.782	0.763
FaceFour	0.200	0.400	10	0.886	0.910	**1.000**	**1.000**
FacesUCR	0.800	1.000	def.	0.912	0.672	**0.941**	0.870
Fish	0.025	0.200	def.	0.840	0.803	0.934	**0.960**
GunPoint	0.025	0.400	10	0.913	0.939	**1.000**	**1.000**
Haptics	0.900	1.000	500	0.412	0.384	0.468	**0.474**
InlineSkate	0.300	0.600	500	0.387	0.259	**0.427**	0.405
ItalyPower	0.900	1.000	100	0.955	0.905	**0.962**	0.940
Lighting2	0.900	1.000	def.	0.869	0.705	**0.869**	0.803
Lighting7	0.700	1.000	100	0.712	0.597	**0.877**	0.699
MALLAT	0.600	1.000	1	0.914	**0.967**	0.954	0.945
MedicalImages	0.800	1.000	def.	**0.747**	0.567	0.734	0.733
MoteStrain	0.025	0.100	def.	0.866	0.783	0.913	**0.921**
OliveOil	0.300	0.600	1	0.833	0.787	0.440	**0.867**
OSULeaf	0.025	0.200	def.	0.616	0.641	0.818	**0.913**
SonyAIBO	0.300	0.600	100	0.695	0.686	**0.952**	0.925
SonyAIBOII	0.025	0.200	500	0.859	0.785	**0.918**	0.897
SwedishLeaf	0.025	0.300	def.	0.843	0.731	**0.913**	0.931
Symbols	0.025	0.300	1	0.938	0.932	0.959	**0.968**
SyntheticControl	0.300	0.600	50	0.983	0.919	**1.000**	**1.000**
Trace	0.025	0.300	100	0.990	0.998	**1.000**	**1.000**
TwoLeadECG	0.025	0.200	100	0.868	0.910	**1.000**	**1.000**
TwoPatterns	0.600	1.000	def.	0.998	0.887	0.997	**0.999**

Table 3 continued

Dataset	Parameters			Accuracy			
	l	u	r	cDTW	SAX	LTS	gRSF
uWav_X	0.300	0.600	100	0.773	0.707	**0.800**	**0.800**
uWav_Y	0.600	1.000	def.	0.699	0.608	0.713	**0.718**
uWav_Z	0.200	0.500	50	0.678	0.636	0.731	**0.744**
Wafer	0.025	0.100	500	0.995	0.996	0.996	**1.000**
WordsSynonyms	0.700	1.000	1	**0.748**	0.406	0.660	0.622
Yoga	0.025	0.300	def.	0.845	0.731	0.850	**0.851**
StarLightCurves	0.025	0.400	50	0.905	0.937	0.967	**0.977**
Thorax1	0.025	0.400	def.	0.815	0.754	0.900	**0.919**
Thorax2	0.025	0.400	100	0.871	0.789	0.850	**0.934**
Average	–	–	–	0.810	0.728	0.847	0.854
Rank	–	–	–	2.689	3.667	1.889	1.756

The parameters (*def.* denotes default value) for gRSF was optimized using the OOB accuracy and the $\bar{d}/s^2$-ratio (see Sect. 5.1)
For each dataset, the highest accuracy is highlighted in bold

Furthermore, for multivariate time series, we here opt for using the default values for upper and lower shapelet length, i.e., all possible lengths. For both univariate and multivariate time series, the number of inspected shapelets is set to 1, 10, 50, 100, 500 and $\sqrt{md(md+1)/2}$, yielding a total of 60 evaluated parameter configurations for univariate and 6 parameter configurations for multivariate time series. By using the out-of-bag error rate, we avoid more costly strategies, such as employing k-fold cross-validation.

5.2 Competing approaches

In this section, we list the baseline algorithms to which the presented approach is compared. In the univariate case, all competing algorithms, except cDTW, are based on shapelets. For multivariate time series, all alternative approaches are based on time series features extraction.

5.2.1 Nearest neighbours (univariate/multivariate)

The most widely adopted time series classifier is the nearest neighbour classifier, which have predominantly and successfully been used together with the constraint dynamic time warping distance measure (Sakoe and Chiba 1978; Ratanamahatana and Keogh 2004). We denote this approach cDTW. The nearest neighbour classifier requires the number of nearest neighbours k and DTW requires one additional parameter: the width of the band. Here, we adopt $k = 1$ using a cross-validation optimized DTW constraint. We opt for not optimizing the number of nearest neighbours, since the gain is often minimal (Bagnall and Lines 2014). For the univariate case, the predictive performance for cDTW in Tables 2 and 3 are taken from Keogh et al. (2015). For

the multivariate case, the distance between two MTS is the sum of DTW distances between the associated UTS, similar to Baydogan and Runger (2014).

5.2.2 Shapelet trees (univariate)

Here we adopt the shapelet-based decision tree classifier with the F-stat measure (FST) proposed by Hills et al. (2014), since, in their work, the F-stat measure is able to both generate trees slightly faster and provide slight benefits in terms of classification accuracy compared to the traditional shapelet tree. The shapelet tree classifier requires two parameters: the upper and lower shapelet size, which were optimized using a subset of time series (Hills et al. 2014). The predictive performance in Table 2 is taken from Hills et al. (2014).

5.2.3 Fast shapelets (univariate)

The fast shapelet algorithm, which we denote SAX, uses symbolic aggregate approximations (SAX) to reduce the dimensionality of the shapelet search problem, improving the run-time of the shapelet-based decision tree algorithm with several orders of magnitude. The employed fast shapelet algorithm requires four parameters: the upper and lower shapelet size, the top-k shapelets for the approximations and a $step$-size. Additionally, SAX requires a number of parameters, e.g., the alphabet size. Here, we use the default setting. Predictive performance in Table 3 is taken from Rakthanmanon and Keogh (2013). Note that the predictive performance of SAX is not included in Table 2, since it provides an approximation of the single shapelet tree (FST) and, hence, gives similar results. Similarly, the single shapelet tree is not included in Table 3 due to computational constraints, instead we here include the approximation provided by SAX. Since the shapelet based ensemble (Cetin et al. 2015), provide comparable results to SAX, we here opt to include only the latter.

5.2.4 Shapelet transformations (univariate)

We also consider shapelet transformations (Hills et al. 2014). Since the transformation is independent from the used classifier, we here opt for the classifier that has been shown to give the best results, namely support vector machines (Hills et al. 2014). We refer to this approach as SVM. The transformation requires two parameters; the top-k shapelets to include and the number of clusters. Additionally, the chosen classifier might have parameters of its own that require tuning, e.g., kernel for SVMs. The predictive performance in Table 2 is taken from Hills et al. (2014), using a linear kernel.

5.2.5 Learning shapelets (univariate)

Finally, for the univariate case, we consider learning shapelets (LTS) (Grabocka et al. 2014), which instead of enumerating shapelets from a restricted pool of candidates in the learning set, consider them as parameters in an optimization problem. The LTS algorithm requires six hyper-parameters; the learning rate η, regularization λ, the

Table 4 Predictive performance, as measured by accuracy, of gRSF compared to SMTS, cDTW, UFS, and LPS for the multivariate time series datasets

Dataset	Parameters	Accuracy				
	r	cDTW	SMTS	LPS	UFS	gRSF
ArabicDigits	100	0.908	0.964	0.971	0.964	**0.975**
AUSLAN	500	0.762	0.947	0.754	**0.972**	0.955
CharacterT.	500	0.960	0.992	0.965	0.993	**0.994**
LIBRAS	500	0.800	0.909	0.903	0.849	**0.911**
ECG	10	0.850	0.818	0.820	0.862	**0.880**
CMU_MOCAP_S16	10	0.931	0.997	**1.000**	**1.000**	**1.000**
uWaveGestureLibrary	10	0.929	0.941	**0.980**	0.929	0.929
Wafer	500	0.977	0.965	0.962	0.976	**0.992**
Japanes Vowels	500	0.649	**0.969**	0.951	0.932	0.800
KickvsPunch	500	0.900	0.850	0.900	0.700	**1.000**
Walkvsrun	500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NetworkFlow	10	0.712	**0.974**	0.965	0.891	0.914
PEMS	100	0.832	0.896	0.844	0.988	**1.000**
Pendigits	10	0.912	0.917	0.931	0.919	**0.932**
Average	–	0.866	0.939	0.925	0.927	0.949
Rank	–	4.177	3.036	2.964	2.964	1.857

For each dataset, the highest accuracy is highlighted in bold

number of iterations $iter$, the soft max parameter α, the scales of pattern lengths r and number of latent patterns k. The predictive performance using optimized parameters in Table 2 is taken from Grabocka et al. (2014) and in Table 3 from their supporting website.[3]

5.2.6 Symbolic representation for multivariate time series (multivariate)

For MTS, we consider a learned symbolic representation for multivariate time series (Baydogan and Runger 2014), which we will refer to as SMTS. SMTS provides a symbolic representation of a multivariate time series based on time index and values which are exploited to generate time segmented features using the terminal nodes in a decision forest. The features are subsequently exploited for making predictions using a second ensemble. The predictive performance in Table 4 is taken from (Baydogan and Runger 2014).

5.2.7 Learned pattern similarity (multivariate)

For the multivariate case, we consider learned pattern similarity (LPS), which models the dependency structure of time series based on the concept of local autopatterns, i.e.

[3] http://fs.ismll.de/publicspace/LearningShapelets.

time segments, learned using an ensemble of regression trees. A similarity measure based on these patterns are subsequently used to make predictions using a nearest neighbour approach (Baydogan and Runger 2015). The feature extraction algorithm has two parameters; the number of regression trees and the number of segments. In Table 4, the predictive performance is taken from Baydogan and Runger (2015).

5.2.8 Ultra fast shapelets (multivariate)

Finally, we also include the the ultra-fast shapelet transformation (UFS) approach, which models the input data as a set of distances from randomly chosen shapelets (Wistuba et al. 2015). The algorithm has only one parameter; the number of random shapelets to sample. The predictive performance in Table 4 is in part taken from (Wistuba et al. 2015) and in part from experiments using our own implementation[4] with the recommended parameter configurations (Wistuba et al. 2015).

5.3 Empirical results

The performance of the gRSF algorithm is compared to the competing approaches both in terms of classification accuracy and run-time. The results are presented in three stages: in Sect. 5.3.1, we compare the predictive performance of gRSF against the competing approaches described in Sect. 5.2 for both *univariate* and *multivariate* time series. For univariate time series the comparison is first conducted for a subset of datasets commonly used when evaluating shapelet classifiers and then for the full set of datasets. In Sect. 5.3.2, the computational performance and scalability of gRSF is evaluated and compared to alternative methods. Finally, in Sect. 5.4, we investigate the effect of gRSF hyper-parameters in terms of the bias-variance decomposition.

5.3.1 Predictive performance

Univariate time series classification Since the computational cost for shapelet based classifiers have been a limiting factor for full utilization of the algorithms, most such classifiers (see Hills et al. 2014; Grabocka et al. 2014) have been evaluated on a subset of smaller datasets only. More specifically, 17 datasets from the UCR time series repository (Keogh et al. 2015) and 11 datasets from Hills et al. (2014) have usually been selected. To give an overview, the results for all algorithms and datasets are presented in Table 2. Statistical tests for determining whether differences in predictive performance are significant or not are undertaken by comparing the performance ranks of the algorithms.

In Table 2, we can see the classification accuracy of the different methods. By inspecting the ranks of the individual algorithms, on average gRFS and LTS are ranked second (with ranks 1.87 and 1.92, respectively) and SVM third (3) and cDTW and FST fourth (3.97 and 4.23). A significance test reveals that the observed differences in accuracy ranks significantly ($p < 0.001$) deviate from what can be expected under the

[4] Available at the supporting website.

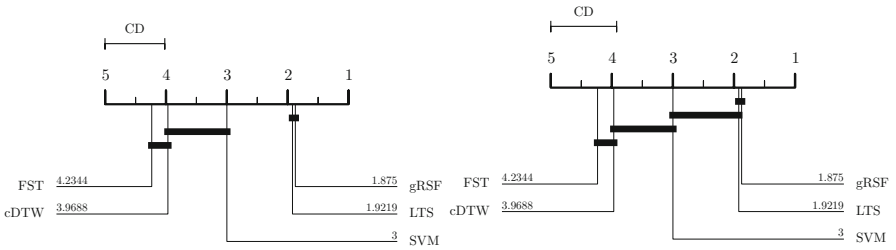

Fig. 2 Comparison of pair-wise Nemenyi tests for all classifiers used for the 28 univariate datasets in Table 2. Classifiers that are not significantly different at $p = 0.1$ (*left*) and at $p = 0.05$ (*right*) are connected

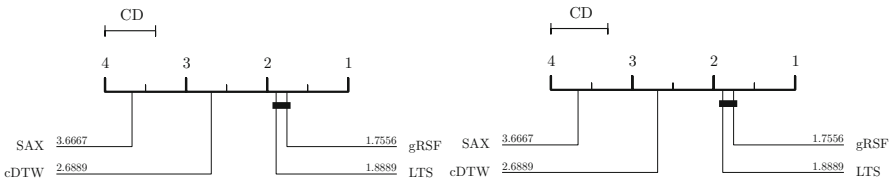

Fig. 3 Comparison of pair-wise Nemenyi tests for all classifiers used for the 45 univariate datasets in Table 3. Classifiers that are not significantly different at $p = 0.1$ (*left*) and at $p = 0.05$ (*right*) are connected

null hypothesis. For a significance level of 0.05 and 0.1, the critical distance is 1.15 and 1.04 respectively. As seen in Fig. 2, a post-hoc Nemenyi test reveals that there is a significant difference between gRSF when comparing against single shapelet-based decision trees (FST), cDTW and SVM for $p < 0.05$ and between gRSF and LTS compared to FST and cDTW for $p < 0.1$. Thus, for the lower significance levels, we can identify two main groups of algorithms; the performance of FST and cDTW are significantly worse than that of gRSF and LTS. The SVM approach, however, falls in between of the two groups; it cannot clearly be separated from the worst (or best) performing group at the lowest significance level. Since the difference between gRSF and LTS is minimal, the empirical findings for univariate time series classification indicate that when aiming for highest accuracy, either gRSF or LTS can be safely recommended.

Since the computational cost for gRSF is generally lower than that of other shapelet based algorithms, e.g., LTS (see Sect. 5.3.2), the second experiment include all 45 datasets from the UCR repository. In Table 3, the results for these datasets are presented for the gRSF algorithm and three state-of-the-art classifiers able to cope with these larger time series datasets, namely: cDTW, SAX, and LTS. As can be seen in Table 3, the two best performing classifiers, in terms of accuracy, are again the LTS and gRSF algorithms. This is also confirmed by the average performance ranks, where gRSF and LTS are ranked second (with ranks 1.76 and 1.89, respectively), while cDTW ranks 2.69 and SAX 3.67. For these datasets, the observed differences in accuracy ranks deviate significantly ($p < 0.001$) from what can be expected under the null-hypothesis of no difference.

To investigate if there are any significant differences between pairs of classifiers, a post-hoc Nemenyi test is again performed (Fig. 3). Given the critical distances for

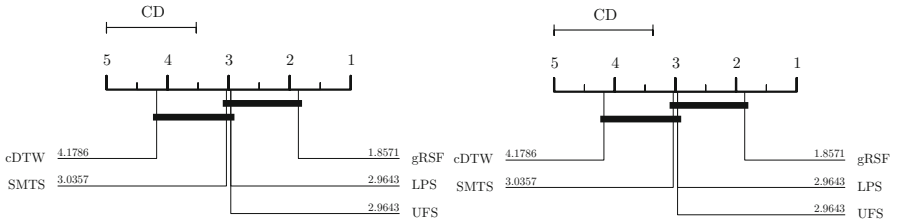

Fig. 4 Comparison of pair-wise Nemenyi tests for all MTS classifiers over the 14 multivariate datasets in Table 4. Classifiers that are not significantly different at $p = 0.1$ (*left*) and at $p = 0.05$ (*right*) are connected

the significance levels of 0.05 and 0.1 as 0.69 and 0.62, the post-hoc test reveals that there is no significant difference between gRSF and LTS for both significance levels. However, SAX is significantly less accurate than all alternatives (again, for both significance levels). Furthermore, at both significance levels, cDTW is less accurate than both LTS and gRSF. Partitioning the algorithms into groups for $p < 0.05$, we can identify two groups; the performances of SAX and cDTW are significantly worse than gRSF and LTS. Thus, since gRSF and LTS cannot be separated in terms predictive performance, either is expected to outperform the alternatives.

Multivariate time series classification Univariate time series classification is a special case of multivariate time series classification where each instance is described by a single time series. In many domains and settings, where more than one variable per instance evolves over time, this is however overly simplistic. Table 4 presents the accuracy for the evaluated multivariate time series classifiers. Looking at the accuracies, we can see that the gRSF algorithm predominantly performs either better or on par with the competing approaches. By inspecting the ranks of the individual algorithms, gRSF is ranked highest (with rank 1.86) and cDTW ranked last (with rank 4.18) and LPS, UTF and SMTS in between (with ranks close to 3). A Friedman test reveals that the observed differences in accuracy ranks deviate significantly ($p < 0.001$) from what can be expected under the null-hypothesis of no difference.

To investigate if there are any significant differences between pairs of algorithms, a post-hoc Nemenyi test is again performed (see Fig. 4). The post-hoc test does not indicate any significant differences between gRSF, LPS, UFS and SMTS for neither the 0.05 nor 0.1 significance levels. cDTW is, however, significantly outperformed by gRSF ($p < 0.05$), but not by LPS, UFS or SMTS. Partitioning the algorithms into groups for $p < 0.05$, we can identify two groups: the performance of cDTW is significantly worse than for gRSF. Since neither LPS, UFS or SMTS clearly belong to any of these groups, the experimental data is not sufficient to reach any conclusions regarding the relative performance of LPS, UFS or SMTS for multivariate time series classification.

5.3.2 Computational performance

In this section, we show that in addition to achieving state-of-the-art predictive performance for univariate and multivariate datasets, the gRSF algorithm is also competitive

in terms of run-time performance. In the univariate case, we include shapelet based classifiers that have been designed for either speed (SAX) or predictive performance (LTS) and show that the gRSF is significantly faster than the latter and perform similar to, albeit is significantly more accurate, than the former. In the multivariate case, we include the classifiers with highest predictive performance, i.e., LPS, UFS and gRSF, and show that the run-time of gRSF is comparable to or better than the alternative approaches. For both UTS and MTS classification, the algorithms have approximately the same number of hyper-parameters and would, hence, be penalized in similar ways when performing grid search for the best setting. This cost cannot, however, be directly compared and is therefore left out.

In the experiments, we report training time relative to that of the gRSF algorithm, hence, a number larger than 1 indicates slower training than gRSF while a number smaller than 1 indicates faster training than gRSF. Since the training time for LPS only includes feature extraction and classification is instance based, i.e., has zero training time, we include both training and testing time in the multivariate case. For the comparison, we select a number of UTS datasets with representative properties and all available MTS datasets.

Univariate time series classification In the comparison of UTS classifiers, we have included SAX and LTS and used the default settings for all algorithms. For LTS this amounts to: $\alpha = -30$, $\eta = -0.1$, $iter = 1000$ and $\lambda = 0.01$, for gRSF: $r = \sqrt{md(md+1)/2}$, $l = 0.025$, $u = 1$ and $p = 500$ and finally, for SAX: $step = 1$ and $topk = 10$. Table 5 shows that in the univariate case, gRSF is systematically at least three times faster than LTS, and in some cases also faster than SAX.[5] On average, over the selected datasets, the performance ratio shows that the gRSF algorithm is approximately 45 times faster than LTS and has a similar performance to SAX-based decision trees. Note, however, that we employ a rather large forest (500 trees) in the comparison, but a smaller one (consisting of less than 100 trees), which is generally faster to train, in most cases still outperforms SAX in terms of classification accuracy (see Karlsson et al. 2015).

Multivariate time series classification Comparing the MTS classifiers in terms of computational cost, we use 500 trees and 100 shapelets for gRSF; 500 trees and 10 segments for LPS and 10,000 shapelets for UFS.[6] Similarly to the univariate case, Table 6 shows that in the multivariate case, gRSF is competitive in terms of computational cost. On average, the gRSF algorithm is six times faster than UFS and eight times faster than the LPS algorithm.[7] One reason for the strong performance of gRSF in the multivariate case, compared to UFS, is that a dynamic (often lower) number of shapelets are required to construct a tree. Another, related, reason is that gRSF only computes the similarity between shapelets and time series from the same dimensions, whereas UFS finds the minimum distance to any dimension, increasing the number of comparisons with a factor d.

[5] We note, however, that since gRSF and LTS are distributed over several cores, the comparison to the non-parallel fast shapelet algorithm is not entirely fair.

[6] For the run-time experiment, we limit the total number of shapelets to 10,000 for computational convenience, i.e., reducing the cost of UFS by a factor d.

[7] Note, however, that the LPS algorithm is run on a single core.

Table 5 Relative training time of gRSF compared to LTS and SAX for univariate time series

Dataset	Properties			Runtime		
	l	n	m	SAX	LTS	gRSF
Adiac	37	390	176	1.19	419.39	**1.00**
DiatomSizeReduction	4	16	345	**0.64**	4.70	1.00
ECGFiveDays	2	23	136	**0.91**	3.44	1.00
fish	7	175	463	**0.11**	6.94	1.00
Gun_Point	2	50	150	**0.76**	2.15	1.00
MedicalImages	10	381	99	2.22	43.60	**1.00**
MoteStrain	2	20	84	**0.50**	3.29	1.00
OSULeaf	6	200	427	**0.81**	3.50	1.00
synthetic_control	6	300	60	2.65	30.11	**1.00**
Trace	4	100	275	**0.94**	4.38	1.00
wafer	2	1000	152	**0.11**	2.45	1.00
Average	–	–	–	**0.98**	47.63	1.00

For each dataset, the fastest method is highlighted in bold

Table 6 Relative training and testing time of gRSF compared to UFS and LPS for multivariate time series

Dataset	Properties				Runtime		
	l	d	n	m	LPS	UFS	gRSF
ArabicDigits	10	13	6600	144	8.06	**0.38**	1.00
AUSLAN	95	22	1140	45–136	2.93	1.59	**1.00**
CharacterTrajectories	20	3	300	109–205	**0.97**	1.27	1.00
LIBRAS	15	2	180	45	11.44	**0.22**	1.00
ECG	2	2	100	39–152	9.03	**0.17**	1.00
CMU_MOCAP_S16	2	62	29	127–580	6.58	23.70	**1.00**
uWaveGestureLibrary	8	3	200	315	**0.53**	2.49	1.00
Wafer	2	6	298	104–198	3.99	2.23	**1.00**
Japanes Vowels	9	12	270	7–79	4.59	1.10	**1.00**
KickvsPunch	2	62	16	274–841	6.60	12.31	**1.00**
Walkvsrun	2	62	28	128–1918	1.78	7.92	**1.00**
NetworkFlow	2	4	803	50-997	**0.15**	0.16	1.00
PEMS	7	963	267	144	51.30	36.87	**1.00**
Pendigits	10	2	300	8	8.50	2.95	**1.00**
Average	–	–	–	–	8.32	6.67	1.00

For each dataset, the fastest method is highlighted in bold

To confirm the theoretical run-time (see Sect. 4.2.1), the computational performance of gRSF is also empirically evaluated in terms of how well the algorithm scales with increasing time series length (m) and increasing number of time series in the data

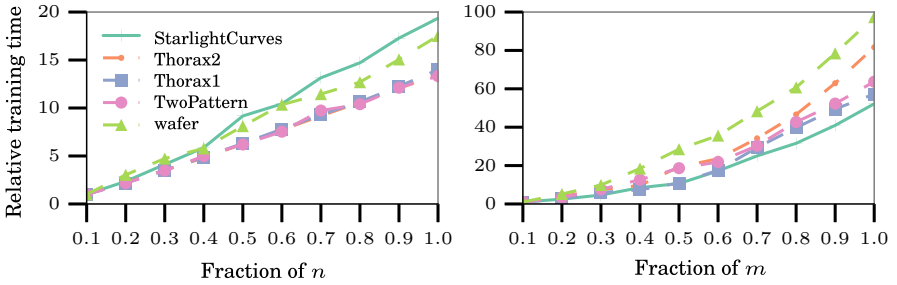

Fig. 5 The reported relative training time (*y-axis*) as a function of increasing time series length (m) (*right*) and of increasing number of time series (n) (*left*)

set (n). For the former, univariate datasets with $m > 1000$ time points, and for the latter univariate datasets with $n > 1000$ instances were selected. Since the run-time of gRSF is unaffected by additional dimensions, only UTS datasets are used. In Fig. 5, the computational performance of running the gRSF algorithm is shown as the size of the training set (left) and time series length (right) increases with an increasing multiplier of $0.1, \ldots, 1$. As we can see, the gRSF algorithm tends to scale approximately linearly in the size of the learning set, i.e. n, and quadratically in time series length, i.e. m.

5.4 The effect of parameters

In this section, we analyze and discuss the effect of the random shapelet forest parameters, including the number of shapelets r, the shapelet length $[l, u]$ and finally the averaging strength of adding more trees to the ensemble, i.e., p. Since the effect of parameters is independent from the number of time series dimensions, we limit the analysis to univariate time series databases.

5.4.1 The shapelet selection strength r

The parameter r denotes the number of random shapelets to evaluate at each node of the forest. For a given problem, the smaller r, the greater the randomization of the trees and, hence, the weaker the output value depends on the structure of the tree. In the extreme, with $r = 1$, the tree is built using a single shapelet sampled independently from the target label, i.e., the trees are grown randomly in an uninformed manner. Figure 6, shows the evolution of the out-of-bag error rate for 9 time series classification data sets as we increase r. The default value of r is shown as a vertical line. In the figure, we see a number of trends, e.g., decreasing (*Fish*, *Adiac*, *OSULeaf*) and increasing (*FaceFour*). For these, the default value is clearly not the optimal choice. For the other data sets, however, the default value provides a good starting point for optimization. Since r provides a mechanism for balancing strength and correlation, it is an important hyper-parameter.

On average, over all datasets, when $r = 500$, the strength (s) and correlation ($\bar{d}$) (see Appendix 1) is $s = 0.36$ and $\bar{d} = 0.12$ and when $r = 1$, $s = 0.26$ and $\bar{d} = 0.1$, i.e., as expected, increasing r strongly increases the strength, but only slightly increases the

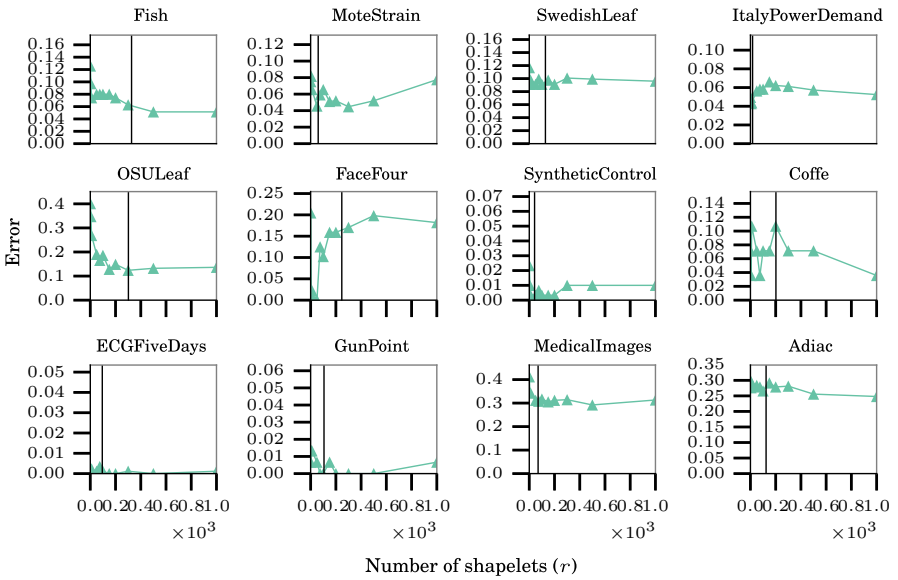

Fig. 6 The evolution of the out-of-bag error rate with $l = 0.025$, $u = 1$ on 9 time series classification tasks. The *vertical line* denotes the default value of $r = \sqrt{md(md+1)/2}$

correlation. By analyzing the bias, Fig. 7 (left) shows that it decreases as r increases, from 0.2 when $r = 1$ to 0.15 when $r = 500$. Similarly, as r increases the variability of the forests decreases, from 0.55 with $r = 1$ to 0.46 with $r = 500$. One reason for the increase in bias for forests with more randomization could be the fact that the tree generation results in trees predominantly voting for the majority class, hence producing highly biased probability estimates.

5.4.2 The shapelet sampling effect of l and u

The parameters l and u denote the fraction of shapelet lengths sampled. For a given problem, l and u defines the search space for suitable features and guides the algorithm towards global or local similarities. In the default case, every possible shapelet length is a possible candidate, but the choice can be specialized using the out-of-bag error rate. In Fig. 7 (right) the average out-of-bag error rate and bias and variance are shown for the average shapelet length when considering $r = \sqrt{md(md+1)/2}$ shapelets at each node.

On average, the error is highest when limiting the search to very short or very long shapelets, indicating that a wider sampling range is needed to get sufficient variability among the trees. An analysis of the bias and the variance, as shown in Fig. 7 (right), shows that the bias decreases from 0.47 when limiting the search to short shapelets to 0.34 when increasing the sampled shapelet sizes. Similarly, the variance decreases as we limit the search to longer shapelets. As seen in Fig. 7 (right), the setting that minimizes the trade-off between bias and variance is, however, inspecting shapelets shapelets of all possible lengths, explaining its superior performance on average. We

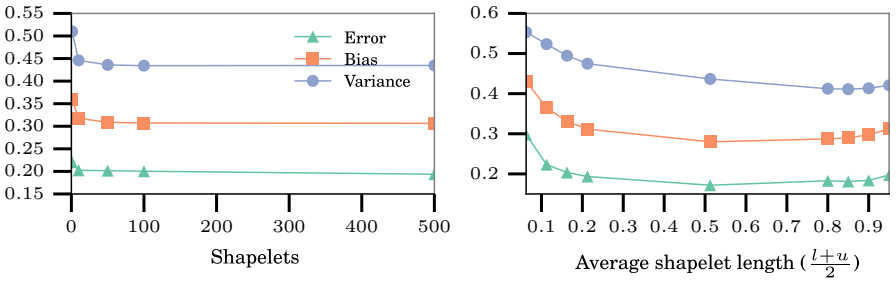

Fig. 7 The shapelet sampling effect of l and u (*right*) and the average effect of increasing the number of shapelets r (*left*)

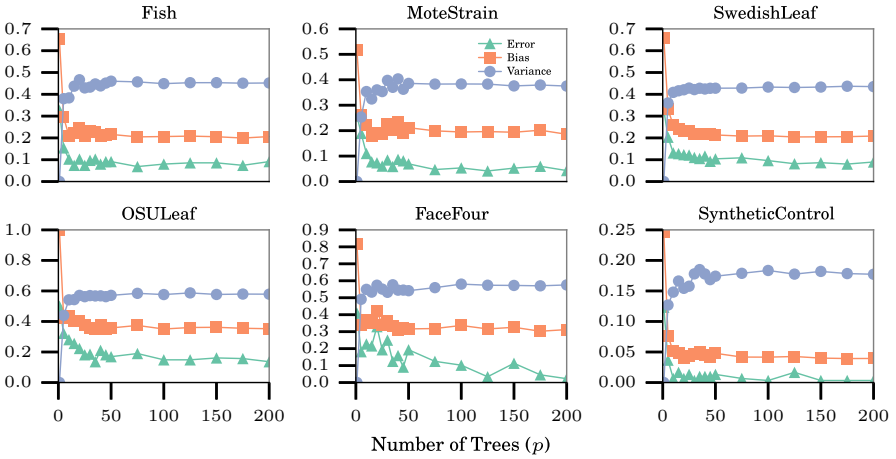

Fig. 8 The averaging strength of p

note, as shown in the result tables, that the parameter requires tuning to find the best performing model.

5.4.3 The averaging strength of p

It is well known that the prediction error is a continual, asymptotically, declining function of the number of members in an ensemble. The latter corresponds to the parameter p, which denotes the number of trees in a random shapelet forest. Hence, the higher the value of p the better the ensemble will perform, essentially rendering the choice of p to be a compromise between computational efficiency and accuracy. Fig. 8 shows the effect of p on the out-of-bag error, bias and variance for a number of data sets (with $r = \sqrt{md(md+1)/2}$). In general, we can see that while randomization increases the bias and variance of individual trees, the variance due to randomization can be canceled out by averaging over a (large) ensemble.

6 Conclusions

In this paper, we have proposed and investigated a novel approach for both univariate and multivariate time series classification; the gRSF algorithm, building on the well-established shapelet tree algorithm (Ye and Keogh 2009) and the random forest algorithm (Breiman 2001). The use of shapelets has been shown to be an important approach for time series classification tasks, but current methods have been either too slow, e.g., learning time series shapelets or shapelet transforms, too inaccurate, e.g., fast shapelets, or both, e.g., single shapelet-based decision trees. We have demonstrated that by combining several weak randomized shapelet-based decision trees into an ensemble, substantial gains in predictive and computational performance can be achieved compared to generating single shapelet-based decision trees. Moreover, the performance of gRSF is comparable to the current state-of-the-art in terms of accuracy for both univariate and multivariate time series classification, but with substantially lower computational cost. The results from the presented extensive empirical investigation suggest that the proposed algorithm is among the strongest classifiers currently available for both uni- and multivariate time series.

Although the proposed algorithm requires tuning of a set of parameters, the number of such parameters are comparable to, or fewer, than for alternative methods. Most importantly, the use of out-of-bag examples allows for tuning the parameters without requiring that a computationally costly cross-validation procedure is employed or that a validation set is put aside. The latter is not as computationally costly as the former, but may negatively affect predictive performance due to leaving fewer examples for model construction. The parameters of gRSF include the number of trees to generate (a common parameter for most ensemble methods), the number of sampled shapelets, for controlling the strength of the random shapelet selection, and also a secondary pair of parameters for controlling the size of the evaluated random shapelets. The empirical investigation has shown that the predictive performance can be greatly improved by optimizing the parameters of the algorithm, reaching state-of-the-art predictive performance for both univariate and multivariate time series classification. Furthermore, by using a bias and variance decomposition of the (squared) prediction error, the ingredients making the algorithm performing well have been investigated, namely the trade-off between the strength of each individual tree and the diversity among the trees.

One advantage of shapelet based decision trees, similar to traditional decision trees, is the possibility to interpret the generated model and gain insights into predictions. For traditional random forests, the importance of factors contributing to predictions can be computed using various methods. A similar procedure was investigated for the univariate version of the random shapelet forest in Karlsson et al. (2015), where the importance of regions and shapelet sizes can be used to support interpreting the model. Since this method is limited to globally important regions and not shapelets, one direction for future work concerns alternative approaches for interpreting random shapelet forests, e.g., by providing the importance of specific shapes, or in the multivariate case, specific dimensions. Other directions for future work include investigating approaches for handling multivariate and heterogeneous time series consisting of both discrete and numerical data streams.

Acknowledgments This work was partly supported by the project High-Performance Data Mining for Drug Effect Detection at Stockholm University, funded by Swedish Foundation for Strategic Research under Grant IIS11-0053.

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

Source code Source code (MIT license) for the Generalized Random Shapelet Forest is available at Github.[8] Instructions and datasets can be found at the supporting website.[9]

Appendix 1: Decomposing the mean squared error in a forest

To investigate whether the observed difference in predictive performance between parameter configurations is mainly due to bias, variance, or both, the mean square error of the forest can be decomposed into those terms. In contrast to regression, where the decomposition of prediction error into bias and variance is well understood and widely used (James 2003), there is no general definition for classification. Hence, several decompositions of the average classification error rate into bias and variance have been proposed (see Friedman 1997; James 2003; Valentini and Dietterich 2004). It is, however, difficult to directly study the effect of bias and variance of randomized algorithms in the context of miss-classification error since a decrease in variability of the predictions can increase the average error, making the averaged model worse than the randomized. Instead, similar to the mean square error (MSE) of a regressor, the mean square error of a forest consisting of trees able to output a conditional class probability estimates for a given example can be computed as follows (Boström 2012). Given the class labels $\{c_1, c_2, \ldots, c_l\}$, let $\bar{b}_k^{(i)} = (b_{k1}^{(i)}, b_{k2}^{(i)}, \ldots, b_{kl}^{(i)})$ be the probabilities assign by k:th random shapelet tree ST_k in the forest for a labeled time series $z^{(i)}$. Furthermore, let $\bar{c}^{(i)} = (\bar{c}_1^{(i)}, \bar{c}_2^{(i)}, \ldots, \bar{c}_l^{(i)})$ represent the true class vector for a labeled time series $z^{(i)}$, where $\bar{c}_j^{(i)}$ is 1 if $y^{(i)} = c_j$ and 0 otherwise, then the mean squared error (mse) of the forest can be defined as:

$$\text{mse} = \frac{1}{n} \sum_{i=0}^{t} \frac{1}{p} \sum_{k=1}^{p} \left(\bar{b}_k^{(i)} - \bar{c}^{(i)} \right)^2 \tag{5}$$

Given the mean class probability vector $\bar{b}_\mu^{(i)}$ for the ith example, the mse can be composed into two parts, the *bias* (left) and *variance* (right):

$$\text{mse} = \frac{1}{n} \sum_{i=0}^{n} \left(\bar{c}^{(i)} - \bar{b}_\mu^{(i)} \right)^2 + \frac{1}{n} \sum_{1=0}^{n} \frac{1}{T} \left(\bar{p}^{(i)} - \bar{b}_\mu^{(i)} \right)^2 \tag{6}$$

[8] http://github.com/briljant/mimir.

[9] http://people.dsv.su.se/~isak-kar/grsf/.

Appendix 2: Internal estimates of strength and correlation

In the original Random Forest, Breiman (2001) proposes internal estimates for the strength (i.e., how accurate the individual classifiers are) and correlation (i.e., the dependence between classifiers) of the forest based on the out-of-bag instances not included during training. Using these measures, an upper bound can be derived for the generalization error. More precisely, for the case of random shapelet forests, each random shapelet tree $ST_k \in \mathcal{R}$ can be seen as a base classifier function f_k. Hence, we can define a set of p base classifier functions $\{f_1(T), \ldots, f_p(T)\}$ as well as the ensemble classifier function $f_{\mathcal{R}}(T)$. Let us denote the set of out-of-bag instances for a classifier ST_k as $\mathcal{D}_{T_k}$. Furthermore, given a class label $c \in \mathcal{C}$, $Q(T, c)$ is an approximation function for $P(f_{\mathcal{R}}(T) = c)$ corresponding to the out-of-bag proportion of votes for class c for the input time series T. More formally:

$$Q(\mathcal{F}, c) = \frac{\sum_{k=1}^{p} \mathbf{1}(ST_k(T) = c; T \in \mathcal{D}_{T_k})}{|T \in \mathcal{D}_{T_k}|}, \tag{7}$$

where $\mathbf{1}(\cdot)$ is the indicator function. Then, the *margin* measures the extent to which the average number of votes for the right class exceeds the average number vote for any other class (Breiman 2001).

Definition 7 (*margin function*) The empirical margin function for a random shapelet forest, similar to a random forest, is

$$\text{mr}(\mathcal{F}, T, c) = P(f_{\mathcal{R}}(T) = c) - \max_{\substack{j=1 \\ j \neq c}}^{|\mathcal{C}|} \{f_{\mathcal{R}}(T) = c_j\} \tag{8}$$

where $P(\cdot)$ is estimated using $Q(\cdot)$.

The expectation over the margin function gives a measure of how accurate, or strong, a set of classifiers are.

Definition 8 (*strength*) The strength of a random shapelet forest is the expected margin, and can be empirically estimated as the average over the training set:

$$s = \frac{1}{n} \sum_{i=1}^{n} mr(T_i, y_i) \tag{9}$$

By computing the variance of the margin, the correlation and interdependence between the individual classifiers can be estimated as the variance of the margin over the squared standard deviation of the random shapelet forest.

Definition 9 (*correlation*) The correlation of the random shapelet forest can be empirically estimated as:

$$\bar{p} = \frac{\text{var}(mr)}{\text{sd}(\mathcal{F})^2} = \frac{\frac{1}{n}\sum_{i=1}^{n} mr(T_i, y_i)^2 - s^2}{\left(\frac{1}{p}\sum_{k=1}^{p}\sqrt{b_k + \hat{b}_k + (b_k - \hat{b}_k)^2}\right)^2} \tag{10}$$

where b_k is an out-of-bag estimate of $P(f_k(T = y) = c)$, with

$$b_k = \frac{\sum_{i=1}^{n} \mathbf{1}(f_k(T_i) = y_i; T_i \in \mathcal{D}_{T_k})}{|\{T \in \mathcal{D}_{T_k}\}|}, \tag{11}$$

and $\hat{b}_k$ is an out-of-bag estimate of $P(f_{\mathcal{F}}(T) = \hat{c}_j)$

$$\hat{b}_k = \frac{\sum_{i=1}^{n} \mathbf{1}(f_k(T_i) = \hat{c}_i)}{|\{T \in \mathcal{D}_{T_k}\}|}, \tag{12}$$

where $\hat{c}_j$ is estimated for every instance in the training set with $Q(T, c)$ as

$$\hat{c}_j = \arg\max_{\substack{j=1 \\ j \neq c}}^{|\mathcal{C}|} Q(T, c_j). \tag{13}$$

Assuming that the strength is $s > 0$, Breiman (2001) showed that a (rather loose) upper bound on the generalization error of a random forest, and by similar argument a random shapelet forest, can be given by $\frac{\bar{p}(1-s^2)}{s^2}$. The bound shows that the two ingredients involved in the generalization error for forests of randomized trees are the strength of the individual classifiers, and the dependencies between them in terms of the margin function (Breiman 2001). Furthermore, the correlation divided with the squared strength, $\bar{p}/s^2$, provides a ratio, where smaller is better, that can be used to understand the functioning of the forest (Breiman 2001).

References

Bagnall A, Lines J (2014) An experimental evaluation of nearest neighbour time series classification. CoRR arXiv:1406.4757

Bankó Z (2012) Correlation based dynamic time warping of multivariate time series. Expert Syst Appl 39(17):12814–12823

Batista GE, Wang X, Keogh EJ (2011) A complexity-invariant distance measure for time series. In: Proceedings of SIAM, SIAM international conference on data mining, pp 699–710

Baydogan MG, Runger G (2014) Learning a symbolic representation for multivariate time series classification. Data Min Knowl Discov 29(2):400–422

Baydogan MG, Runger G (2015) Time series representation and similarity based on local autopatterns. Data Min Knowl Discov 30(2):1–34

Baydogan MG, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Trans Pattern Anal Mach Intell 35(11):2796–2802

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop, knowledge discovery and data mining, pp 359–370

Boström H (2011) Concurrent learning of large-scale random forests. In: Proceedings of the Scandinavian conference on artificial intelligence, pp 20–29
Boström H (2012) Forests of probability estimation trees. Int J Pattern Recognit Artif Intell 26(02):125–147
Bradley AP (1997) The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159
Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
Breiman L (2001) Random forests. Mach Learn 45(1):5–32
Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, Boca Raton
Cetin MS, Mueen A, Calhoun VD (2015) Shapelet ensemble for multi-dimensional time series. In: Proceedings of SIAM international conference on data mining, SIAM, pp 307–315
Chen L, Ng R (2004) On the marriage of l_p-norms and edit distance. In: Proceedings of the international conference on very large data bases, ACM, pp 792–803
Chen L, Özsu MT (2005) Robust and fast similarity search for moving object trajectories. In: Proceedings of the ACM SIGMOD international conference on management of data, ACM, pp 491–502
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153
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
Friedman JH (1997) On bias, variance, 0/1–loss, and the curse-of-dimensionality. Data Min Knowl Discov 1(1):55–77
Fulcher BD, Jones NS (2014) Highly comparative feature-based time-series classification. IEEE Trans Knowl Data Eng 26(12):3026–3037
Gordon D, Hendler D, Rokach L (2012) Fast randomized model generation for shapelet-based time series classification. arXiv:1209.5038
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
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
Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
Hu B, Chen Y, Keogh EJ (2013) Time series classification under more realistic assumptions. In: Proceedings of SIAM international conference on data mining, SIAM, pp 578–586
James GM (2003) Variance and bias for general loss functions. Mach Learn 51(2):115–135
Kampouraki A, Manis G, Nikou C (2009) Heartbeat time series classification with support vector machines. Inf Technol Biomed 13(4):512–518
Karlsson I, Papapetrou P, Boström H (2015) Forests of randomized shapelet trees. In: Proceedings of statistical learning and data sciences, Springer, pp 126–136
Keogh E, Zhu Q, Hu B, Y H, Xi X, Wei L, Ratanamahatana CA (2015) The ucr time series classification/clustering homepage. www.cs.ucr.edu/~eamonn/time_series_data/
Lines J, Bagnall A (2014) Time series classification with ensembles of elastic distance measures. Data Min Knowl Discov 29(3):565–592
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
Maier D (1978) The complexity of some problems on subsequences and supersequences. J ACM 25(2):322–336
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
Nanopoulos A, Alcock R, Manolopoulos Y (2001) Feature-based classification of time-series data. Int J Comput Res 10:49–61

Patri OP, Sharma AB, Chen H, Jiang G, Panangadan AV, Prasanna VK (2014) Extracting discriminative shapelets from heterogeneous sensor data. In: Proceedings of IEEE international conference on big data, IEEE, pp 1095–1104

Quinlan JR (1993) C4.5: programs for machine learning. Elsevier, Amsterdam

Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of SIAM international conference on data mining, SIAM

Ratanamahatana CA, Keogh E (2004) Everything you know about dynamic time warping is wrong. In: 3rd workshop on mining temporal and sequential data, pp 22–25

Rebbapragada U, Protopapas P, Brodley CE, Alcock C (2009) Finding anomalous periodic time series. Mach Learn 74(3):281–313

Rodríguez JJ, Alonso CJ (2004) Interval and dynamic time warping-based decision trees. In: Proceedings of the 2004 ACM Symposium on applied computing, ACM, pp 548–552

Rodríguez JJ, Alonso CJ, Maestro JA (2005) Support vector machines of interval-based features for time series classification. Knowl Based Syst 18(4):171–178

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. In: Transactions on ASSP, IEEE, pp 43–49

Schmidhuber J (2014) Deep learning in neural networks: an overview. arXiv:1404.7828

Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423

Shokoohi-Yekta M, Wang J, Keogh E (2015) On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In: Proceedings of SIAM international conference on data mining, SIAM, pp 289–297

Valentini G, Dietterich TG (2004) Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. J Mach Learn Res 5:725–775

Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Discov 26(2):275–309

Wistuba M, Grabocka J, Schmidt-Thieme L (2015) Ultra-fast shapelets for time series classification. CoRR arXiv:1503.05018

Wu Y, Chang EY (2004) Distance-function design and fusion for sequence data. In: Proceedings of ACM international conference on information and knowledge management, ACM, pp 324–333

Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on machine learning, ACM, pp 1033–1040

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

Ye L, Keogh E (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min Knowl Discov 22(1–2):149–182