Knowledge and Information Systems (2020) 62:1671–1700
https://doi.org/10.1007/s10115-019-01389-4

REGULAR PAPER

Locally and globally explainable time series tweaking

Isak Karlsson[1] · Jonathan Rebane[1] · Panagiotis Papapetrou[1] · Aristides Gionis[2]

Received: 4 January 2019 / Revised: 14 July 2019 / Accepted: 22 July 2019 / Published online: 30 August 2019

Abstract
Time series classification has received great attention over the past decade with a wide range of methods focusing on predictive performance by exploiting various types of temporal features. Nonetheless, little emphasis has been placed on interpretability and explainability. In this paper, we formulate the novel problem of explainable time series tweaking, where, given a time series and an opaque classifier that provides a particular classification decision for the time series, we want to find the changes to be performed to the given time series so that the classifier changes its decision to another class. We show that the problem is **NP**-hard, and focus on three instantiations of the problem using *global* and *local* transformations. In the former case, we investigate the k-nearest neighbor classifier and provide an algorithmic solution to the global time series tweaking problem. In the latter case, we investigate the random shapelet forest classifier and focus on two instantiations of the local time series tweaking problem, which we refer to as reversible and irreversible time series tweaking, and propose two algorithmic solutions for the two problems along with simple optimizations. An extensive experimental evaluation on a variety of real datasets demonstrates the usefulness and effectiveness of our problem formulation and solutions.

Keywords Time series classification · Interpretability · Explainability · Time series tweaking

1 Introduction

Time series classification has been the center of attention in the time series community for more than a decade. The problem typically refers to the task of inferring a model from

✉ Isak Karlsson
isak-kar@dsv.su.se

Jonathan Rebane
jonathan@dsv.su.se

Panagiotis Papapetrou
panagiotis@dsv.su.se

Aristides Gionis
aristides.gionis@aalto.fi

1 Stockholm University, Stockholm, Sweden

2 Aalto University, Espoo, Finland

a collection of labeled time series, which can be used to predict the class label of a new time series. Examples applications of time series classification include historical document or projectile point classification [42], classification of electrocardiograms (ECGs) [18], and anomaly detection in streaming data [29].

Several time series classification models have been proposed in the literature, including distance-based classifiers (see, e.g., Ding et al. [11] for a thorough review), shapelet-based classifiers [42,42] along with optimizations for shapelet selection or generation [15–17,39], and ensemble-based classifiers [2]. Recently, the random shapelet forest classifier (RSF) [20] has been proposed for classifying univariate and multivariate time series. The main idea is to build a set of decision trees, where each feature corresponds to a *shapelet*. The decision condition on an internal node is the presence or absence of a shapelet in a test time series example.

Despite its competitive performance in terms of classification accuracy on a large collection of time series datasets, RSF is an opaque classification model. It is, hence, not feasible to come up with any reasoning behind the predictions that could possibly be helpful to domain experts and practitioners. *Interpretability* studies within the time series domain have been largely dominated by the explanatory power provided by shapelets, which are class-discriminatory subsequences extracted from training examples [23,40,42]. However, a clear gap has been present within the time series domain regarding *explainability*, which this study has sought to address.

Consider the task of binary time series classification, where a times series may belong to either the positive ('+') or negative class ('−'). Our main objective in this paper is to study the following simple problem: given a time series $\mathcal{T}$ and an opaque classification model (e.g., an RSF or the k-nearest neighbor classifier) expressed by function $f(\cdot)$, such that $f(\mathcal{T}) =$ '−', we want to identify the minimum number of changes that need to be applied to $\mathcal{T}$ in order to switch the classifier's decision to the positive class. That is, we want to define a transformation of $\mathcal{T}$ to $\mathcal{T}'$, such that $f(\mathcal{T}') =$ '+'. We call this problem *explainable time series tweaking* and propose two methods that can provide *global* transformations or a series of *local* transformations. By solving this problem, practitioners will not only be able to understand the reasoning behind decisions produced by an opaque time series classification model, but will also be able to take action to *change* a given time series instance from an undesired state (e.g., *sick*) to a desired state (e.g., *healthy*).

To motivate the problem of explainable time series tweaking, we present two examples: one from the biomedical domain and one from the biomechanical domain.

Example I: Abnormal versus normal heartbeats. Consider an electrocardiogram (ECG) recording, such as the one shown in Fig. 1. The original signal (blue curve), denoted as $\mathcal{T}$, corresponds to a patient suffering from a potential myocardial infarction. An explainable time series tweaking algorithm would suggest a transformation of the original time series to $\mathcal{T}'$ (yellow curve), such that the classifier considers it normal. In the figure, we are showing a series of *local* transformations that would change the prediction of the opaque classifier from one class to the other.

Example II: Gun-draw versus finger-point. Consider the problem of distinguishing between two motion trajectories, one corresponding to a gun-draw and the other to a finger-point. In Fig. 2, we can see the trajectory of a regular finger-pointing motion (blue time series), denoted as $\mathcal{T}$. The objective of explainable time series tweaking would be to suggest a transformation of $\mathcal{T}$ to $\mathcal{T}'$ (yellow curve), such that the classifier considers it a gun-point motion instead. Suppose we have an actor making a motion with her hand. The objective is to be able to

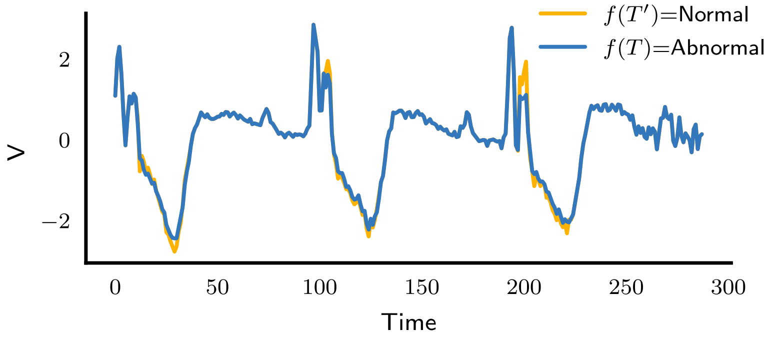

Fig. 1 Abnormal versus normal heartbeat identification. The original time series is depicted in blue. We observe that a classifier f classifies the input time series $\mathcal{T}$ as *Abnormal* (blue curve). By applying time series tweaking, we change the classifier's decision to the normal class (yellow curve) (color figure online)

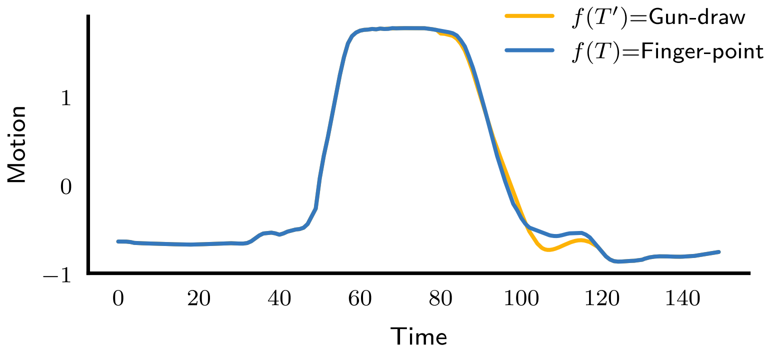

Fig. 2 Gun-draw identification. The original time series is depicted in blue. We observe that a classifier f classifies the input time series $\mathcal{T}$ as class *Finger-point*. When transforming $\mathcal{T}$ to $\mathcal{T}'$ by changing two small segments (indicated in yellow) converts it to class *Gun-draw* (color figure online)

distinguish whether that motion corresponds to a gun-draw or to pointing. In Fig. 2, we can see the trajectory of a regular finger-pointing motion (blue time series).

1.1 Contributions

The main **contributions** of this paper are summarized as follows:

- we formulate the novel problem of explainable time series tweaking, and focus on two instantiations of the problem using the k-nearest neighbor and the random shapelet forest classifiers;
- we propose a generalized method for solving the problem for the k-nearest neighbor classifier;[1]
- we propose two methods for solving the problem for the random shapelet classifier, both of which are based on *shapelet feature tweaking*, along with optimization techniques;
- for the random shapelet forest, we show that the problem is **NP**-hard by reduction to the Hitting Set problem;

[1] An earlier version of the algorithm, restricted to $k = 1$, was presented together with a limited empirical evaluation in Karlsson et al. [21].

– we provide an extensive experimental evaluation of the two proposed methods and compare them in terms of three metrics: cost, compactness, and speed of transformation.

The remainder of this paper is organized as follows: in Sect. 2, we discuss the related work in the area of time series classification with emphasis on interpretability, while in Sect. 3 we provide the formal problem formulation. In Sect. 4, we describe the two proposed methods, along with optimization strategies and theoretical properties, while in Sect. 5, we present our experimental evaluation and results. Finally, in Sect. 6, we conclude the paper and provide directions for future work.

2 Related work

The majority of time series classification methods typically rely on instance-based classification techniques. For example, the k-nearest neighbor (k-NN) classifier employs various 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), dynamic state warping [14] or longest common subsequence [25] and variants, e.g., cDTW [32], EDR [7], ERP [6], which are robust to misalignment and time warps. By regularization using, e.g., a band [28], the search performance and generalization behavior of k-NN can be greatly improved [11]. For a more complete overview of instance-based univariate time series classifiers, the reader is referred to, e.g., Ding et al. [11].

A growing body of research is related to the domain of interpretable models, in which investigators have sought to provide greater clarity to decisions made by machine learning classifiers [22,30,38]. Such a need for interpretability often stems from a stakeholder desire to trust a model in order to find it useful; a trust which can be built both through the transparency of the model itself and post hoc interpretability such as from local explanations [24]. As mentioned, a variety of studies in the time series domain highlight shapelets as the main vehicle for providing interpretability [23,40,42] with at least one study providing an alternative Symbolic Aggregate approximation (SAX) combined with a vector space approach [33].

Moreover, instance-based classifiers are supplemented by feature-based classifiers that typically use class-discriminant features, called shapelets [42], which correspond to time series subsequences with high utility, measured by different discriminative measures, such as information gain [34]. 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 [42].

Shapelet transformation is one instance of a more general concept of feature generation, which has been thoroughly investigated for time series classification. For example, the generated features can range from statistical features [10,26] to interval-based features [31] or other interpretable features, such as correlation or entropy [13]. A typical grouping of features produced by these transformations includes: 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. [10] and a transformation based on time series bag-of-words Baydogan et al. [3]. Moreover, in order to achieve performance improvements, Hills et al. [17] introduce a heuristic approach for providing an estimation of the shapelet length. The described optimization algorithm repeatedly selects the ten best shapelets in a subset of ten randomly selected time series, searching for subsequences of all possible lengths.

Regarding multivariate time series classification methods, a shapelet forest approach has been introduced by Patri et al. [27] for heterogeneous time series data. The algorithm employs the Fast Shapelet selection approach for extracting the most informative shapelets per dimension. In a similar manner, a shapelet tree is built from each time series dimension using several additional techniques for providing search speedups [5]. Moreover, various voting approaches are evaluated for providing the final classification label, demonstrating that one shapelet tree per dimension outperforms shapelets defined over multiple dimensions [5]. More recently, the generalized random shapelet forest has been proposed for univariate and multivariate time series classification, by expanding the idea of random shapelet trees and randomly selecting shapelet features per dimension [20]. While this approach can achieve competitive performance against existing classifiers in terms of classification accuracy, it is a black-box classifier with limited interpretability and explainability of the predictions.

Complementary to interpretability, a number of studies have focused on actionable knowledge extraction[35,37], where the focus is placed on identifying a transparent series of input feature changes intended to transform particular model predictions to a desired output with low cost. Many actionability studies exist with a business and marketing orientation, investigating actions necessary to alter customer behavior for mostly tree-based models [19,41]. In addition, several studies place particular focus on actionability which can be performed in an efficient and optimal manner [12,36]. For example, Cui et al. specified an algorithm to extract a knowledgeable action plan for additive tree ensemble models under a specified minimum cost for a given example [9]. Similarly, an actionability study by Tolomei et al. investigated actionable feature tweaking in regards to converting true negative instances into true positives; employing an algorithm which alters feature values of an example to the point that a global tree ensemble prediction is switched under particular global cost tolerance conditions [35].

Despite the expansion of explainability, this is an unexplored prospect within the time series domain. In this paper, we study the problem of altering the prediction of examples, through the alteration of examples themselves, such that the prediction of a tree ensemble is changed with minimal cost. Moreover, we achieve such class alterations in an effective and efficient manner, proving and addressing the **NP** hard nature of the problem in accord with several optimization strategies. We then examine the real-world relevancy of this approach in regard to both medical and biomechanical time series datasets.

3 Problem formulation

In this section, we present our notation and formally define the problem of explainable time series tweaking.

Definition 1 (*Time series*) A time series $\mathcal{T} = \{T_1, \ldots, T_m\}$ is an ordered set of real values, sampled at equal time intervals, where each $T_i \in \mathbb{R}$.

In this paper, we only consider univariate time series, but the proposed framework and methods can be easily generalized to the multivariate case. For the remainder of this paper, we will refer to univariate time series simply as time series.

A local, continuous segment of a time series is called a *time series subsequence*.

Definition 2 (*Time series subsequence or shapelet*) Given a time series $\mathcal{T}$, a *time series subsequence or shapelet* [42] of $\mathcal{T}$ is a sequence of ℓ contiguous elements of $\mathcal{T}$, denoted as $\mathcal{T}[s,\ell] = \{T_s, \ldots, T_{s+\ell-1}\}$, where s is the starting position and ℓ is its length.

Time series classification mainly relies on the chosen distance or similarity measure used to discriminate between instance pairs. The main task is to employ a distance function $d(\cdot)$ that compares two time series of equal length, and then given a time series subsequence (corresponding to a candidate discriminant shapelet) identify the closest subsequence match in the target time series. Depending on the application domain and the nature of the time series, various distance measures can be used.

Definition 3 (*Time series subsequence distance*) Given two time series $\mathcal{S}$ and $\mathcal{T}$ of lengths ℓ and m, respectively, such that $\ell \leq m$, the *time series subsequence distance* between $\mathcal{S}$ and $\mathcal{T}$, is the minimum distance between $\mathcal{S}$ and any subsequence of $\mathcal{T}$ of length ℓ, i.e.,

$$d_s(\mathcal{S}, \mathcal{T}) = \min_{s=1}^{m-\ell+1} \{d(\mathcal{S}, \mathcal{T}[s,\ell])\}\,. \tag{1}$$

A typical instantiation of $d(\cdot)$, given two time series $\mathcal{T}$ and $\mathcal{T}'$ of equal length ℓ, is the Euclidean distance, i.e.,

$$d(\mathcal{T}, \mathcal{T}') = d_E(\mathcal{T}, \mathcal{T}') = \sqrt{\sum_{i=1}^{\ell} (\mathcal{T}_i - \mathcal{T}_i')^2}. \tag{2}$$

A collection of n time series $\mathcal{X} = \{\mathcal{T}^1, \ldots, \mathcal{T}^n\}$ defines a *time series dataset.*

Definition 4 (*Time series classification function*) Given a time series $\mathcal{T}$ and a finite set of class labels $\mathcal{C}$, a classification function is a mapping f from the set of all possible time series to the set $\mathcal{C}$, such that:

$$f(\mathcal{T}) = \hat{y} \in \mathcal{C}.$$

Note that $\hat{y}$ denotes the predicted class for $\mathcal{T}$, and f can be any type of time series classification function.

In this paper, we study the problem of *explainable time series tweaking*, which is formulated below.

Problem 1 (*Explainable time series tweaking*) Given a time series $\mathcal{T}$, a desired class y', and a classifier f, such that $f(\mathcal{T}) = \hat{y}$, with $\hat{y} \neq y$, we want to find a transformation function τ, such that $\mathcal{T}$ is transformed to $\mathcal{T}' = \tau(\mathcal{T})$, with $f(\mathcal{T}') = y'$, and $c(\mathcal{T}, \mathcal{T}')$ is minimized, where $c(\mathcal{T}, \mathcal{T}')$ defines the cost of the transformation. We call a transformation that changes the class *successful* and the transformation that minimizes the cost the *most successful* transformation.

Any distance or similarity measure can be employed as a cost function. In this paper, we use the Euclidean distance, and consider two instantiations of Problem 1. In the first, f is a k-nearest neighbor classifier, and in the second f is the random shapelet forest (RSF) classifier [20].

4 Explainable time series tweaking

In this section, we instantiate the problem of explainable time series tweaking as either *global* or *local*, and provide three algorithms for solving the problem. In the former case, we provide a solution for the k-nearest neighbor classifier (Sect. 4.1) and in the latter case we introduce two solutions for the random shapelet forest [20] algorithm (Sect. 4.2).

4.1 Global tweaking: *k*-nearest neighbor

We define the problem of global explainable time series tweaking for the k-nearest neighbor classifier[2] and present a simple solution to tackle this problem. Eventually, we show that our algorithm for finding a transformation $\mathcal{T}'$ for the k-nearest neighbor classifier is a generalization of the 1-nearest neighbor approach presented by Karlsson et al. [21].

The most widely adopted time series classifier is the nearest neighbor classifier, which has been predominantly and successfully used together with the Euclidean distance measure [28]. In short, the k-nearest neighbor classifier is a proximity-based model, that assigns the majority class among the k *closest* time series in the training data. Although any distance measure can be used to define the proximity between time series, in this work we opt to use the Euclidean distance for ease of comparison, as well as its simplicity and often state-of-the-art predictive performance. We note, however, the proposed algorithm can be applied along with other distance measures, since superior predictive performance can often be achieved with alternative measures, such as dynamic time warping [32]

The first step is to define a transformation function $\tau(\cdot)$ for global explainable time series tweaking. Given a desired number of nearest neighbors k, a training set of time series $\mathcal{X}$ with corresponding class labels $\mathcal{Y}$, and a target time series $\mathcal{T}$, we define the transformation function τ_{NN} with the goal of suggesting a transformation of $\mathcal{T}$, such that the transformation cost is minimized and the classifier changes its decision to the desired class label. In this case, the smallest cost corresponds to the transformation that imposes the lowest Euclidean distance between the original and transformed time series.

More formally, the global explainable time series tweaking subproblem is defined as follows.

Problem 2 (*Global time series tweaking*) Given a time series $\mathcal{T}$, a target class y', and a k-nearest neighbor classifier $\mathcal{R}$, such that $f(\mathcal{T}, \mathcal{R}) = \hat{y}$, with $\hat{y} \neq y'$, we want to transform $\mathcal{T}$ to $\mathcal{T}' = \tau(\mathcal{T})$, such that $f(\mathcal{T}', \mathcal{R}) = y'$, the Euclidean distance $d_E(\mathcal{T}, \mathcal{T}')$ is minimized, and $\tau(\mathcal{T})$ defines a transformation $\mathcal{T} \rightarrow \mathcal{T}'$.

Algorithm 1 outlines the k-nearest transformation procedure. The first step, Line 1, is to define the number of cluster centroids, C, to use in the transformation. In this paper, we suggest to use a simple heuristic: given that centroids are placed uniformly, we allocate k points to each centroid, allowing those centroids to be selected with a majority of the desired class. The next step of the algorithm is to cluster the training data into C partitions using the k-means clustering algorithm, and to select those centroids that fulfill the majority class condition (Line 3). More specifically, the majority class condition states that the majority of the time series are labeled as y' and that the number of time series labeled as y' should be greater than or equal to $\frac{k}{|C|} + 1$. The former condition is required, since we do not guarantee

[2] Note that this is a generalization of the baseline method for 1-nearest neighbor presented in Karlsson et al. [21].

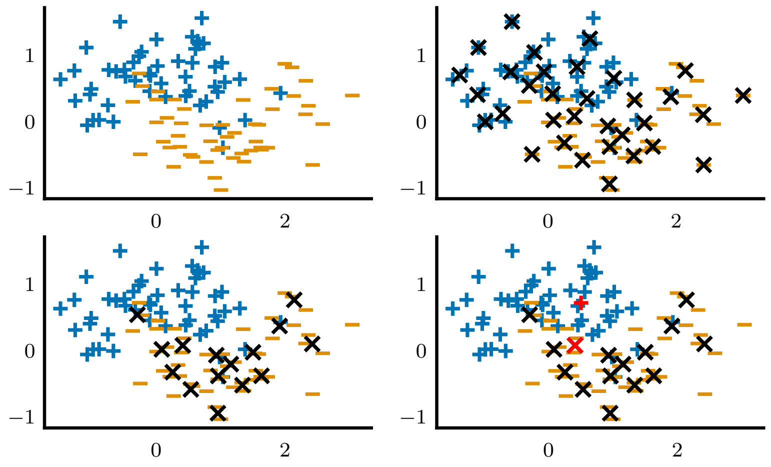

Fig. 3 Example of transforming an example of the blue class to an example of the yellow class using the k-nearest neighbor algorithm with $k = 3$. The top-right figure shows the clustering of the k-means algorithm with $\lfloor \frac{100}{3} = 33 \rfloor$ centroids, marked as black crosses. The bottom-left figure shows the centroids that can guarantee a transformation from blue to yellow. Finally, the bottom-right figure shows the result of transforming a new point '+' to a point of the desired class (red cross) (color figure online)

that a cluster cannot contain more than k time series, while the latter condition is required, since a cluster may contain less than k time series.

To exemplify the algorithm, using $k = 3$ neighbors, consider Fig. 3. In Fig. 3 (top-right), we see the $\lfloor \frac{100}{3} = 33 \rfloor$ centroids (black) identified by the k-means algorithm. Moreover, in Fig. 3 (bottom-left), we select the centroids that fulfill the conditions defined in Line 3 of Algorithm 1. Finally, in Fig. 3 (bottom-right), we transform the new time series, denoted as '+', to the closest centroid of the desired class (i.e., the red cross).

Note that for $k = 1$, the algorithm proposed here is equivalent to the baseline algorithm for the 1-nearest neighbor proposed by Karlsson et al. [21]. The explanation to this is simply that for the case of 1-nearest neighbor, where, by definition, $k = 1$, we have that $C = |\mathcal{X}|$. As a result, the centroids selected in Line 3 are the centroids of the target class y', which

Algorithm 1: Global time series tweaking algorithm (τ_{NN})

```
input : A desired number of $k$ nearest neighbors, a training set of time series $\mathcal{X}$ with corresponding
        labels $\mathcal{Y}$, a time series $\mathcal{T}$ to be transformed and a desired class $y'$
output: A transformed time series $\mathcal{T}'$
1 $C \leftarrow \lfloor \frac{|\mathcal{X}|}{k} \rfloor$
2 Apply $k$-means clustering with $C$ clusters to $\mathcal{X}$, resulting in a set of centroids $\mathcal{K}$
3 Select the centroids in $\mathcal{K}$ such that a majority of the time series closest to $K \in \mathcal{K}$ are labeled as $y'$ and
  the number of time series labeled as $y' > \frac{k}{|\mathcal{C}|} + 1$, i.e.,
      $$\mathcal{K}' \leftarrow \left\{ K \in \mathcal{K} \mid \text{majority}(K, \mathcal{X}, \mathcal{Y}) = y' \wedge \text{count}(K, \mathcal{X}, \mathcal{Y}, y') > \frac{k}{\mathcal{C}} + 1 \right\}$$
4 $\mathcal{T}' \leftarrow \arg\min_{K \in \mathcal{K}'} d_E(K, \mathcal{T})$
5 return $\mathcal{T}'$
```

results in the 1-nearest neighbor (1-NN) under the Euclidean distance, among the time series labeled as the target transformation label, i.e.,

$$\underset{\{\mathcal{T}' | (\hat{y}, \mathcal{T}') \in \mathcal{X}, \hat{y} = y'\}}{\arg\min} \quad d_E(\mathcal{T}, \mathcal{T}'). \tag{3}$$

Overall, the computational complexity of the global tweaking algorithm is $O(nmC)$, where n is the number of time series in the training set, m the number of time points and C the number of cluster centroids. Also note that Algorithm 1 can be extended to support multivariate time series transformation by defining a multivariate distance measure, e.g., the dimension-wise sum of Euclidean distances.

4.2 Local tweaking: random shapelet forest

In this section, we define local explainable time series tweaking for the random shapelet algorithm and describe the shapelet transformation function, which is the primary building block of our solution. Next, we describe two algorithms to tackle the problem and present simple optimization strategies for both algorithms. Finally, we prove that the problem we study is **NP**-hard, when considering forests of shapelet trees.

In short, an RSF, denoted as $\mathcal{R} = \{F_1, \ldots, F_{|\mathcal{R}|}\}$, is a shapelet tree ensemble of size $|\mathcal{R}|$, where each F_j denotes a shapelet tree, constructed using a random sample of shapelet features [42]. Each shapelet tree $F_j \in \mathcal{R}$ comprises a set of t decision paths $\{P_1^{(y_1)j}, \ldots, P_t^{(y_t)j}\}$, where y_i is the decision class of path i.

Let p_{ik}^j denote the kth non-leaf node in the ith path $P_i^{(y_i)j}$, such that

$$P_i^{(y_i)j} = \{p_{i1}^j, \ldots, p_{iu}^j\} \rightarrow y_i \ ,$$

where u is the length of path $P_i^{(y_i)j}$, i.e., $|P_i^{(y_i)j}| = u$ and each p_{ik}^j is described by a tuple

$$\langle \mathcal{S}_k^j, \theta_k^j, \delta_{ik}^j \rangle$$

defining a condition over shapelet $\mathcal{S}_k^j$ using a distance threshold $\theta_k^j \in \mathbb{R}$ and a comparison operator $\{\leq, >\}$, such that δ_{ik}^j equals -1 or 1 if the comparison operator is $\leq$ or $>$, respectively.

Definition 5 (*Condition test*) Given a non-leaf node $p_{ik}^j = \langle \mathcal{S}_k^j, \theta_k^j, \delta_{ik}^j \rangle$ of shapelet tree F_j and a time series $\mathcal{T}$, a condition test of path i on non-leaf node k is defined as:

$$\phi(\mathcal{T}, p_{ik}^j) = \begin{cases} \text{true}, & \text{if } (d_s(\mathcal{S}_k^j, \mathcal{T}) - \theta_k^j)\delta_{ik}^j \leq 0 \\ \text{false}, & \text{otherwise.} \end{cases} \tag{4}$$

More concretely, $\phi(\cdot)$ returns true if $\mathcal{T}$ fulfills the kth condition of the ith path of the jth tree.

To clarify the notation, consider the simple tree in Fig. 4, with two internal nodes and three terminal nodes. This tree can be converted into a set F_1 of 3 distinct paths:

$$\begin{aligned} P_1^{(\text{`+'})1} &= \{\langle S_1^1, \theta_1^1, 1\rangle\} \\ P_2^{(\text{`-'})1} &= \{\langle S_1^1, \theta_1^1, -1\rangle, \langle S_2^1, \theta_2^1, 1\rangle\} \\ P_3^{(\text{`+'})1} &= \{\langle S_1^1, \theta_1^1, -1\rangle, \langle S_2^1, \theta_2^1, -1\rangle\}. \end{aligned}$$

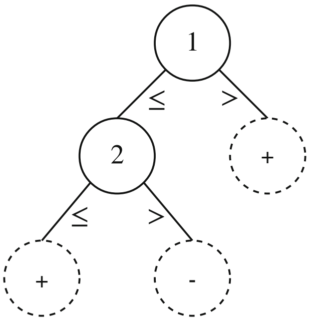

Fig. 4 A simple decision tree example of two internal nodes and three leaf nodes

Finally, observe that each non-leaf node performs a binary split depending on whether the time series subsequence distance between $\mathcal{S}^{(i,k)}$ and $\mathcal{T}$ is within a distance range θ. The decision label of F_j for $\mathcal{T}$ is denoted as $y^j = f(\mathcal{T}, F_j)$, while the decision label of $\mathcal{R}$ for $\mathcal{T}$ is defined as $\hat{y} = f(\mathcal{T}, \mathcal{R}) = M(y^1, \ldots, y^{|\mathcal{R}|})$, where $M(\cdot)$ is the majority function. For more details on the actual structure and implementation of RSF, the reader may refer to [20].

The final step is to define a suitable transformation function $\tau(\cdot)$ for explainable time series tweaking. Given a time series example $\mathcal{T}$ and an RSF classifier $\mathcal{R}$, we define the transformation function $\tau(\cdot)$ used at each conversion step while traversing a decision path in each tree of the ensemble. Recall that our goal is to suggest the transformation of $\mathcal{T}$, such that the transformation cost is minimized and the classifier changes its classification decision. Again, remember that the smallest cost corresponds to the transformation that imposes the *lowest Euclidean distance* between the original and transformed time series.

We study two versions of $\tau(\cdot)$, hence defining the following two subproblems: *reversible time series tweaking* (τ_{RT}) and *irreversible time series tweaking* (τ_{IRT}).

Problem 3 (*Local time series tweaking*) Given a time series $\mathcal{T}$, a target class y', and an RSF classifier $\mathcal{R}$, such that $f(\mathcal{T}, \mathcal{R}) = \hat{y}$, with $\hat{y} \neq y'$, we want to transform $\mathcal{T}$ to $\mathcal{T}' = \tau(\mathcal{T})$, such that $f(\mathcal{T}', \mathcal{R}) = y'$, the Euclidean distance $d_E(\mathcal{T}, \mathcal{T}')$ is minimized, and $\tau(\mathcal{T})$ defines a sequence of transformations $\mathcal{T} \to \mathcal{T}^1 \to \mathcal{T}^2 \to \ldots \to \mathcal{T}'$, where each subsequent transformation $\mathcal{T}^i$ <u>*can override*</u> (reversible time series tweaking) or <u>*cannot override*</u> (irreversible time series tweaking) any earlier transformation $\mathcal{T}^j$, with $j \leq i$.

Note that reversible time series tweaking is a more general version of Problem 3 as it allows any change applied to the time series to be overridden by a later change, while irreversible time series tweaking *locks* the time series segments that have already been changed, hence not allowing for any change to be reversed. By restricting overriding transformations in Problem 3, the Euclidean distance between the current and transformed time series is guaranteed to be monotonically increasing as more transformations are applied. Since this monotonicity property is guaranteed, transformations can be **abandoned early** if the cumulative cost of a transformation is above the currently best successful transformation so far. In contrast, reversible time series tweaking does not guarantee that the Euclidean cost is monotonically increasing and, as a consequence, does not allow for early abandoning of the transformation. Despite this, we will show in Sect. 4.2.2 that a simple optimization can achieve substantial speedups for the reversible time series tweaking algorithm.

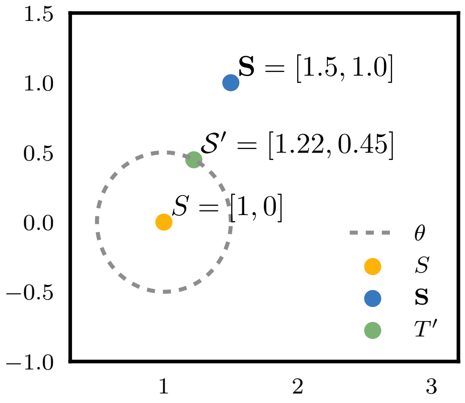

Fig. 5 Example of moving the point $\mathcal{T}$ to the closest point on the circle representing the distance threshold θ, where the distance between $d(\mathcal{S}, \mathcal{T}) = \theta$

4.2.1 Shapelet tweaking

Given a non-leaf node p_{ik}^{j} containing a shapelet $\mathcal{S}_k^j$, and a threshold θ_k^j, we define two types of time series tweaks:

- *increase distance*: if $\mathcal{S}_k^j$ exists in the current version of the time series (i.e., $d_s(\mathcal{S}_k^j, \mathcal{T}) \leq \theta_k^j$) and the current kth condition demands that $\mathcal{S}_k^j$ does not (i.e., demanding that $d_s(\mathcal{S}_k^j, \mathcal{T}) > \theta_k^j$), we want to increase the distance of all matches falling below θ_k^j to $> \theta_k^j$;
- *decrease distance*: if $\mathcal{S}_k^j$ does not exist in the current version of $\mathcal{T}$ (i.e., $d_s(\mathcal{S}_k^j, \mathcal{T}) > \theta_k^j$) and the current kth condition demands that $\mathcal{S}_k^j$ does (i.e., it demands that $d_s(\mathcal{S}_k^j, \mathcal{T}) \leq \theta_k^j$), we want to decrease the distance of its best match to $\leq \theta_k^j$.

As depicted in Fig. 5, these time series tweaks can be achieved by considering any shapelet $\mathbf{S}$ as an m-dimensional point, and by defining an m-sphere with point $\mathcal{S}_k^j$ as its center and radius θ_k^j.

Intuitively, if $d_E(\mathbf{S}, \mathcal{S}_k^j) \leq \theta_k^j$, then $\mathbf{S}$ falls inside the circle, and hence the resulting time series corresponds to the point on the circle that intersects the line connecting the two points. Given a target distance threshold (radius) θ_k^j, the transformed time series that satisfies θ_k^j exactly, is given by:

$$\tau_{\mathcal{S}}(\mathbf{S}, p_{ik}^j, \epsilon) = \mathcal{S}_k^j + \frac{\mathcal{S}_k^j - \mathbf{S}}{\|\mathcal{S}_k^j - \mathbf{S}\|_2}(\theta_k^j + (\epsilon\delta_{ik}^j)) \tag{5}$$

where $\epsilon \in \mathbb{R}$, $\epsilon \geq 0$ is a parameter that controls if the transformed time series distance falls inside the m-sphere ($\epsilon < 0$), outside the m-sphere ($\epsilon > 0$), or exactly on the circumference ($\epsilon = 0$). Note that in Eq. 5, we use δ_{ik}^j to control the direction of the move, i.e., if the test of condition k is $\leq$, then ϵ is negated, and if the test is $>$, then ϵ is not negated.

In summary, transforming a time series $\mathcal{T}$ predicted as $\hat{y}$ to a time series $\mathcal{T}'$ predicted as y' for a single decision tree is a matter of changing the time series such that all conditions of the decision path, resulting in a transformation with the lowest cost, are successful. Next, we will present two greedy algorithms for giving an approximate solution to Problem 3 using forests of randomized shapelet trees.

Algorithm 2: Reversible time series tweaking algorithm (τ_{RT})

```
input  : A shapelet forest $\mathcal{R}$, a time series $\mathcal{T}$ and a desired class $y'$ and transformation strength $\epsilon$
output: A transformed time series $\mathcal{T}'$
$\mathcal{T}' \leftarrow \mathcal{T}.copy$
$c_{min} \leftarrow \infty$
for $j \leftarrow 1$ to $|\mathcal{R}|$, $k \leftarrow 1$ to $|F_j|$ do
    for $i \leftarrow 1$ to $u$ do
        if $y^k = y' \wedge \phi(\mathcal{T}', p^j_{ik})$ is false then
            $\mathbf{T} \leftarrow \mathcal{T}'.copy$
            if $d_s(\mathcal{S}^j_k, \mathcal{T}) \leq \theta^j_k$ then
                while $d_s(\mathcal{S}^j_k, \mathcal{T}) \leq \theta^j_k$ do
                    $idx \leftarrow$ start index of subsequence with lowest distance, $d_s(\mathcal{S}^j_k, \mathbf{T})$
                    $\mathcal{S}' \leftarrow \tau_{\mathcal{S}}(\mathbf{T}[idx : idx + |\mathcal{S}^j_k|], p^j_{ik}, \epsilon)$
                    Assign $\mathcal{S}'$ to $\mathbf{T}[idx : idx + |\mathcal{S}^j_k|]$
            else
                $idx \leftarrow$ start index of subsequence with lowest distance, $d_s(\mathcal{S}^j_k, \mathbf{T})$
                $\mathcal{S}' \leftarrow \tau_{\mathcal{S}}(\mathbf{T}[idx : idx + |\mathcal{S}^j_k|], p^j_{ik}, \epsilon)$
                Assign $\mathcal{S}'$ to $\mathcal{T}[idx : idx + |\mathcal{S}^j_k|]$
    if $c(\mathbf{T}, \mathcal{T}) < c_{min} \wedge f(\mathbf{T}, \mathcal{R}) = y'$ then
        $\mathcal{T}' \leftarrow \mathbf{T}$
        $c_{min} \leftarrow d(\mathcal{T}', \mathcal{T})$
return $\mathcal{T}'$
```

4.2.2 Reversible shapelet tweaking

Given an ensemble $\mathcal{R}$ of shapelets trees, where each tree F_j is converted to a set of decision paths $\{P^{(y_1)}_{ij}, \ldots, P^{(y_t)}_{tj}\}$, a desired class label y' and a transformation strength parameter ϵ, which controls the amount of transformation applied, Algorithm 2 enumerates and applies the changes recommended by each condition k, for each path i, of all trees in the forest that are labeled with the target class label y'.

In Algorithm 2, transformations are applied one condition at a time, for each a path with the desired class label. Consequently, the first step, in Line 5, is to check if the current condition test k is fulfilled for the time series $\mathcal{T}'$, whose label we want to transform to y'. This check investigates the need for applying any of the two tweaks for the current test condition to hold. In the case where the condition does not hold, i.e., if $\phi(\cdot, \cdot)$ returns false, we check, in Line 7, if there is a need to *increase* or *decrease* the distance to fulfill the kth condition. In the first case, i.e., the closest distance is larger than the threshold, while it needs to be smaller, the transformation is simple: we find the shapelet (starting at idx and ending at $idx + |\mathcal{S}^j_k|$) with the closest distance and apply Eq. 5 to tweak the shapelet so that its distance is slightly smaller than θ^j_k, hence replacing the shapelet in $\mathbf{T}[idx : idx + |\mathcal{S}^j_k|]$ with the new subsequence, $\mathcal{S}'$. In the second case, i.e., the closest distance is smaller than or equal to the threshold, while it needs to be larger, the transformation is slightly more convoluted, since there might exist many positions where the distance is smaller than θ^j_k. In the proposed algorithm, we find and transform each time series subsequence corresponding to the lowest distance position incrementally (Line 8) until there exists no subsequence in the transformed time series with a distance smaller than θ^j_k.

After all conditions $k, \ldots, u$ of the ith path have been applied, the algorithm computes the cost of transforming $\mathcal{T}$ to $\mathcal{T}'$, i.e., $c(\mathcal{T}, \mathcal{T}')$, and if this cost is lower than the *best so far* **and** the classification according to $f(\mathbf{T}, \mathcal{R})$ has changed to y', we record the current score as the lowest and keep track of the best transformation. This procedure is repeated for all paths, until the path with the lowest cost is returned.

The computational complexity of the greedy local tweaking algorithm is, assuming a given random shapelet forest, $O(|\mathcal{R}|n\log(n)m^2)$, where n is the number of examples and m the number of time points. More concretely, in the worst case where each leaf consists of one example, the number of paths in a forest is $|\mathcal{R}|n$. Moreover, since each path has $\log n$ conditions and we need to compute the minimum distance between time series of size m, we have for each path a cost of $m \log n$. Finally, we have the additional cost of ensemble prediction, which for an ensemble of size $\mathcal{R}$, is $O(|\mathcal{R}|m \log n)$ for each n paths.

Finally, Algorithm 2 can trivially be extended to allow for transforming multivariate time series by constructing a multivariate random shapelet forest [20] and for each condition transform the time series dimension corresponding to the dimension from which the condition shapelet was extracted. As such, the only alteration introduced in Algorithm 2, similar to the global tweaking algorithm, that need to be introduced is a multivariate cost measure, e.g., the dimension-wise sum of Euclidean distances.

Early abandoning of transformations. Since the prediction cost of the random shapelet forest is higher than the cost of transforming a time series, one possible optimization is to compute all transformations, $\mathcal{T}'_1, \ldots, \mathcal{T}'_I$ for a particular time series $\mathcal{T}$ and order the transformed time series in increasing order of transformation cost, $c(\mathcal{T}, \mathcal{T}'_i)$, where $i \in \{1, \ldots, K\}$. The first transformation for which $f(\mathcal{T}', \mathcal{R}) = y'$ is true, is by definition the transformation with the lowest cost that also changes the class label. Although this might seem a simple optimization, the pruning power and runtime reduction are very significant in practice, as seen in Sect. 5.

4.2.3 Irreversible shapelet tweaking

The irreversible tweaking algorithm (τ_{IRT}) introduces a *locking* data structure that stores the start and end positions (i.e., *idx* and $idx + |S_k^j|$ in Algorithm 2) of the transformed regions of the time series $\mathcal{T}$. Consequently, we modify Algorithm 2 to store these locked regions after the transformations have been applied (Lines 11, 15). We also ensure that the subsequence with the lowest distance does not overlap with a region that has been previously *locked* by introducing additional checks in Line 8 and after Line 13. Note that by introducing the irreversible criterion, it is not guaranteed that the changes introduced by the algorithm will change the prediction of even the current tree j. However, as we show in Sect. 5 this does not significantly affect the transformation cost and allows the algorithm to produce more *compact* transformations.

Early pruning of predictions and transformations. For the irreversible tweaking problem, early abandoning of transformations is not possible. However, if we specifically lock regions (as for τ_{IRT}) of the time series that have already been transformed by an earlier condition $p_{ik}^{\cdot}$, the cost is guaranteed to be monotonically increasing as we progress with further transformations. As soon as a transformation is successful, i.e., $f(\mathcal{T}', \mathcal{R}) = y'$ (Line 7), the conditions for which the partial cost is greater than or equal to $c(\mathcal{T}, \mathcal{T}')$ can be safely ignored by introducing a partial cost indicator and checking if the current cost is increased above this value after each transformation, i.e., after Line 15. Using this simple technique, we can prune both predictions and transformations, reducing the computational burden.

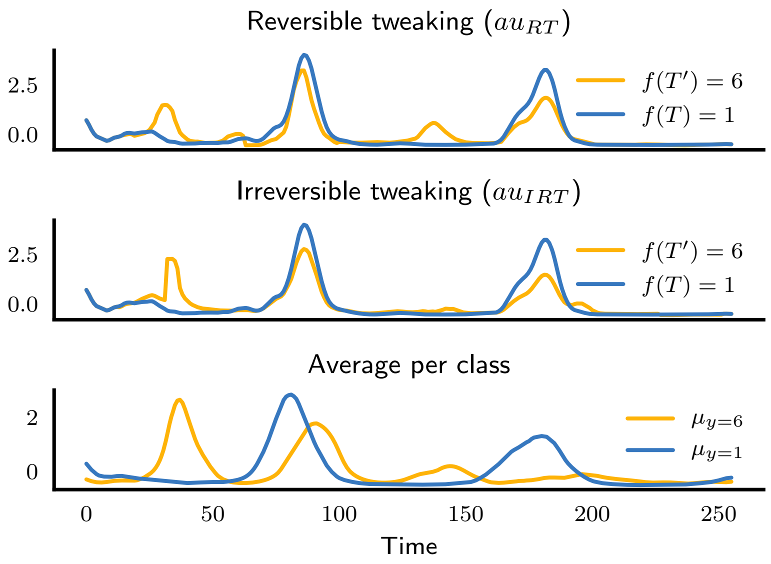

Fig. 6 (Top/middle) Example of transforming time series (blue) classified as 1 by the classifier transformed to time series (yellow) labeled as 6 by the classifier, using both the reversible and irreversible tweaking algorithms. (bottom) Average time series belonging to each of the classes, used for comparison (color figure online)

Examples of both shapelet transformation algorithms are provided in Fig. 6 (top/middle). In the figure, a time series representing different insects flying through an audio recording device are transformed from being predicted as class 1 (blue) to class 6 (yellow). We can note that both tweaking algorithms increase the amplitude around time $t = 35$ and reduce the amplitude around $t = 100$ and $t = 175$. All changes seem to correspond well with the intuition provided by the average time series for each class (Fig. 6 (bottom)).

4.2.4 NP-hardness

Let us consider a very simple model where time series are sequences of binary values, and the tree classifier tests whether certain elements of the time series have a certain value.

Let $\mathcal{T}$ be the time series, and let $\mathcal{R} = \{F_1, \ldots, F_m\}$ be the set of all decision trees in the ensemble.

Theorem 1 *Given a time series $\mathcal{T}$ and an ensemble $\mathcal{R}$, the problem of making the smallest number of changes in the time series so as to change the ensemble prediction is* **NP***-hard.*

Proof We consider the decision version of the problem, where a number k is given and we ask whether there exists a solution that requires at most k changes in the time series.

We reduce a variant of the Hitting Set problem to the problem of explainable time series tweaking (Problem 1). An instance of the Hitting Set problem is the following: We are given a ground set U of n elements, subsets $S_1, \ldots, S_m \subseteq U$, and an integer k. We ask whether there is a set $H \subseteq U$ of cardinality $|H|$ at most k, so that $H \cap S_j \neq \emptyset$, for all $j = 1, \ldots, m$, that is, whether there are at most k elements in U that "hit" all the sets $S_1, \ldots, S_m$. Here we consider the variant where we ask whether there are at most k elements in U that hit *at least half* of the sets $S_1, \ldots, S_m$. This is also an **NP**-hard problem, as it is equivalent to Maximum Cover.

Given an instance of this variant of Hitting Set problem, we create an instance to the explainable time series tweaking problem as follows. We first create a time series of length n, where all its entries are 0s, that is, $\mathcal{T}[i] = 0$, for all $i = 1, \ldots, n$. Then we create an ensemble $\mathcal{R} = \{F_1, \ldots, F_m\}$, so that there is a tree F_j for each subset S_j. In particular, the tree F_j is constructed as follows. If $i \in S_j$, then the tree F_j contains a node of the form "if $\mathcal{T}[i] = 1$, then $\mathcal{T}$ is classified to class 1, otherwise ⟨pointer to another node⟩". The tree F_j is organized in an left-unbalanced manner, so that if none of these rules are satisfied, they will all be checked. The last (leftmost) leaf has the form "if $\mathcal{T}[i] = 1$ then $\mathcal{T}$ is classified to class 1, otherwise to class 0". It follows that the tree F_j classifies the time series to 1 if and only if the series has a value equal to 1 in at least one position that corresponds to an element of the input set S_j.

We see that $\mathcal{T}$ is initially classified to class 0. We ask whether it is possible to change at most k positions in $\mathcal{T}$ so that it is classified to class 1. One can easily see that the answer to this question is affirmative, if and only if there exists a solution to the instance of the Hitting Set variant that is given as input. Thus, we conclude that the explainable time series tweaking problem is **NP**-hard. □

Note that we prove **NP**-hardness for a very special case of our problem. As a result, the most general case of our problem, where we have real-valued time series, complex shapelets, and arbitrary decision trees in the ensemble, is also **NP**-hard.

Finally, note that when the forest consists of a single shapelet tree, Problem 1 is trivial and can be solved by enumerating all paths leading to a desirable output and choosing the path with the lowest cost. However, if the forest consists of more than one tree, simply choosing a path with the lowest cost does not necessarily change the prediction for a majority of the trees in the forest.

5 Experimental evaluation

5.1 Experimental setup

We evaluate the proposed algorithms on datasets from the UCR Time Series repository [8]. The datasets represent a wide range of different classification tasks varying from motion classification, e.g., `GunPoint` to sensor reading classification, e.g., `ECG200`. In the paper, we have selected all *binary classification tasks* and the *multiclass classification tasks* with fewer than 10 classes to empirically evaluate the proposed time series tweaking algorithms. Moreover, to reduce the computational burden we limit the multiclass tasks to datasets with a maximum of 1000 time series and 600 time points. Specifically, the task is to convert time series that are correctly classified as $\hat{y} \neq c$ to $y' = c$, which amounts to converting each true negative classification to each of the other classes. As such, the results presented in the paper are the *average* of all $|\mathcal{C}|$ transformations. In the experiments, we set aside 20% of the data for transformation and testing and use the remaining 80% for training the models. We also note that, in terms of actionability, not every dataset examined possesses a domain where the benefits or realism of tweaking examples apply. We still include such datasets for the sake of comprehensive experimentation.

5.1.1 Parameters

The random shapelet algorithm requires several hyper-parameters to be set, e.g., the number of shapelets to sample at each node, the number of trees in the forest, and the minimum and maximum shapelet sizes. Since our goal is not to evaluate the effectiveness of the shapelet forest algorithm, the hyper-parameters are set to their default values, which amounts to 100 random shapelets at each node and shapelets of all possible sizes. To have a viable number of paths to use for transformation, we let the learning algorithm grow 100 trees. Moreover, we set the transformation strength for both the reversible and irreversible tweaking algorithms to $\epsilon = 1$,[3] which corresponds to relatively small changes. Similarly, the nearest neighbor classifier requires the number of nearest neighbors k to be set. Here, we adopt $k = 1$ since the gain is often minimal for optimizing the number of nearest neighbors [1]. For completeness, however, we show how the cost of transformations is affected by the number of nearest neighbors k for τ_{NN}. Similarly, we show how τ_{RT} and τ_{IRT} are effected by ϵ.

5.2 Performance metrics

We compare the algorithms using the average cost over the test set, which we define as:

$$c_\mu(\tau, y') = \frac{1}{n}\sum_{i=1}^{n} c(\mathcal{T}_i, \tau(\mathcal{T}_i, y'))$$

where n is the number of time series in the test set not classified as y'. We report the average of $c_\mu(\cdot, y')$ with $y' \in \mathcal{C}$. Moreover, we examine which fraction of the original time series must be altered. Given $\mathcal{T}$, its transformation $\mathcal{T}'$, and a threshold $e \in \mathbb{R}$,[4] assuming that $|\mathcal{T}| = |\mathcal{T}'|$ we define the compactness of a transformation of $\mathcal{T}$ to $\mathcal{T}'$ as

$$compact(\mathcal{T}, \mathcal{T}') = \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} diff(T_i, T_i')\ ,$$

where

$$diff(T_i, T_i') = \begin{cases} 1, & \text{if } |T_i - T_i'| \leq e \\ 0, & \text{otherwise.} \end{cases}$$

Note that a compactness of 1 means that the entire time series is *changed*, whereas a compactness of 0 indicates that the transformed and original time series are identical. We report the average compactness, defined as:

$$compact_\mu(\tau, y') = \frac{1}{n}\sum_{i=1}^{n} compact(\mathcal{T}_i, \tau(\mathcal{T}_i, y'))$$

where n is the number of time series in the test set not classified as y'. We report the average of $compact_\mu(\cdot, y')$ with $y' \in \{\text{'+'}, \text{'-'}\}$ for the binary datasets and $y' \in \mathcal{C}$ for the multiclass datasets.

Finally, we examine the fraction of correct predictions, i.e., the accuracy, produced by our classifiers as a means of judging the trustworthiness of the classification approaches, and consequently the trustworthiness of the transformations.

[3] We also evaluate the impact of ϵ in subsequent experiments.

[4] We set e to 1×10^{-8}.

5.3 Empirical results

The performance of the globally and locally explainable time series tweaking algorithms is compared both in terms of classification accuracy, cost, compactness and runtime of performing transformations. The results are presented in four stages: in Sect. 5.3.1, we investigate the effect of the number of neighbors, k, for the global tweaking approach using the nearest neighbor algorithm for both *binary* and *multiclass* time series. In Sect. 5.3.2, we explore the effect of ϵ for transformations using the local tweaking algorithms τ_{RT} and τ_{IRT}. In Sect. 5.4, the performance, measured by cost and compactness of transformation, of the two local tweaking approaches and the global tweaking approach is evaluated and compared. We also investigate the computational performance and scalability of the three algorithms, as measured by the time it takes to transform a single time series. Finally, in Sect. 5.5, we show the applicability of the algorithms as well as motivation for their usefulness.

5.3.1 Global time series tweaking

In this section, we explore the effect of the number of neighbors k for the k-nearest neighbor transform algorithm (τ_{NN}). As seen in Figs. 7 and 8, the cost of transformation increases as the number of neighbors increases. One possible explanation for the increased cost of transformation can, as seen in Figs. 7 and 8, be attributed to the fact that the accuracy decreases with increasing k. Moreover, to guarantee that a cluster centroid has at least $\frac{k}{|\mathcal{C}|}+1$, and a majority, of the target class examples in its neighborhood, the transformation is pushed further from the decision boundary and, as such, closer to the major density of the class, i.e., increasing the cost and, equivalently, the distance. Similarly, in Table 1, we can see that both

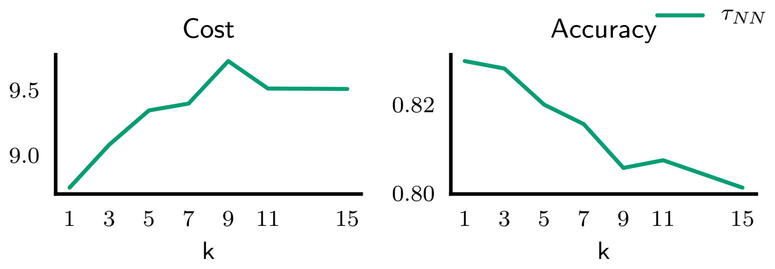

Fig. 7 Cost and accuracy, averaged of over all binary datasets ($|\mathcal{C}| = 2$), as a function of the number of k neighbors for the k-nearest neighbor transform algorithm

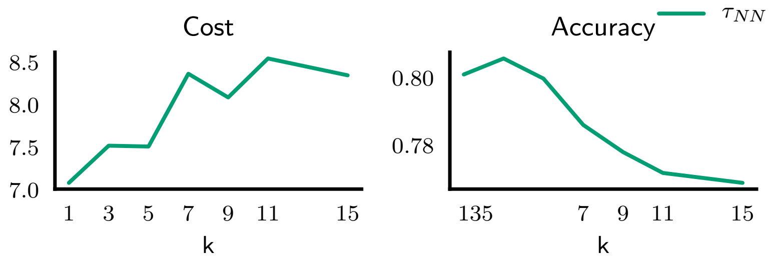

Fig. 8 Cost and accuracy, averaged over all multiclass ($|\mathcal{C}| > 2$) datasets, as a function of the number of k neighbors for the k-nearest neighbor transform algorithm

Table 1 Average cost and compactness for different numbers of neighbors (k) for the k-nearest neighbor transformation algorithm

k	Multiple classes ($\lvert\mathcal{C}\rvert > 2$)		Binary classes ($\lvert\mathcal{C}\rvert = 2$)	
	Cost	Compactness	Cost	Compactness
1	**8.7491**	**0.9999**	**7.0824**	**0.9997**
3	9.0793	1.0000	7.5192	0.9999
5	9.3427	1.0000	7.5096	0.9999
7	9.3954	1.0000	8.3643	1.0000
9	9.7221	1.0000	8.0862	0.9999
11	9.5110	0.9999	8.5439	1.0000
15	9.5078	1.0000	8.3459	1.0000

Note that the cost is generally increasing as the number of nearest neighbors increases. The best score is highlighted in bold

the minimum cost and the minimum compactness are achieved by setting $k = 1$, for both the multiclass datasets and the binary datasets, confirming our intuition.

5.3.2 Local time series tweaking

In this section, we analyze and discuss the effect of the ϵ parameter on the reversible and irreversible shapelet tweaking algorithms. In the experiments, we set $\epsilon \in \{0.01, 0.05, 0.1, 0.5, 1, 5\}$, where a smaller value, as seen in Fig. 5, results in a smaller local transformation and a larger value results in a larger local transformation. That is, the larger ϵ is set, the farther from the decision condition the new shapelet resides. In Figs. 9 and 10, we can see that

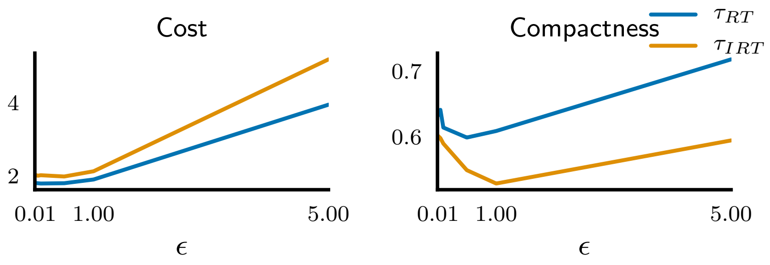

Fig. 9 Cost and compactness, averaged of over all binary datasets ($\lvert\mathcal{C}\rvert = 2$), as a function of the transformation strength ϵ for both the reversible and irreversible tweaking algorithm

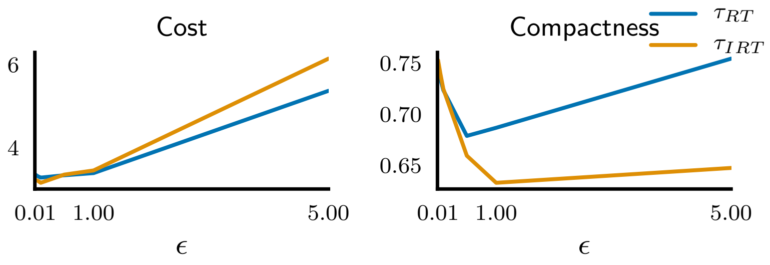

Fig. 10 Cost and compactness, averaged of over all multiclass datasets ($\lvert\mathcal{C}\rvert > 2$), as a function of the transformation strength ϵ for both the reversible and irreversible tweaking algorithm

Table 2 Average cost and compactness while adjusting the transformation strength (ϵ) for the both the reversible and irreversible tweaking algorithms

ϵ	Multiple classes ($\vert\mathcal{C}\vert > 2$)				Binary classes ($\vert\mathcal{C}\vert = 2$)			
	Cost		Compactness		Cost		Compactness	
	τ_{RT}	τ_{IRT}	τ_{RT}	τ_{IRT}	τ_{RT}	τ_{IRT}	τ_{RT}	τ_{IRT}
0.01	3.353	**3.228**	**0.742**	0.752	**1.796**	2.017	0.632	**0.601**
0.05	3.315	**3.207**	**0.734**	0.739	**1.788**	1.994	0.641	**0.597**
0.1	3.283	**3.158***	**0.723**	0.724	**1.781***	2.008	0.614	**0.589**
0.5	**3.339**	3.354	0.679	**0.659**	**1.789**	1.973	0.598	**0.549**
1	**3.391**	3.454	0.687	**0.633**	**1.891**	2.117	0.608	**0.528***
5	**5.364**	6.136	0.754	**0.647***	**3.943**	5.186	0.718	**0.594**

The lowest score for each level is highlighted with bold text, and the lowest score is marked by a star

the cost of transformation increases for both τ_{RT} and τ_{IRT} as we increase the value of ϵ. The reason for the increased cost can be attributed to the fact that *each local* transformation incurs a larger change to the time series, i.e., each adjusted segment is moved farther from the decision boundary. Consequently, the transformed time series deviates more from the original time series. Interestingly, the compactness of the solutions is, in both experiments, minimized when we set $\epsilon = 1$. This can possibly be explained by the fact that by limiting the *strength* of transformation, more local segments must be adjusted for the transformation to be successful, i.e., change the decision of the ensemble. Again, by *increasing* the strength of transformations more local segments need to be adjusted.

Another interesting finding, as seen in Table 2, is that for the multiclass datasets the irreversible transformation algorithm has a lower transformation cost than the reversible tweaking algorithm for smaller values of ϵ. This finding strengthens the intuition that weak transformation strength requires more overlapping local segments to be adjusted, which as a result increases the cost. In conclusion, there seems to be a trade-off between moving the new shapelet *too far* or *too close* to the decision threshold, to achieve the optimal transformation.

5.4 Comparing global and local tweaking

Figures 11 and 12 show a comprehensive comparison of the global and local transformation approaches, with $k = 1$ and $\epsilon = 1$, in terms of the solution quality measured by the cost and compactness for binary and multiclass datasets, respectively.[5] We can observe in regard to cost and compactness, that both the reversible tweaking τ_{RT} and irreversible tweaking τ_{IRT} approaches greatly outperform the nearest neighbor τ_{NN} approach, with τ_{RT} demonstrating the best average cost by a small degree compared to τ_{IRT}, and τ_{IRT} showing the best level of compactness by a small degree compared to τ_{RT}. In fact, τ_{RT} has the lowest cost for 22/26 datasets and τ_{IRT} in 4/26 datasets; similarly τ_{IRT} produces the most compact solutions with 23/26 wins compared to τ_{RT} with 3/26 wins.

Although these results are mostly pronounced for the binary datasets, the multiclass datasets show similar tendencies, i.e., that τ_{RT} produces transformations with lowest cost and τ_{IRT} the most compact transformations. In fact, by inspecting Table 3 we can see that τ_{RT}

[5] A complete overview of the solution quality is available in Tables 3 and 4.

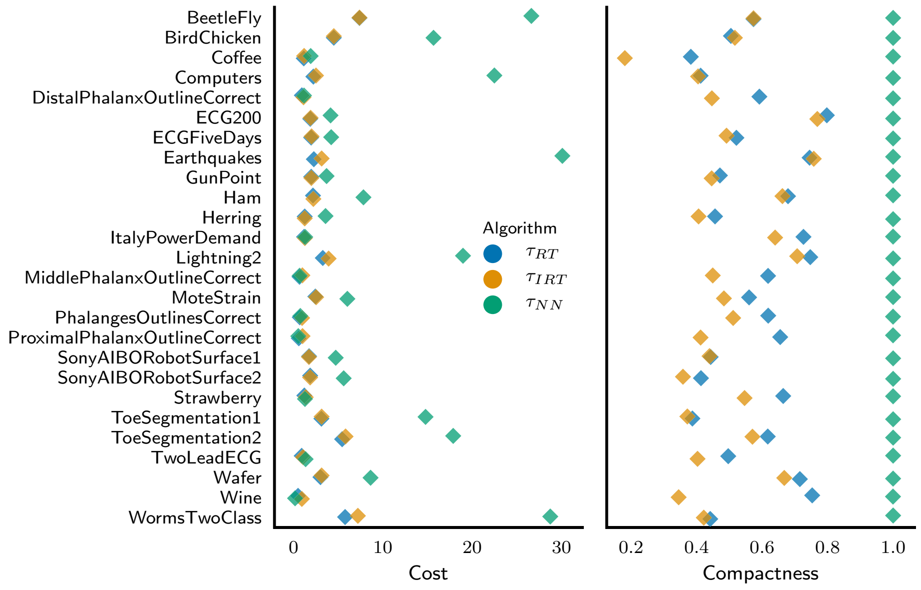

Fig. 11 Cost and compactness for datasets with $|\mathcal{C}| = 2$. For an exact overview of the performance of the different methods, please refer to Table 4

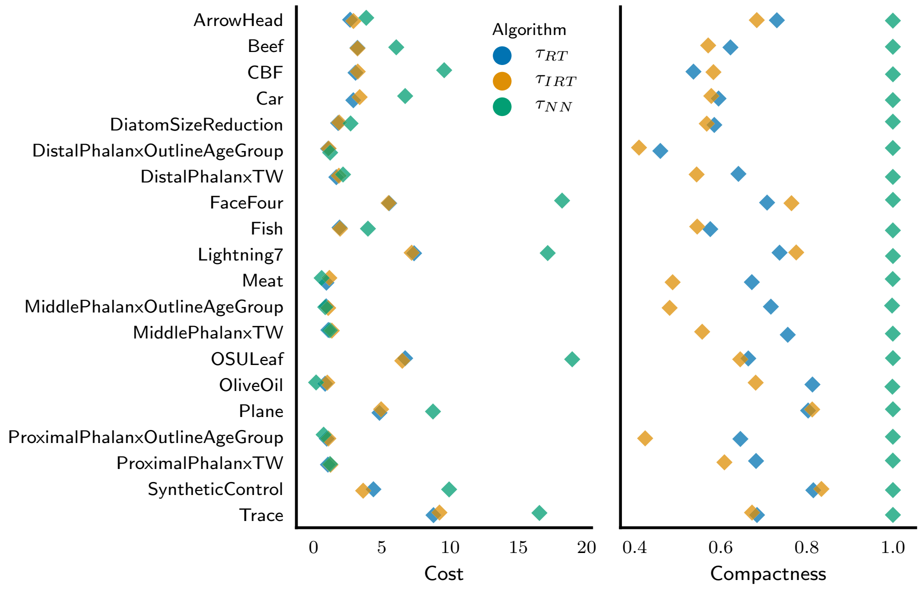

Fig. 12 Cost and compactness for datasets with $|\mathcal{C}| > 2$. For an exact overview of the performance of the different methods, please refer to Table 3

has the lowest cost in 12/20 datasets, τ_{IRT} in 4/20 datasets and τ_{NN} in 4/20 datasets. On the other hand, the most compact solutions are still provided by the τ_{IRT} algorithm, with 15/20 wins compared to 5/20 wins for the τ_{RT} algorithm.

As seen in Tables 3 and 4, RSF consistently and on average provides more trustworthy predictions, i.e., has higher accuracy, compared to τ_{NN} and thus the explainable tweaking produced by RSF would not only result in less costly and more compact transformations but potentially also be considered more trustworthy by domain experts.

5.4.1 Computational performance

In Table 5, we present a runtime comparison of τ_{IRT} against τ_{RT} with and without pruning, limiting this discussion to binary datasets only since the runtime is not affected by the number of classes. In Table 5, we can observe that τ_{RT} with pruning provides the best runtime performance on average. The superior runtime of τ_{RT} with pruning can be explained by the fact that the relative cost of an *ensemble prediction* by the RSF is, on average, more costly than a *transformation* using the proposed algorithm. As such, the improved performance of τ_{RT} can be attributed to its ability to prune more predictions than τ_{IRT}. In fact, for datasets with costly transformations (e.g., PhalangesOutlinesCorrect), the τ_{IRT} algorithm, which is able to prune 90% of the transformations, outperform τ_{RT}. As a result, one should prefer τ_{IRT} when transformations are complex and the compactness of transformations are deemed important.

5.5 Use-case examples

In this section, we provide two use-case examples of the proposed time series tweaking framework by revisiting the motivating examples from Sect. 1. The examples originate in two domains: electrocardiogram classification and motion detection.

5.5.1 Electrocardiograms

Revisiting the problem of heartbeat classification (Example I), we demonstrate a use-case example from the ECG200 dataset, which contains measurements of cardiac electrical activity as recorded from electrodes at various locations on the body; each time series contains the measurements recorded by one electrode. The binary classification objective is to distinguish between *Normal* and *Abnormal* heartbeats.

In Fig. 13, we observe that the original time series $\mathcal{T}$(blue curve) exhibits a low-amplitude QRS complex, which may suggest a pericardial effusion or infiltrative myocardial disease [4], and is hence classified as *Abnormal* by the RSF classifier. Our locally explainable time series tweaking algorithm τ_{RT} suggests a transformation of the original time series to $\mathcal{T}'$ (yellow curve), such that the low-amplitude QRS complex is changed, by increasing the amplitude of the S-wave. This is illustrated by the yellow curve. Since τ_{RT} is the best performing transformation of the two proposed ones in terms of cost for this dataset, we apply it to $\mathcal{T}$ resulting in the classifier to label $\mathcal{T}'$ as *Normal*. Moreover, we observe that the baseline competitor τ_{NN} suggests a much costlier transformation (dotted curve).

5.5.2 Gun-draw versus finger-point

Revisiting the problem of motion recognition (Example II), we demonstrate a use-case example from the *GunPoint* dataset, which contains motion trajectories of an actor making a motion with his or her hand. The objective is to distinguish whether that motion corresponds to a gun-draw or to finger-pointing.

Table 3 Summary of the results for the evaluation metrics

Dataset	# classes	**Cost**			**Compactness**			**Accuracy**	
		IE	LIE	NN	IE	LIE	NN	RSF	NN
ArrowHead	3	**2.6784**	2.9195	3.8583	0.7302	**0.6835**	0.9999	**0.9762**	0.9286
Beef	5	**3.2143**	3.2193	6.0560	0.6223	**0.5706**	0.9998	**0.6667**	**0.6667**
CBF	4	**3.0737**	3.2354	9.5646	**0.5363**	0.5828	1.0000	**1.0000**	0.9892
Car	3	**2.9132**	3.3751	6.7069	0.5946	**0.5778**	0.9998	**0.8333**	0.7500
DiatomSizeReduction	4	**1.7998**	1.8778	2.7101	0.5848	**0.5672**	0.9998	0.9844	**1.0000**
DistalPhalanxOutlineAgeGroup	3	**1.0615**	1.1065	1.2154	0.4594	**0.4098**	0.9998	**0.7963**	0.7315
DistalPhalanxTW	6	**1.6680**	1.8534	2.1582	0.6408	**0.5437**	0.9999	**0.7870**	0.7222
FaceFour	4	5.5396	**5.4917**	18.1979	**0.7075**	0.7642	1.0000	**1.0000**	0.9545
Fish	7	**1.9022**	1.9409	3.9817	0.5754	**0.5450**	0.9999	**0.8857**	0.8000
Lightning7	7	7.3598	**7.1822**	17.1438	**0.7364**	0.7752	1.0000	**0.6897**	0.6207
Meat	3	0.9401	1.1562	**0.5917**	0.6720	**0.4879**	0.9995	**1.0000**	**1.0000**
MiddlePhalanxOutlineAgeGroup	3	0.9232	1.0719	**0.8548**	0.7163	**0.4810**	0.9980	**0.7297**	0.6667
MiddlePhalanxTW	6	**1.0838**	1.3358	1.1948	0.7555	**0.5566**	0.9997	**0.6486**	0.6216
OSULeaf	4	6.7071	**6.4971**	18.9391	0.6637	**0.6451**	1.0000	**0.7841**	0.6705
OliveOil	6	0.8466	1.0056	**0.1857**	0.8129	**0.6809**	0.9987	**1.0000**	0.9167
Plane	7	**4.8264**	4.9480	8.7450	**0.8028**	0.8124	0.9999	**0.9762**	0.9524
ProximalPhalanxOutlineAgeGroup	3	0.9672	1.0850	**0.7372**	0.6454	**0.4240**	0.9998	**0.8595**	0.7603
ProximalPhalanxTW	6	**1.0516**	1.2403	1.2081	0.6818	**0.6083**	0.9998	**0.8512**	0.7273
SyntheticControl	6	4.3820	**3.6315**	9.9254	**0.8150**	0.8337	1.0000	**0.9833**	0.9167
Trace	4	**8.7819**	9.2175	16.5373	0.6840	**0.6725**	1.0000	**1.0000**	0.8750
Avg.	–	**3.0860**	3.1695	6.5256	0.6719	**0.6111**	0.9997	**0.8726**	0.8135

The best score is highlighted in bold

Table 4 Summary of the results for the evaluation metrics

Dataset	**Cost**			**Compactness**			**Accuracy**	
	τ_{RT}	τ_{IRT}	τ_{NN}	τ_{RT}	τ_{IRT}	τ_{NN}	RSF	τ_{NN}
BeetleFly	**7.3810**	**7.3810**	26.6223	**0.5737**	**0.5737**	1.0000	**0.8750**	0.7500
BirdChicken	**4.5071**	4.5098	15.6695	**0.5048**	0.5169	1.0000	**1.0000**	0.6250
Coffee	**1.1447**	1.1846	1.9178	0.3824	**0.1809**	1.0000	**1.0000**	**1.0000**
Computers	**2.2197**	2.5132	22.4809	0.4123	**0.4044**	1.0000	**0.7000**	0.4900
DistalPhalanxOutlineCorrect	**0.9314**	1.1150	1.1704	0.5917	**0.4466**	0.9999	**0.7886**	0.7143
Earthquakes	**2.2725**	3.1455	30.0943	**0.7449**	0.7577	1.0000	**0.7826**	0.6630
ECG200	**1.8730**	1.9080	4.1428	0.7976	**0.7686**	1.0000	**0.8750**	0.9500
ECGFiveDays	**1.9722**	2.0158	4.2143	0.5215	**0.4913**	1.0000	**1.0000**	0.9944
GunPoint	**1.9787**	1.9942	3.6975	0.4712	**0.4460**	0.9998	**1.0000**	0.9250
Ham	**2.1744**	2.2187	7.8253	0.6791	**0.6621**	0.9999	**0.8605**	0.7907
Herring	1.2492	**1.2488**	3.5817	0.4563	**0.4060**	0.9999	**0.5000**	0.3846
ItalyPowerDemand	**1.1791**	1.2645	1.3088	0.7262	**0.6397**	0.9998	**0.9726**	0.9589
Lightning2	**3.2741**	3.9266	18.9703	0.7470	**0.7071**	1.0000	**0.6667**	0.6667
MiddlePhalanxOutlineCorrect	**0.6685**	0.9877	0.6791	0.6182	**0.4493**	0.9999	**0.8258**	0.7753
MoteStrain	**2.4413**	2.5313	6.0249	0.5602	**0.4834**	1.0000	**0.9685**	0.9213
PhalangesOutlinesCorrect	**0.6979**	0.9568	0.7574	0.6186	**0.5116**	0.9998	**0.8421**	0.7782
ProximalPhalanxOutlineCorrect	**0.5895**	1.0056	0.5326	0.6552	**0.4121**	0.9997	**0.8315**	0.8090
SonyAIBORobotSurface1	1.7384	**1.7260**	4.7213	0.4429	**0.4394**	1.0000	0.9919	**1.0000**
SonyAIBORobotSurface2	1.8601	**1.8566**	5.6126	0.4133	**0.3584**	1.0000	0.9796	**0.9949**
Strawberry	**1.2082**	1.3628	1.2802	0.6644	**0.5464**	0.9999	0.9695	**0.9797**
ToeSegmentation1	**3.1200**	3.1436	14.7768	0.3871	**0.3718**	1.0000	**0.9259**	0.7407

Table 4 continued

Dataset	**Cost**			**Compactness**			**Accuracy**	
	τ_{RT}	τ_{IRT}	τ_{NN}	τ_{RT}	τ_{IRT}	τ_{NN}	RSF	τ_{NN}
ToeSegmentation2	**5.4407**	5.8238	17.8733	0.6173	**0.5705**	1.0000	**0.9697**	0.7879
TwoLeadECG	**0.9112**	1.0671	1.3517	0.4966	**0.4028**	0.9999	**1.0000**	0.9957
Wafer	**3.0135**	3.1419	8.6207	0.7152	**0.6676**	0.9999	0.9958	**0.9979**
Wine	**0.5052**	0.9301	0.1708	0.7529	**0.3452**	0.9996	**1.0000**	**1.0000**
WormsTwoClass	**5.7723**	7.2023	28.7383	0.4416	**0.4219**	1.0000	**0.8269**	0.7308
Avg.	**2.3132**	**2.5329**	**8.9552**	**0.5733**	**0.4942**	**0.9999**	**0.8924**	**0.8240**

The best score is highlighted in bold

Table 5 Summary of the runtime of the two algorithms including the pruning power of the proposed optimization protocols

Dataset	$\lvert\mathcal{T}\rvert$	Runtime			Pruning	
		No pruning	τ_{RT}	τ_{IRT}	τ_{RT}	τ_{IRT}
BeetleFly	512	1.763	**0.622**	0.646	**0.181**	0.142
BirdChicken	512	1.966	**0.576**	0.622	**0.328**	0.261
Coffee	286	1.099	**0.067**	0.092	**0.774**	0.677
Computers	720	194.992	**13.676**	29.358	**0.902**	0.738
DistalPhalanx…	600	14.271	0.918	**0.772**	**0.941**	0.863
Earthquakes	512	117.447	**23.904**	46.285	**0.724**	0.474
ECG200	96	2.194	**0.234**	0.269	**0.719**	0.588
ECGFiveDays	136	1.173	**0.083**	0.104	**0.718**	0.601
GunPoint	150	1.966	**0.144**	0.197	**0.748**	0.598
Ham	431	17.898	**2.270**	3.494	**0.761**	0.605
Herring	512	14.642	**1.710**	2.556	**0.829**	0.725
ItalyPowerDemand	24	3.217	0.294	**0.232**	**0.917**	0.837
Lightning2	637	16.765	**3.867**	5.479	**0.612**	0.390
MiddlePhalanx…	80	17.903	1.404	**1.158**	**0.949**	0.906
MoteStrain	84	11.668	**1.062**	1.478	**0.779**	0.457
PhalangesOutlinesCorrect	80	69.298	7.515	**5.690**	**0.949**	0.904
ProximalPhalanx…	80	16.599	1.351	**1.160**	**0.948**	0.881
SonyAIBORobot…1	70	1.641	**0.111**	0.134	**0.847**	0.692
SonyAIBORobot…2	65	3.842	**0.373**	0.460	**0.821**	0.592
Strawberry	235	22.119	**1.336**	1.727	**0.923**	0.864
ToeSegmentation1	277	3.208	**0.582**	0.755	**0.600**	0.442
ToeSegmentation2	343	2.574	**0.695**	0.790	**0.350**	0.252
TwoLeadECG	82	2.400	0.158	**0.144**	**0.941**	0.871
Wafer	152	18.709	**2.163**	34.402	**0.815**	0.558
Wine	234	2.766	**0.147**	0.233	**0.917**	0.788
WormsTwoClass	900	105.508	**28.217**	40.302	**0.566**	0.333
Avg.		25.678	**3.595**	6.867	**0.752**	0.617

While τ_{RT} pruning does not prune any transformations, the τ_{IRT} pruning algorithm does. Hence, for τ_{IRT} the fraction of early abandoned transformations is the same as the fraction of pruned predictions. Moreover, note that the runtime of the nearest neighbor approach is omitted, since a transformation is a simple nearest neighbor search and as such only takes fractions of a second for the included datasets. Please note that due to space constraints some dataset names have been truncated with '…'. The best score is highlighted in bold

In Fig. 14, we observe the trajectory of a pointing motion (blue curve), for three consecutive motions recording, classified as *finger-point* by the `RSF` classifier. By observing the bottom recording (yellow curve), we see the transformations needed to change the decision of `RSF` to *Gun-point*, using τ_{RT} (which has again achieved the lowest transformation cost, according to Table 4). Following our experimental findings, we also observe that the baseline competitor τ_{NN} suggests a much costlier transformation (dotted curve).

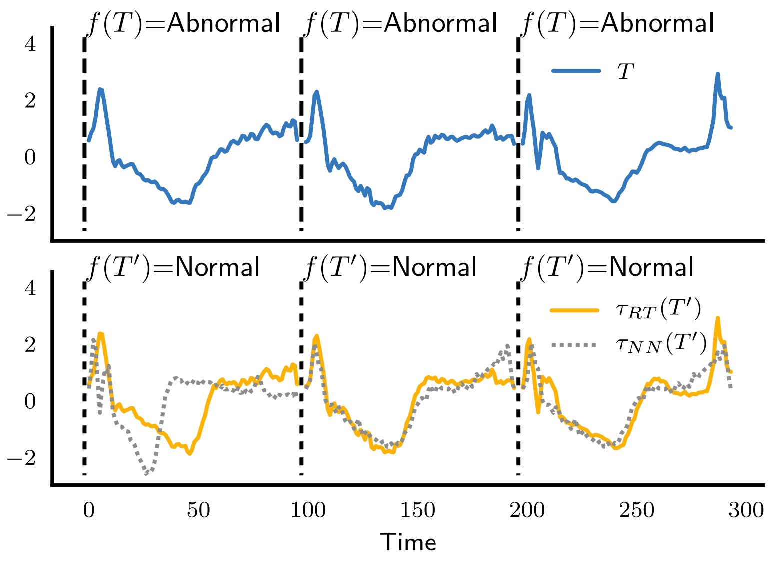

Fig. 13 Abnormal versus normal heartbeat identification: the original time series is depicted in blue. We observe that a classifier f labels the three segments of the input time series $\mathcal{T}$ as *Abnormal* (top). By applying τ_{RT}, we can transform these heartbeats to the normal class (bottom). We also show the transformations using τ_{NN} (bottom) (color figure online)

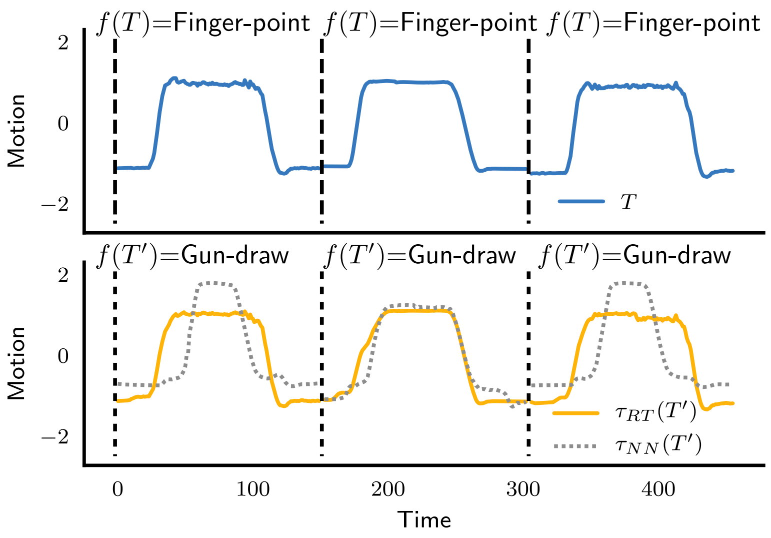

Fig. 14 Gun-draw versus finger-point identification. The original time series is depicted in blue. We observe that RSF classifies the three segments of the input time series $\mathcal{T}$ as *finger-point* (top). By applying τ_{RT}, we can transform these finger-point motion trajectories to gun-draw trajectories (bottom). We also show the transformations using τ_{NN} (bottom) (color figure online)

6 Conclusions

In this study, we have sought to exploit and expand upon the interpretability afforded by shapelets in the time series domain as a means of permitting explainability. We defined the problem of locally and globally explainable time series tweaking and provided two solutions for the k-nearest neighbor algorithm and the random shapelet forest algorithm. Moreover, for the random shapelet forest we showed that the proposed problem formulation is **NP**-hard and provided two instantiations of the problem.

Experiments were performed to examine our approaches in-depth, in terms of Euclidean distance cost, compactness of transformations, and time needed for altering time series examples. We have demonstrated that the local explainable algorithms using the random shapelet forest outperform the global k-nearest neighbor solution in terms of cost and compactness, both of which are important factors in permitting actions pertaining to time series that are actually feasible in the sense that alterations can be realistically performed in a given domain. Future work includes the investigation of alternative distance measures, such as dynamic time warping, as well as expanding our approach to permit transformations exploiting trade-offs between cost and trustworthiness of classifier predictions.

Reproducibility. Source code and data are available at: http://github.com/isakkarlsson/tsexplain.

Acknowledgements Open access funding provided by Stockholm University. This work was partly supported by grants provided by the Stockholm County Council and partly by the project Temporal Data Mining for Detective Adverse Events in Healthcare, Ref. No. VR-2016-03372. The work was also supported by three Academy of Finland projects (286211, 313927, and 317085), and the EC H2020 RIA project "SoBigData" (654024). Finally, we acknowledge the UCR Time Series Classification Archive [8] for the datasets used in this paper.

References

1. Bagnall A, Lines J (2014) An experimental evaluation of nearest neighbour time series classification. CoRR arXiv:1406.4757
2. Bagnall A, Lines J, Hills J, Bostrom A (2016) Time-series classification with cote: the collective of transformation-based ensembles. In: IEEE international conference on data engineering (ICDE), pp 1548–1549
3. 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
4. Bejar D, Colombo PC, Latif F, Yuzefpolskaya M (2015) Infiltrative cardiomyopathies. J Clin Med Insights 9(Suppl 2):29–38
5. 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
6. 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
7. 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
8. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The ucr time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/

9. Cui Z, Chen W, He Y, Chen Y (2015) Optimal action extraction for random forests and boosted trees. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 179–188
10. Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153
11. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. Proc VLDB Endow 1(2):1542–1552
12. Du J, Hu Y, Ling CX, Fan M, Liu M (2011) Efficient action extraction with many-to-many relationship between actions and features. In: International workshop on logic, rationality and interaction. Springer, Berlin, pp 384–385
13. Fulcher BD, Jones NS (2014) Highly comparative feature-based time-series classification. IEEE Trans Knowl Data Eng 26(12):3026–3037
14. Gong Z, Chen H (2018) Sequential data classification by dynamic state warping. Knowl Inf Syst 57(3):545–570
15. 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
16. Grabocka J, Wistuba M, Schmidt-Thieme L (2016) Fast classification of univariate and multivariate time series through shapelet discovery. Knowl Inf Syst 49(2):429–454
17. 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
18. Kampouraki A, Manis G, Nikou C (2009) Heartbeat time series classification with support vector machines. Inf Technol Biomed 13(4):512–518
19. Karim M, Rahman RM (2013) Decision tree and naive bayes algorithm for classification and generation of actionable knowledge for direct marketing. J Softw Eng Appl 6(04):196
20. Karlsson I, Papapetrou P, Boström H (2016) Generalized forests. Data Min Knowl Discov 30(5):1053–1085
21. Karlsson I, Rebane J, Papapetrou P, Gionis A (2018) Explainable time series tweaking via irreversible and reversible temporal transformations. In: International conference on data mining (ICDM), pp 1–7
22. Koh PW, Liang P (2017) Understanding black-box predictions via influence functions. arXiv preprint arXiv:1703.04730
23. 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, pp 289–297
24. Lipton Z (2016) The mythos of model interpretability. CoRR arXiv:1606.03490
25. Maier D (1978) The complexity of some problems on subsequences and supersequences. J ACM 25(2):322–336
26. Nanopoulos A, Alcock R, Manolopoulos Y (2001) Feature-based classification of time-series data. Int J Comput Res 10:49–61
27. 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
28. 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
29. Rebbapragada U, Protopapas P, Brodley CE, Alcock C (2009) Finding anomalous periodic time series. Mach Learn 74(3):281–313
30. Ribeiro M, Singh S, Guestrin C (2016) why should I trust you?: Explaining the predictions of any classifier. CoRR arXiv:1602.04938
31. 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
32. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. In: Transactions on ASSP, IEEE, pp 43–49
33. Senin P, Malinchik S (2013) Sax-vsm: interpretable time series classification using sax and vector space model. In: Data mining (ICDM), 2013 IEEE 13th international conference on, IEEE, pp 1175–1180
34. Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423
35. Tolomei G, Silvestri F, Haines A, Lalmas M (2017) Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: ACM international conference on knowledge discovery and data mining, pp 465–474
36. Turney PD (1994) Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. J Artif Intell Res 2:369–409

37. Van Linh N, Anh NK, Than K, Dang CN (2017) An effective and interpretable method for document classification. Knowl Inf Syst 50(3):763–793
38. Vellido A, Martín-Guerrero JD, Lisboa P (2012) Making machine learning models interpretable. ESANN 12:163–172
39. Wistuba M, Grabocka J, Schmidt-Thieme L (2015) Ultra-fast shapelets for time series classification. CoRR arXiv:1503.05018
40. Xing Z, Pei J, Yu PS, Wang K (2011) Extracting interpretable features for early classification on time series. In: Proceedings of the 2011 SIAM international conference on data mining, SIAM, pp 247–258
41. Yang Q, Yin J, Ling C, Pan R (2007) Extracting actionable knowledge from decision trees. IEEE Trans Knowl Data Eng 19(1):43–56
42. 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

Isak Karlsson is an Assistant Professor at the Department of Computer and Systems Sciences at Stockholm University. His area of expertise is in machine learning and data science with a particular focus on ensemble methods, complex and temporal data and interpretable and explainable learning. His main application domain of focus is healthcare, finance and regulatory technologies. His previous appointments include being a researcher at University of Gävle and postdoctoral researcher at Stockholm University. His current research is partly funded by Handelsbankens forskningsstiftelser (project Audit Automation Using Machine Learning).

Jonathan Rebane is currently a PhD student at the Department of Computer and Systems Sciences at Stockholm University in Stockholm, Sweden. He is originally from Toronto, Canada, with a background in cognitive science and health informatics. His current research interests are focused on the development and application of data mining methods related to complex temporal data. His main application domain of focus is healthcare, with emphasis on the detection of adverse drug events from electronic healthcare records.

Panagiotis Papapetrou is a Professor at the Department of Computer and Systems Sciences at Stockholm University and Adjunct Professor at the Computer Science Department at Aalto University. His area of expertise is algorithmic data mining with particular focus on mining and indexing temporal data and healthcare data. Panagiotis received his PhD in computer science at Boston University in 2009, was a post-doctoral researcher at Aalto University during 2009–2013, and was a lecturer at the University of London during 2012–2013. He has participated in several national and international research projects. He is a board member of the Swedish AI Society.

Aristides Gionis is a Professor at the Department of Computer Science in Aalto University. His previous appointments include being a Visiting Professor in the University of Rome and a Senior Research Scientist in Yahoo! Research. He is currently serving as an Action Editor in the Data Management and Knowledge Discovery journal (DMKD), an Associate Editor in the ACM Transactions on Knowledge Discovery from Data (TKDD), and an Associate Editor in the ACM Transactions on the Web (TWEB). He has contributed to several areas of data science, such as algorithmic data analysis, web mining, social-media analysis, data clustering, and privacy-preserving data mining. His current research is funded by the Academy of Finland (projects Nestor, Agra, AIDA, MLDB) and the European Commission (project SoBigData).