Agnostic local explanation for time series classification

Maël Guillemé
Energiency, Univ Rennes, Inria, CNRS, IRISA
Rennes, France
mael.guilleme@irisa.fr

Véronique Masson
Univ Rennes, Inria, CNRS, IRISA
Rennes, France
veronique.masson@irisa.fr

Laurence Rozé
Univ Rennes, Insa, Inria, CNRS, IRISA
Rennes, France
laurence.roze@irisa.fr

Alexandre Termier
Univ Rennes, Inria, CNRS, IRISA
Rennes, France
alexandre.termier@irisa.fr

***Abstract*—Recent advances in Machine Learning (such as Deep Learning) have brought tremendous gains in classification accuracy. However, these approaches build complex non-linear models, making the resulting predictions difficult to interpret for humans. The field of *model interpretability* has therefore recently emerged, aiming to address this issue by designing methods to explain a posteriori the predictions of complex learners. Interpretability frameworks such as LIME and SHAP have been proposed for tabular, image and text data.**

Nowadays, with the advent of the Internet of Things and of pervasive monitoring, time-series have become ubiquitous and their classification is a crucial task in many application domains. Like in other data domains, state-of-the-art time-series classifiers rely on complex models and typically do not provide intuitive and easily interpretable outputs, yet no interpretability framework had so far been proposed for this type of data.

In this paper, we propose the first agnostic Local Explainer For TIme Series classificaTion (LEFTIST). LEFTIST provides explanations for predictions made by any time series classifier. Our thorough experiments on synthetic and real-world datasets show that the explanations provided by LEFTIST are at once faithful to the classification model and understandable by human users.

***Index Terms*—Interpretability, Time series classification, Local explanations**

I. INTRODUCTION

Recent progresses in the Machine Learning field, especially owed to Deep Learning approaches, have allowed to considerably increase the performance of ML systems for supervised tasks such as classification or regression. As a result, such ML systems are becoming ubiquitous, for applications as diverse as credit scoring, fraud detection or chat bots. As these systems influence more and more important decisions, the need to explain their predictions also increases, to ensure that 1) the predictions do not exhibit unacceptable biases and 2) the user has received the necessary and sufficient information to adequately interpret the results and draw her conclusions. Such explanations are hard to obtain from high accuracy approaches such as Deep Learning or ensemble methods, which rely on complex numerical models.

This has sparked a new line of research on *interpretability*, where novel algorithms are designed to explain the predictions of high-accuracy complex classifiers, seen as "black boxes". Prominent approaches such as LIME [1] and SHAP [2] solve a *local interpretability* problem: given a black box classifier and its prediction for a particular data point, they provide an explanation for this specific classifier output. This explanation is usually of the form of a simple linear model that approximates the decision surface of the classifier around the data point of interest.

Previous approaches have focused on text, image and tabular data and the problem of generating explanations for the time series data has not received attention so far. The importance of being able to handle such data cannot be understated, as more and more connected devices collect possibly critical data in the form of time series, and actions might be triggered in response to the predictions made on those observations (in near real-time). Among such devices are medical sensors monitoring, e.g., sugar levels, heart rate or brain activity, with obvious health implications. In houses and offices, consumption of various fluids (electricity, gas or water) can be monitored to optimize their usage.

Like with other data types, the best classifiers for time series are based on non-interpretable approaches [3], [4], while the number and potential criticality of their applications make interpretability an especially important property. In this paper we present the LEFTIST approach (Local Explainer For TIme Series classificaTion). To the best of our knowledge, this is the first agnostic local interpretability approach for time series. Our main contributions are:

- we build on state-of-the-art interpretability framework and design the LEFTIST approach, which is tailored to time series data;
- we break down existing interpretability frameworks to elementary components, show how they can be adapted to time series and reassembled, resulting in a modular and flexible approach;
- we present the first experimental study of local explanations in the context of time series data. Our extensive experiments cover both synthetic and real-world dataset and include a detailed user study.

2375-0197/19/$31.00 ©2019 IEEE
DOI 10.1109/ICTAI.2019.00067

The rest of the paper is organized as follows. In section II we give an outline of existing systems and explain how our proposed method relates to and builds on these precursors. Section III introduces the framework for agnostic local explanations and LEFTIST. Section IV presents our experimental evaluation of our agnostic local explainer for time series classification. Finally, we conclude and discuss future research directions in section V.

II. Related Work

Our focus is on time series classification methods and their interpretability.

1) Time series classification (TSC): With the increase of available temporal data, many TSC algorithms have been proposed in the last decade [3]. Recent approaches have focused on developing ensemble methods, such as for example Shapelet Transform [5], BOSS [6], COTE [7] or HIVE-COTE [8]. These methods significantly outperform older ones [3] but the use of an ensemble of different classifiers reduces the interpretability of the system prediction. More recently, the success of Deep Learning in various classification tasks has motivated its application in TSC [4], [9]. Fawaz et al [4] show that ResNet, the most accurate Deep Neural Network of their study, reaches similar performance as COTE, the best performing classifier in the great time series classification bake off [3]. The use of Deep Neural Networks for TSC will keep increasing but the black-box effect of deep models renders them uninterpretable.

Recently, few TSC models have been designed to be inherently interpretable [10], [11]. Lee et al. [10] use a grammar-based decision tree for interpretable TSC and Teijeiro et al. [11] propose a classifier following the abductive reasoning paradigm rather than the more common deductive reasoning paradigm. These original approaches are not yet well compared to the state-of-the-art TSC methods. Note that shapelet-based time series classifiers might be seen as interpretable as their own. Shapelets are time series subsequences that are discriminative of class membership. Then, the shapelets used for the classification of an instance could be presented to the user, for example, in order to explain the classifier reasoning. But the state-of-the-art in shapelet-based methods [3], Shapelet Transform (ST) and Learning Shapelet (LS), use respectively, an ensemble of different classifiers and shapelets not restricted to real subseries of the data (LS learns near-to-optimal shapelets by exploring shapelet interaction [12]). These characteristics do not allow to consider them as interpretable.

The lack of interpretability can lead to many issues: technical (e.g. for tuning or debugging the classifier), ethical (e.g. for critical decisions provided in medical domain) or legal (e.g. GDPR[1] in Europe). Especially, not providing explanations that are interpretable to a domain expert is critical to the acceptance and the deployment of safety-critical systems [1], [10]. Therefore, a major concern is to design an interpretation layer between accurate, but uninterpretable, classifiers and the user.

2) Interpretability: Intuitively, a classifier is often seen as interpretable if the rationale behind its answers can be easily explained. A formal definition of interpretability remains elusive [13], [14]. Lipton [13] suggests that interpretability refers to more than one concept. To gain conceptual clarity, he proposes to contextualize the motives of interpretability and to consider what properties of models might render them with respect to these motives. For Guidotti et al. [14], an interpretable model is required to provide an explanation. An explanation is considered as an "interface" between humans and a decision maker. Such an interface is both an accurate proxy of the decision maker and comprehensible to humans.

Few papers address a crucial but difficult aspect: the model comprehensibility. Comprehensibility is typically estimated as a function of the model size, based on the number of rules, the number of features or the tree depth [14]. Recently, another type of evaluation of comprehensibility involves user studies [1], [2].

The first dimension of interpretability to be taken into consideration is usually identified as global/local interpretability [14].

3) Global interpretability: Global interpretable models aim to describe the underlying system as a whole, over the entire domain, by means of a simpler, more easily understandable model, that approximates the behavior of the model of interest. RxREN [15], for example, is obtained by reverse engineering a Neural Network and produces a rule-based explanation. As far as we know, no global interpretable model handles time series data. Even though these models are widely used, they are very sensitive to the interpretability-accuracy trade-off [16]. The simpler the proxy, the lower its accuracy and the fidelity to the original model are. Improved accuracy and fidelity come at the cost of increasing the complexity of the proxy, and reducing the interpretability of the obtained explanations.

4) Local interpretability: The most recent body of research on interpretable modules concentrate on local explanations [1], [2], [4], [17]. The interpretable approximation is a classifier that provides explanations for the answer of the black-box in the vicinity of an individual instance of interest. By focusing on a region of the instance space, the complexity of target model can be reduced and the impact of the interpretability-accuracy trade-off decreases.

Most of local explainers are Neural Network-dependant [14]. The explanation is obtained by using a Saliency Mask, i.e. a subset of the original record which is mainly responsible for the prediction. The function to extract the local explanation is typically not generalizable and often strictly tied to a particular type of network. Wang et al. [9] and Fawaz et al. [4] propose an interpretable analysis of TSC with Deep Neural Networks. Wang et al. investigate the use of Class Activation Maps (CAM) to provide an interpretable feedback that highlights the subsequence of time series that contributed the most to a certain classification. Fawaz et al. propose

[1]http://www.privacy-regulation.eu/en/article-14-information-to-be-provided-where-personal-data-have-not-been-obtained-from-the-data-subject-GDPR.html

another visualization technique, based on Multi-Dimensional Scaling, to understand the latent representation learned by the DNNs. The aim is to gain some insights on the spatial distribution of the input time series belonging to different classes in the dataset. Note that employing the CAM or Multi-Dimensional Scaling is only possible with Neural Networks with a Global Average Pooling layer.

Other approaches are model-agnostic and generalized to any black-box classifier. LIME [1] proposes to use an interpretable set of features to ease the explanations for users. The main intuition behind LIME is that the explanation may be derived locally from instances randomly generated in the neighborhood of the instance to be explained. The contribution of the instances to the explanation is weighted according to their proximity to it. In their experiments, the authors adopt linear models as comprehensible local predictor returning the importance of the features as explanation. SHAP [2] introduces the concept of Shapley values from game theory to ensure more specific properties to the explanation. With ANCHORS [17], a variation of LIME, Ribeiro et al. propose to use a set of association rules instead of a linear model as explanation.

III. Agnostic local explainer framework

This section presents our main contribution, the LEFTIST algorithm for agnostic local explanation of black-box time series classifiers. Agnostic means that it can be used for any kind of classifier. The basic principles of agnostic local explanations are first presented, then the LEFTIST approach is detailed.

A. Agnostic local explanations in a nutshell

Let X be a dataset, Y be a set of classes, and $f : X \mapsto [0,1]^{|Y|}$ be a classification model seen as a black-box (f returns a probability distribution vector over classes Y). The goal of local explainer approaches is, given a data point $x \in X$ and a classifier f, to provide an explanation allowing a human to understand why classifier f chose class c for data point x (where c is the most probable class from $f(x)$).

To allow such understanding, it is assumed that for each x, a set of *interpretable components* $\{x'_i\}_{i=1...m}$ can be computed. These components represent, in an human-understandable way, a part of the information conveyed by x. For example, in the image setting, the x'_i can be regions of the original image, and in the text setting they can be words or set of words, possibly enriched with some grammatical information. A *local explanation* for the classification of point x indicates how much each interpretable component contributes to the classification. For example, for classifying an image as a dog, the component with the mouth of the dog is likely to contribute more than a component with the background. To build the local explanation, a simple proxy classifier g_x is trained on the interpretable components, which approximates the decision surface of f around x. The importance of each interpretable component for the classification decision of f over x is then extracted from the model of g_x (as weights $\phi_0, \phi_1, \ldots, \phi_m$). This general approach is illustrated in Figure 1. There are

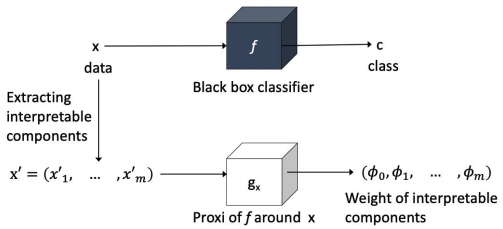


Fig. 1: Agnostic interpretation classification

thus two important elements in this approach: 1) extracting the interpretable components 2) building of the proxy g_x.

Let $x \in X$ be a data point (starting point of Figure 2). Let $x' = \{x'_1, ..., x'_m\}$ be the m interpretable components of x.

In order to build a classifier g_x local to x, a significant portion of the training examples must be neighbors of x. These neighbors are defined in the interpretable components space, by removing one or more interpretable components from x'. The practical methods to choose the components to remove are described in [1], [2]. To represent the neighbors, we introduce a domain of masks over the interpretable components $IF = \{0,1\}^m$, where the i^{th} element indicates the presence or absence of x'_i. For example x is written as $m_x = (1, ..., 1)$ (a vector with only ones). In the following, we denote the masks of nn neighbors of x by $m_x^1, ..., m_x^{nn}$.

The neighbors found lie in the interpretable components space, however their class is not known: a way to compute the class of these neighbors with f is required. The main "trick" of local explainer approaches is thus to define a function $h_x : IF \mapsto domain(X)$ that maps from the masks domain to the original data space (named $domain(X)$). With this function, it is possible to get $z_x^j = h_x(m_x^j)$ the representation of m_x^j, the j^{th} neighbor of x, in the original data space, and thus to compute the class probabilities of m_x^j with $c_x^j = f(z_x^j)$. A training set can then be formed with the m_x^j and their class probabilities $f(h_x(m_x^j))$, from which the local model g_x is learnt.

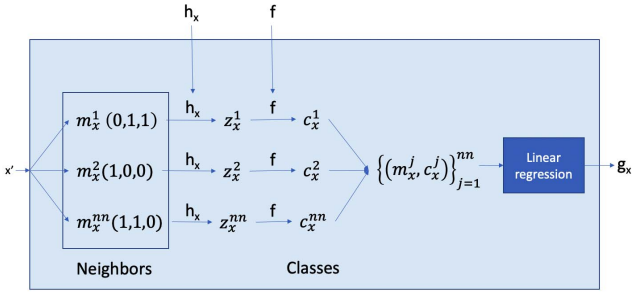


Fig. 2: Building the proxy g_x

This computation of the proxy model is shown Figure 2. It synthesizes the approaches presented in papers describing LIME [1] and SHAP [2].

This model is learnt by selecting K components with Lasso (using the regularization path [18]) and then learning

the weights (ϕ_i) via least squares. This algorithm is called K_LASSO and is described in [1].

Note that for simplicity, this description corresponds to the approach in LIME [1]. In SHAP [2] the approach is similar but h_x is a relation and not a function: it produces several z_x^j, and the mean of their classifications by f is considered to associate a class to m_x^j.

B. *The* LEFTIST *approach*

Now that the general principles of local explanations have been exposed, it is possible to present the LEFTIST approach for local explanations of time series.

A time series t is a sequence of pairs $\langle(ts_1, v_1), ..., (ts_p, v_p)\rangle$ where $\forall i \in [1, p]$ $ts_i \in \mathbb{N}$ is a *timestamp* and $v_i \in \mathbb{R}$ is a value.

The first step is to define the interpretable components of a time series. We consider that some segments of the time series can have interpretable value: they can exhibit particular shapes (ex: peaks) that are easily understandable by humans. This is the basic idea of approaches based on *shapelets*. Note that this also corresponds to the idea of superpixels in the image domain [1].

We thus define the interpretable components of a time series t as the elements of an arbitrary segmentation of t. We denote this segmentation by $S(t)$, with $S(t) = \{S_t^1, ..., S_t^m\}$, and: $S_t^1 = \langle(ts_1, v_1), \ldots, (ts_{t^1}, v_{t^1})\rangle$,..., $S_t^m = \langle(ts_{t^{m-1}+1}, v_{t^{m-1}+1}), \ldots, (ts_p, v_p)\rangle$ where t^i is the end timestamp of the i^{th} segment.

In the interpretable domain, t is represented as $m_t = (1, \ldots, 1)$: it means that t has all the segments of S. In other words, given $\oplus$ the sequence concatenation operator, $t = \bigoplus_{i=1}^m S_t^i$.

Now let m_t^j be the j^{th} neighbor of m_t, where an arbitrary numbers of ones have been flipped to zero. This means that some segments of t are no longer present. The learned model g_t can thus be expressed over $m_t^j = (m_t^{j,1}, \ldots, m_t^{j,m})^T$ (with $m_t^{j,i} \in \{0, 1\}$) $g_t(m_t^j) = \phi_0 + \phi_1 m_t^{j,1} + \ldots + \phi_m m_t^{j,m}$

where $\phi_0, \ldots, \phi_m$ are coefficients in $\mathbb{R}$.

As explained in Section III-A, a function h_t is needed to build g_t. h_t reconstructs a complete time series t_j from m_t^j, according the following formula:

$$h_t(m_t^j) = \bigoplus_{i=1}^{m} \begin{cases} S_t^i \text{ if } m_t^{j,i} = 1 \\ transform_t(S_t^i) \text{ if } m_t^{j,i} = 0 \end{cases}$$

It means that if $m_t^{j,i}$ is equal to 1 the segment i of t is kept, otherwise it is replaced by $transform_t(S_t^i)$, where various transformation functions can be plugged in. The interesting part is to define $transform_t(S_t^i)$ in the above formula. We propose several possible definitions of transform $S_t^i = \langle(ts_{t^i}, v_{t^i}), \ldots, (ts_{t^{i+1}}, v_{t^{i+1}})\rangle$, each of them will be illustrated with the time series t, where $m_t = (1, 1, 1, 1, 1)$, given by Figure 3, each segment is represented by a different color. Each $transform_t$ function concerns $m_t^j = (0, 0, 1, 1, 1)$.

- **Linear interpolation** Let $d_{t,i}(x) = a_{t,i}x + b_{t,i}$ be the line going through $(ts_{t^{i-1}}, v_{t^{i-1}})$ and $(ts_{t^{i+1}}, v_{t^{i+1}})$, meaning going through the last point before S_t^i and the first point after S_t^i.

$$transform_t(S_t^i) = \langle(ts_i, d_{t,i}(ts_i), \ldots, (ts_{t^i}, d_{t,i}(ts_{t^i})\rangle \quad (1)$$

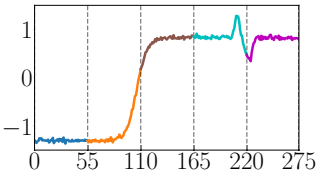
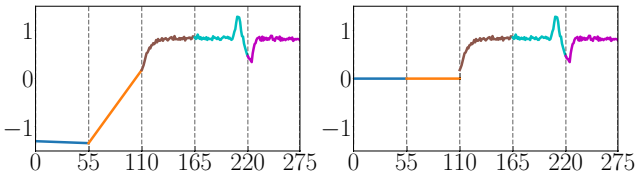


Fig. 3: Initial time series

- **Constant** Let $d_{t,j}(x) = constant$. The constant should be a parameter or it should be computed from the time series (average). In this constant case, $transform_t(S_t^i)$ is also defined by equation (1). A similar function has been used in the image domain [1] where grey pixels replace the removed ones.



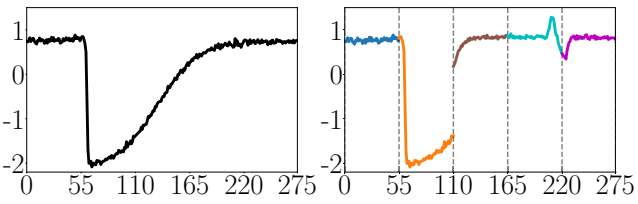
(a) Linear interpolation (b) Constant

Fig. 4: Example of transform function with equation (1).

- **Random BackGround** Here, a set of data BGS (BackGround Set) has to be available and will be used to compute $transform_t$. An example $tr \in BGS$ is randomly chosen ($tr = \bigoplus_{i=1}^m S_{tr}^i$), and the removed segment S_t^i of t is replaced by S_{tr}^i (the i^{th} segment of tr). This function has been used in [17] in the image domain.

$$transform_t(S_t^i) = S_{tr}^i \quad (2)$$

For example, if the random time series tr is given by Figure 5a then Figure 5b shows $h_x(0, 0, 1, 1, 1)$ for the time series t represented in Figure 3.



(a) Time series tr randomly chosen (b) Random Background

Fig. 5: Example of transform function with equation (2).

IV. EXPERIMENTS

The goal of our experiments is to answer the following questions:

- Question 1: Are our explanations faithful to an interpretable classifier?
- Question 2: Are our explanations faithful to complex non-interpretable classifiers?
- Question 3: Are our explanations understandable by humans?

Questions 1 and 2 are related to the **fidelity** metric, used to evaluate interpretable models [15], [14]. Fidelity is the measure of the agreement between the initial (black-box) classifier and the explanations provided by the local interpretable approximation. We first evaluate fidelity with a decision tree based classifier which as the advantage of being a white-box model (section IV-B). Fidelity is then evaluated with three complex black-box classifiers (section IV-C).

Question 3 addresses the **comprehensibility** of an interpretable model. Evaluating the comprehensibility is still an outstanding and difficult problem (cf section II). Comprehensibility concerns human-understandability and thus involves cognitive science [19]. User studies appear to enable an evaluation of comprehensibility [1], [2], [17]. Our user study is detailed in subsection IV-D.

We first present our general experimental setup.

A. Experimental setup

We select 25 diverse datasets from the original 128 datasets of the UCR Archive [20], a well-known database of time series classification tasks.

As seen in III-B, $Background\ transform$ requires to have a pool of time series for function $transform$ (the BackGroundSet BGS). In all datasets, we thus reserve 25% of the training set for BGS. The other 75% are used to train the black-box classifier.

In our experiment we build an interpretable local model for each instance of the test set (test set directly given in the UCR Archive), for the main class c. The parameters for LEFTIST are :

- the method for decomposing the time series in segments. For simplicity we have chosen a uniform decomposition of the data into m segments;
- a number of neighbors, fixed to 1000 (our empirical study has shown that the results did not improve with more neighbors);
- a $transform$ function (cf subsection III-B), chosen in $Linear\ Interpolation$, $Random\ Background$ and $Constant$ (fixed to the average of the input time series);
- the learning process proposed in LIME and SHAP, that we called respectively $learn_{LIME}$ and $learn_{SHAP}$.

Our code and results are available online [2].

B. Fidelity and interpretable classifier (Q1)

The goal of this experiment is to measure the fidelity of LEFTIST against a white-box classifier, where the actual features that led to the classifier prediction can easily be recovered. For such interpretable classifier we use Fast Shapelet (FS) [21], that builds a shapelet-based decision tree. Thus, for each instance, we can compare the shapelets used by FS to classify the instance with the relevant segments (S_t^j) identified by LEFTIST (those with a positive weight).

The experiment protocol is the following: for a given time series t, the path for its classification in the FS decision tree is recovered, let S^+ be the set of shapelets of this path where the decision tree test is positive (shapelets present in t). For each shapelet of S^+, the sum of its coverages with the segments having a positive contribution in the local explanation model g_t is computed. For t we thus have one coverage value by shapelet. This is repeated for all the time series of the test dataset. For each shapelet of the decision tree, the median of the coverages over the test dataset is computed, and last the "score" for the dataset is the median of all these shapelets coverage values. This final score gives an idea of how much the shapelets of FS are covered in the segments of the local explanation.

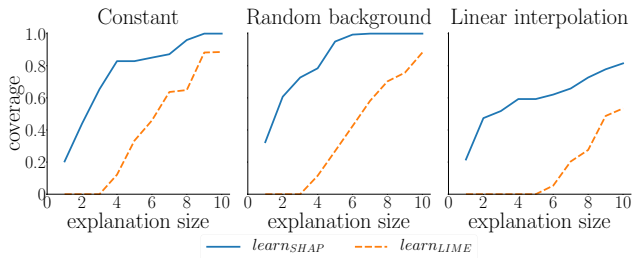


Fig. 6: Median coverage of FS shapelets by local explanations

Figure 6 shows the coverage of FS shapelets by local explanations on all datasets, w.r.t. the size of the explanation (i.e. the number of segments returned by our approach). There is one graphic per transform function, each graphic using both LIME and SHAP learning processes. The most important conclusion is that good coverage values can be obtained with a limited number of segments in the local explanation (≈ 4 segments). This means that the explanation is smaller than the complete time series, and thus that some patterns are captured. The classifier learned with $Random\ Background$ produces local explanations that agree the most with the shapelets of FS, while $Linear\ Interpolation$ produces local explanations having the least agreement with the shapelets of FS. Our interpretation is that the goal of $transform$ is to provide, in the original data space, some representation for "missing" parts of the data, in order for the classifier to put more importance on the "present" interpretable components. As actual data must be used to represent these missing parts, $Random\ Background$ seems to be a good choice, as it provides many different data parts that are far from the original value of the component. On the other hand, $Constant$ is a bit worse as it provides some averaging of the values of the component, and $Linear\ Interpolation$ is the worst as it may be too close from the original component.

[2]https://www.dropbox.com/s/y1xq5bhpf0irg2h/code_LEFTIST.zip?dl=0

C. Fidelity and complex classifier (Q2)

We now evaluate the fidelity of LEFTIST with complex classifiers. This time, we only have access to the results of f: the goal of the fidelity experiment is to determine, for a time series t, if f and the proxy classifier g_t give similar predictions in the vicinity of t. The main difficulty is that g_t operates in the interpretable mask space, whereas we would like to recover its prediction on arbitrary pertubations of t. For this, we define the function go_t that is a mapping of g_t in the original data space:

$$go_t(t') = \phi_0 + \phi_1 cov_{S_1}(t, t') + \ldots + \phi_m cov_{S_m}(t, t')$$

where t' is a time series and $cov_{S_i}(t, t') \in [0, 1]$ is a function that computes the percentage of exact coverage of t and t' in the segment S_i (given segmentation defined before).

With go_t, it is possible to compute class assignments for any time series, which are consistent with the decision surface defined by g_t. This will allow to estimate how much g_t is a good local approximation of f, and how the quality of the approximation degrades when getting further away from t.

A time series t', perturbation of t is built by selecting a number of points of t and altering their value. There are two parameters:

- $ratio_{nbpoint}$ corresponding to the percentage of data point to modify in t.
- $ratio_{amplitude}$ corresponding to the maximum amplitude of modification on the modified data point.

In practice, $ratio_{nbpoint} * size(t)$ data points of t are modified. Each of these points is added a value uniformly drawn from the interval $[-((max(t) - min(t)) * ratio_{amplitude}), (max(t) - min(t)) * ratio_{amplitude}]$. Figure 7 shows, on one dataset (GunPoint), that all the new generated time series are close to t.

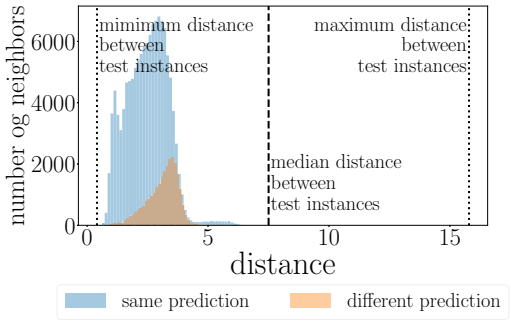


Fig. 7: Proximity of the generated time series

Three black-box models are used for this experiment: a classic support vector machine (SVM) from [22], a shapelet-based classifier Learning Shapelet [12] (LS) and a neural network classifier based on the ResNet architecture (RESNET) [18].

We set $ratio_{amplitude} \in \{5, 10, 15, 20, 25\}$ and $ratio_{nbpoints} \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. Then, we classify the generated instances by both the LEFTIST (g_t) model and the black box model (f).

The experiments are conduced over the 25 datasets mentioned above, where a set of 100 perturbed time series is built for each time series of the test set. For each instance of the test set, an accuracy value is obtained by measuring the percentage of generated time series where the classification of the black box model and of the LEFTIST model agree. The median of these accuracy values is computed over the test set, and further the median over all datasets is computed.

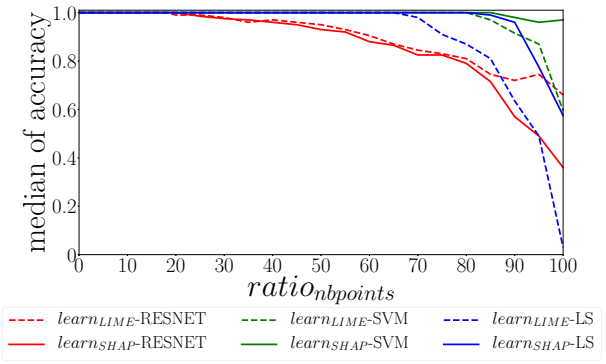


Fig. 8: Evolution of accuracy of black-box approximation with percentage of modified points and different classifiers

Figure 8 shows the evolution of the median of accuracy according to the percentage of modified points. The amplitude of modification is set to 20%: we selected a high value (lots of perturbations) in order to ease the observation of drops in accuracy (low values for this amplitude led to constantly high accuracy values). The number of segments is $m = 10$. One can observe for all classifiers, accuracy remains high up to 40 or 50% of modified points: this means that the LEFTIST local model can approximate the black-box in a fairly large vicinity. Among the classifiers, the accuracy of the approximation is lost first on ResNet: as this is the classifier with the most complex decision surface, this was expected.

We thus fix the classifier as ResNet, which is the most difficult to approximate, and modify the $transform$ function. The results are shown in Figure 9.

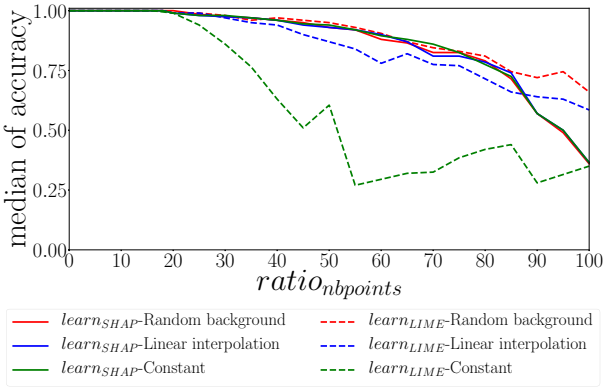


Fig. 9: Evolution of accuracy of black-box approximation with percentage of modified points and different $transform$ functions

Here it is interesting to note that the different $transform$

functions have nearly no influence on the SHAP learner, while they have a strong influence on the LIME learner. This can be easily explained: with a high percentage of modified points, in the formula of go_t, all the cov_{S_i} terms tend to 0, thus $go_t(t') \approx \phi_0$. In the SHAP approach, ϕ_0 does not depend on the $transform$ functions, whereas in LIME, ϕ_0 depends on the $transform$ functions. For LIME, the worst performing $transform$ function is $Constant$. Our experiments show that this crude transformation function brings the neighbors too far from the original instance t, not allowing g_t to approximate well the complex decision surface of ResNet.

Last, we fix the $transform$ function to *Random Background* as it gives the best results overall, and we study in Figure 10 the influence of the number of segments m.

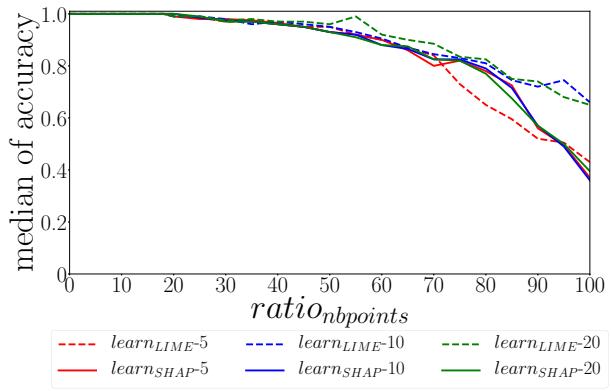


Fig. 10: Evolution of accuracy of black-box approximation with percentage of modified points and different values of m

The number of segments has a limited influence on SHAP, for the same reason as above: go_t tends towards ϕ_0, which does not depend on m in SHAP. For LIME, which is more influenced by m, having more segments leads to a better approximation accuracy. This can be easily explained: with $m = 5$, there are only $2^5 = 32$ different neighbors in the space of interpretable features: this makes few distinct neighbors to learn g_t. For $m = 10$ and $m = 20$ the number of distinct neighbors is much higher, always exceeding the 1000 neighbors that are considered by the algorithm in practice.

D. User study (Q3)

This experiment aims to evaluate the comprehensibility of LEFTIST explanations for users.

The question we are interested in is "Does an explanation *help* a domain expert to understand the class assignment of a black-box classifier?". As the notion of "help" is difficult to answer, we translate this question by "Does an explanation alone allow a domain expert to find the class found by a black-box classifier?".

Asking real domain experts for time series is difficult and time consuming, as this may require considerable background (ex: doctor for ECG signals). Thus, we focused on more general time series datasets, in which there exists characteristic patterns that are relatively easy to identify on visualizations.

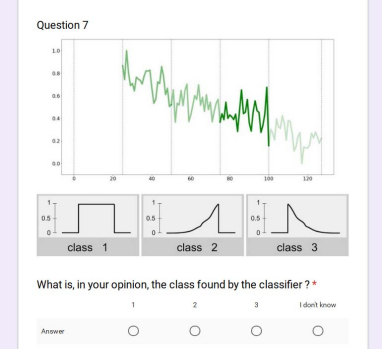


Fig. 11: An example of question in the user study.

These patterns are known beforehand and are provided to the users in order to enable them to become immediately "experts" of these time series (see questionnaire in Dropbox).

Each explanation tested provides the user segments of the time series identified by LEFTIST to explain the prediction of a black-box classifier. Segments are highlighted in green color in a time series visualization, and the more dark green they are, the more relevant LEFTIST estimates them to be. We ask the user to select the class which seems to correspond best to the provided explanation as illustrated in Figure 11. Note the answer "I don't know" is always possible. We then compare the users' class assignment with the classifier one and compute an accuracy score.

Explanations tested are extracted from experiments developed in section IV-C, where we fixed ResNet as the black-box classifier and *Random Background* as the $transform$ function. The questionnaire has not to be too long for a human user, thus we choose to ask about 30 questions with an estimated time of completion of the questionnaire of 15 min. Explanations provided concern three datasets from UCR Archive: CBF (simulated, time series length = 128 points, 3 classes), Trace (simulated, time series length = 275 points, 4 classes) and GunPoint (real, time series length = 150 points 2 classes). Note that the ResNet used in IV-C is very accurate and its class assignment is very good for these three datasets. As we wanted to have cases where the classifier made mistakes, we voluntarily learned another model of ResNet with degraded parameters, which makes a wrong class assignment in 3 of the 33 questions we ask.

We collect answers from 214 users: most of them work or study in computer science (75% vs 15% from another scientific domain and 10% from a non-scientific one).

To ensure users understand time series classification, the questionnaire begins with a short training where some questions are asked about time series classification. Users who fail to answer this test are not kept in the results of the user study (194 users kept).

Results The global accuracy is 0.77 when considering all 33 questions of the questionnaire and 194 users that passed the initial test. This accuracy rises to 0.82 when considering only the 30 questions where the classifier did make a correct

prediction. For all these questions, it is easy for the users to identify, thanks to the segments found by LEFTIST, the class found by the classifier.

However, there may be an ambiguity in these results: from the part of the time series shown, the users may be trying to do the classification task themselves, and answer a different question: *"What is the true class of the shown time series?"*. This is the interest of the questions where we intentionally allowed the classifier to do a mistake: here we can see if the users follow the classifier or what they guess from the time series. There are three such questions:

- On a question for the Trace dataset, the correct class is 1, ResNet found class 3, and no users found class 3. 40% of them find the correct class 1, 16% found class 2, 28% found class 4, and 17% answered "I don't know". Here the majority of users answered the wrong question and LEFTIST did not help. One can note that in this case, most classes exhibit characteristic patterns with some shapes in common, however for this question the distinction occurred on the Y-values in the time series. This misled many users that focused on shapes.
- On a question for the GunPoint dataset, the correct class is 1, ResNet found class 2, and the majority of users (48%) answer "I don't know". The remaining mostly find class 2 (43%) so in this case LEFTIST really helped a significant number of users to understand the prediction of the classifier.
- On a question for the CBF dataset, the correct class is 1, ResNet found class 3, and the majority of users (59%) found class 3 (and 21% found class 1). In this case LEFTIST helped most users to understand the prediction of the classifiers.

In conclusion, this user study shows in easy cases (classifier does not make mistakes), the explanations of LEFTIST allow to understand the prediction of the classifier. And in case where the classifier makes mistakes, there is still a significant number of users which understand the prediction of the classifier thanks to the explanations of LEFTIST.

V. CONCLUSION

In this paper, we presented LEFTIST, the first agnostic local explainer for time series. To build LEFTIST, we analyzed the existing interpretability frameworks and could break them down into elementary components, which we then adapted to time series. The experiments show that our approach has a high fidelity: locally, the explanations provided approximate well complex black-box classifiers such as SVM, Learning Shapelets and ResNet. We also performed a thorough experimental study with 194 users, which demonstrates that the explanations of LEFTIST are understandable in many cases.

A short term perspective is to exploit our framework to deepen the comparison between LIME-based and SHAP-based approaches. A simple improvement would be to explore more complex segmentation choices than uniform segmentation. We would also like to exploit LEFTIST in real applications, such as understanding energy consumption.

REFERENCES

[1] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, 2016, pp. 1135–1144.

[2] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[3] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, May 2017.

[4] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *International Joint Conference on Neural Networks (IJCNN)*, Mar 2019.

[5] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining Knowledge Discovery*, vol. 28, no. 4, Jul. 2014.

[6] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, 11 2015.

[7] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: The collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, pp. 1–1, 09 2015.

[8] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 5, Jul. 2018.

[9] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585, 2017.

[10] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, and J. Silbermann, "Interpretable categorization of heterogeneous time series data," in *SDM*, 2017.

[11] T. Teijeiro and P. Félix, "On the adoption of abductive reasoning for time series interpretation," *Artificial Intelligence*, vol. 262, pp. 163–188, 2018.

[12] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, 2014, pp. 392–401.

[13] Z. C. Lipton, "The mythos of model interpretability," *Commun. ACM*, vol. 61, no. 10, pp. 36–43, 2018.

[14] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, Aug. 2018.

[15] M. G. Augasta and T. Kathirvalavakumar, "Reverse engineering the neural networks for rule extraction in classification problems," *Neural Processing Letters*, vol. 35, no. 2, pp. 131–150, Apr 2012.

[16] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, 2016.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[18] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[19] J. Fürnkranz and T. Kliegr, "The need for interpretability biases," in *Advances in Intelligent Data Analysis XVII - 17th International Symposium, IDA, 24-26, 2018, Proceedings*, 2018, pp. 15–27.

[20] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," October 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[21] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 668–676.

[22] R. Tavenard, "tslearn: A machine learning toolkit dedicated to time-series data," 2017, https://github.com/rtavenar/tslearn.