Random-shapelet: an algorithm for fast shapelet discovery

Xavier Renard*†, Maria Rifqi‡, Walid Erray† and Marcin Detyniecki*§
*Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris.
†Arcelormittal Research, Maizieres-les-Metz, France.
‡Universite Pantheon Assas, Univ Paris 02, LEMMA, Paris, France.
§Polish Academy of Sciences, IBS PAN, Warsaw, Poland.
{xavier.renard, marcin.detyniecki}@lip6.fr, maria.rifqi@u-paris2.fr, walid.erray@arcelormittal.com

***Abstract*—Time series shapelets proposes an approach to extract subsequences most suitable to discriminate time series belonging to distinct classes.**

Computational complexity is the major issue with shapelets: the time required to identify interesting subsequences can be intractable for large cases. In fact, it is required to evaluate all the subsequences of all the time series of the training dataset. In the literature, improvements have been proposed to accelerate the process, but few provide a solution that dramatically reduces the time required to find a solution.

We propose a random-based approach that reduces the time necessary to find a solution, in our experimentation until 3 orders of magnitude compared to the original method.

Based on extensive experimentations on several data sets from the literature, we show that even with a few time available, random-shapelet algorithm is able to find very competitive shapelets.

I. Introduction

Handling time series in the field of machine learning is not an easy task, in particular due the high dimensionality of the data and because the information is temporal. Among the approaches that have been proposed in the last decade, time series shapelets [13], [14] is a promising technique, in particular because it gives competitive accuracies for time series classification tasks.

A shapelet is a subsequence, extracted from a time series, that is the most representative of a class: its occurrence in a time series is characteristic of a class label. For example, we can think about the patterns in heart-rate time series associated with a particular pattern of regular functioning and another pattern associated with a disease.

The shapelet discovery relies on two main steps:

- The enumeration of all possible subsequences of a training set of time series. These subsequences are called shapelet candidates.
- The evaluation of all the shapelet candidates. Shapelet candidates that best separate the classes on the training set are retained.

This approach holds desirable properties such as interpretability and velocity to produce a classification, once the training phase has been achieved.

However, the major weakness of a shapelet-based methods is the time required during the training phase, when shapelets are discovered. The time needed increases faster than the size of the training set. Despite various propositions developed in order to increase the performances, shapelet discovery remains intractable for large data sets and thus, many real life applications.

Often, we have no prior knowledge how the useful information inside time series appears. For the process of shapelet discovery it is undesirable to set up hypotheses to direct the finding process in order to accelerate it, or establish a heuristic.

In this work, we propose the random-shapelet algorithm in order to generate a small fraction of the whole set of shapelet candidates while covering a wide variety of shapes. This is performed by picking in a random order the shapelet candidates to be evaluated. We show on classical datasets from the literature that random-shapelet produces comparable classification results with exhaustive shapelets, while drastically reducing the time required by the discovery process.

The method presents several advantages:

- The random-shapelet algorithm is able to give a rapid valid solution, with only a small fraction of the calculation performed. With more calculation allocated, the shapelet retained is ensured to improve and then converge to the solution of the exhaustive shapelet algorithm. This is possible thanks to randomization: an accurate distribution of the shapelet candidates quickly is obtained by randomly picking in the exhaustive set containing many similar candidates.
- There is no additional parameterization required, except one parameter related to the time available for the calculation. In fact, the size of the subset of shapelet candidates can be adapted to fit the time available for the training phase.
- The velocity of the random-shapelet avoids making assumptions about the shapelet to be discovered. In fact, setting bounds on the shapelet length can be necessary in order to execute the exhaustive shapelet discovery in a reasonable time. The time-saving of the random-shapelet allows to avoid this parameterization and to check more variate shapelet candidates with potentially a higher classification accuracy.

In the following section, we present a brief state of the art on time series shapelet. Section III analyzes the calculation complexity of the shapelet discovery. In section IV, we describe the random-shapelet algorithm. In section V we detail the experimental plan, and in section VI we present the results of the experimentation. Finally, in section VII we discuss the results and form our conclusions in section VIII.

II. Background

A. Time series mining

There are numerous sources of time series in science, industry, health, finance, etc. Thus, applications of every data mining task on this specific kind of data are wide. However, in order to perform classification, clustering, prediction or anomaly detection, typical learning algorithms need to be adapted or a suitable representation of the time series has to be provided. In fact, while static information is meaningful, we are also interested in mining the temporal information.

Many strategies have been proposed in the literature to tackle this problem, reviews are proposed in [12], [2], [1]. Among them, time series shapelet is a promising technique to perform time series classification [13] or other time series mining tasks through a transformation [9].

B. Shapelets

In time series classification, the nearest neighbor algorithm has been shown as particularly effective and difficult to beat in terms of accuracy. Time series shapelet, initially introduced in [13], aims to address three drawbacks of the nearest neighbor algorithm : its space complexity (it is required to build a large corpus of examples), its time complexity to produce a classification (to browse all the individuals of the database), and finally its poor interpretability (there is no explicit insight given by the nearest neighbor).

A shapelet has been informally described as a time series subsequence that is the most representative of a class. Its selection process has similarities to the one use to select a node in a decision tree.

Formally, the discovery process of a shapelet is defined as follows.

1) Notations: We have a dataset D containing n time series T_i with $1 \leq i \leq n$.

A *time series* is an ordered sequence of real values, such as

$$T = (t_1, ..., t_m) \text{ with } (t_1, ..., t_m) \in \mathbb{R}^m$$

where $m = |T|$ is the length of the time series.

A *subsequence* S of a time series T is a sequence of contiguous elements from T. S has length l with $l \leq m$. Then, we get

$$S = (t_p, ..., t_{p+l})$$

with $1 \leq p \leq m - l + 1$.

with m the length of the time series and l the length of the subsequence.

2) Shapelet discovery algorithm: The shapelet discovery remains on two main steps. Once all the subsequences S of the training set D composed of time series have been enumerated with the method *GenerateCandidates*, all of the candidates are checked in order to evaluate their discriminatory power in the method *CheckCandidate*. The best one is returned.

Algorithm 1 ShapeletDiscovery(D, $lmin$, $lmax$)

```
W ← GenerateCandidates(D, lmin, lmax)
best_gain ← 0
gain ← 0
splitPoint ← −1
for all S in candidates do
    spliPoint, gain ← CheckCandidate(D, S)
    if gain > best_gain then
        best_gain ← gain
        best_shapelet ← S
        best_splitPoint ← splitPoint
    end if
end for
return best_shapelet, best_splitPoint
```

a) Shapelet candidates: The shapelet discovery requires the extraction of all the subsequences S_i^l of length $l \in [lmin; lmax]$ for each time series T_i from the dataset D. This is performed in *GenerateCandidates*, see algorithm 2.

The enumeration of all the subsequences S^l of length l of a time series T gives

$$S_T^l = \{S_1^l(T) \ ... \ S_p^l(T) \ ... \ S_{m-l}^l(T)\}$$

Then, the enumeration of all the subsequences of a time series T in the range of lengths allowed $[lmin; lmax]$ gives

$$S_T = \{S_T^{lmin} \ ... \ S_T^{lmax}\}$$

with $1 \leq lmin < lmax \leq m$

These bounds on the length are not easy to set-up but they are essentials for the success of the process to get pertinent shapelets:

- If the interval $[lmin; lmax]$ is too small or badly located, representative subsequences will be missed.
- If the interval is too wide and the exhaustive discovery process is used, the calculations may be intractable, because of the dramatic increase of the shapelet candidates subset that has to be evaluated.

Finally, over the time series of the training set D we get a set of shapelet candidates

$$W = \{S_{T_1} \ ... \ S_{T_n}\}$$

b) Shapelet candidate evaluation: Once the set of shapelet candidates W has been compiled, every candidate of W are checked to evaluate their discriminatory power, in other words, how well each shapelet candidate is able to separate time series associated with different classes.

For each shapelet candidate, the algorithm seeks if we observe a similar subsequence in every time series of D.

Algorithm 2 GenerateCandidates(D, $lmin$, $lmax$)

```
W ← ∅ // Shapelet candidates set
for all T_i in D do
    l ← lmin
    while l ≤ lmax do
        p ← 1
        while p ≤ m − l do
            W ← W + S_p^l(T_i)
            p ← p + 1
        end while
        l ← l + 1
    end while
end for
return W
```

The Euclidean distance calculations d between all the shapelet candidates S of W and every subsequence of every time series T of D are performed.

$$d(S, S_p^l(T)) = \sqrt[2]{\sum_{i=1}^{l}(S_i - S_p^l(T)_i)^2}$$

The minimal distance $min(d(S, T_p))$ over all the possible lengths by time series is retained as exposed in algorithm 3 with method *SubsequenceTimeSeriesDistance*.

Algorithm 3 SubsequenceTimeSeriesDistance(S, T)

```
l ← length(S)
m ← length(T)
p ← 1
while p ≤ m − l do
    dist ← d(S, T_p^l)
    if p = 1 OR dist < minDist then
        minDist = dist
    end if
    p ← p + 1
end while
l ← l + 1
return minDist
```

For each shapelet candidate, we get a table with one distance per time series. This table is ordered increasingly. Then, an optimal split point is sought in order to split the table in two, while maximizing the separation of the classes.

In the original method [13] the discriminatory power of each shapelet candidate is evaluated by the information gain.

The entropy measures how uniform is a set, in our case the class labels associated with the time series in a set. For a set of time series D with a table L_D of classes associated with the time series of D, the entropy is given by

$$H(L_D) = -\sum_{y=1}^{K} p_y.log(p_y)$$

where $y \in \{1, ..., K\}$ is the class label associated with a time series of the data set D, and p_y is the proportion of elements with class y in the set D.

The information gain measures the difference of entropy of a set before and after being split. If we have a table of classes L of time series that is split into two smaller ones L_1 and L_2, we get the information gain IG with

$$IG(L, [L_1, L_2]) = H(L) - (p_1.H(L_1) + p_2.H(L_2))$$

where $H(L)$, $H(L_1)$ and $H(L_2)$ are the entropy of L, L_1 and L_2. p_1 and p_2 are the proportions of time series from L into L_1 and L_2 respectively.

Algorithm 4 CalculateInformationGain($dist$, $splitPoint$)

```
L ← classes of the time series in dist
L1 ← ∅
L2 ← ∅
for all key in keys of dist do
    if key ≤ splitPoint then
        L1 ← L1+class of the time series dist[key]
    else
        L2 ← L2+class of the time series dist[key]
    end if
end for
return IG(L, [L1, L2])
```

By combining the steps of distance and information gain calculations, we get the algorithm 5 named *CheckCandidate*. The minimal distances between every shapelet candidate and all the time series of D are computed and put as keys in a hash table with the time series as values. The hash table is then sorted increasingly. Successive thresholds between two minimal distances are set in order to split the time series associated with their labels in two groups. The information gain is then calculated to evaluate the pertinence of the split regarding the classes. The shapelet candidate with the highest information gain is kept in [13], reflected by the algorithms presented here. In [9], the k best shapelet candidates are retained.

Time series classification using shapelets: Once the shapelet discovery process is completed, shapelets obtained can be used to perform the desired time series mining tasks. In the literature on shapelets, an emphasis has been put on the classification, through the building of a *shapelet-tree* in [13], inspired by the decision tree approach. Extensions have been developed, such as the transformation of time series in [9] into a vector of distances between the time series and the k-shapelets found.

In our work, we use the shapelet-tree algorithm to perform the experimentation. The algorithm 6 gives an overview of the procedure of training.

III. The performance challenge of shapelet discovery

In the shapelet discovery process described previously, the most demanding step in terms of calculations for the overall procedure is the distance calculation performed between all the shapelet candidates and the time series of D.

In fact, the number of distance calculations is large and implies a high time complexity. In [14] this complexity has

Algorithm 5 CheckCandidate(S, D)

```
dist is an empty hash table
for all T_i in D do
    key = SubsequenceTimeSeriesDistance(S, T_i)
    dist[key] ← T_i
end for
Order dist increasingly by the key
best_informationGain = 0
optimalSplitPoint = −1
for all splitPoint calculated as the average between two successive keys do
    informationGain ← CalculateInformationGain(dist, splitPoint)
    if informationGain > best_informationGain then
        best_informationGain = informationGain
        optimalSplitPoint = splitPoint
    end if
end for
return optimalSplitPoint, best_informationGain
```

Algorithm 6 shapeletTree(D, $lmin$, $lmax$)

```
shapelet, splitPoint ← ShapeletDiscovery(D, lmin, lmax)
Split D in two sub-sets D1 and D2 using shapelet and splitPoint
if H(L1) > 0 then
    leftNode ← shapeletTree(D1)
end if
if H(L2) > 0 then
    rightNode ← shapeletTree(D2)
end if
node ← {shapelet, splitPoint, leftNode, rightNode}
return node
```

been evaluated as $O(m^4n^2)$ and qualified as intractable for real-world problems.

For a better view, we can estimate the number of calculations involved in the exhaustive shapelet discovery, while varying the parameters of the method : the range of lengths of the shapelet sought described by $lmin$ and $lmax$ and the size of the time series database D.

The number of shapelet candidates is given by the following formula :

$$Card(ShapeletCandidates) = \sum_{l=lmin}^{lmax} \sum_{T_i \in D} (m_i - l + 1)$$

Then, we can evaluate the number of distance calculations to perform in the evaluation step :

$$Card(DistanceCalculations) = \sum_{l=lmin}^{lmax} \sum_{T_i \in D} (m_i - l + 1)^2$$

In the previous formulas, for a sake of simplicity, the length of all time series is equal to m. A generalization to arbitrary

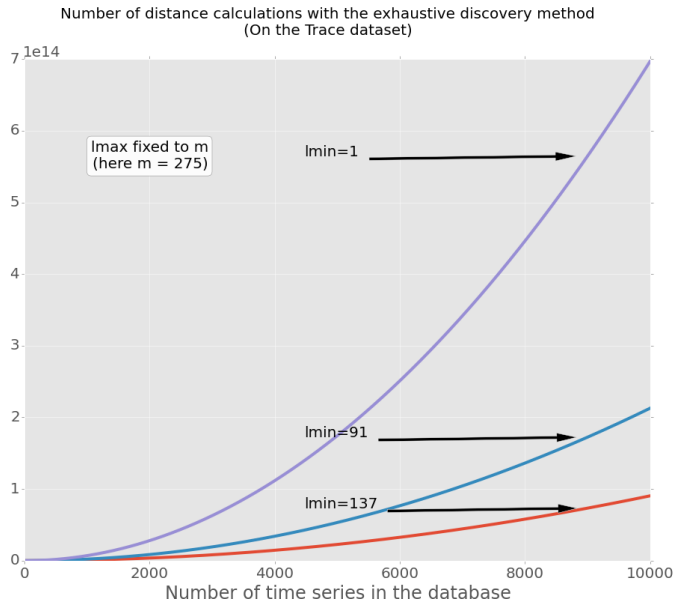

Figure 1. Evolution of the number of distance calculations performed by the exhaustive shapelet discovery algorithm on the Trace dataset. The upper bound $lmax$ for the length of the shapelet candidates extracted is fixed. 3 different values are tested for the lower bound $lmin$. We can observe both the impact of a larger data set and a wider range of possible lengths for the candidates.

lengths m per time series is trivial. The number of distance calculations is considerable and increases quadratically with the number of shapelet candidates, and thus the size of the data set and the bounds on the lengths of the shapelet candidates. Figure 1 shows the number of distance calculations as a function of the size of the training set.

Strategies to reduce the complexity of the discovery process have been proposed:

- To early discard a shapelet candidate as soon as it appears that it will not outperform a previous candidate in terms of information gain [13].
- To early stop a distance computation between a shapelet candidate and a time series, as soon as it appears that a previous shapelet candidate from the same time series produces a lower distance [13].
- To use other quality measures than information gain [9].
- To use classic pruning technique based on triangular inequality to prune candidates and to use caching tricks [10].
- To use a discrete representation of lower dimension of the time series [11], based on SAX [7].
- To use a classification objective function and a tailored stochastic gradient learning algorithm [4].

In terms of time complexity reduction, the method proposed by [11] based on dimensionality reduction and the use of SAX seems to be the most promising, with speedups between 10 and 2000 times faster than the state of the art.

In [3] an approach was proposed to generate quickly a shapelet-based classifier called SALSA. The shapelet discovery, which involves a dose of random, is integrated in a

global method that builds a modified version of the shapelet-tree[13]. The *SALSA* algorithm is not compatible with some other approaches of acceleration for shapelet discovery. Also, the method comes with its own classifier. For example, it is not possible to perform a shapelet transform [9] in order to use a classical classifier.

Our proposition to reduce the time complexity of the shapelet discovery algorithm is a general principle that only affects the shapelet candidate generation and thus is potentially generalizable to all shapelet discovery methods that involve an enumeration of candidates. The random-shapelets approach is generic and can be plugged into any shapelet evaluation algorithms and thus any classifier. It can be considered as an overlayer and is complementary with most of the methods of time complexity reduction detailed previously.

IV. Random-shapelet algorithm

We propose to decrease the time complexity of the shapelet discovery by reducing the size of the shapelet candidate set with a slight modification of the discovery process.

In order to reduce the size of the shapelet candidate set, it is desirable and often necessary to avoid assumptions on the kind of shapelets to discover, such as their possible length, position and shape. While performing data mining tasks, we have no prior knowledge on the data and the phenomenon occurring.

Without hypothesis on the shapelet candidates in order to decrease the time complexity, we have chosen to explore the randomization of the shapelet discovery algorithm.

Instead of checking shapelet candidates with the exhaustive algorithm, we propose to pick randomly candidates in the exhaustive candidate set W produced by algorithm 2[1]. Each picked candidate is then evaluated like in the original discovery method with algorithm 5. When a candidate has been picked and evaluated, it is not replaced in the set of candidates to ensure it will not be evaluated twice. Each candidate is given the same probability of being picked : the random function assumes to have a uniform distribution on candidates.

By sampling the exhaustive shapelet candidate set W randomly, we get an approach distribution $\hat{W}$ of all the possible subsequences in the dataset of time series D, and thus an approach distribution of the shapes contained in D.

$$\hat{W} = \text{randomly select } R\% \text{ of W}$$

The random-shapelet is described in algorithm 7. The only modification of the exhaustive shapelet discovery algorithm concerns line 2 where the randomization step is added. An additional parameter R has to be supplied to the *randomShapelet-Discovery* method. This parameter tells the algorithm the fraction of shapelet candidates from the whole set W that has to be picked to be then evaluated. The value adopted for R should impact directly the time complexity of the method.

The random-shapelet discovery algorithm can be integrated in the shapelet-tree algorithm described in algorithm 6 without any other modification.

[1]While implementing the random-shapelet, it is not necessary to generate all the shapelet candidates but only those picked up randomly.

Algorithm 7 randomShapeletDiscovery(D, $lmin$, $lmax$, R)

```
1:  W ← GenerateCandidates(D, lmin, lmax)
2:  candidates ← Pick randomly R% of candidates in W
3:  best_gain ← 0
4:  gain ← 0
5:  splitPoint ← −1
6:  for all S in candidates do
7:      gain ← CheckCandidate(D, S)
8:      if gain > best_gain then
9:          best_gain ← gain
10:         best_shapelet ← S
11:         best_splitPoint ← splitPoint
12:     end if
13: end for
14: return best_shapelet, best_splitPoint
```

V. Experimentation

The purpose of the experimentation is to evaluate the impact of the generation of a small set of shapelet candidates by random picking of the shapelet candidates using the random-shapelet technique. The impact evaluation is performed according to two criteria: the quality of the returned shapelets and the gain in term of velocity for the whole process of shapelet discovery.

We framed our experimentation in the experimentation plan proposed in [5]. Thus, we have investigated the competitiveness of the classification accuracy provided by our proposition of random generation compared to what can be considered as the standard: the exhaustive generation of candidates. This was performed with the original algorithm, the shapelet-tree, without any modification on the evaluation and classification steps. We could have used other approaches for candidate evaluation or classification, but this choice is not important for our goal. In fact, the idea is to compare the random generation of candidates with the exhaustive generation while the other steps remain unchanged (candidate evaluation and classification).

A. Objectives formally defined

Experimentation of the random-shapelet has to figure out if the use of a fraction of the whole shapelet candidates set W does not degrade the quality of the shapelets returned in terms of accuracy, while being small enough to reduce the time complexity of the shapelet discovery. Experimentation focuses on the evaluation of the random-shapelet discovery method with several values of the proportion R of shapelet candidates, from very small ones to the whole set W.

More specifically, experiments aim to evaluate the random-shapelet on two axis :

- The gain in velocity provided by the random-shapelet versus the exhaustive discovery, which is performed by comparing the time required to produce shapelets.
- The assessment of the quality of the shapelet extracted using random-shapelets discovery, which is performed by measuring the classification accuracy produced by a shapelet-tree as described in [13].

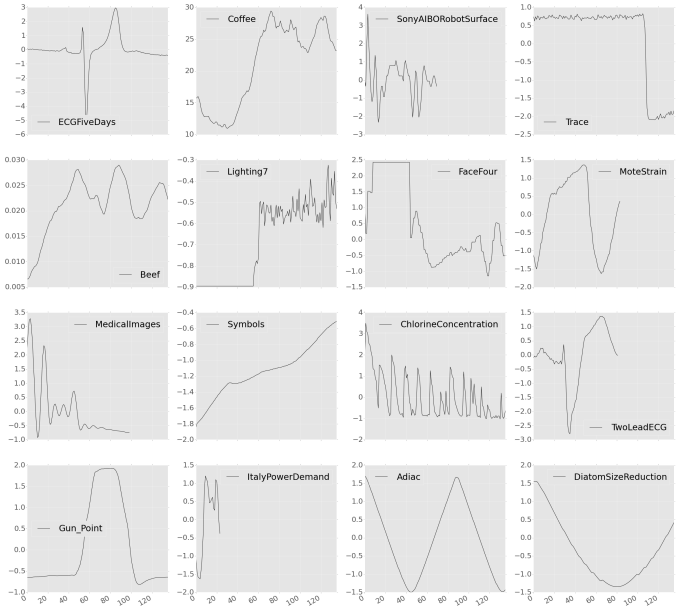

Figure 2. Datasets used for the experimentation are extracted from the UCR time series repository [6]. The first time series of each dataset is represented to give an overview of the type of signals.

B. Data sets

We set up our evaluation in the framework proposed in [5] where they tested the impact on the classification accuracy of the shapelet transform versus the shapelet-tree. We used the same approach to compare the impact of the random candidate generation on the classification accuracy and the shapelet discovery time. Therefore we used the same 16 data sets from the UCR time series repository [6], the same parameters for the minimum and maximum lengths allowed for the shapelet candidates and the shapelet-tree.

The datasets and the parameters used are described in table I and an overview of the profiles of the time series is shown on figure␣2. The UCR data sets contain an interesting diversity of time series, from various origins. Evolution of the signals is also diverse (See figure 2), with some of them monotonous, some more erratic, with more or less velocity of evolution, and different lengths. 7 datasets have 2 classes, while the 9 others have more classes, until 37 for the Adiac data set. The length of the data sets is contained between 24 points per time series until 470. Most of the data sets contains time series longer than 150 points. This variety of characteristics is interesting in order to assess the random-shapelet.

The train/test split proposed by the UCR repository has been used. The training set is used to perform the shapelet discovery and to build the classifier. The test set is only used to evaluate the final trained classifiers. The training sets contain between 16 and 467 time series, while the test set contain between 30 and 3840.

Time series are not normalized before the shapelet discovery is performed and the shapelet-tree is built.

Table I. DESCRIPTION OF THE DATASETS FROM UCR REPOSITORY [6] USED TO PERFORM THE EXPERIMENTATION.

Dataset	Training / Test set size	No. classes	Length	Shapelets min / max
Adiac	390/391	37	176	3/10
Beef	30/30	5	470	8/30
Chlorine Concentration	467/3840	3	166	7/20
Coffee	28/28	2	286	18/30
Diatom Size Reduction	16/306	4	345	7/16
ECG Five Days	23/861	2	136	24/76
Face Four	24/88	4	350	20/120
Gun Point	50/150	2	150	24/55
Italy Power Demand	67/1029	2	24	7/14
Ligthing 7	70/73	7	319	20/80
Medical Images	381/760	10	99	9/35
Mote Strain	20/1252	2	84	16/31
Sony AIBO Robot Surface	20/601	2	70	15/36
Symbols	25/995	6	398	52/155
Trace	100/100	4	275	62/232
Two Lead ECG	23/1139	2	82	7/13

C. Measures

The assessments of the quality of the shapelet extracted is performed measuring the classification accuracy of a shapelet-tree [13], which embed the random-shapelet discovery for its training. The shapelet-tree with the shapelet discovery is built on a training subset, specific for each dataset, while the tests are performed on a separate set.

The gain in velocity provided by the random-shapelet is measured on the time required to train the shapelet-tree and thus discover the shapelets.

Several proportions R of the whole set of shapelet candidates are tried for the random-shapelet discovery. R has been set to 0.1%, 0.2%, 0.5%, 1%, 2%, 10%, 20%, 25%, 33.3% and 50% of the whole set of shapelet candidates for all datasets. The exhaustive discovery, 100% of the shapelet candidates, is also performed and is our baseline.

Because the random-shapelet algorithm introduces uncertainty, inherent to any randomized algorithms, it is required to set up a solid experimentation plan. The building of the shapelet-tree embedding the discovery of the random-shapelet has been performed 100 times for each dataset and each value of R. This is performed in order to evaluate the variability of the method and to ensure significant results.

Experiments and algorithms have been written in Python, using the standard libraries and Numpy, Scipy and Pandas. The code and the details of the experimentation will be available.

VI. RESULTS

Experiments aim to compare the random-shapelet discovery with the exhaustive discovery in terms of classification accuracy and time required by the process.

Table II summarizes the classification accuracies obtained with the mean accuracy obtained per test and the standard deviation associated. A test is defined as the building of a shapelet-tree embedding the shapelet discovery, for a dataset and a proportion R of the whole shapelet candidates set. Each test is run 100 times. For $R = 100\%$, the exhaustive discovery,

the test is run one time : there is no standard deviation associated with the classification accuracy. Table III shows the time required per test. In order to highlight and easily compare the time calculation between the different values of R and over the dataset, the time is normalized with $R = 0.1\%$ as reference. For $R = 0.1\%$, the calculation time is set to 1.

Results are also presented with box-plots. A box-plot represents a test : all the runs for one dataset and one value of R, in order to compare easily the classification accuracy and to evaluate the variability of the results.

Box-plots for some datasets are available on figure 3.

c) Classification accuracy: On table II we observe that the mean classification accuracies produced by the shapelet-tree using random-shapelet discovery remain close to the classification accuracy produced by the shapelet-tree using exhaustive discovery. This observation is valid even for the smallest proportion R tested of the whole set of shapelet candidates. For a value of $R = 0.1\%$, the worse accuracy obtained is 10.1%, lower than the reference based on the exhaustive discovery (ie. when $R = 100\%$). For the same value of $R = 0.1\%$, for most datasets the mean accuracy is less than 5% under the reference. The gap between the random-shapelet and the exhaustive method reduces while R increases. With a proportion $R = 1\%$ for the random-shapelet discovery, mean classification accuracies are competitive with the classification accuracies obtained with the exhaustive discovery.

For $R = 1\%$ the mean accuracy obtained for 5 data sets with the random-shapelet discovery is even higher than the classification accuracy obtained with the exhaustive discovery.

We observe the variability of the results induced by the randomness of the algorithm. The mean standard deviation for the classification accuracy is equal to 5.35% for $R = 0.1\%$. The mean standard deviation decreases with higher values of R : it decreases from 4.2% when $R = 1\%$ to 1.4% when $R = 50\%$.

d) Calculation time : We observe on table III the mean time required for the training phase of the shapelet-tree. The time required for the training coherently increases while R takes higher values.

The gain in velocity obtained using the random-shapelet discovery is considerable : using random-shapelet with a proportion of shapelet candidates $R = 0.1\%$ is 3 orders of magnitude faster than using the exhaustive discovery.

VII. Discussion

We made the hypothesis that building a small set of shapelet candidates picked randomly inside the exhaustive set would tackle the time complexity while preserving the quality of the shapelet discovered.

Experimentation has shown that random-shapelet reduces considerably the time required to discover pertinent time series subsequences.

Random-shapelet should be especially helpful to tackle the problematic of shapelet discovery on large datasets. The method proposed in this article is complementary with other time complexity reduction methods proposed in the shapelet literature. In particular approaches that operate at different stages of the shapelet discovery. Random-shapelet can be seen as an overlayer.

Random-shapelet still presents two difficulties.

The first is related to the randomness of the algorithm, which introduces variability of the accuracy in the result obtained. A way to counter the variability consists in executing the random-shapelet algorithm several times in order to retain the best runs. This strategy is possible because of the speed of the method. Additionally, experiments showed that variability decreases with the proportion R of shapelet candidates.

The second deals with the question of the proportion R of shapelet candidates to use for the shapelet discovery. However, experimentation showed that even on very small values of R, random-shapelet gives good results.

Both difficulties, number of executions of the random-shapelet and proportion of shapelet candidates, imply a trade-off that is not hard to find thanks to the velocity of the method.

A. Intuitive explanation of random-shapelet results

Algorithm 2 of exhaustive generation of shapelet candidates introduces redundancy with strong similar candidates. For example, two consecutive subsequences from the same time series with about the same length contain similar information. In many cases, the minimal distance $min(d(S, T_p))$ between these related shapelet candidates and a new time series should not produce distances significantly different. This difference could even become insignificant in the presence of noise, for example.

However, this intuition is difficult to evaluate with precision. It is also difficult to generalize it to every possible case. We can attempt an illustration of this remark on figure 4.

This intuitive explanation could explain why the random-shapelet works. By picking randomly subsequences, a reasonably representative set of shapelet candidates is built, with less similar candidates. The shapelet discovery process will equally select the most pertinent subsequences among the approximate shapelet candidate set.

B. Impact of results

1) Shapelet candidate length: The exhaustive shapelet discovery requires bounds $lmin$ and $lmax$ to be parameterized, for the length of the shapelet seeked, in order to produce results in an admissible time. The range of lengths authorized for the shapelet discovery is a difficult question to answer.

If we rule out the random guessing for the choice of $lmin$ and $lmax$, bounds can be set up with *a priori* knowledge on the dataset, but this pre-requisite is questionable while using a data mining algorithm, precisely used to discover novelties.

A strategy has been proposed in [9] based on the execution of the shapelet discovery algorithm on a small sample of time series from the whole dataset D while setting $lmin = 1$ and $lmax = |T_i|$ (all the possible lengths are evaluated).

This is an interesting approach, and we propose here to go further and sample randomly the whole set of shapelet candidates from every time series of the training set. Our method is shown to be reliable in comparison with the exhaustive shapelet discovery, with the same bounds .

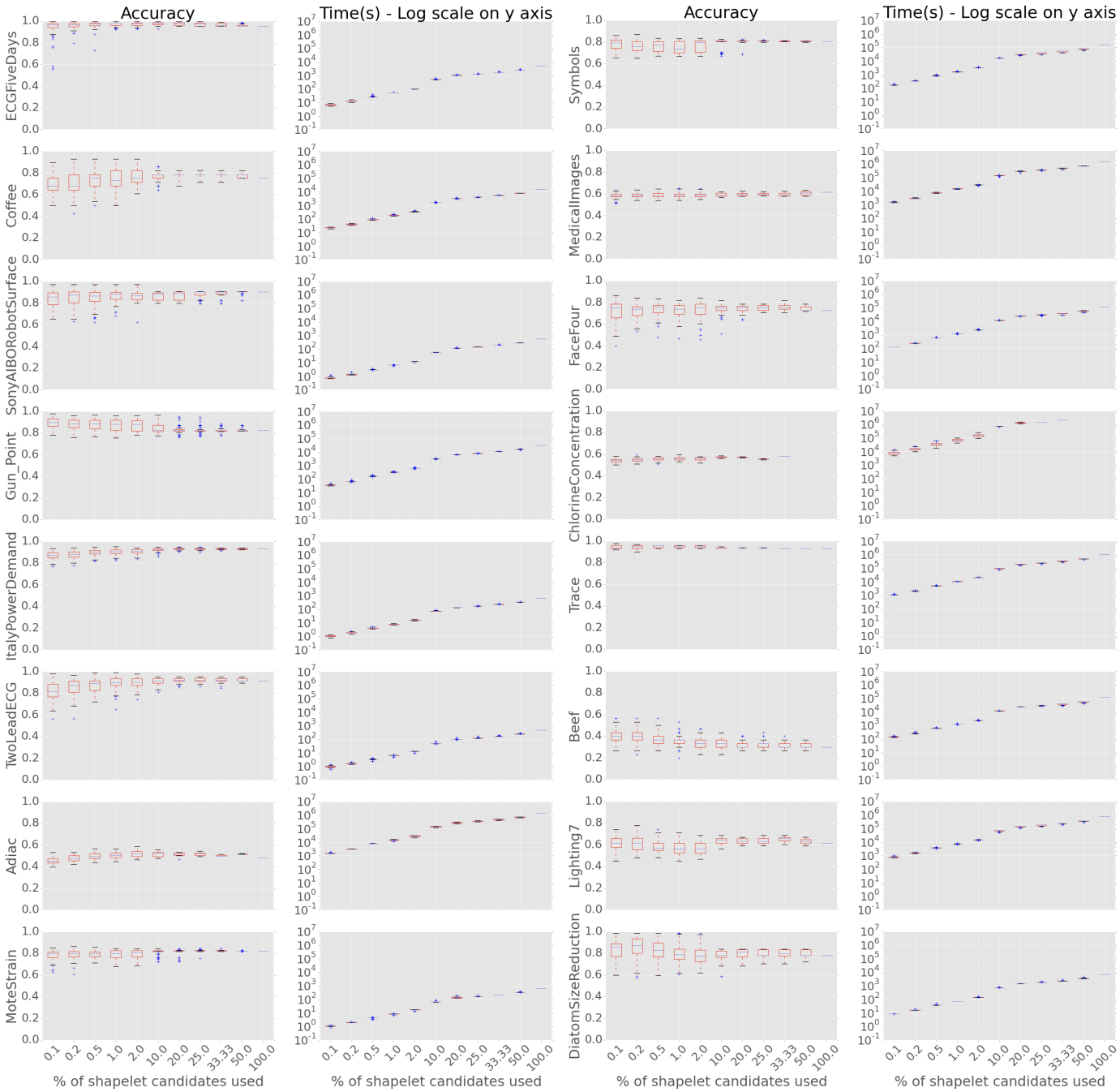

Figure 3. Experimentation results of random-shapelet as a function of R, the fraction of the whole set of shapelet candidates W. Accuracy and time (log scale) are represented in the form of box-plots. The reference is exhaustive discovery where $R = 100\%$.

Table II. ACCURACY WITH STANDARD DEVIATION OF THE SHAPELET-TREE WITH RANDOM-SHAPELET DISCOVERY, FUNCTION OF THE % OF THE WHOLE SET OF SHAPELET CANDIDATES USED. CLASSIFICATION ACCURACY OF THE SHAPELET-TREE WITH EXHAUSTIVE DISCOVERY RELY ON COLUMN 100%. STANDARD DEVIATION IS NOT DISPLAYED : ONE RUN IS NECESSARY FOR EXHAUSTIVE DISCOVERY. THE REFERENCE IS EXHAUSTIVE DISCOVERY WHERE $R = 100\%$. * REFERS TO TESTS THAT DID NOT CONVERGE IN A REASONABLE TIME (SEVERAL WEEKS ON A HIGH PERFORMANCE COMPUTER).

	0.10	0.20	0.50	1.00	2.00	10.00	20.00	25.00	33.33	50.00	100.00
ECGFiveDays	94.2% ±7.9	96.2% ±2.8	96.7% ±3.2	97.2% ±1.6	97.3% ±1.5	97.7% ±1.4	97.3% ±1.4	97.0% ±1.4	96.7% ±1.3	96.1% ±1.0	95.4%
Coffee	69.0% ±8.2	70.3% ±9.4	73.1% ±9.6	74.1% ±8.7	76.4% ±7.8	76.9% ±4.8	78.1% ±2.5	78.3% ±2.0	78.3% ±1.9	78.3% ±2.4	75.0%
SonyAIBORobotSurface	83.8% ±7.7	84.9% ±8.4	85.5% ±7.5	86.9% ±5.6	86.3% ±4.1	86.8% ±3.5	87.2% ±3.8	87.9% ±3.6	88.6% ±3.2	89.8% ±1.5	90.3%
Trace	94.4% ±1.4	94.5% ±1.5	95.0% ±1.3	95.0% ±1.1	94.6% ±1.1	93.4% ±0.6	93.1% ±0.2	93.1% ±0.3	93.0% ±0.0	93.0% ±0.0	93.0%
Beef	40.0% ±6.5	39.3% ±6.3	36.8% ±5.9	35.5% ±5.0	33.6% ±5.1	32.4% ±3.8	31.6% ±3.7	32.0% ±3.5	31.8% ±2.7	31.2% ±1.8	30.0%
Lighting7	61.9% ±5.9	61.0% ±6.8	58.1% ±5.8	56.9% ±6.1	56.9% ±5.6	63.5% ±3.1	63.5% ±2.4	63.9% ±2.9	64.9% ±2.4	63.0% ±2.6	61.6%
FaceFour	72.1% ±9.3	71.6% ±6.8	73.8% ±5.9	72.9% ±6.2	72.8% ±7.4	74.2% ±4.1	74.6% ±2.8	75.0% ±2.2	75.5% ±2.2	74.4% ±2.2	72.7%
MoteStrain	78.4% ±4.3	78.9% ±4.0	79.2% ±3.4	79.2% ±3.8	79.2% ±3.9	81.5% ±2.7	81.8% ±2.2	82.3% ±0.9	82.2% ±0.4	82.0% ±0.3	82.0%
MedicalImages	58.3% ±2.2	58.5% ±1.9	58.8% ±2.2	58.9% ±2.2	58.8% ±2.0	59.2% ±1.7	59.6% ±1.4	59.8% ±1.4	59.7% ±1.9	60.6% ±2.4	61.4%
Symbols	78.0% ±5.1	76.1% ±4.8	76.0% ±4.9	75.2% ±5.3	76.8% ±5.5	79.5% ±4.0	80.6% ±1.7	80.8% ±0.4	80.7% ±0.3	80.6% ±0.4	80.2%
ChlorineConcentration	54.2% ±1.8	54.8% ±1.6	55.7% ±1.6	55.9% ±1.4	56.0% ±1.6	57.2% ±1.0	57.2% ±1.0	55.5% ±0.8	58.1% ±*	*	*
TwoLeadECG	82.2% ±8.5	85.6% ±7.2	87.4% ±6.1	89.5% ±5.3	90.2% ±4.2	91.4% ±3.0	92.1% ±2.0	92.1% ±2.1	92.1% ±2.0	92.0% ±1.5	91.4%
Gun_Point	89.3% ±4.9	88.3% ±4.6	87.9% ±5.1	87.3% ±5.3	87.3% ±5.1	84.4% ±5.3	82.9% ±3.3	82.7% ±2.7	82.8% ±1.6	82.9% ±1.3	82.7%
ItalyPowerDemand	87.1% ±3.5	87.8% ±3.2	90.0% ±2.8	90.4% ±2.5	90.8% ±2.3	92.4% ±1.4	93.0% ±0.8	93.0% ±0.7	93.1% ±0.5	93.2% ±0.4	93.3%
Adiac	45.4% ±2.8	47.7% ±2.8	49.6% ±2.8	50.6% ±2.6	51.9% ±3.0	51.6% ±2.5	51.3% ±2.5	51.7% ±2.2	50.3% ±1.0	51.8% ±0.5	48.1%
DiatomSizeReduction	83.6% ±8.1	85.4% ±9.6	82.5% ±9.5	80.5% ±9.8	78.6% ±7.3	77.4% ±5.3	79.5% ±4.2	79.8% ±3.5	79.9% ±3.7	79.7% ±3.1	77.8%
Mean standard deviation	±5.52%	±5.11%	±4.85%	±4.54%	±4.22%	±3.02%	±2.25%	±1.90%	±1.67%	±1.43%	±0.0%

Table III. NORMALIZED TIME OF SHAPELET DISCOVERY AND THE SHAPELET-TREE BUILDING WITH $R = 0.1\%$ AS REFERENCE SET TO 1, WITH RESPECT TO THE % OF THE WHOLE SET OF SHAPELET CANDIDATES USED. THE REFERENCE IS EXHAUSTIVE DISCOVERY WHERE $R = 100\%$. * REFERS TO TESTS THAT DID NOT CONVERGE IN A REASONABLE TIME (SEVERAL WEEKS ON A HIGH PERFORMANCE COMPUTER).

	0.10	0.20	0.50	1.00	2.00	10.00	20.00	25.00	33.33	50.00	100.00
ECGFiveDays	1	1.8	3.9	7.6	15.0	75.1	149.9	186.5	249.4	373.4	761.1
Coffee	1	1.8	4.2	8.1	15.8	74.9	149.0	184.6	246.5	367.2	735.6
SonyAIBORobotSurface	1	1.7	3.7	8.1	14.1	70.3	145.2	176.2	235.7	350.7	715.9
Trace	1	2.0	4.8	9.6	19.2	91.2	183.5	230.4	304.9	460.5	973.9
Beef	1	1.9	4.4	8.6	16.9	84.1	165.2	204.3	260.8	388.5	820.1
Lighting7	1	1.9	4.5	8.9	17.7	83.2	168.2	209.2	282.1	423.8	887.9
FaceFour	1	1.9	4.7	9.4	18.7	93.2	186.9	231.5	293.5	443.2	945.8
MoteStrain	1	1.8	4.1	7.8	15.3	65.5	128.6	152.7	201.9	307.1	612.8
MedicalImages	1	2.0	4.9	9.8	19.1	91.7	190.8	236.4	312.4	480.5	1000.0
Symbols	1	1.9	4.7	9.2	18.3	91.6	174.4	214.7	288.7	431.4	916.2
ChlorineConcentration	1	2.0	4.5	8.6	19.9	87.4	166.7	174.9	268.7	*	*
TwoLeadECG	1	1.7	3.5	6.5	12.6	54.9	104.5	130.6	173.3	261.2	504.1
Gun_Point	1	1.9	4.6	9.2	18.4	90.7	181.9	227.3	302.9	443.5	907.6
ItalyPowerDemand	1	1.8	3.9	7.2	14.6	66.4	128.3	160.7	217.0	322.7	634.5
Adiac	1	2.0	4.8	9.2	17.3	87.7	174.5	224.3	294.4	434.8	866.6
DiatomSizeReduction	1	1.8	4.4	8.1	15.8	78.6	156.9	196.1	257.7	378.8	796.7
Mean time factor	1.	1.9	4.4	8.5	16.8	80.4	159.7	196.3	261.9	391.2	805.3

Thanks to the time complexity reduction obtained with the random-shapelet, it should be possible to explore a wider range of shapelet candidate lengths directly during the shapelet discovery. It is even conceivable to avoid any assumption on the shapelet length and to be liberated from setting the minimum and maximum length allowed for the shapelet.

2) Exhaustive shapelet discovery and overfitting: On some runs of the experimentation, we observe that shapelet-tree based on random-shapelet discovery produces higher classification accuracy than shapelet-tree based on the exhaustive discovery.

In these cases, the random-shapelet algorithm is probably able to discover shapelets that generalize better on the test set than the shapelets discovered by the exhaustive shapelet algorithm.

3) Anytime random-shapelet discovery: A simple modification of the *randomShapeletDiscovery* algorithm can make it *anytime*. The whole set W is evaluated by picking randomly shapelet candidates without replacement. Whenever a better shapelet is found, the algorithm returns it. Even if the algorithm is stopped at any time during the calculations, the user is in possession of the best-so-far solution. The parameter R that controls the proportion of shapelet candidates to pick is no longer necessary. The anytime version and the classical random-shapelet algorithm rely on the same principle. Thus, conclusions made on the experimentation of the random-shapelets algorithm are identical. The anytime random-shapelet algorithm can be useful when the time available for the calculations is not known at the beginning of the execution.

C. On-going work and further improvements

The random-shapelet has been shown pertinent and further improvements have to be assessed.

The combination of the random-shapelet with other methods of time complexity reduction proposed has to be evaluated.

The impact of exploring wider ranges of lengths for the shapelet candidates on the classification accuracy and the relevancy of the shapelet discovered has to be quantify.

One future direction of work could consist in refining the random-shapelet discovery using local search methods, such as tabu search or other optimization tools. Another direction concerns the trade-off between the proportion R of shapelet candidates and the variability of the accuracy exposed previously. It should be interesting to evaluate the use of cross-validation during the discovery phase of the random-shapelet in order to stop the running when the accuracy is stabilized or

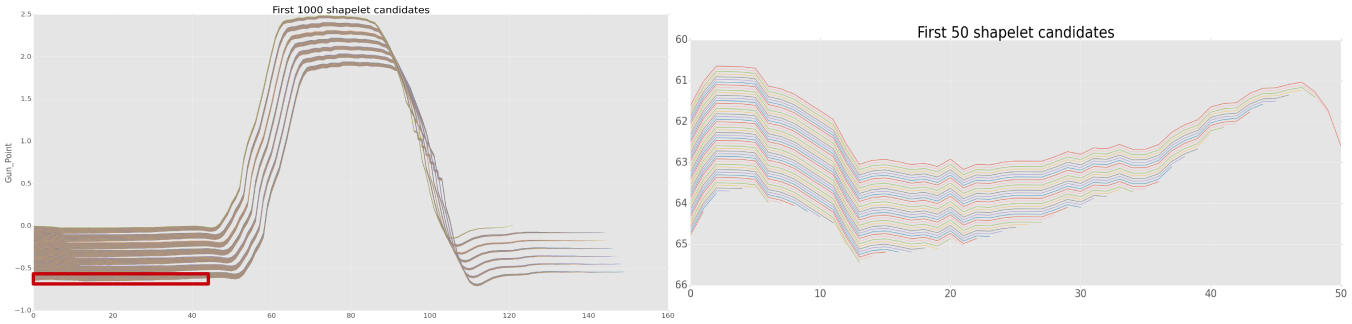

Figure 4. Illustration of the redundancy of the shapelet candidates generated by the exhaustive algorithm. Example provided from the Gun Point dataset on the first 1000 shapelet candidates (left) and zoom on the first 50 shapelet candidates (right)

reach a desired level. The proportion R of shapelet candidates to pick would then be no longer necessary.

VIII. Conclusion and Further Work

In this paper, we have proposed the random-shapelet to reduce dramatically the time required to discover pertinent time series shapelets. The method is based on the randomization of the discovery process.

We have applied the random-shapelet on 16 data sets of the literature. We have shown the relevancy of the approach in terms of quality of the shapelet discovered, evaluated with the classification accuracy, and time complexity reduction, until 3 orders of magnitude.

The method proposed preserves the properties of the shapelet and can be combined with the extensions around the shapelet. Optimizations proposed in the literature to decrease the time complexity of the shapelet could be easily combined with the random-shapelet to go further in the time reduction of the discovery process.

Future directions of work detailed in the discussion concern mainly the assessment of:

- The combination of the random-shapelet with other optimization methods.
- The possibility offers by the random-shapelet to explore longer shapelet candidates.
- The refining of random-shapelet using methods of local search, such as *tabu* search or optimization methods.

We have also planned to use the random-shapelet on real-world cases from the industry, with large data sets.

Acknowledgements

This work is the result of a partnership between UPMC Paris/CNRS (LIP6) and Arcelormittal. Xavier Renard is supported by the French national association for research and technology (ANRT), through the CIFRE convention num. 2014/0068.

References

[1] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):1–34, November 2012.

[2] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, February 2011.

[3] Daniel Gordon, Danny Hendler, and Lior Rokach. Fast Randomized Model Generation for Shapelet-Based Time Series Classification. *arXiv*, 2012.

[4] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 392–401, 2014.

[5] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, May 2013.

[6] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering, 2011.

[7] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, April 2007.

[8] Jason Lines and Anthony Bagnall. Alternative quality measures for time series shapelets. *Intelligent Data Engineering and Automated Learning-* . . . , pages 475–483, 2012.

[9] Jason Lines, Luke M. Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*, page 289, 2012.

[10] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-Shapelets: An Expressive Primitive for Time Series Classification. pages 1154–1162.

[11] Thanawin Rakthanmanon and Eamonn Keogh. Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. pages 668–676.

[12] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40, November 2010.

[13] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. *Conference on Knowledge discovery and data mining*, 2009.

[14] Lexiang Ye and Eamonn Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1-2):149–182, June 2010.