End-of-year Project Report
3rd year GS

Subject:

Clustering Time Series using Unsupervised-Shapelets

Students
Sonokoli Bamara Soro
Yasmine Tarchouna

Academic supervisor
Julien Pelamatti

School supervisor
Sébastien Da Veiga

2024-2025

Acknowledgments

We would like to express our deepest gratitude to our academic supervisor, Julien Pelamatti, for his continuous guidance and invaluable insights throughout the development of this project. His expertise and constructive feedback have been instrumental in shaping the direction of this work.

We also wish to thank our school supervisor, Sébastien Da Veiga, for his helpful advice and input, which have greatly contributed to the refinement of our research.

Finally, we are grateful to Emily Burmeister, our English supervisor, for her careful proofreading and for helping us improve the clarity and quality of our writing, during the first phase of the project.

Abstract

This report explores the Unsupervised-Shapelets (U-Shapelets) algorithm, a subsequence-based method for clustering time series data. The algorithm focuses on identifying discriminative subsequences within time series, reducing the dimensionality of the clustering problem compared to traditional whole time series approaches. U-Shapelets enhances interpretability by focusing on significant temporal patterns and demonstrates robustness to variations such as shifts and changes in scale. In this report, we experiment with various datasets to evaluate the algorithm's performance and practical utility. However, the algorithm encounters substantial computational challenges, particularly during the shapelet extraction phase. Furthermore, U-Shapelets may struggle in cases where time series differences arise from phase shifts or magnitude variations, which the algorithm does not address effectively. The algorithm's performance is also highly dependent on the tuning of hyperparameters. This report suggests that while U-Shapelets is a promising tool for clustering time series, further optimization is needed to improve its efficiency and facilitate its application in real-world scenarios.

Glossary

Algorithm: A step-by-step procedure designed to solve a specific problem or perform a task, producing an output from given inputs. It serves as a fundamental tool in computing and data processing

Centroid: The central point of a cluster, calculated as the mean position of all points in the cluster

Clustering: Grouping data into subsets or clusters where members of each cluster share similar characteristics

Euclidean Distance: A measure of the straight-line distance between two points in Euclidean space, often used to assess similarity between data points

Hyperparameters: Configurable parameters that affect the performance and outcome of an algorithm

k-means clustering: A widely used clustering algorithm that partitions data into K clusters based on proximity to centroids

Rand Index (RI): A measure of similarity between two clusterings. It is computed as the ratio of the number of pairs of elements that are either in the same cluster in both clusterings or in different clusters in both clusterings, to the total number of pairs. The Rand Index ranges from 0 (no agreement) to 1 (perfect agreement).

Subsequence: A subset of consecutive data points from the time series

Time series : A sequence of data points ordered in time, often used for analyzing patterns and trends

U-shapelet(Unsupervised-Shapelets): Subsequence patterns extracted from time series data without labeled information, used for clustering

Contents

List of Figures

Introduction

In an era of big data and increasingly large datasets, clustering techniques have become very popular for classifying data without having any initial knowledge about the class labels. In particular, time series clustering aims to group together time series with similar patterns, considering each time series as a single data object. Time series clustering is popular in many fields, such as finance, health-care, climatology and electrical engineering.

According to a review by Paparrizos et al. (2024) [7], time series clustering methods can be categorized into four main types: Distance-based, Distribution-based, Subsequence-based, and Representation-learning-based. In this report, we will be focusing on a subsequence clustering method called *Unsupervised-Shapelets* or *U-Shapelets*, introduced in 2012 by J. Zakaria, A. Mueen, and E. Keogh [10]. Unlike whole time series clustering, this method purposefully "ignores" some of the data and only focuses on meaningful subsequences, to allow efficient time series clustering in realistic problem settings, according to the authors.

In this report, we begin by defining the problem setting and explaining the motivation for using U-Shapelets. We then introduce the U-Shapelets algorithm, including key definitions and a description of the algorithm. Additionally, we present an experimental evaluation of the algorithm on synthetic datasets to demonstrate its effectiveness and robustness in practical scenarios, as well as explore cases where the algorithm may not perform optimally. Finally, we conclude by summarizing the key findings and suggesting potential improvements to the algorithm.

Chapter 1

Problem setting and motivation

1.1 Problem setting

A *time series* $T = \{t_1, t_2, \ldots, t_n\}$ is a sequence of data points ordered in time, where n is the length of the time series. Examples of time series include daily temperature readings, stock prices over time, heart rate measurements in medical monitoring, etc.

Let $D = \{T_1, T_2, \ldots, T_N\}$ be a dataset of N time series, where $T_i = \{t_{i,1}, t_{i,2}, \ldots, t_{i,n_i}\}$, and n_i is the length of T_i. The *time series clustering problem* is the task of partitioning D into K clusters, where $K \leq N$, such that the time series in each cluster should be as "similar" as possible, while different clusters should be as "dissimilar" as possible.

Figure 1.1 illustrates an example of time series clustering applied to an ECG dataset, which consists of recordings of electrical activity in the heart. In this visualization, each small circle represents a time series, and the data is grouped into two distinct clusters: the blue cluster represents normal heartbeats, while the red cluster corresponds to ischemic heartbeats. This example highlights how time series clustering can effectively differentiate between physiological and pathological patterns in ECG data.

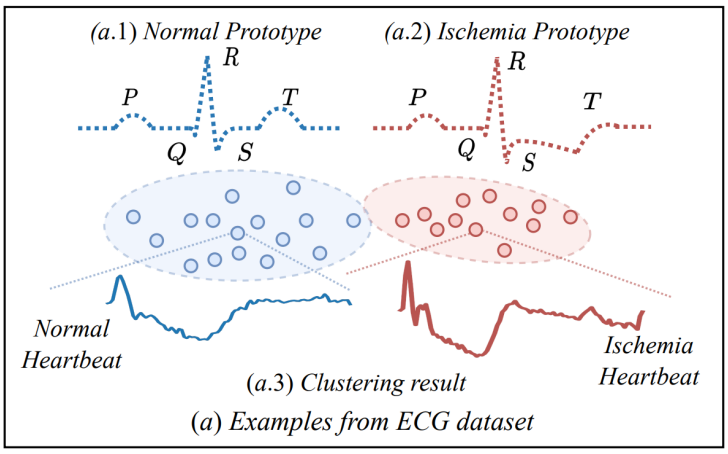

Figure 1.1: A visual example of time series clustering on ECG data. Taken from "Bridging the Gap: A Decade Review of Time-Series Clustering Methods" [7]

Effectively clustering time series data involves measuring similarity between time series and selecting an appropriate clustering algorithm. The similarity measure between time

series depends on the method being used.

As we have mentioned in the introduction, time series clustering methods can be classified into four main families (see Figure 1.2): Distance-based, Distribution-based, Subsequence-based, and Representation-learning-based. An example of a Distance-based method is Dynamic Time Warping (DTW), which calculates the optimal alignment between time series. A Distribution-based method, such as Gaussian Mixture Models (GMM), assumes that data points are generated from a mixture of normal distributions. Subsequence-based methods are further divided into two subfamilies: sliding-window-based methods (which faced criticism in the past for producing results indistinguishable from random clusters [6]) and shapelet-based methods. Finally, Representation-learning-based methods use deep learning techniques to learn representations of time series for clustering. For further details, the reader may refer to the review by Paparrizos et al. (2024) [7].

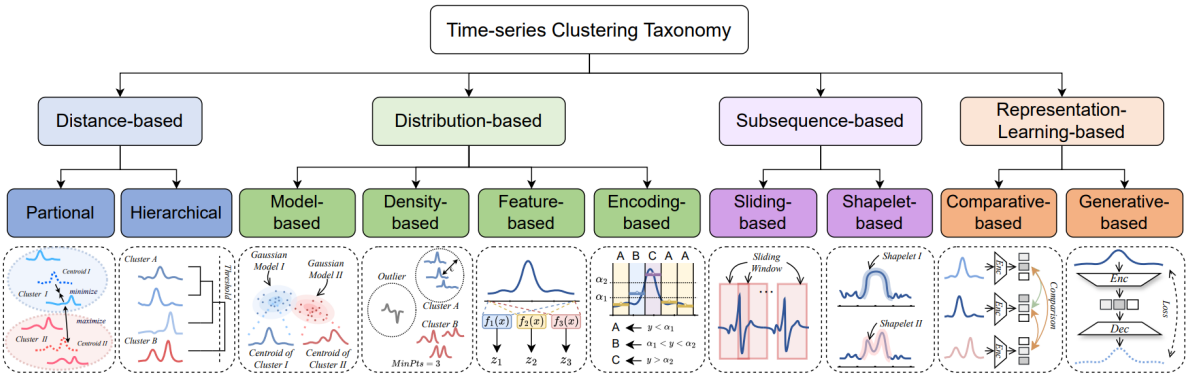

Figure 1.2: The taxonomy of time-series clustering algorithms, taken from "Bridging the Gap: A Decade Review of Time-Series Clustering Methods" [7]

For the purpose of this report, we will be using the k-**means algorithm**, one of the most widely used clustering algorithms. The k-means algorithm iteratively partitions a dataset into K clusters by assigning points to the nearest[1] centroid and updating centroids based on the mean of the assigned points until convergence (see Algorithm 5 in Appendix A).

1.2 Motivation for using U-Shapelets

As explained previously, clustering time series requires grouping together "similar" time series, and separating different ones. Since time series lie in a high-dimensional space, we should define appropriate distance measures that are adapted to this type of data.

An intuitive distance measure that we may be tempted to consider is the Euclidean distance measure, which is given by the formula $d(T_1, T_2) = \sqrt{\sum_{i=1}^{n}(t_{1,i} - t_{2,i})^2}$ for two time series T_1 and T_2 of equal length n.

It should however be noted that this measure gives very poor results if we consider two

[1] With respect to a predefined distance metric

time series with similar shapes but with different orders of magnitude or phases, as it was demonstrated by Keogh et al. [5]. A crucial step is thus to *z-normalize*[2] the time series before computing the Euclidean distance. Figure 1.3 illustrates this: sequences Q and C on the left have similar shapes but differ in Y-axis offsets. As shown on the right, Euclidean distance (visualized with gray hatch lines) exaggerates dissimilarity without normalization, while normalization reveals their true similarity.

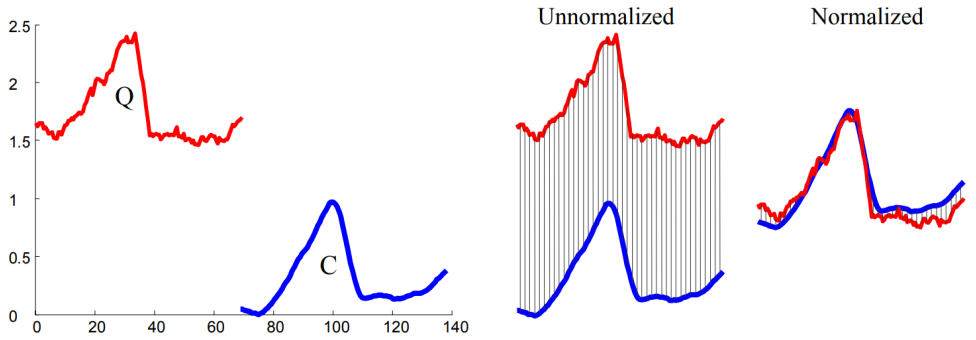

Figure 1.3: A visual intuition of the necessity to normalize time series before measuring the distance between them, taken from "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration."[5]

If we keep the previous notations, we can define the z-normalized time series $\hat{T}_1 = \{\hat{t}_{1,1}, \hat{t}_{1,2}, \ldots, \hat{t}_{1,n}\}$, where each element $\hat{t}_{1,i}$ is computed as:

$$\hat{t}_{1,i} = \frac{t_{1,i} - \mu_{T_1}}{\sigma_{T_1}}, \quad for\ i = 1, 2, \ldots, n$$

such that the mean of the time series, denoted μ_{T_1}, is defined as $\mu_{T_1} = \frac{1}{n}\sum_{i=1}^{n} t_{1,i}$, and the standard deviation σ_{T_1} is given by $\sigma_{T_1} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_{1,i} - \mu_{T_1})^2}$.

Similarly, for a second time series $T_2 = \{t_{2,1}, t_{2,2}, \ldots, t_{2,n}\}$, we compute its z-normalized form $\hat{T}_2$ in the same way.

Once both time series are z-normalized, the *z-normalized Euclidean distance* between T_1 and T_2 is given by: $d(\hat{T}_1, \hat{T}_2) = \sqrt{\sum_{i=1}^{n}\left(\hat{t}_{1,i} - \hat{t}_{2,i}\right)^2}$

While this z-normalized Euclidean distance measure may give decent results, it is still a limited method because it only works when the time series are of equal length. *Dynamic Time Warping* or *DTW* can solve this issue and compute distances between time series that may vary in length as well as frequency. DTW is a method where the time axis is stretched (or compressed) to find an optimal alignment between two time series [2]. Figure 1.4 visually compares the two methods. The Euclidean distance assumes a direct alignment between corresponding points in two time series, which may lead to inaccurate dissimilarity measurements when the series differ in length or speed. In contrast, DTW allows for flexible alignment by adjusting for variations in speed and length, providing a more accurate distance measure.

[2]normalize the time series such that the mean of all of the values is 0 and the standard deviation is 1

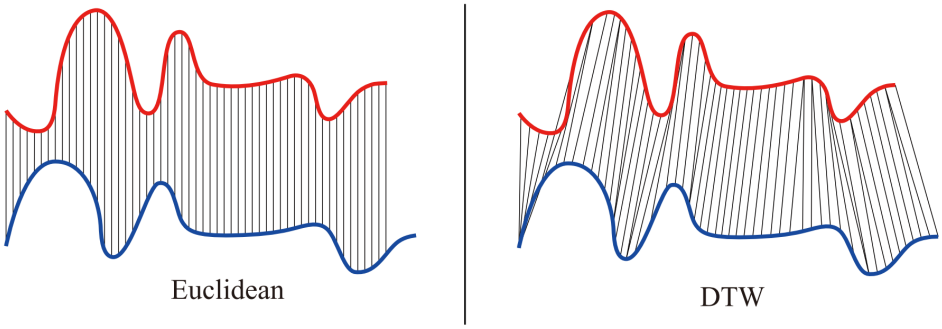

Figure 1.4: An illustration of the difference between Euclidean distance and Dynamic Time Warping (DTW) for time series alignment. Taken from "Dynamic Time Warping for Lead-Lag Relationship Detection in Lagged Multi-Factor Models." [11]

Both the Euclidean distance and Dynamic Time Warping rely on whole time series to assess similarities between them. However, it could be argued that in real world settings, time series data often contains noise or unnecessary parts that could make it difficult to efficiently evaluate similarities between time series using those methods. Zakaria et al. argue that the Euclidean distance and DTW methods work well in previous research because of an overdependence to the UCR[3] time series archive, where data had been manually modified to obtain "cleaner" time series that almost perfectly align [10].

Moreover, time series data often contains redundant patterns, which justifies using a set of "significant" segments of a time series instead of using a single long time series. These significant segments are usually called *shapelets*, and using them can have the advantage of reducing the dimensionality of the clustering problem. The concept of shapelets was first introduced by Ye and Keogh (2009) in their paper "Time Series Shapelets: A New Primitive for Data Mining" [9]. Zakaria et al. (2012) adapted this concept to define *Unsupervised-Shapelets*, which are shapelets that are found without relying on labeled data. We will formally define Unsupervised-Shapelets in the following chapter.

To illustrate this concept, Figure 1.5 shows a potential shapelet, which is a sinusoidal pattern present in two time series. Notice how the shapelet appears in different locations along the X-axis, and exhibits variations in magnitude and length. Despite these differences, the shapelet captures the same underlying pattern. This shows the power of the *Unsupervised-Shapelets* method, which is invariant to shifts, scales, and distortions, allowing it to detect similar patterns across time series even when they differ in these respects.
Using shapelets also makes the results more interpretable. A pattern extracted from a time series often corresponds to a real-world phenomenon, such as a specific economic trend or a physical process. For example, in financial data, a shapelet might represent a recurring market pattern, while in environmental data, it could indicate a seasonal change. By focusing on these meaningful patterns, the clustering process becomes easier to understand and relate to actual events, improving its interpretability. Besides, the *Unsupervised-Shapelets* method addresses the criticisms of traditional subsequence clustering methods. Unlike the sliding window approach, which often produces random and meaningless clusters due to constraints unlikely to be satisfied by any dataset [6], the

[3]University of California, Riverside

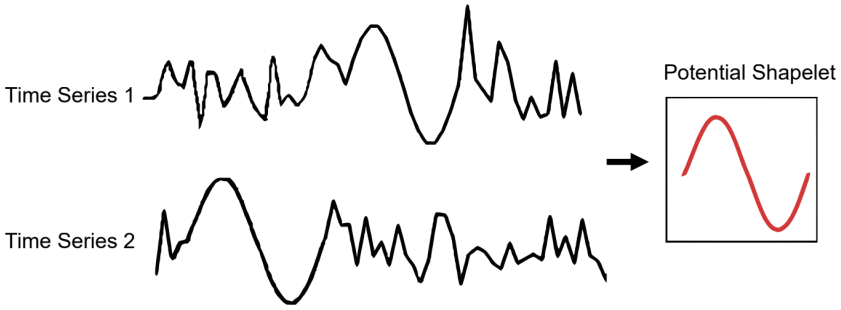

Figure 1.5: An illustration of a potential shapelet extracted from two time series, showing a sinusoidal pattern present in both series. Taken from "Applications of Shapelet Transform to Time Series Classification of Earthquake, Wind, and Wave Data."[1]

Unsupervised-Shapelets method effectively identifies the most informative subsequences and solves these problems.

To sum up, the *Unsupervised-Shapelets* method extracts *subsequences*[4] with significant patterns from unlabeled time series, ignoring irrelevant or noisy parts of the data. By focusing on these meaningful subsequences, the method allows for more efficient distance computation between time series, rather than relying on the full data. By reducing dimensionality and concentrating on meaningful patterns, it not only enhances the efficiency of the clustering process but also makes the results more interpretable.

In the field of hydrology, a promising application of the Unsupervised-Shapelets algorithm is in analyzing water flow data during flood periods, which is especially relevant for companies like EDF (Électricité de France) in managing hydroelectric operations. Flood periods typically feature a peak in water height followed by a decline, and it's essential to detect and isolate abnormal measurements that might arise from sensor errors or handling mistakes. By utilizing this algorithm, we can potentially distinguish plausible hydrographs from those that are not, thereby improving the accuracy of flood monitoring and prediction systems. This capability is vital for enhancing water management strategies and ensuring the reliability of data used in decision-making during critical flood events, which directly impacts the efficiency and safety of hydroelectric operations.

[4]We will formally define subsequences in the next chapter

Chapter 2

The Unsupervised-Shapelets algorithm

Now that we have established the relevance of Unsupervised-Shapelets in time series clustering, we will define key concepts and explain the U-Shapelets algorithm.

2.1 Definitions and key ideas

A *subsequence* $S_{i,l}$ of a time series $T = \{t_1, t_2, \ldots, t_n\}$ is defined as a subset of consecutive data points from the time series. Formally, a subsequence $S_{i,l}$ of length $l \leq n$ can be represented as: $S_{i,l} = \{t_i, t_{i+1}, \ldots, t_{i+l-1}\}$ where $1 \leq i \leq n - l + 1$.
Here, i is the starting position of the subsequence in the time series T, and l is the length of the subsequence.

The subsequences of each time series are the primary candidates for identifying shapelets. As we have seen in Section 1.2, it is crucial to z-normalize time series before computing their Euclidean Distance. Since we will be comparing subsequences of different lengths, we also need a length-independent distance measure. This is critical because in most cases we expect the distance between shorter subsequences will be less than the distance between longer subsequences [10].

The *length-normalized Euclidean distance* between two time series $T_1 = \{t_{1,1}, t_{1,2}, \ldots, t_{1,n}\}$ and $T_2 = \{t_{2,1}, t_{2,2}, \ldots, t_{2,n}\}$ of same length n, can be computed as:

$$dist(T_1, T_2) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(t_{1,i} - t_{2,i})^2}$$

Since we aim to find the distance between candidate shapelets and time series, it is essential to calculate the distance between a short subsequence and a much longer time series. The key idea is to "slide" the subsequence over the time series, comparing it to each possible subsequence of the same length within the time series, and identifying the alignment that minimizes the distance.

The *subsequence distance* between a subsequence S of length m and a time series T of length n is defined as the smallest distance between S and any subsequence of T that

has the same length m. This is mathematically expressed as:

$$\text{sdist}(S, T) = \min_{1 \leq i \leq n-m} \text{dist}(S, T_{i,m})$$

where $T_{i,m}$ represents the subsequence of T starting at position i with length m, and $\text{dist}(S, T_{i,m})$ is the length-normalized Euclidean distance between S and $T_{i,m}$.

Now we can finally define what we have been calling "Unsupervised-Shapelets". Technically, a shapelet could be any time series. However, to reduce the candidate space and facilitate the search, we focus only on subsequences of time series from the dataset. This implicitly assumes that the patterns which best separate the different clusters are contained within the time series being considered. Formally, an *Unsupervised-Shapelet* is a subsequence of a time series T for which the subsequence distances (sdists) to the time series from a group D_A are "much smaller" than the sdists to the time series in the rest of the dataset, D_B. In other words, the subsequence S' is a good shapelet if it represents a pattern that strongly distinguishes the group D_A from the rest of the data in D_B. This can be mathematically expressed as:

$$sdist(S', D_A) \ll sdist(S', D_B)$$

where S' is the subsequence, D_A is the subset of time series to which S' is most similar, and D_B is the remaining subset of time series in the dataset D.

The *separation gap*, denoted as *gap*, is a measure of how well the subsequence S' distinguishes the two groups. S' is considered a good U-Shapelet if it results in a large separation gap, indicating that the subsequence distances to D_A are much smaller than those to D_B. The separation gap is defined as:

$$gap = (\mu_B - \mu_A) - (\sigma_B + \sigma_A)$$

where:
- μ_A and μ_B are the means of the subsequence distances to the time series in D_A and D_B, respectively.
- σ_A and σ_B are the standard deviations of the subsequence distances to the time series in D_A and D_B, respectively.

The difference in means, $\mu_B - \mu_A$, indicates how far apart the clusters are, while the sum of standard deviations, $\sigma_B + \sigma_A$, reflects the spread within each cluster. By maximizing this gap score, we favor u-shapelets that create well-separated and compact clusters, ensuring clear and distinct groupings in the data.

In most practical scenarios, extracting multiple U-Shapelets is essential for obtaining meaningful clusterings, as a single pattern may not be sufficient to distinguish between all the time series. To handle this, we define a *Distance map*. A *Distance map* is a matrix that contains the subsequence distances (sdists) between each of the U-Shapelets and all the time series in the dataset. If we have m U-Shapelets and N time series in the dataset, the size of the Distance map is $N \times m$. Each column in the matrix represents the distance vector of a single U-Shapelets, where each entry in that column corresponds to the subsequence distance between the U-Shapelets and a particular time series. A distance vector can be represented as a schematic line, called an *orderline*. The orderlines are

analyzed to identify the point, denoted as dt, that maximizes the previously defined gap function. Distances to the left of dt correspond to sdist(S', D_A), while distances to the right correspond to sdist(S', D_B).

To visualize this, let's look at Figure 2.1, which uses data from a synthetic dataset introduced in [4]. The dataset consists of four classes of time series, each embedding class-specific signals with significant random variations in length and height, along with additional noise. Figure 2.1 illustrates a U-Shapelet extracted from class 1 and its cor-

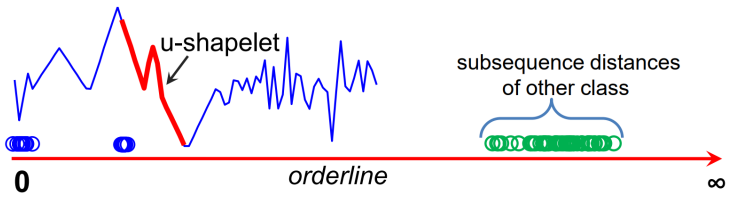

Figure 2.1: U-Shapelet extracted from class 1 and the corresponding orderline. The figure was taken from "Clustering Time Series Using Unsupervised-Shapelets"[10]

responding orderline[1]. This U-Shapelet is the distinguishing pattern between class 1 and class 4. In the orderline, distances to the left of the threshold d_t represent sdist(S', D_A), i.e., distances between the U-Shapelet and time series in class 1, while distances to the right represent sdist(S', D_B), i.e., distances to time series in class 4 [2]. The separation gap highlights the effectiveness of the U-Shapelet in distinguishing between these two classes.

2.2 Description of the algorithm

The Unsupervised-Shapelets algorithm consists of several interconnected components, each with a specific role in identifying and utilizing U-Shapelets for clustering. The algorithm consists of two main steps: shapelet extraction and clustering.

2.2.1 Shapelet extraction

Shapelet extraction is ensured by three algorithms, with Algorithm 1: *Extraction of U-Shapelets* being the main driver of the process. This step focuses on identifying and extracting a set of u-shapelets from the time series dataset.

Algorithm 1: Extraction of U-Shapelets

The primary purpose of this algorithm is to iteratively select the best u-shapelet at each step based on its ability to divide the dataset into two groups: one group that is similar to the u-shapelet and another that is not.

The process begins by initializing an empty set for storing u-shapelets and selecting the first time series from the dataset (Lines 1-2). It then considers all possible subsequences of this time series as candidate u-shapelets (Lines 4-5). For each candidate, the algorithm

[1] A visual line representing the subsequence distances between this U-Shapelet and the time series from classes 1 and 4 in this case

[2] while usually D_B consists of all the time series in the dataset after exclusing D_A, in this example, only the time series in class 4 are considered for simplification

computes a gap score, which quantifies how well the candidate separates the data (Line 6). The gap score is calculated using Algorithm 2: *Compute Gap*, which evaluates the separation quality based on the distances between the candidate u-shapelet and all other time series in the dataset.

Once the gap scores are computed for all candidates, the algorithm selects the candidate with the maximum gap score and adds it to the set of u-shapelets (Line 16). To avoid redundancy, the algorithm removes time series similar to the selected u-shapelet from the dataset before proceeding to the next iteration (Lines 11-18). This removal is based on a threshold θ.

The process is repeated until no more useful u-shapelets can be found, ensuring that the final set of u-shapelets is both effective and non-redundant.

Algorithm 1 Extraction of u-shapelets

Require: D: The dataset, L: Shapelet length, k: The number of classes
Begin:

1: Initialize an empty list to store the u-shapelets.
2: Consider the first time series ts from the dataset.
3: **while true do**
4: **for** length $i = 1$ to L **do**
5: **for** each subsequence of ts of length i **do**
6: Compute the gap and the distance dt using **Compute Gap** (see Algorithm 2).
7: **end for**
8: **end for**
9: Select the subsequence with the maximal gap and its corresponding distance dt.
10: Compute the distance (dis) between this subsequence and each time series in the dataset using **Compute Distance** (see Algorithm 3).
11: Find DA, the set of time series with a distance to the subsequence less than dt.
12: Find $disDA$, the set of corresponding distances.
13: **if** length of $DA = 1$ **then**
14: **break**.
15: **else**
16: Set ts to the time series corresponding to $\arg\max(dis)$.
17: Retain only time series in D with distances higher than $\theta = \text{mean}(disDA) + \text{std}(disDA)$.
18: Remove the time series with a distance less than θ. The aim is to keep the time series corresponding to the U-shapelet not already selected, avoiding redundancy
19: **end if**
20: **end while**

End.

Algorithm 2: Compute Gap

This algorithm is designed to evaluate the quality of a candidate u-shapelet by computing its gap score. The gap score measures how well the candidate separates the dataset into two distinct clusters. The algorithm begins by computing the distance vector between the candidate u-shapelet and all time series in the dataset (Line 1). This distance vector

is then sorted to facilitate the calculation of the gap score (Line 2).

The algorithm iterates over all possible locations for the threshold dt that separates the data into two clusters (Lines 4-15). For each location, it calculates the gap score, which is based on the difference between the mean distances of the two clusters and their standard deviations. The goal is to find the location that maximizes this gap score, indicating the best separation between the clusters.

The algorithm also checks that the ratio of the sizes of the two clusters is within a certain range to ensure that the u-shapelet has discriminative power and is not merely identifying outliers or universal patterns (Line 9). The maximum gap score and the corresponding threshold dt are returned, providing a measure of the candidate u-shapelet's effectiveness in separating the data (Line 16).

Algorithm 2 Compute Gap

Require: `subsequence`: The subsequence for which the gap is to be calculated, D: The dataset, k: The number of classes

Begin:

1: Compute the distances *dis* between `subsequence` and each time series in the dataset D using **Compute Distance**.
2: Sort the distances in ascending order.
3: Initialize `maxGap` $\leftarrow 0$ and `dt` $\leftarrow 0$.
4: **for** each pair of successive distances in the sorted list **do**
5: Compute the midpoint d as the average of the two successive distances.
6: Identify DA: The subset of distances less than d.
7: Identify DB: The subset of distances greater than or equal to d.
8: Compute the ratio $r \leftarrow \frac{\text{size of } DA}{\text{size of } DB}$ (if DB is empty, set $r \leftarrow \infty$).
9: **if** $\frac{1}{k} < r < 1 - \frac{1}{k}$ **then**
10: Compute the gap as: $gap \leftarrow \text{mean}(DB) - \text{std}(DB) - \text{mean}(DA) + \text{std}(DA)$.
11: **if** $gap >$ `maxGap` **then**
12: Update `maxGap` $\leftarrow gap$ and `dt` $\leftarrow d$.
13: **end if**
14: **end if**
15: **end for**
16: **Return** `maxGap` and `dt`.

End.

Algorithm 3: Compute Distance

The purpose of this algorithm is to compute the distance vector between a candidate u-shapelet and all time series in the dataset. This distance vector is crucial for evaluating the quality of the candidate u-shapelet and is used in Algorithm 2: Compute Gap to calculate the gap score.

The algorithm begins by initializing an empty set for storing distances and normalizing the candidate u-shapelet (Line 1). For each time series in the dataset, it computes the minimum distance between the u-shapelet and all possible subsequences of the time series (Lines 4-8). This involves sliding the u-shapelet along the time series and calculating the

Euclidean distance at each position. The minimum distance is then normalized by the length of the u-shapelet to ensure comparability across different lengths.

The set of distances is returned, providing a measure of the similarity between the candidate u-shapelet and each time series in the dataset (Line 11).

Algorithm 3 Compute Distance

Require: *shap*: A subsequence, D: A dataset containing time series.
Begin:

1: Initialize an empty list *distances* to store the minimum distances for each time series.
2: **for** each row i in D **do**
3: Extract the time series $ts \leftarrow D[i]$.
4: Set $min\_distance \leftarrow \infty$ (to track the smallest distance).
5: **for** each subsequence of ts of length equal to *shap* **do**
6: Compute the Euclidean distance *dist* between the subsequence and *shap*.
7: Update $min\_distance \leftarrow \min(min\_distance, dist)$.
8: **end for**
9: Add *min_distance* to *distances*.
10: **end for**
11: **return** *distances*.

End

Figure 1.5 illustrates two examples of shapelets, one "good" (in green) and one "bad" (in yellow) for clustering four time series. Time series with upward peaks are shown in blue, while those with downward peaks are shown in red. As seen, the computed distance (Dist) between the "good" shapelet and each time series is small for the series with upward peaks (0 and 2.59) and larger for the series with downward peaks (8.5 and 8.86). However, for the "bad" shapelet, the distances are similar across all time series, indicating that this shapelet lacks discriminative power.

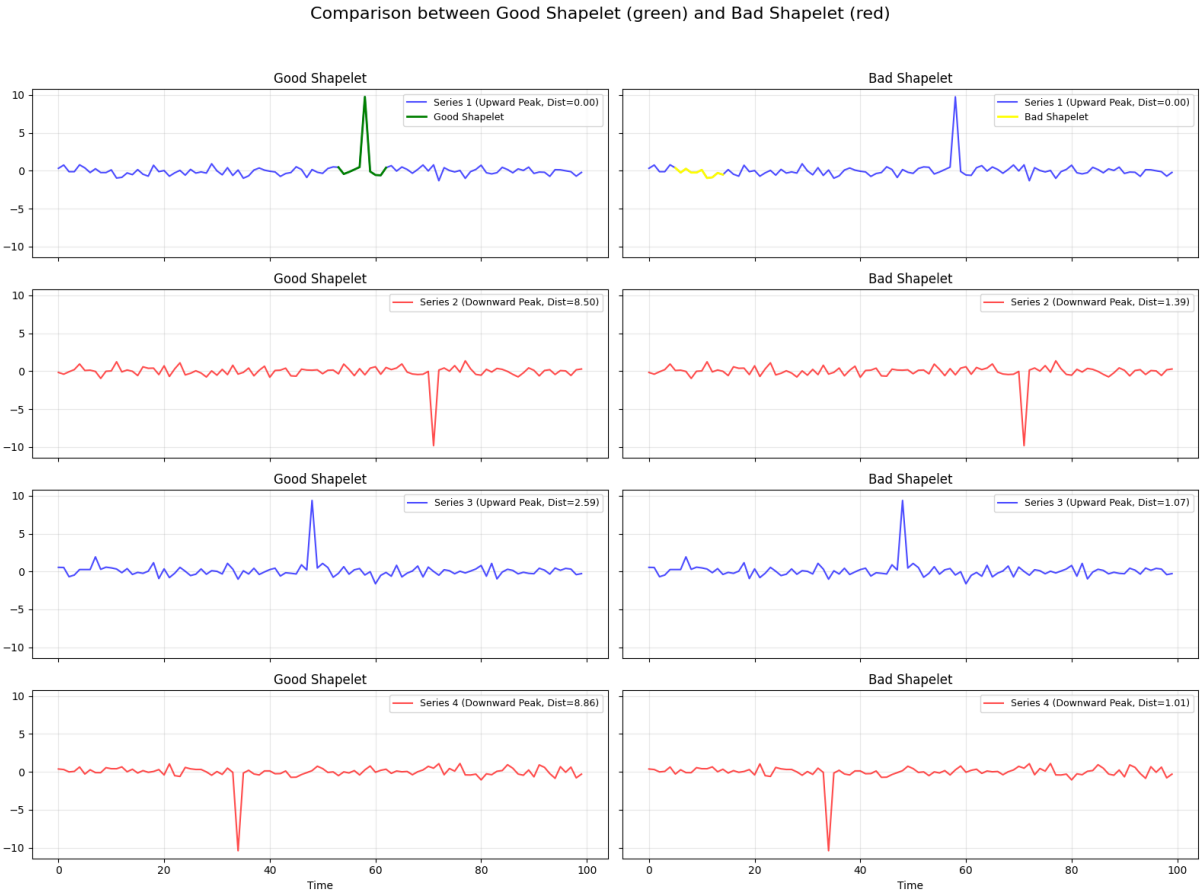

Figure 2.2: Comparison between good and bad shapelets in clustering time series.

2.2.2 Clustering

Once shapelets have been extracted, the algorithm proceeds to the clustering phase. Here, algorithm 4 uses the identified shapelets to group the time series into clusters. For each time series in the dataset, the algorithm calculates its distance to each of the extracted shapelets. Time series that are similar to a shapelet (i.e., those with smaller distances) are grouped together. Through this clustering process, the time series are assigned to clusters based on their similarity to the identified shapelets. The algorithm uses k-means clustering to finalize the assignment of time series to clusters.

During the clustering iterations, the algorithm also tracks the Rand Index (RI), a measure of the similarity between two clustering results. In particular, the complement of the Rand Index ($1-RI$) is recorded to evaluate the consistency of clustering across iterations. This helps in identifying whether further iterations improve the clustering.

Algorithm 4 Cluster Data using Shapelets and K-Means

Require: D: The dataset, $Shapelets$: Set of shapelets, k: Number of clusters

Begin:

Initialize an empty list DIS to store distances between each shapelet and the time series.
Initialize a zero vector to manage the cluster labels for each series.
Set $sumDis \leftarrow \infty$, representing the minimum cluster inertia.
Initialize an empty list CRI to store $1 - RI$ values (Rand Index complement).
Set $cls \leftarrow$ **None** to hold current cluster labels.
if $Shapelets$ is empty **then**
  return {}.
end if
for each shapelet s in $Shapelets$ **do**
  Compute distances between s and all time series in D using **computeDistance**.
  Append the computed distances to DIS.
  Transpose DIS to form DIS_matrix.
  Perform k-means clustering on DIS_matrix with k clusters.
  Retrieve cluster labels IDX and inertia $SUMD$.
  if $SUMD < sumDis$ **then**
    Update $sumDis \leftarrow SUMD$ and $cls\_current \leftarrow IDX$.
  end if
  if iteration > 1 **then**
    Compute Rand Index RI between cls and $cls\_current$.
    Append $1 - RI$ to CRI.
  end if
  Update $cls \leftarrow cls\_current$.
end for
if CRI is not empty **then**
  Identify the iteration a with minimal change in $1 - RI$.
  Set $final\_labels \leftarrow cls\_current$ from iteration a.
else
  Set $final\_labels \leftarrow cls$ (last iteration labels).
end if
Create a dictionary mapping each time series to its cluster label.
return The resulting dictionary.

End.

Chapter 3

Experimental evaluation

In this chapter, we describe the implementation of the U-Shapelets algorithm, and explore its application across various datasets to evaluate its performance and robustness.

3.1 Algorithm implementation

We implemented the Unsupervised-Shapelets algorithms as described in the original article using Python. Initially, we followed the pseudocode outlined in the paper exactly. However, we quickly realized that the algorithms were computationally expensive, even for relatively small datasets. This was due to the high time complexity of the Shapelet Extraction algorithm, as detailed in Appendix B.

To tackle this issue, we made several modifications to improve efficiency and scalability. First, we introduced a parameter for the minimum length of u-shapelets to consider. The original algorithm only defines the maximum length and sets the minimum length to 1 by default, which is often impractical for identifying meaningful patterns. By allowing a user-defined minimum length, we can focus on more relevant subsequences that are likely to capture significant patterns in the data.

Additionally, we used vectorization techniques to speed up the algorithm. One key technique we employed was the `as_strided` function from the NumPy library. The `as_strided` function allows to create sliding window views over arrays without actually copying the data. By using this technique, we we able to efficiently compute distances between subsequences and time series without the need for explicit loops, significantly reducing computational cost.

3.2 Application to a synthetic dataset with peaks

We began by generating synthetic time series data containing peaks, either pointing upwards or downwards. This initial example aimed to verify that the algorithm could successfully detect two "simple" shapelets, corresponding to an upward peak and a downward peak, respectively (see Figure 3.1).

The algorithm distinctly recognized the trivial shapelets representing upward and downward peaks (see Figure 3.2). This demonstrates the algorithm's ability to identify and

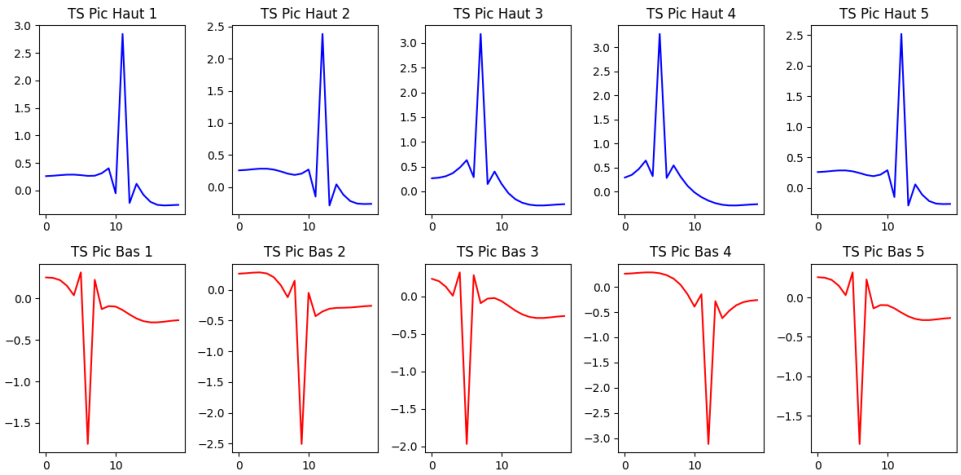

Figure 3.1: Visualisation of 5 examples for each class of time series with peaks

differentiate between simple, yet significant, patterns in time series data. Furthermore, the clustering was successful as it perfectly separated the two classes. In Figure 3.3, we visualize the clustering results in the space of the two extracted shapelets. We can observe that the time series in Cluster 1 are closer to Shapelet 1. These correspond to series 1 to 5, each having an upward peak. Similarly, series 6 to 10, belonging to Cluster 2, are the series with a downward peak and are closer to Shapelet 2.

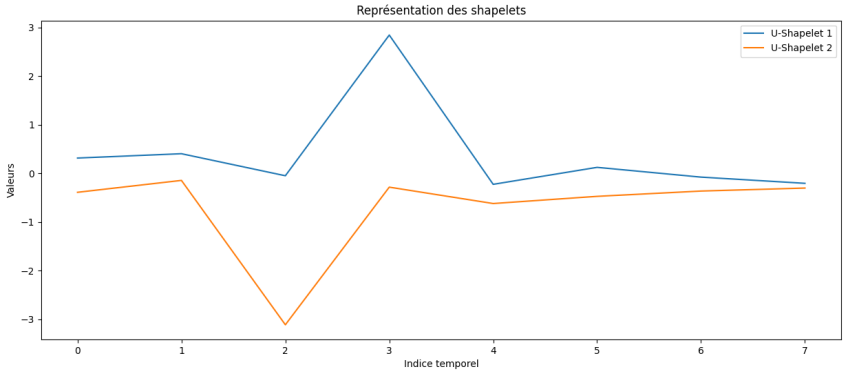

Figure 3.2: Extracted U-Shapelets (Peaks dataset)

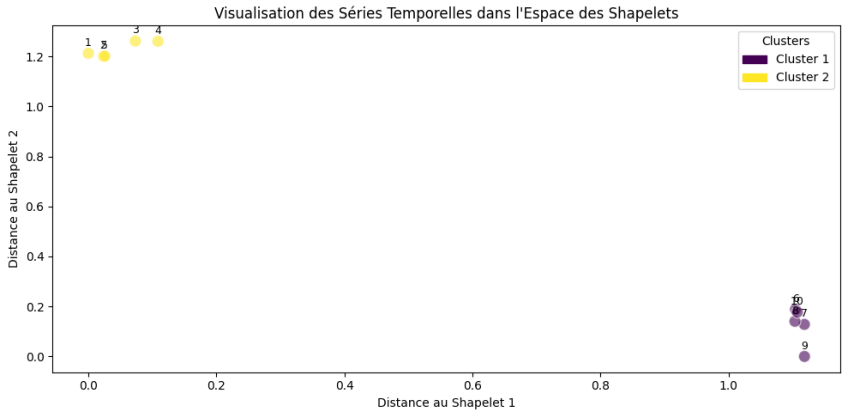

Figure 3.3: Clustering results (Peaks dataset)

3.3 Application to the Trace dataset

After successfully demonstrating the algorithm's capability with the synthetic dataset containing peaks, we extended our evaluation to a more complex dataset: the Trace dataset [3]. The Trace dataset is derived from the Transient Classification Benchmark, and simulates instrumentation failures in a nuclear power plant. The subset we use includes specific features from 4 classes, resulting in 200 instances with 50 per class. Each instance is linearly interpolated to 275 data points and is z-normalized.

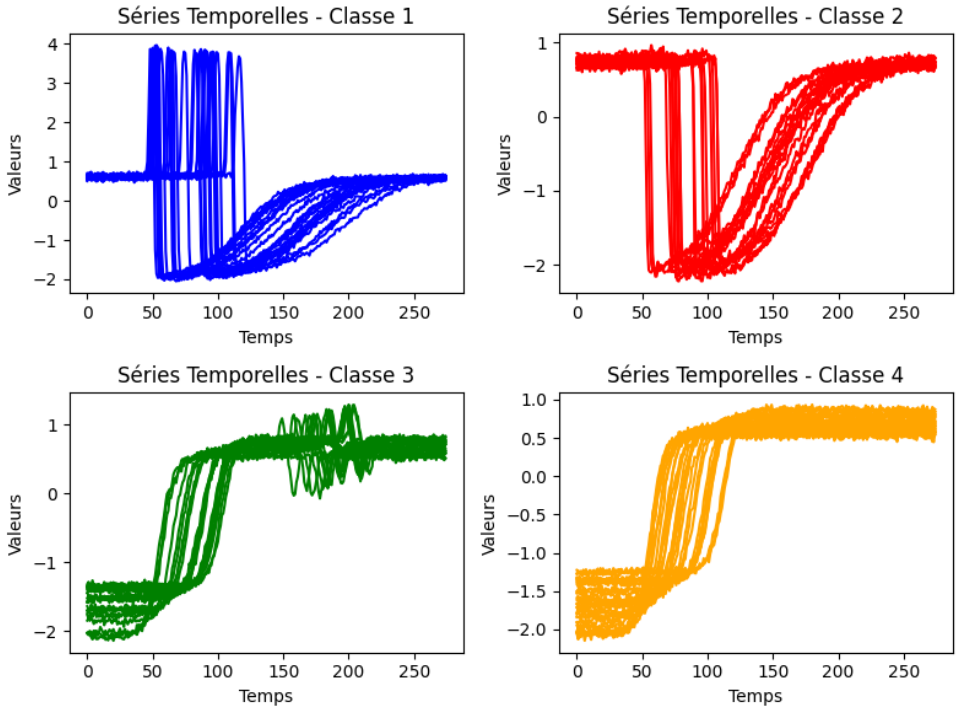

Figure 3.4: Visualisation of the 4 classes of the Trace dataset

This dataset was also used by the authors of the Unsupervised-Shapelets algorithm, and we aimed to replicate their results to validate our implementation. Figure 3.4 shows the distinct patterns for each class of time series in the Trace dataset.

In our experiments, we identified 5 shapelets that effectively captured the distinguishing features of the different classes in the dataset (see figure 3.5). Following the approach of the authors, we selected only 2 shapelets (see Figure 3.6) for the clustering, which resulted in a Rand Index of 1, indicating that the clustering was perfect.

In Figure 3.7, we visualize the clustering results in the space of the two extracted shapelets. We can observe that the time series in Cluster 2 are closer to Shapelet 1. These correspond to series from Class 2 in our case. Similarly, the series from Cluster 3, are closer to Shapelet 2 (which is an oscillation), indicating that these series correspond to Class 3 from the dataset. Moreover, the series in Cluster 4 have nearly equal and large distances to both shapelets, suggesting that they correspond to Class 4, as their characteristic shape is not represented by either of the shapelets. Finally, the series from Cluster 1 are the most challenging to identify clearly, as they are somewhat close to Cluster 2. These series correspond to Class 1, which has a shape similar to Class 2 but with an additional peak.

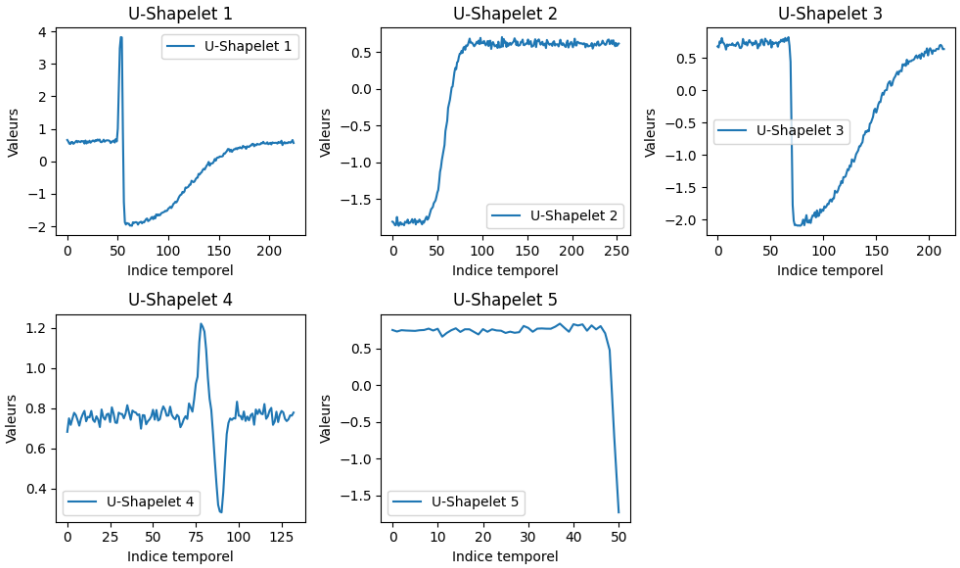

Figure 3.5: Visualisation of the 5 shapelets extracted from the Trace dataset

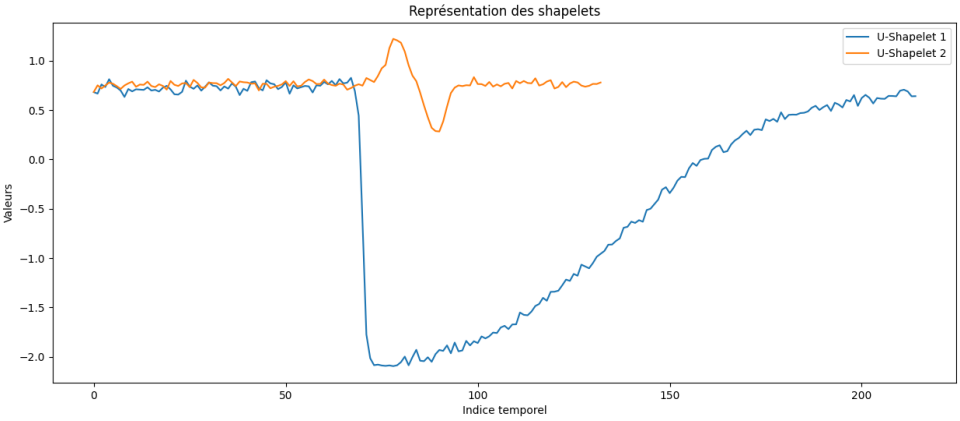

Figure 3.6: Visualisation of the 2 selected shapelets extracted from the Trace dataset

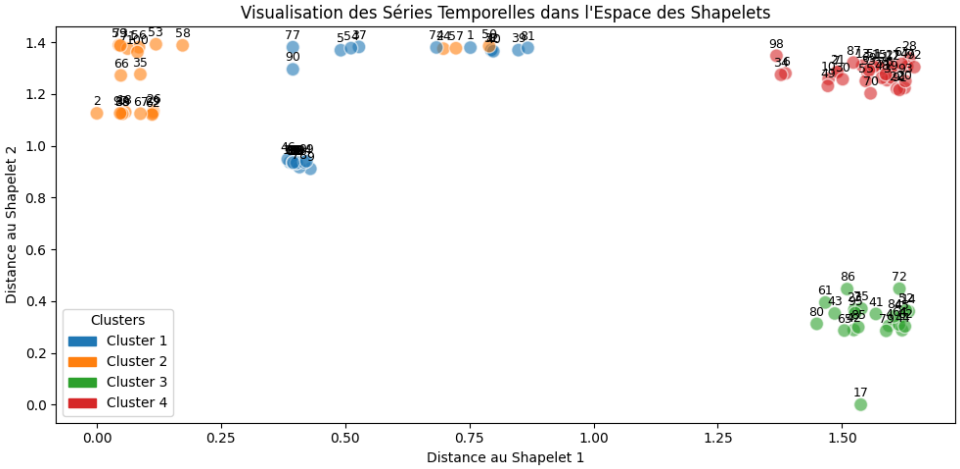

Figure 3.7: Visualisation of the time series based on their distance to the two selected shapelets in a two-dimensional space

3.4 Application to the Gun Point dataset

Continuing our exploration of the Unsupervised-Shapelets algorithm, we applied it to the Gun Point dataset [3]. The Gun Point dataset consists of motion tracking data captured from actors performing a gesture with their hand, simulating the motion of drawing a gun. It includes two classes of time series data: one representing the motion of drawing a gun, and the other representing the motion of pointing with the index finger. Each time series captures the hand movement over time.

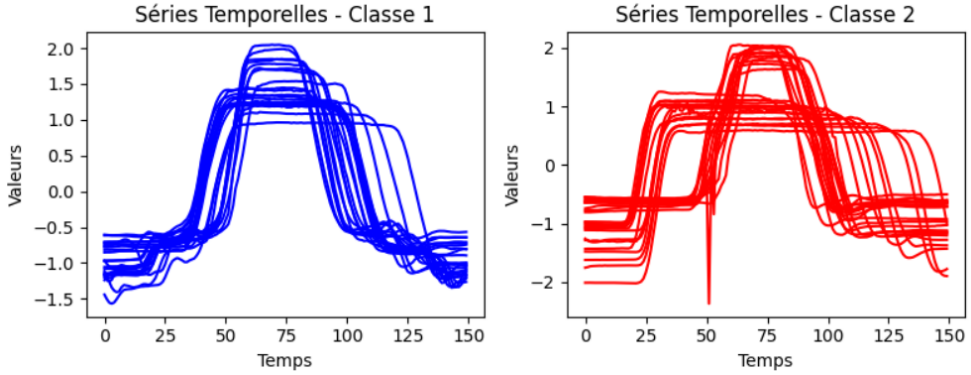

Figure 3.8: Visualisation of the 2 classes of the Gun Point dataset

In our analysis of the Gun Point dataset, we extracted six shapelets, as visualized in Figure 3.9. These shapelets capture the essential patterns and characteristics of the different gestures represented in the dataset. The clustering performance was evaluated using the Rand Index, achieving a score of 0.64.

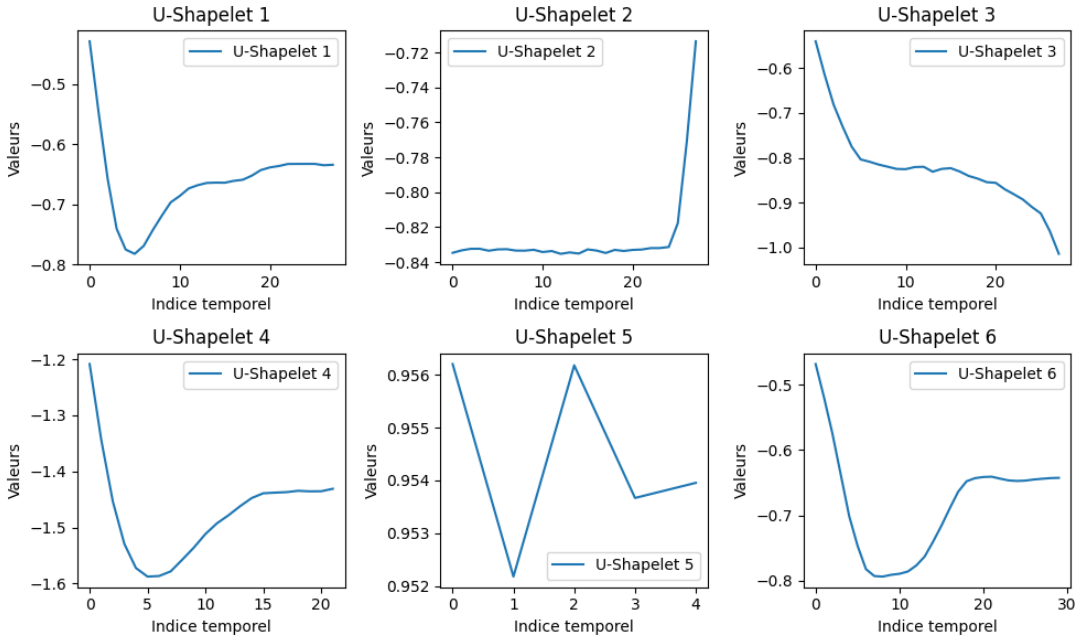

Figure 3.9: Visualisation of the 6 shapelets of the Gun Point dataset

3.5 Application to the SyntheticControl dataset

To further evaluate the capabilities of the Unsupervised-Shapelets algorithm in handling a larger number of classes, we applied it to the SyntheticControl dataset [3]. This dataset simulates various scenarios, and consists of six classes of control charts: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift (see Figure 3.10).

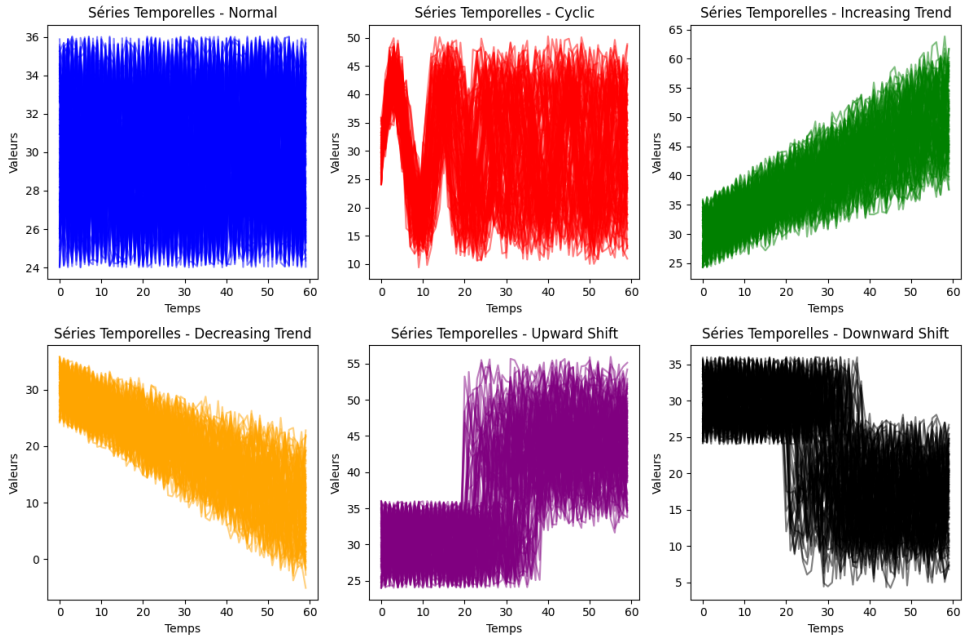

Figure 3.10: Visualisation of the 6 classes of the SyntheticControl dataset

The shapelet extraction algorithm identified 10 distinct shapelets (see Figure 3.11). These shapelets enabled the clustering of the time series data, achieving a Rand Index of 0.94, which aligns with the performance reported by the original authors.

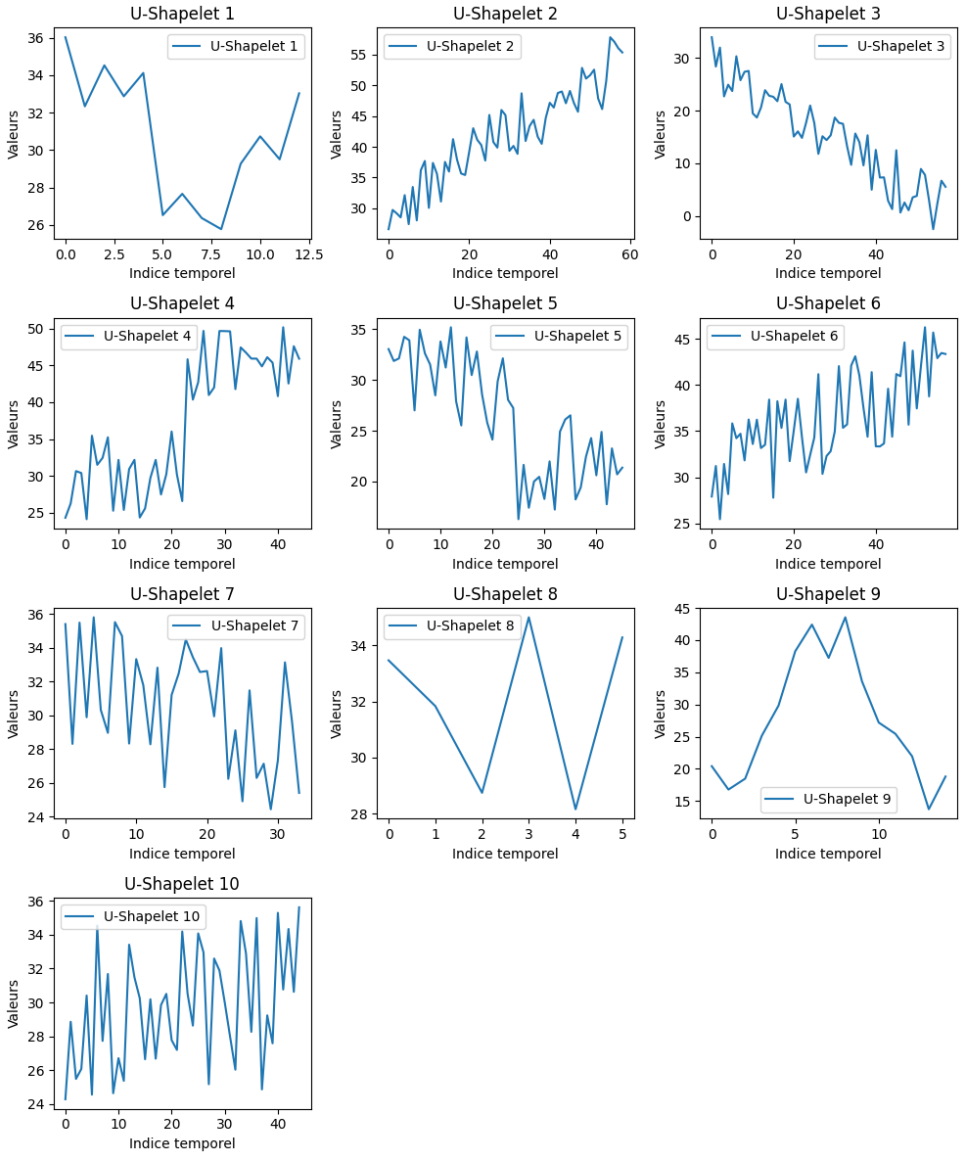

Figure 3.11: Visualisation of the 10 shapelets of the SyntheticControl dataset

3.6 Clustering precision and computational time

Our results are consistent with those presented in the original paper in most cases (See Table 3.1). The result is lower than expected in the case of Gun Point dataset (RI = 0.74 in the paper). It can be explained by having different values of hyperparameters (especially sLen).

Dataset	Rand Index	Rand Index in Paper	Shapelet Extraction Time
Trace	1.00	1.00	85 min 19 s
ECG	0.73	0.70	146 min 7 s
Gun Point	0.64	0.74	3 min 1 s
Syn-Control	0.94	0.94	49 min 28 s

Table 3.1: Rand Index and shapelet extraction times for different datasets, with comparison to Rand Index in the paper

We can see that the extraction of shapelets takes considerable time, despite the significant improvement compared to the original algorithm. This highlights the trade-off between the increased efficiency of the shapelet extraction process and the computational cost required to obtain high-quality shapelets.

Discussion

The Unsupervised-Shapelets algorithm is a method for time series clustering, designed to identify key subsequences that distinguish different patterns in the data. It addresses the limitations of previous subsequence-based clustering approaches, and reduces the dimentionality of the clustering problem, compared to whole time series clustering.

However, like many algorithms, U-Shapelets comes with significant computational costs. The most computationally expensive part of the algorithm is shapelet extraction, which involves considering subsequences and calculating distances to identify the subsequences that best separate groups in the data. The time complexity of this phase is $O(L \times m^2 \times n)$, where L is the maximum subsequence length, m is the length of each time series, and n is the number of time series (see Appendix B). Despite efforts to optimize the algorithm, such as caching to reduce the time complexity of calculating the length-normalized Euclidean distance between time series, the computational cost remains high. For example, for a relatively moderate dataset with 100 time series, each of length 500, and a maximum subsequence length of 10, the shapelet extraction phase would require approximately 250 million operations. This makes the algorithm challenging to scale for large datasets.

This theoretical complexity was confirmed in our experiments on synthetic datasets, where even with optimization efforts such as vectorization techniques, the algorithm remained computationally expensive.

Another limitation of the U-Shapelets algorithm arises in certain cases where time series differences are due to phase shifts or magnitude variations rather than shape differences. For example, in datasets containing sinusoidal signals with differing phase shifts or varying amplitudes, U-Shapelets may struggle to distinguish between classes based on these factors. Since the algorithm focuses on subsequences with distinct shapes, it may fail to capture the differences in phase or magnitude, which could be crucial for accurate clustering. In these situations, U-Shapelets might not be the most suitable approach.

The performance of this method is also influenced by the choice of parameters. Hyperparameters like subsequence length and the threshold for selecting discriminative shapelets must be carefully tuned for good results. Poor tuning can result in selecting too many shapelets, including noise, or selecting too few, leading to inefficient clustering.

Further work should focus on addressing the algorithm's computational limitations. Despite these challenges, U-Shapelets remains a promising approach for clustering time series when temporal shape patterns are the primary focus.

Conclusion

This paper focuses on the interest of Unsupervised-shapelets for clustering time series. It provides a solid theoretical base for using (U-shapelet) in this way. By focusing on significant subsequences, U-Shapelets enhance interpretability and improve robustness to variations such as shifts and changes in scale. The algorithm is thoroughly described, with theoretical advantages emphasized over some traditional methods like Euclidean distance and DTW.

Our experimental evaluation further validates the practical utility of U-Shapelets across various datasets, including synthetic datasets with peaks, the Trace dataset, the Gun Point dataset, and the SyntheticControl dataset. These experiments demonstrate the algorithm's effectiveness in distinguishing between different time series patterns and successfully clustering time series.

However, the study also identifies key challenges, particularly the high computational cost associated with shapelet extraction. This limitation, confirmed through our experiments, underscores the need for further optimization to enhance scalability.

In conclusion, while the Unsupervised-Shapelets algorithm presents computational challenges, its ability to identify meaningful patterns makes it a valuable tool for time series clustering.

Appendix A

Additional algorithms

Algorithm 5 k-means Clustering Algorithm [8]

Require: K, number of clusters; D, a data set of N points
Ensure: A set of K clusters
1: Initialization of the centers of the clusters
2: **repeat**
3: **for** each point p in D **do**
4: Find the nearest center and assign p to the corresponding cluster
5: **end for**
6: update clusters by calculating new centers using the mean of their members
7: **until** stopping criteria met (typically convergence or max iterations)
8: **return** clustering result

Appendix B

The time complexity of the U-Shapelets algorithm

The overall time complexity of the Unsupervised-Shapelets algorithm can be broken down into its two main parts:

1. Shapelet Extraction:
For each time series, we generate subsequences of varying lengths and compute the gap for each subsequence. For each subsequence, the compute distance operation runs in $O(n \times m)$, where n is the number of time series and m is the length of each series. The total number of subsequences generated is proportional to $O(L \times m)$, where L is the maximum length of a subsequence. The overall complexity for shapelet extraction is $O(L \times m^2 \times n)$, assuming we check all subsequences across all time series.

2. Clustering Using K-Means:
The k-means clustering algorithm is applied to the distance matrix, which has a complexity of $O(k \times n \times t)$, where k is the number of clusters, n is the number of time series, and t is the number of iterations. The total complexity of clustering is $O(k \times n \times t)$.

Thus, the overall complexity of the algorithm is:

$$O(L \times m^2 \times n + k \times n \times t)$$

where n is the number of time series, m is the time series length, L is the maximum shapelet length, k is the number of clusters, and t is the number of iterations in k-means clustering.

Example: We will consider that $n = 100$ (time series), $m = 500$ (length per time series), $L = 10$ (max subsequence length), $k = 5$ (clusters) and $t = 50$ (iterations)

For the first step, the complexity is: $O(L \times m^2 \times n) = O(10 \times 500^2 \times 100) = O(250,000,000)$
For the second step, the complexity is: $O(k \times n \times t) = O(5 \times 100 \times 50) = O(25,000)$
The overall time complexity is: $O(250,000,000 + 25,000) = O(250,025,000)$

In this example, the total complexity is about **250 million operations**, with the shapelet extraction step being the most computationally expensive part.

Bibliography

[1] Monica Arul and Ahsan Kareem. *Applications of shapelet transform to time series classification of earthquake, wind and wave data*. Dec. 2020. DOI: 10.1016/j.engstruct.2020.111564.

[2] Donald J. Berndt and James Clifford. "Using dynamic time warping to find patterns in time series". In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. AAAIWS'94. Seattle, WA: AAAI Press, 1994, pp. 359–370.

[3] Hoang Anh Dau et al. *The UCR Time Series Classification Archive*. 2019. URL: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[4] B. Hartmann, I. Schwab, and N. Link. "Prototype Optimization for Temporarily and Spatially Distorted Time Series". In: *AAAI Spring Symposium Series (SSS 2010)*. 2010, pp. 15–20.

[5] Eamonn Keogh and Shruti Kasetty. "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration". In: *Data Mining and Knowledge Discovery* 7.4 (Oct. 2003), pp. 349–371. DOI: 10.1023/A:1024988512476. URL: https://doi.org/10.1023/A:1024988512476.

[6] Eamonn Keogh and Jessica Lin. "Clustering of time-series subsequences is meaningless: Implications for previous and future research". In: *Knowledge and Information Systems* 8 (Aug. 2005), pp. 154–177. DOI: 10.1007/s10115-004-0172-7.

[7] John Paparrizos, Fan Yang, and Haojun Li. *Bridging the Gap: A Decade Review of Time-Series Clustering Methods*. Dec. 2024. DOI: 10.48550/arXiv.2412.20582.

[8] Claude Sammut and Geoffrey I. Webb, eds. *Encyclopedia of Machine Learning*. Berlin: Springer, 2011. ISBN: 978-0-387-30164-8.

[9] Lexiang Ye and Eamonn Keogh. "Time series shapelets: a new primitive for data mining". In: June 2009, pp. 947–956. DOI: 10.1145/1557019.1557122.

[10] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. "Clustering Time Series Using Unsupervised-Shapelets". In: Dec. 2012, pp. 785–794. ISBN: 978-1-4673-4649-8. DOI: 10.1109/ICDM.2012.26.

[11] Yichi Zhang et al. "Dynamic Time Warping for Lead-Lag Relationship Detection in Lagged Multi-Factor Models". In: *Proceedings of the Fourth ACM International Conference on AI in Finance*. ICAIF '23. Brooklyn, NY, USA: Association for Computing Machinery, 2023, pp. 454–462. ISBN: 9798400702402. DOI: 10.1145/3604237.3626904. URL: https://doi.org/10.1145/3604237.3626904.