UNIVERSITY OF TURIN

DEPARTMENT OF MATHEMATICS GIUSEPPE PEANO

M.Sc. in Stochastics and Data Science

Final Dissertation

Anomaly Detection for Time Series using Shapelets

Thesis Supervisor:
Prof.ssa Elena Cordero

Candidate:
Ludovico Bartoli

Research Supervisors:
Dott. Antonio Mastropietro
Dott.ssa Rosalia Tatano

ACADEMIC YEAR 2021/2022

Abstract

In time series classification literature, *shapelets* are subsequences able to discriminate the different classes. It has been shown that classifiers achieve high accuracy when taking in input the distances from the time series and a properly chosen set of shapelets. Additionally, these subsequences can be easily compared visually to input signals: this characteristic makes such algorithms naturally interpretable. More recent methods propose to parametrize the shapelets instead of searching them through all the possible subsequences, which is computationally expensive. The optimal shapelets are learned through the minimisation of a loss function; this approach typically sacrifices their interpretability in favour of a higher accuracy.
This thesis has been carried on in collaboration with *Addfor Industriale S.r.l.* towards the goal of developing shapelets-based algorithms for unsupervised anomaly detection in univariate and multivariate time series datasets. We propose and analyse three algorithms and compare their results on five benchmark datasets, focusing on the difference between searching and learning approaches. The experiments confirm the main advantages of shapelets methods, which work well when recurring short patterns distinguish the shape of normal and anomalous time series. In particular, the proposed shapelets-learning algorithm manages to learn different discriminating shapelets, while maintaining their interpretable characteristic.
Keywords: Interpretable Machine Learning, Time Series Analysis, Anomaly Detection, Support Vector Machines.

Acknowledgements

To my mum and dad

Contents

List of Figures

List of Algorithms

List of Symbols

The next list describes several symbols that will be later used within the body of the document

α	Ratio between the number of normal time series and anomalies in a dataset
ν	Proportion of anomalies in a dataset
AUC	Area under ROC curve
w^{-1}	Reversed-time signal w
$y * w$	Discrete convolution between the signals y and w.
$\langle a, b \rangle$	Dot product of vectors a and b
$\sigma_{\mathbf{S}}$	Shapelet transform w.r.t. **S** set of shapelets
$\{t_i\}_{i=1}^{Q}$	Time Series elements
$I[f]$	Expected risk of the function f
$I_D[f]$	Empirical risk of the function f w.r.t. the dataset D
$mean$	mean of a discrete set of real numbers
std	standard deviation of a discrete set of real numbers
OCSVM	One Class Support Vector Machine
ROC	Receiving Operating Characteristic Curve
SVDD	Support Vector Data Description

Chapter 1

Introduction

1.1 Background

An outlier or anomaly is a data point that is significantly different from the remaining data; in [Haw80] Hawkins defined an outlier as follows:

> *"An outlier or anomaly is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."*

The recognition of such unusual characteristics may provide useful understanding of the data generation process.

Many real-world data are in longitudinal form, depending on time, i.e. *Time series*; they differ from tabular data, since a temporal dependency is present where each value in the time series correlates with its previous values. Detecting anomalies in time series is important in various domains ana applications; to name a few:

- *Sensor events*: nowadays, consequently to the rise of the Internet-of-Things (IoT), sensors are largely used to monitor environmental and location parameters.

- *Medical diagnosis*: anomalies in electrocardiograms (ECG), brain or physical activity signals are of fundamental importance in diagnosis.

- *Earth science*: anomalies in spatio-temporal data about weather patterns, climate changes, or seismic data can provide crucial comprehension about human activities or environmental trends.

Machine learning (ML) is "a set of methods that computers use to make and improve predictions or behaviours based on data" ([Mol20]): for example, to predict if a certain time series is anomalous, the computer would learn patterns from the ones that are generated by a normal process.

Non-mathematically, *interpretability* can be defined ([Mol20]) as "the degree to which a human can understand the cause of a decision". The interpretability of a ML outlier detection model is extremely important from the point of view of an analyst: it is highly desirable to determine why a particular data point should be considered an outlier.

Typically, models that work with the original attributes and use fewer transforms on the data (e.g., principal component analysis [Jol05]) have higher interpretability. The trade-off is that data transformations usually intensify the contrast between outliers and normal data points, achieving better perfomance at the expense of interpretability.

The task of anomaly detection may become arduous when dealing with time series, since a single series is composed by many observations, which are temporally dependent to each other.

In time series classification domain, *shapelets* ([YK09]) are subsequences that are peculiar of a certain class; it has been shown that classifiers are able to achieve state-of-the-art results by taking the distances from the input time series to a set of shapelets, which need to be chosen properly. Since only the best matching subsequence is considered when calculating distances, the time series segments not relevant for classification are deliberately ignored. Such algorithms possess two fundamental characteristics: shapelets represent a method for *feature extraction* in time series data; moreover they can be easily plotted and compared to any input signal, allowing interpretability by the user. These key qualities make them appealing in an anomaly detection scenario.

1.2 Thesis outline

This thesis has been carried on in collaboration with Addfor Industriale S.r.l. towards the goal of developing unsupervised shapelets-based anomaly detection algorithms. The first phase consisted in reviewing the current state-of-the-art literature; after the subsequent implementation of the most suitable algorithms, we compared them on various benchmark time series datasets, both univariate and multivariate.

The thesis is organised as follows: in Chapter 2 we will give an introduction of statistical learning theory as formulated by Vapnik in [Vap99] with focus on Support Vector Machines (SVM) [SSB$^+$02]; an in-depth theoretical study about Kernel Methods in Machine Learning is given, for the sake of completeness about the review of SVM algorithm. In Chapter 3 we will discuss the anomaly detection data mining problem and show two algorithms inspired by SVM for estimating the support of the normal class distribution. In Chapter 4 shapelets are introduced in detail, starting from their first definition, then explaining their extension to the unsupervised case and anomaly detection problem. The shapelets learning problem is also presented formally and a link is established between shapelets and filter masks of convolutional neural networks. In Chapter 5 three algorithms for unsupervised anomaly detection using shapelets are presented; they will be tested and compared through various experiments, illustrated in Chapter 6.

Chapter 2

Statistical Learning

2.1 Statistical Learning Theory

The basic problem of supervised learning is to estimate a function **f** of interest, defined on a set $\mathscr{X}$ called the space of the data and taking values in $\mathscr{Y}$, the target variable space, given a (training) set of N input/output pairs $D_N = (x_1, y_1), ..., (x_N, y_N) \in (\mathscr{X} \times \mathscr{Y})$. We assume that $\mathscr{X} \times \mathscr{Y}$ is a probability space with distribution P and that the training set is sampled identically and independently. Usually $\mathscr{X} = \mathbb{R}^p$, $p > 0$ and $\mathscr{Y} = \mathbb{R}$ in case of regression problem, $\mathscr{Y} = \{1, ..., c\}$ in case of a classification problem with $c \geq 2$ classes. We denote by T the *target space*, that is the largest space of functions where **f** belongs.

Definition 2.1.1. *(Loss Function) A map $L : \mathscr{Y} \times \mathscr{Y} \mapsto [0, \infty[$ with the property $L(y, y) = 0, \ \forall y \in \mathscr{Y}$ is called a loss function.*

Given a loss function L, the function of interest is the one minimizing the **expected risk** I (true error):

$$\mathbf{f} = \arg\min_{f \in T} I[f], \quad I[f] = \int_{\mathscr{X} \times \mathscr{Y}} L(y, f(x)) dP(x, y), \tag{2.1}$$

which can be interpreted as an idealization of the notion of test error, since it is the expected loss value over all the possible input/output pairs. In practice, the minimization of the expected risk is unfeasible for two reasons:

firstly the measure P is known only through the training set D_N. The second one is that, in practice, the search of a solution needs to be restricted to some smaller class of functions $H \subset T$, called **hypothesis space**.

Definition 2.1.2. *A learning algorithm A is a map from the data space to H:*

$$A(D_N) = f_N \in \boldsymbol{H}. \tag{2.2}$$

In order to have an algorithm that can approximate the target function at the best with the data available, the natural approach is to consider the **empirical risk** minimization (ERM) induction principle, which aims to approximate **f** through f_{D_N}, where

$$f_{D_N} = \arg\min_{f \in H} I_{D_N}[f], \quad I_{D_N}[f] = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) \tag{2.3}$$

and D_N is a set of N training data points. It is the expected loss under the empirical measure of the data. If it exists, a good estimator f_{D_N} of the target function should satisfy some requirements:

- **Generalization**: $\lim_{N\to\infty} |I_{D_N}[f_{D_N}] - I[f_{D_N}]| = 0$ in P-probability
- **Consistency**: $\forall \varepsilon > 0$, $\lim_{N\to\infty} P(|I[f_{D_N}] - \inf_{f\in H} I[f]| > \varepsilon) = 0$
- **Stability**

In other words, generalization means that the training error for the solution must converge to the expected error. Consistency amounts to saying that as the number of examples N tends to infinity, we want f_{D_N} to lead to a test error which converges to the lowest achievable value. We will not formalize the notion of stability; intuitively it means that f_{D_N} should depend continuously on the training set D_N. In particular, changing one of the training points should affect less and less the solution as N goes to infinity.

Definition 2.1.3. *([Kab08]) A problem is well-posed if its solution:*

- *exists*
- *is unique*

- *depends continuously on the data (i.e. it is stable).*

A problem is ill-posed if it is not well-posed.

Example 2.1.1. *(Ill-posed ERM problem) Assume that we want to solve a regression problem with a quadratic loss function* $L(y, f(x)) = (y - f(x))^2$ *and a linear class of functions as hypothesis space:*

$$H = \left\{ f,\ f(x) = \sum_{i=1}^{m} \alpha_i f_i(x)\ with\ \alpha_i \in \mathbb{R} \right\} \tag{2.4}$$

where f_i *are mapping from* $\mathscr{X}$ *to* $\mathbb{R}$. *We want to find the minimizer of the empirical risk:*

$$\arg\min_{f \in H} I_D[f] = \arg\min_{\alpha \in \mathbb{R}^m} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{i=j}^{m} \alpha_j f_j(x_i) \right)^2 . \tag{2.5}$$

Define $F_{ij} = f_j(x_i) \in \mathbb{R}^{N \times m}$, *and compute the derivative w.r.t.* α. *It can be proved ([SSB⁺02]) that the minimum is achieved if:*

$$F^T y = F^T F \alpha. \tag{2.6}$$

- *If* $F^T F$ *is invertible, it could have a bad condition number (quotient between largest a lowest eigenvalues) leading to numerical difficulties.*

- *If* $m > N$ *(i.e. there are more basis functions than training data), there will exist a whole space of solutions of dimension at least* $m - N$.

- *If* H *is too rich, it could happen that the discrepancy between* $I_{D_N}[f]$ *and* $I[f]$ *is too large.*

Since Tikhonov, it is well-known that a generally ill-posed problem such as ERM can be guaranteed to be well-posed and therefore stable by an appropriate choice of H. It can be shown (see [EPP00]) that the two desirable conditions for a supervised learning algorithm, generalization and stability, are equivalent and they correspond to the same constraints on H. **Regularization** in general means ”restricting H” and is the classical way to restore well posedness: the idea is that the regularizer should help to find stable solutions. For arbitrary hypothesis space H the ERM problem is not well-posed; there are two classical ERM regularization ways, which can be proved are equivalent (see [ORA16]):

- **Ivanhov** regularization: minimize

$$I_D[f] = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) \tag{2.7}$$

  while satisfying $R(f) \leq r$, where R is a positive function on H called ”regularizer” and $r > 0$;

- **Tikhonhov** regularization: minimize over all H the regularized expected risk

$$\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \gamma R(f), \qquad \gamma > 0. \tag{2.8}$$

A classical example ([SSB$^+$02]) are Reproducing Kernel Hilbert Spaces (RKHS) as hypothesis spaces and their norm as regularization; in this case Ivanov regularization is the restriction of H to a ball of radius R and Tikhonhov's can be seen as the Lagrangian formulation of the Ivanov constrained optimization problem. It can be shown that on RKHS hypothesis space, Tikhonhov regularization ensures generalization and therefore stability. See [SSB$^+$02] for a more detailed review of statistical learning theory.

2.2 Kernel Methods in Machine Learning

2.2.1 Introduction

Given some empirical data $\{(x_i, y_i)\}_{i=1}^{N}$ we want a predictor of the target variable y being able to generalize to new unseen data. To do so, a measure of similarity in the data domain X is needed in order to establish if a new data point x is similar to the training ones. Citing [SSB$^+$02] "Characterizing the similarity of the output is rather easy, while the choice of the similarity measure for the inputs is a deep question that lies at the core of the field of machine learning". The 'kernel trick' arises from the necessity to construct algorithms in feature spaces different from the original X: we therefore require a function $k : X \times X \to \mathbb{R},\ \ (x, x') \mapsto k(x, x')$ satisfying $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ where $\langle \cdot, \cdot \rangle$ is a dot product in a Hilbert space. The function Φ is called **feature map** and maps the set X into a Hilbert space called feature space. We want to be able to define a class of kernels that satisfy such a requirement. A Kernel can be seen as inner product between feature maps of points; hence, kernel is a measure of similarity between points and this similarity is computed in the feature space rather than input space.

2.2.2 Reproducing Kernel Hilbert Spaces

Definition 2.2.1. *(Gram Matrix) Let X be a nonempty set, k a kernel on X and inputs $x_1, .., x_n$. The $n \times n$ matrix $K := (k(x_i, x_j))$ is called Gram matrix of k w.r.t. $x_1, .., x_n$.*

Definition 2.2.2. *(Positive Definite Kernel) A kernel is positive definite if its Gram Matrix is symmetric positive definite for every $x_1, .., x_n \in X$ and $n \in \mathbb{N}$, i.e. it satisfies $c^T K c \geq 0,\ \ \forall c \in \mathbb{R}^n$.*

Observation 2.2.1. *Kernels defined through feature maps are positive definite; let $c \in \mathbb{R}^n$:*

$$\sum_{i,j} c_i c_j \langle \Phi(x_i), \Phi(x_j) \rangle = \langle \sum_i c_i \Phi(x_i), \sum_j c_j \Phi(x_j) \rangle = \left\| \sum_i c_i \Phi(x_i) \right\|^2 \geq 0, \tag{2.9}$$

where $\|\cdot\|$ is the norm induced by the scalar product in the feature space. In particular if X is a Hilbert space we may choose Φ to be the identity.

Kernels can thus be regarded as generalized dot products. While they are not generally bilinear, they share important properties with dot products, such as the Cauchy-Schwarz (C-S) inequality, which follows directly from Theorem 2.2.2: if k is a positive definite kernel, and $x_1, x_2 \in X$, then

$$k(x_1, x_2)^2 \leq k(x_1, x_1)k(x_2, x_2). \tag{2.10}$$

We now define a map from X into the space of functions mapping X into $\mathbb{R}$, denoted as $\mathbb{R}^X$, via $\Phi : X \to \mathbb{R}^X$, $x \mapsto k(\cdot, x)$. The goal is to construct a dot product space containing the images of the inputs under Φ; to this end first extend it into a vector space H_0 by forming linear combinations:

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i) \tag{2.11}$$

where $n \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$ and $x_i \in X$ are arbitrary. The dot product between f and $g(\cdot) = \sum_{j=1}^{n'} \beta_j k(\cdot, x'_j)$ is defined as

$$\langle f, g \rangle := \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(x_i, x'_j). \tag{2.12}$$

Note that $\langle f, g \rangle = \sum_{i=1}^{n} \alpha_i g(x_i) = \sum_{j=1}^{n'} \beta_j f(x'_j)$ so the product does not depend on the particular coefficients of f and g. This shows that is bilinear. Symmetricity and positive definiteness follow directly from Equation (2.12) and that k is symmetric positive definite.

Proposition 2.2.1. *(Reproducing property)*

$$\langle k(\cdot, x), f \rangle = f(x) \text{ and in particular } \langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x') \tag{2.13}$$

Proof. Using Equations (2.12) and (2.11):

$$\langle k(\cdot, x), f \rangle = \sum_{i=1}^{n} \alpha_i k(x, x_i) = f(x). \tag{2.14}$$

□

By C-S inequality applied to $\langle \cdot, \cdot \rangle$ and the reproducing property, we have

$$|f(x)|^2 = |\langle k(\cdot, x), f \rangle|^2 \leq k(x, x)\langle f, f \rangle \tag{2.15}$$

therefore $\langle f, f\rangle = 0 \Rightarrow f = 0$ and we can conclude that $\langle \cdot, \cdot\rangle$ is a dot product. One can complete the space H_0 in the norm corresponding to the dot product and gets a Hilbert space H called **Reproducing Kernel Hilbert Space** (RKHS) associated to the kernel k.

Example 2.2.1. *(Some positive definite kernels)*

- *Linear kernel:* $k(x, x') = x^T x'$.
- *Polynomial kernel:* $k(x, x') = (x^T x')^d,\ \ d \in \mathbb{N}$.
- *Inhomogeneous polynomial:* $k(x, x') = (c + x^T x')^d,\ \ d \in \mathbb{N},\ \ c \in \mathbb{R}$.
- *Gaussian kernel:* $k(x, x') = e^{-\frac{\|x-x'\|^2}{\sigma^2}},\ \ \sigma > 0$.

Since sum and products of kernels are still kernels, all of them can be proved to be kernels derived from the linear one (See [SC08]).

Citing B. Scholkopf, ”Choosing different kernels, one can show that the norm in the corresponding RKHS encodes different notions of smoothness”, cf. [SSB$^+$02].

A RKHS is a Hilbert space of point-wise defined functions and can be defined equivalently to be a Hilbert space of functions where the **point evaluations** $F_x(f) := f(x)$ are continuous linear functionals.

Definition 2.2.3. *A Hilbert space of real-valued functions on a set X is a RKHS is there exists $M > 0$ such as:*

$$|F_x(f)| \leq M \|f\|_H,\ \ \forall f \in H. \tag{2.16}$$

Theorem 2.2.1. *(Riesz Representation Theorem) If T is a bounded linear functional on a Hilbert space H, then there exists a unique $g_T \in H$ such that, for every $f \in H$, we have $T(f) = \langle f, g\rangle$.*

Moreover, $\|T\| = \|g_T\|_H$, where $\|T\|$ denotes the operator norm of T, i.e. $\|T\| = \inf\left\{M > 0, |Tf| \leq M \|f\|_H,\ \ \forall f \in H\right\}$.

The reproducing kernel can be constructed using Riesz's theorem: for $x \in X$, if F_x is bounded, then there exists $k_x \in H$ such that the reproducing property holds

$$F_x(f) = f(x) = \langle f, k_x \rangle_H. \tag{2.17}$$

Define $k(x, y) = k_x(y) = \langle k_x, k_y \rangle$; it is clearly symmetric and positive definite because

$$\sum_{i,j=1}^{n} c_i c_j k(t_i, t_j) = \sum_{i,j=1}^{n} c_i c_j \langle k_{t_i}, k_{t_j} \rangle_H = \left\| \sum_{j=1}^{n} c_j k_{t_j} \right\| \geq 0. \tag{2.18}$$

We just proved the following Theorem:

Theorem 2.2.2. *(Moore-Aronszaj, [SSB$^+$02]) Suppose k is a symmetric, positive definite kernel on a set X. Then there is a unique Hilbert space of functions on X for which k is a reproducing kernel.*

Putting together all the results, we can conclude that for every positive definite kernel on $X \times X$, there exists a unique RKHS and vice versa.

Example 2.2.2. *(Linear Case) Consider the space of 1-dimensional lines in $\mathbb{R}^p$, $p > 0$*

$$H = \{f(x) = w^T x, \;\; x, w \in \mathbb{R}^p, \;\; w \;\; constant\}, \tag{2.19}$$

and $k(x, x_i) = x^T x_i$; the RKHS norm is simply

$$\|f\|_H^2 = \langle f, f \rangle_H = \langle k_w, k_w \rangle = k(w, w) = \|w\|^2, \tag{2.20}$$

so the measure of complexity of the set of functions H given by the norm in the RKHS is the slope of the line.

Theorem 2.2.3. *(Representer Theorem [PR09]) Let $D = (x_1, y_1), ..., (x_N, y_N)$ be a dataset of N points, with $x_i \in X$ data space. Let H be a RKHS defined on X with positive definite kernel k, L a non negative loss function. Then* *any minimizer f^* of the Tikhonhov regularization problem (regularized empirical risk):*

$$\min_{f \in H} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_H^2 \tag{2.21}$$

can be represented by

$$f^*(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i), \qquad \alpha_i \in \mathbb{R}, \tag{2.22}$$

for some N-tuple $\alpha = (\alpha_1, \cdots, \alpha_N) \in \mathbb{R}^N$, provided that $\lambda > 0$. Hence, minimizing over the (possibly infinite dimensional) Hilbert space, boils down to minimizing over $\mathbb{R}^N$ and the solution can be expressed as a kernel expansion in terms of training set data D.

It can be shown that if the loss is strictly convex w.r.t. the second term and coercitive (grows rapidly at extrema) then it will exist a unique minimizer.

Proof. Define the linear subspace of H spanned by representers of the training set:

$$W = \left\{ f \in H, \;\; f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i), \;\; (\alpha_1, \cdots, \alpha_N) \in \mathbb{R}^N \right\}; \tag{2.23}$$

since W has finite dimension, we have

$$H = W + W^{\perp} \tag{2.24}$$

and

$$W^{\perp} = \{ f \in H, \;\; \langle f, \sum_{i=1}^{N} \alpha_i k(\cdot, x_i) \rangle_H = 0 \}. \tag{2.25}$$

For each $f \in H$, $f = \bar{f} + \bar{f}^{\perp}$, $\bar{f} \in W$, $\bar{f}^{\perp} \in W^{\perp}$, and the empirical risk becomes:

$$I_D[f] = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \bar{f}(x_i) + \bar{f}^{\perp}(x_i)) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \bar{f}(x_i)) = I_D[\bar{f}]; \tag{2.26}$$

the second equality follows by the reproducing Property (2.13) and the orthogonality, since

$$\bar{f}^{\perp}(x_i) = \langle k(\cdot, x_i), \bar{f}^{\perp}\rangle = 0. \tag{2.27}$$

Also because of orthogonality:

$$\|f\|_H = \left\|\bar{f}\right\|_H + \left\|\bar{f}^{\perp}\right\|_H. \tag{2.28}$$

Minimizing the regularized empirical risk over H gives:

$$\min_{f\in H}\{I_D[f] + \lambda \|f\|_H^2\} = \min_{f\in H}\{I_D[\bar{f}] + \lambda(\left\|\bar{f}\right\|_H^2 + \left\|\bar{f}^{\perp}\right\|_H^2)\} \tag{2.29}$$

Since

$$\lambda(\left\|\bar{f}\right\|_H^2 + \left\|\bar{f}^{\perp}\right\|_H^2) \geq \lambda \left\|\bar{f}\right\|_H^2, \tag{2.30}$$

the resulting minimizer must have $\left\|\bar{f}^{\perp}\right\|_H^2 = 0$ and belong to the subspace W. □

Corollary 2.2.4. *Using Representer theorem and reproducing propery, expression* (2.21) *becomes*

$$\min_{\alpha} \frac{1}{N}\sum_{i=1}^{N} L(y_i, \langle \alpha, K_i\rangle) + \lambda\alpha^T K\alpha \tag{2.31}$$

where K is the Gram Matrix evaluated in the training points and K_i is its i-th row. Simple numerical algorithms can be used to optimize it.

Example 2.2.3. *(Common Loss Functions) We list below some examples of common loss functions mainly employed in applications.*

- *Regularized least squares (RLS):*

$$L(y, f(x)) = (y - f(x))^2; \tag{2.32}$$

in this case Equation (2.31) *becomes*

$$\min_{\alpha}(y - K\alpha)^T(y - K\alpha) + \lambda\alpha^T K\alpha \tag{2.33}$$

and it is called "Generalized Ridge Regression" problem. It has a unique solution

$$\alpha^* = (K + \lambda I)^{-1}y. \tag{2.34}$$

- *Support vector machines for classification:*

$$L(y, f(x)) = (1 - yf(x))_+ \tag{2.35}$$

where $(k)_+ = \max(0, k)$ *and* $f(x) = \alpha_0 + \sum_{i=1}^{N} \alpha_i k(x, x_i)$*. The parameters are then chosen to minimize:*

$$\arg\min_{\alpha,\alpha_0} \frac{1}{N}\sum_{i=1}^{N}(1 - y_i f(x_i))_+ + \lambda\alpha^T K\alpha, \tag{2.36}$$

This can be viewed as a quadratic optimization problem with linear constraints, and requires a quadratic programming algorithm for its solution. The name support vector arises from the fact that typically many of the $\alpha_i^* = 0$ *due to the piecewise-zero nature of the loss function, and so f is an expansion only in a subset of* $\{k(\cdot, x_i)\}_{i=1}^{N}$.

2.3 Support Vector Machines

Suppose our training data consist of N pairs (x_i, y_i) with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, +1\}$. Suppose the two classes are linearly separable, we are looking for a separating hyperplane

$$\{x : f_{w,b}(x) = w^T x + b,\ \|w\| = 1,\ y_i f_{w,b}(x_i) > 0\}, \tag{2.37}$$

with induced classification rule $G(x) = sgn(f(x))$, such that the margin M between the classes in maximal:

$$\arg\max_{w,b} M \tag{2.38}$$

$$subject\ \ to\ \ \|w\| = 1, \quad y_i(w^T x_i + b) \geq M \quad \forall i \tag{2.39}$$

We can get rid of the constraint $\|w\| = 1$ by replacing the condition with $y_i(w^T x_i + b) \geq M\,\|w\|$. Since for any w, b satisfying the inequalities, any positive scaled multiples satisfy them too, we can set $\|w\| = 1/M$ and the problem becomes:

$$\arg\min_{w,b} \|w\| \tag{2.40}$$

$$subject\ \ to \quad y_i(w^T x_i + b) \geq 1 \quad \forall i \tag{2.41}$$

We call "**soft margin**" the case when classes overlap in the data space: it is convenient to introduce N positive "**slack variables**" $\xi = (\xi_1, \ldots, \xi_N)$ to allow for some points to be on the wrong side of the margin and still trying to maximize M. There are two natural ways of modeling the errors by modifying the constraints:

$$y_i(w^T x_i + b) \geq M - \xi_i\ \ or \tag{2.42}$$

$$y_i(w^T x_i + b) \geq M(1 - \xi_i), \tag{2.43}$$

which lead to different solutions; the first one measures overlap in actual distance from the margin, the second measures the overlap in relative distance, which changes with the width of the margin M. Only the second one leads to a convex optimization problem.

As before we can drop M and the norm constraint and the problem becomes:

$$\arg\min_{w,b} \|w\| \;\; subject\ to \tag{2.44}$$

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0 \;\; \forall i \;\; and \;\; \sum_{i=1}^{N} \xi_i \leq constant \tag{2.45}$$

Bounding the sum $\sum \xi_i$ (Ivanhov regularization) we bound the total proportional amount by which predictions fall on the wrong side of the margin, since misclassification occur when $\xi_i > 1$, we are bounding the total number of misclassifications to a costant value. An equivalent and more convenient form is:

$$\arg\min_{w,b,\xi} 1/2 \|w\|^2 + C \sum_{i=1}^{N} \xi_i \tag{2.46}$$

$$subject\ to \;\; y_i(w^T x_i + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0 \;\; , \forall i, \tag{2.47}$$

where C replaces the constant (Tykhonhov regularization). For sufficiently large C the vector w and constant b that minimize under constraints determine the hyperplane that minimizes the number of errors on the training set and separate the rest of the elements with maximal margin.

Note that, for fixed (w, b), the optimal solution necessarily satisfy

$$\xi_i = (1 - y_i(b + w^T x_i))_+, \tag{2.48}$$

where we recall $(x)_+ = \max(0, x)$. Indeed, from the initial constraints we have

$$\xi_i \geq (1 - y_i(x_i^T w + b))_+. \tag{2.49}$$

Setting ξ_i as in Equation (2.48) the constraints are satisfied and any other optimal solution would have a greater value, therefore would not be optimal for the minimization of the objective.

Consequently the soft margin SVM can be formulated as a penalization problem

$$\arg\min_{w,b} \sum_{i=1}^{N} (1 - y_i(b + w^T x_i))_+ + \frac{\lambda}{2} \|w\|^2 \tag{2.50}$$

with $\lambda = 1/C$; this is Equation (2.35) with linear kernel.

The problem in expression 2.46 is quadratic with linear inequality constraints, hence it is a convex optimization problem. We describe a quadratic programming solution using KKT multipliers, defined below, from [SSB$^+$02].

Definition 2.3.1. *A function f defined on a set X is called* convex *if, for any $x, x' \in X$, and any $\lambda \in [0, 1]$ such that $\lambda x + (1-\lambda)x' \in X$, we have*

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x'). \tag{2.51}$$

Consider the following nonlinear minimization problem:

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{2.52}$$

$$subject\ to\ c_i(x) \leq 0 \tag{2.53}$$

where we assume f (called *objective* or utility function) and the $c_i, i = 1, .., m$ (called *inequality constraints*) are all real convex functions. The goal is to find some sufficient and necessary conditions for optimality: clearly $f'(x) = 0$ is a too restrictive condition since f' could point in a direction forbidden by the constraints and we could have optimality even if $f'(x) \neq 0$. Some of the most important are the Karush-Kuhn-Tucker (KKT) saddle point conditions [SSB$^+$02], which do not require differentiability or convexity of the functions.

Theorem 2.3.1. *Assume an optimization problem in the form of expression* (2.52) *and* (2.53) *and define the Lagrangian*

$$L(x, \alpha, \lambda) = f(x) + \alpha^T c(x) \tag{2.54}$$

where $c(x) = (c_1(x), \cdots, c_m(x))$ and $\alpha_i \geq 0 \ \forall i \in 1, \cdots, m$. If a pair (x^*, α^*) exists such that $\forall x \in \mathbb{R}^n, \ \ \alpha_i \geq 0$:

$$L(x^*, \alpha) \leq L(x^*, \alpha^*) \leq L(x, \alpha^*) \tag{2.55}$$

(saddle point), then x^* is a solution of (2.52) *and* (2.53)*.*

The coefficients α in (2.54) are called **Lagrange KKT Multipliers**; in the SVM problem, they will become the coefficients of the kernel expansion of the solution.

Proof. From the first inequality, it follows

$$\sum_{i=1}^{m}(\alpha_i - \alpha^* i)c_i(x^*) \leq 0. \tag{2.56}$$

Since we are free to choose $\alpha_j \geq 0$, we can set all of them but one equal to α_j^* and the remaining one to $\alpha_i = \alpha_i^* + 1$, this shows that $c_i(x^*) \leq 0$ for all $i = 1, \cdots, m$. In the same way if we set the remaining $\alpha_i = 0$ we get $\alpha_i^* c_i(x^*) \geq 0$. The only way to satisfy this is by having:

$$\alpha_i^* c_i(x^*) = 0 \;\; \forall i. \tag{2.57}$$

Combining equation (2.57) and $c_i(x^*) \leq 0$ with the second inequality we get $f(x^*) \leq f(x)$ for all feasible x. □

Denote by $X = \{x \in \mathbb{R}^n, \;\; c_i(x) \leq 0, \;\; \forall i\}$ the *feasible region* of the problem; under little assumpions on X the KKT saddle point conditions become also necessary:

Theorem 2.3.2. *(Necessary KKT Conditions [SSB+02]) Under the hypothesis of Theorem 2.3.1 with additional conditions that f and c_i, $\forall i$, are convex defined on a convex set $X \subset \mathbb{R}^n$ and the $\{c_i\}$ satisfy other light conditions, the saddle point criterion* (2.55) *is necessary for optimality.*

Theorem 2.3.3. *(KKT for differentiable convex problems) A solution of 2.52 with convex, differentiable f, $\{c_i\}$ is given by x^* if there exists some α^* such that the following conditions are satisfied:*

- $\nabla_x L(x^*, \alpha^*) = \nabla_x f(x^*) + \sum_{i=1}^{m} \alpha_i^* \nabla_x c_i(x^*) = 0$

- $\nabla_{\alpha_i} L(x^*, \alpha^*) = c_i(x^*) \leq 0$
- $\alpha^{*T} c(x^*) = 0$

Proof. By convexity and differentiability of f and c_i, $i = 1, ..., m$

$$\begin{aligned} f(x) - f(x^*) &\geq (\nabla_x f(x^*))^T (x - x^*) \\ &= -\sum_{i=1}^{m} \alpha_i^* \big(\nabla_x c_i(x^*)\big)^T (x - x^*) \\ &\geq -\sum_{i=1}^{m} \alpha_i^* (c_i(x) - c_i(x^*)) \\ &= -\sum_{i=1}^{m} \alpha_i^* c_i(x) \geq 0 \end{aligned} \tag{2.58}$$

by the last KKT condition. □

Note that the problem of minimizing functions is transformed into one of solving a set of equations.

From the primal minimization problem (2.52) expressed in terms of x, we can find a dual maximization problem in terms of α by computing the saddle point of the Lagrangian and eliminating the variable x.

Definition 2.3.2. *The **dual** maximization problem from 2.52 is: maximize* $L(x, \alpha)$ *where*

$$(x, \alpha) \in Y := \big\{x \in \mathbb{R}^n, \alpha_i \geq 0 \ and \ \nabla_x L(x, \alpha) = 0\big\} \tag{2.59}$$

In our case, keeping the notation of (2.46) $h = 0$, $c_i = y_i(w^T x_i + b) \geq 1 - \xi_i$ and $c_{(2i)} = \xi_i \geq 0$ for $i = 1, .., N$ (therefore c consists of $2N$ constraints) with $x^* = (w^*, b^*, \xi^*) \in \mathbb{R}^{p+1+N}$. The (primal) Lagrange function is

$$L_P(w, b, \xi, \alpha, \mu) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i \xi_i \tag{2.60}$$

where we denote as α the first N multipliers and as μ the last N ones. Differentiating w.r.t. w, b, ξ and setting the derivatives to zero we get, $\forall i$:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i; \tag{2.61}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i; \tag{2.62}$$

$$\alpha_i = C - \mu_i. \tag{2.63}$$

Since $\mu_i \geq 0$ by definition, we can drop the multipliers μ_i by adding the constraint $0 \leq \alpha_i \leq C$. Substituting the results in L_P we get (see [HTFF09]) the Lagrangian (Wolfe) dual objective function, which does not depend on w and μ_i anymore:

$$L_D(b, \xi, \alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i . y_{i'} x_i^T x_{i'} \tag{2.64}$$

subject to

$$0 \leq \alpha_i < C, \ \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{2.65}$$

The last KKT condition becomes:

$$\mu_i \xi_i = 0, \tag{2.66}$$

$$\alpha_i [y_i (w^T x_i + b) - (1 - \xi_i)] = 0. \tag{2.67}$$

Maximizing the dual is a simpler convex quadratic programming problem than the primal, and can be solved with standard techniques (for example see python's library *cvxopt* documentation [ADV20]). Keeping in mind that the optimal $\hat{\xi}_i$ is $(1 - y_i(w^T x_i + b))_+$, the last condition tells us that the optimal $\hat{\alpha}_i$ are not 0 in two cases:

- $\hat{\xi}_i > 0$
- $\hat{\xi}_i = 0$ and $1 = y_i(w^T x_i + b)$

The training observations with $\alpha_i > 0$ are called **support vectors**. Moreover, since $\mu_i \xi_i = 0$ and $\alpha_i = C - \mu_i$, we can infer that if $\xi_i > 0$ then $\alpha_i = C$ so the support vectors in the boundary of the margin will have their corresponding multiplier such as $0 < \alpha_i < C$ while the support vectors which violate the margin constraint will have $\alpha_i = C$.

Since the Dual depends only on dot products $x^T x$ in the feature space, it is easy to solve the problem in the RKHS defined by a positive definite kernel. Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space. The idea is to fit the SVM classifier using the feature map $h(x_i) = (h_1(x_i), ..., h_M(x_i))$ (in finite dimensional case) and produce the non linear function $f(x) = b + w^T h(x)$ which can be written (after solving analogous dual problem) as $f(x) = \sum_{i=1}^{N} \alpha_i y_i k(x, x_i) + b$, requiring only the knowledge of the kernel function.

Chapter 3

Anomaly detection

3.1 Introduction

In most applications, the data is created by one or more generating processes; when the generating process behaves differently, it results in the creation of outliers. Traditional anomaly detection techniques ignore the sequential aspect of the data; despite this, it can happen that anomalies in sequences can be detected only by analyzing data instances together as a sequence, and hence cannot be detected by the traditional techniques. For a detailed review of anomaly detection methods, see [Agg13].

We will address the problem of **unsupervised anomaly detection** for time series data. In Chapter 2 we presented the problem of supervised learning or ”learning with a teacher”; in the case of unsupervised learning one has a set of N observations $(x_1, x_2, \cdots, x_N)$ of a random p-vector X having joint density $P(X)$. The goal is to directly infer the properties of this probability density without the help of a supervisor providing correct answers or degree-of-error for each observation.

Following Beggel’s [BKS$^+$19] and Yamaguchi [YN18] approach, in this work we will make the further hypothesis that the data set is mostly made up of normal instances, which share common characteristics, and only a small percentage of instances will be anomalous. Note that anomalies are rare by definition ([Agg13]) and there may be many diverse types; therefore it cannot be assumed that the space of all possible anomalies can be exhaustively sampled.

3.2 Anomaly Detection in Time Series Data

Definition 3.2.1. *A **time series** T with length Q and number of channels C is an ordered set of values in $\mathbb{R}^C$: $T = t_1, t_2, \ldots, t_Q$, where $t_i \in \mathbb{R}^C$, $\forall i$. A dataset of time series is a collection of N time series: $D = \{T_1, \ldots, T_N\}$.*

Detecting anomalies in time series is a particularly challenging topic, because abnormal behavior may manifest itself in any short sub-sequence, as well as in longer trends. Two major types of anomaly detection problems for time series can be distinguished (Beggel, [BKS$^+$19]):

- identify entire time series as anomalous in relation to a set of other time series;
- identify short parts or even single points within a time series as anomalous.

We will focus on the first scenario of detecting entire time series as anomalous. Since mining of entire time series, especially if multivariate, may lead to unfeasible memory and time requirements, it is desirable to have models that work on a smaller number of features. Our interest is in extracting those features using shapelets methods, which are presented in Chapters 4 and 5.

3.3 Outlier Evaluation Techniques

Evaluating the effectiveness of an anomaly detection algorithm is not immediate since the imbalanced nature of the problem. We will restrict ourselves to the case when a labeled ground truth, used exclusively for evaluating the algorithm's outcome, is available. Most outlier-detection algorithms output an outlier score [Agg13], and a threshold on this score is used to convert the scores into outlier labels. For example, in Figure 3.1 the squared distances (see Section 3.4.1) to the SVDD boundary from each shapelet-transformed time series in ECG dataset (Section 6.5) are plotted; different thresholds can be chosen to separate the two clusters.

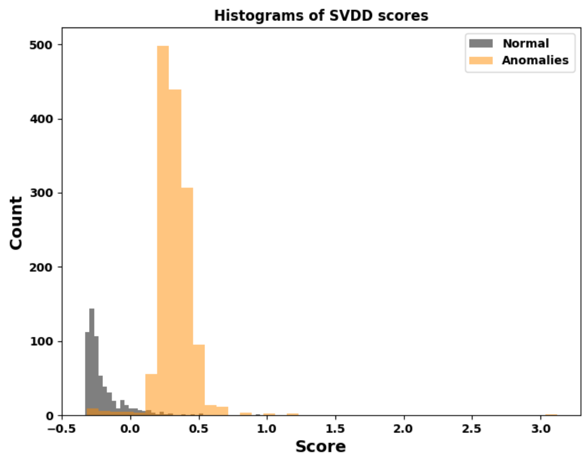


Figure 3.1: Histogram of SVDD scores for ECG dataset.

If the threshold is selected too restrictively to minimize the number of declared outliers, then the algorithm will miss true outlier points (false negatives). On the other hand, if the algorithm declares too many data points as outliers, then it will lead to too many false positives. This trade-off can be measured in terms of precision and recall. For any given threshold t on the outlier score, the declared outlier set is denoted by $S(t)$. G represents the true set (ground-truth set) of outliers in the data set.

Definition 3.3.1. *The **Precision** is defined as the percentage of reported outliers that truly turn out to be outliers*

$$Precision(t) = 100\ \frac{|S(t) \cup G|}{|S(t)|}; \tag{3.1}$$

the value of $Precision(t)$ is not necessarily monotonic in t, because both the numerator and denominator may change with t differently.

*The **Recall** or True Positive Rate (TPR) is defined as the percentage of ground-truth outliers that have been reported as outliers at threshold t*

$$Recall(t) = 100\ \frac{|S(t) \cup G|}{|G|}. \tag{3.2}$$

*The $F1$ **score** or F-measure is the harmonic mean of precision and recall:*

$$F1(t) = 2\ \frac{Precision(t)Recall(t)}{Precision(t) + Recall(t)}. \tag{3.3}$$

*Eventually, the **False Positive Rate** or "bad Recall" reports the percentage of the normal instances that are wrongly reported as outliers. For a data set D with ground truth G is defined as:*

$$FPR(t) = 100\ \frac{|S(t) \cup G|}{|D - G|}. \tag{3.4}$$

By varying the parameter t, it is possible to plot a curve between the Recall (X-axis) and the FPR (Y-axis), this is defined as the **Receiver Operating Characteristic Curve** (ROC) which is a visually intuitive tool to characterize the trade offs. Note that both "good" and "bad" Recall increase monotonically with the more relaxed values of the threshold t at which more outliers are reported. Therefore, the end points of the ROC curve are always at $(0, 0)$ and $(100, 100)$, and a random method is expected to exhibit performance along the diagonal line connecting these points, as in Figure 3.2.

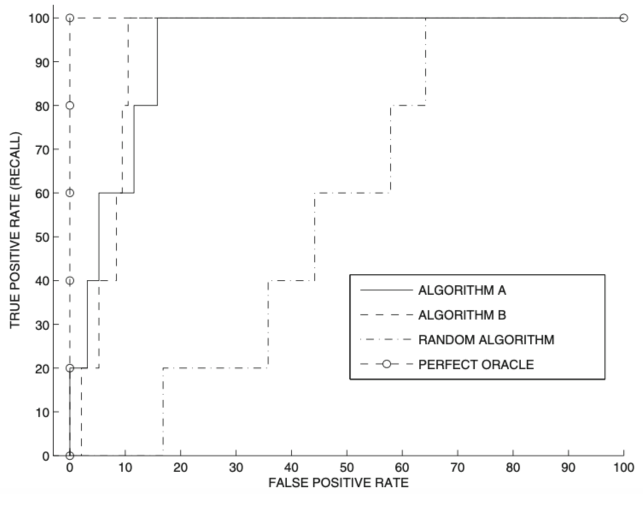


Figure 3.2: Example of ROC curves from [Agg13].

The lift obtained above this diagonal line provides an idea of the additional accuracy of the approach over a random method. The Area Under Curve

(AUC) can then be used as a proxy of the overall effectiveness of the algorithm; it has a simple probabilistic interpretation (see ([Agg13])):

Theorem 3.3.1. *Given a ranking or scoring of a set of points in order of their propensity to be outliers (with higher ranks/scores indicating greater outlierness), the AUC is equal to the probability that a randomly selected outlier-inlier pair is ranked correctly (or scored in the correct order).*

$$AUC = Mean(\{f(x_i, x_j)\}_{x_i \in G,\ x_j \in D-G}),\ \ where \tag{3.5}$$

$$f(x_i, x_j) = \begin{cases} 1 & \textit{if } x_i \textit{ scored higher than } x_j \\ 0.5 & \textit{if } x_i \textit{ scored equal to } x_j \\ 0 & \textit{if } x_i \textit{ scored lower than } x_j \end{cases} \tag{3.6}$$

Note that AUC measure should be used very carefully, because all parts of the ROC curve may not be equally important for different applications. This is because the initial part of the ROC curve is usually far more important. For example, for a data set containing 1000 points, it makes little difference whether an outlier data point is ranked at the 501th or 601th position. On the other hand, it makes a lot of difference whether an outlier data point is ranked at the 1st or 101th position.

3.4 Novelty Detection Algorithms

In this section we will describe two natural extensions of SVM algorithm to the unsupervised **One-class classification** problem [SPST$^+$01]. Given a dataset drawn from an underlying probability distribution P, the goal of One-class classification is to estimate a ”simple” subset S of the input space such that the probability that a test point drawn from P lies outside S equals some a priori specified value between 0 and 1. Note that this is an easier task than estimating the density of the distribution P generating the data, which most of the times is not necessary for detecting anomalies. If all the data are from the same distribution, the single class problem can be regarded as a quantile estimation problem:

Definition 3.4.1. *Let $x_1, ..., x_n$ be i.i.d. random variable in $\mathbb{R}^p$ with distribution P. Let **C** be a class of measurable sets (e.g. class of Borel closed convex sets) w.r.t. λ Lebesgue measure. The **quantile** function w.r.t. $(P, \lambda, \boldsymbol{C})$ is*

$$U(\alpha) = inf\{\lambda(C) : \ P(C) > \alpha, \, C \in \boldsymbol{C}\}, \ \ \alpha \in]0,1]; \tag{3.7}$$

where P is the empirical distribution of the data, we obtain the empirical quantile function.

Denote by $C(\alpha)$ the not necessarily unique $C \in \boldsymbol{C}$ that attains the infimum (when achievable) in (3.7); note that if λ is the Lebesgue measure, $C(\alpha)$ is the minimum volume $C \in \boldsymbol{C}$ that contains at least a fraction α of the probability mass. The goal of the next two algorithms is to find regions close to $C(\alpha)$ for a given $0 < \alpha \leq 1$.

3.4.1 SupportVectorDataDescription (SVDD)

Given a kernel function k defined on the space of the data, in the SVDD problem we are looking for an hypersphere in the feature space F with minimum radius that incorporates our data; in the data space the hypershpere includes the following boundary for the "description" of the data:

$$\{x \in \mathbb{R}^p, f_{a,R}(x) = R^2 - k(x-a, x-a) = 0\}. \tag{3.8}$$

The induced classification rule is therefore:

$$y = sgn(f_{a,R}(x)) = sgn(R^2 - (k(x,x) - 2k(x,a) + k(a,a))). \tag{3.9}$$

For simplicity we will consider the kernel to be linear in $F = X = \mathbb{R}^p$ (for some $p > 0$) Hilbert space of the data. Introducing the slack variables to penalize distances from the center a which are larger than R, we define the error function to minimize:

$$F(R,a) = R^2 + C \sum_{i=1}^{N} \xi_i \tag{3.10}$$

with constraints that almost all objects are within the sphere:

$$\|x_i - a\|^2 \leq R^2 + \xi_i, \qquad \xi_i \geq 0 \qquad \forall i \tag{3.11}$$

The parameter C controls the trade-off between the volume of the sphere and the errors; if C is defined through $C := 1/\nu N,\ \nu \in]0,1]$ the same result as will be seen in Theorem 3.4.2 applies to this case (see [TD04]). As in Section 2.3 it is possible to show that the optimal ξ_i satisfy $\xi_i = (\|x_i - a\|^2 - R^2)_+$.

Note that the set of functions we are looking the solution into is non linear and therefore it is not a case covered by representer theorem. To solve (3.10) and (3.11), we can define the Lagrangian function with nonnegative multipliers α_i, μ_i:

$$\begin{aligned} L(R, a, \xi_i, \alpha_i, \mu_i) = R^2 - \sum_{i=1}^{N} \alpha_i (R^2 - (\langle x_i, x_i \rangle - 2\langle x_i, a \rangle - \langle a, a \rangle)) + \xi_i) \\ + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \mu_i \xi_i. \end{aligned} \tag{3.12}$$

L should be minimized with respect to R, a, ξ_i and maximized with respect to α_i and μ_i. Setting partial derivatives w.r.t. R, a and ξ_i to zero gives [TD04]:

$$\sum_{i=1}^{N} \alpha_i = 1 \tag{3.13}$$

$$a = \sum_{i=1}^{N} \alpha_i x_i \tag{3.14}$$

$$\alpha_i = C - \mu_i \tag{3.15}$$

As before, substituting (see [TD04]) leads to the Dual problem:

$$\arg\max_{\alpha} L(\alpha) = \sum_{i=1}^{N} \alpha_i \langle x_i, x_i \rangle - \sum_{i,j=1}^{N} \alpha_i \alpha_j \langle x_i, x_j \rangle \tag{3.16}$$

Same considerations as standard SVM hold for the relationship between support vectors and Lagrange multipliers. Let z be any support vector on the boundary ($0 < \alpha(z) < C$); the radius of the description can then be computed (reminding that $a = \sum_{i=1}^{N} \alpha_i x_i$):

$$R^2 = \left\|(z-a)^2\right\| = \langle z, z\rangle - 2\sum_{i=1}^{N} \alpha_i \langle z, x_i\rangle + \sum_{i,j=1}^{N} \alpha_i \alpha_j \langle x_i, x_j\rangle \quad (3.17)$$

Maximizing expression (3.16) with constraints (3.13) and (3.15) is a quadratic programming with linear constraint problem; we used python's library *convxopt* [ADV20] to solve it.

3.4.2 Modified SVDD

A modified version of the SVDD algorithm with the center forced to be fixed in the origin is proposed in [BKS+19]. The corresponding constrained optimization problem becomes

$$\arg\min_{R,\xi} R^2 + C\sum_{i=1}^{N} \xi_i \quad (3.18)$$

$$\text{s.t. } \|x_i\|^2 < R^2 + \xi_i, \;\; \xi_i \geq 0 \quad (3.19)$$

and after analogous computations the dual becomes

$$\arg\max_{\alpha} L(\alpha) = \sum_{i=1}^{N} \alpha_i \langle x_i, x_i\rangle \quad (3.20)$$

$$\text{s.t. } \sum_{i=1}^{N} \alpha_i = 1, \;\; 0 < \alpha_i < C \quad (3.21)$$

which in this case is a linear problem and can be optimized through linear programming methods. For instance, we used python's library *convxopt* [ADV20].

3.4.3 OneClassSupportVectorMachine (OCSVM)

The strategy in OCSVM is to map the data into the feature space corresponding to some kernel, and to separate them from the origin with maximum margin by an hyperplane with generic equation

$$\{\phi(x)\ , \langle w, \phi(x)\rangle - \rho = 0\} \tag{3.22}$$

with parameters $w \in F$ feature space and $\rho \in \mathbb{R}$. For simplicity we will consider ϕ to be the identity, the kernel to be linear in $F = X = \mathbb{R}^p$ (for some p) Hilbert space of the data. Note that the distance of the hyperplane from the origin from the origin is $|\rho| / \|w\|_2$ under such a parametrization,

Definition 3.4.2. *A dataset $D = x_1, ..., x_N \subset \mathbb{R}^p$ is called separable if there exists some $w \in X$ such that*

$$\langle w, x_i\rangle > 0 \quad \forall i. \tag{3.23}$$

Note that if we use a Gaussian kernel in ex 2.2.1, any dataset is separable in the feature space; indeed, $k(x_i, x_j) > 0\ \forall i, j$, implying that all the data lie inside the same orthant. Moreover, since $k(x_i, x_i) = 1$ they all have unit length, hence they are separable from the origin.

Theorem 3.4.1. *(Supporting Hyperplanes, [Cal20]) Suppose C and D are two convex sets in $\mathbb{R}^n$ that do not intersect. Then there exist $a \neq 0$ and b such that $a^T x \leq b\ \ \forall x \in C$ and $a^T x \geq b \forall x \in D$. In other words, the affine function $a^T x - b$ is nonpositive on C and nonnegative on D. The hyperplane $\{x | a^T x = b\}$ is called a separating hyperplane for the sets C and D, or is said to separate the sets C and D.*

Proposition 3.4.1. *If a data set is separable, then there exists a unique supporting hyperplane with the properties that*

- *it separates all data from the origin*
- *its distance to the origin is maximal among all such hyperplanes*

. For any $\rho > 0$ fixed, it is given by

$$\min_{w \in X} \frac{1}{2} \|w\|^2 \ \ subject\ \ to\ \ \langle w, x_i\rangle \geq \rho \tag{3.24}$$

Proof. Due to the separability, the convex hull of the data does not contain the origin. The existence and uniqueness of the hyperplane then follows from theorem 3.4.1. Moreover, separability implies that there actually exists some $\rho > 0$ and w such that $\langle w, x_i \rangle \geq \rho \ \forall i$; by rescaling w this can be seen to work for arbitrarily large ρ. The distance of the hyperplane $\{x \in X, \ \langle w, x \rangle = \rho\}$ to the origin is $\frac{\rho}{\|w\|}$. Therefore the optimal hyperplane is obtained by minimizing $\|w\|$. □

To maximize the margin we want to maximize ρ and the objective function becomes

$$\min_{w \in X} \left[\frac{1}{2} \|w\|^2 - \rho\right] \ \ subject \ \ to \ \langle w, x_i \rangle \geq \rho. \tag{3.25}$$

Introducing the slack variables ξ_i to model the errors with coefficient C:

$$\min_{w \in X} \left[\frac{1}{2} \|w\|^2 - \rho + C \sum_{i=1}^{N} \xi_i\right] \ \ subject \ \ to \ \langle w, x_i \rangle \geq \rho - \xi_i, \ \ \xi_i \geq 0. \tag{3.26}$$

Define $\nu \in]0, 1]$ and $C := 1/\nu N$; ν controls the tradeoff between margin to be high (which implies the regularization term $\|w\|$ to be small) and the penalization induced by the slack variables.

As we did before, to solve the convex optimization problem we introduce the Lagrangian function and after setting to zero the derivatives w.r.t. the primal variables w, ξ, ρ it yields to the following dual problem:

$$\min_{\alpha} \frac{1}{2} \sum_{ij} \alpha_i, \alpha_j \langle x_i, x_j \rangle \ \ subject \ \ to \ \ 0 \leq \alpha_i \leq \frac{1}{\nu N}, \ \ \sum_{i=1}^{N} \alpha_i = 1 \tag{3.27}$$

with $w = \sum_{i=1}^{N} \alpha_i x_i$. Using KKT conditions 2.3.3 it is straightforward to see that the observations with $0 < \alpha_i < \frac{1}{\nu N}$ satisfy $\langle w, x_i \rangle = \rho, \ \ \xi_i = 0$ and ρ can be recovered using $\rho = \sum_{i=1}^{N} \alpha_i \langle x_i, x_i \rangle$ for such x_i's.

We state the following theorem without proof (see [SSB$^+$02]):

Theorem 3.4.2. *(ν-Property) Assume the solution of 3.26 satisfies $\rho \neq 0$. The following statements hold:*

- *ν is an upper bound on the fraction of outliers.*

- *ν is a lower bound on the fraction of SVs.*

- *Suppose the data were generated independently from a distribution P which does not contain discrete components. Suppose, moreover, that the kernel is analytic and non-constant. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of outliers.*

Chapter 4

Shapelets

In this Chapter we will review some relevant papers to introduce the concept of shapelets and how they are applied, with the goal of implementing such methods in Chapter 5 for the Anomaly detection task. We will discuss briefly about supervised (Keogh, [YK09]) and unsupervised (Zakaria, [LDHB12]) shapelets, both defined as real subsequences to be searched through the dataset. Next we will introduce the first algorithm for shapelets learning (Grabocka, [GSWST14]) and a general form for the learning problem; shapelets' interpretability and connection with Convolutional Neural Networks are then discussed. Lastly, we present Beggel et al.'s algorithm for unsupervised anomaly detection using shapelets, proposed in [BKS+19], and discuss about Gradient Descent optimization techniques.

4.1 Definitions and background

Definition 4.1.1. *Given a time series $T = \{t_k\}_{k=1}^{Q}$ with length Q and C channels, its subsequence with length L and starting position $s < Q-L+1$ is $T^{s,L} = t_s, t_{s+1}, \dots, t_{s+L-1}$, where each element t_k is a vector in $\mathbb{R}^C$.*

Definition 4.1.2. *Let $\{T_i\}_{i=1}^{N}$ be a set of N time series of the same length Q with C channels. We define a* ***shapelet*** *as any sequence in $\mathbb{R}^C$ of length*

$L < Q$.

Definition 4.1.3. *(l^2 discrepancy) Given a shapelet $S = \{s_j\}_{j=1}^{L}$ and a time series $T = \{t_j\}_{j=1}^{Q}$, the l^2* ***discrepancy*** *between S and T is defined as*

$$d(T, S) = \min_{s=1,\ldots,J} \sqrt{\frac{1}{L} \sum_{j=1}^{L} \|t_{s+j} - s_j\|_2^2} \tag{4.1}$$

where $J = Q - L + 1$ is the number of subsequences of T with length L, s is the starting index of each subsequence and $\|\cdot\|_2$ is the euclidean norm in $\mathbb{R}^C$.

Definition 4.1.4. *Let $S = \{s_i\}_{i=1}^{L}$ be a time series segment with length L. Its mean μ and variance σ^2 are*

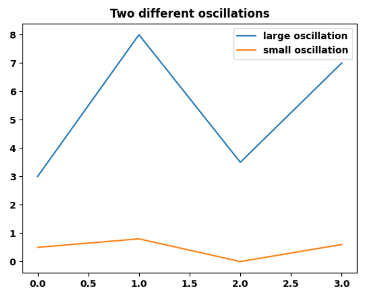
$$\mu = \frac{1}{L} \sum_{i=1}^{L} s_i; \quad \sigma^2 = \frac{1}{L} \sum_{i=1}^{L} (s_i - \mu)^2. \tag{4.2}$$

The standardized or z-normalized time series is

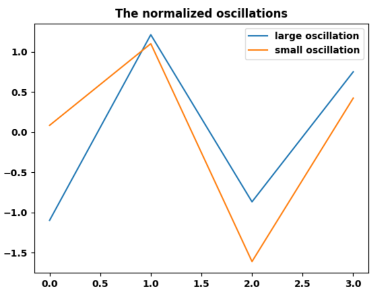
$$\bar{S} = \left\{ \frac{s_i - \mu}{\sigma} \right\}_{i=1}^{L} \tag{4.3}$$

where $\sigma = \sqrt{\sigma^2}$ is the standard deviation.

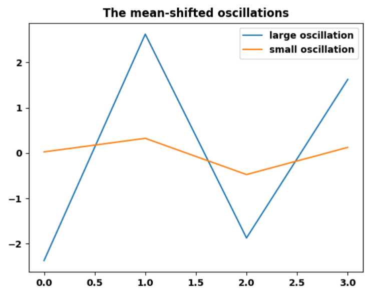
Euclidean distance is used to compute the distance between two subsequences of equal length. In [ZMK12] it is suggested to z-normalize the time series before computing their Euclidean distance, in order to make the distance measure invariant to scale and offset. A note here has to be made about z-normalization of subsequences: often if can happen that there is noise in time series measurements, which translates in small oscillations around a mean value. If we divide by the variance before calculating euclidean distance, the small oscillations become comparable to a significant one, as in Figure 4.1. Therefore we propose only to shift the sequences by their mean in order to make them comparable but not to alter them.



(a) Two oscillations significantly different in magnitude



(b) The oscillations after z-normalization



(c) The oscillations after mean-shift

Figure 4.1: Two oscillations after z-normalization and mean-shift

Definition 4.1.5. *(Shapelet Transform) Given a set of K shapelets* **S**$=\{S_k\}_{k=1}^{K}$ *with length Q and a time series T the shapelet transform σ of T w.r.t.* **S** *is the vector in $\mathbb{R}^K$*

$$\sigma_{\mathbf{S}}(T) = (d(T,S_1),\dots,d(T,S_K)). \tag{4.4}$$

The shapelet transform is a feature vector of the representation of the time series T w.r.t. the set of shapelets **S**. The goal of the shapelets-based algorithms is to find shapelets that enable to:

- reduce the dimensionality, since tipically $K << Q$;

- make data mining tasks easier in the transformed space;
- contribute to have interpretable results.

4.2 Keogh's Shapelets

Shapelets were first introduced in [YK09] as a primitive for time series classification: informally, they are time series subsequences which are in some sense maximally representative of a class. The algorithm proposed by Keogh adressed some limitations of the most accurate and robust time series classification algorithm at that time, the nearest neighbor with either Dynamic Time Warping or eucliden distance [DTS$^+$08], such as the use of the entire time series (gloabal features) and the slowness in classifying new ones. To introduce the first definition of shapelets, we give some definitions:

Definition 4.2.1. *Given a random variable X taking values in a discrete space* $\mathscr{X} = \{x_1, \dots, x_n\}$*, with distribution* $p : X \mapsto [0,1]$ *its* ***entropy*** $I(X)$ *is defined as:*

$$I(X) = -\sum_{i=1}^{n} p(x_i) log(p(x_i)) \tag{4.5}$$

Note that if $n = 1$, $I(X) = 0$. Suppose we have a dataset $D = (T_1, y_1), \dots, (T_N, y_N)$ of N time series with labels belonging to c different classes. If X is the random variable which maps each series to its label, its entropy w.r.t. the empirical distribution of the data is $I_D = I_D(X) = -\sum_{i=1}^{c} p_i log(p_i)$ where p_i is the proportion of the i-th class in D. Suppose we have a splitting strategy that divides D into two complementary subsets, D_1 and D_2, the total entropy of D after splitting is defined as

$$\hat{I}_D = f(D_1) I_{D_1} + f(D_2) I_{D_2}, \tag{4.6}$$

where $f(D_k) = \frac{|D_k|}{|D|}$ is the fraction of time series in the k-th subset, $k = 1, 2$.

Definition 4.2.2. *(Information Gain) Given a certain split strategy sp which divides D into two subsets* D_1 *and* D_2*, the entropy before and after splitting* *is $I(D)$ and $\hat{I}(D)$. The information gain for this splitting rule is:*

$$Gain(sp) = I(D) - \hat{I}(D) \tag{4.7}$$

The discrepancy to a shapelet is used as the splitting rule: shapelets are defined as subsequences such that most of the time series objects in one class of the dataset are close to the shapelet under l^2 discrepancy, while most of the time series objects from the other class are far away from it.

Definition 4.2.3. *(Optimal Split Point (OSP)) For a shapelet candidate S, each distance threshold d_{th} splits D into D_1 and D_2, such that for every time series object T_i^1 in D_1, $d(T_i^1, S) < d_{th}$ and for every time series object T_j^2 in D_2, $d(T_i^2, S) \geq d_{th}$. An Optimal Split Point is a distance threshold such that*

$$Gain(S, d_{OSP}(D, S)) \geq Gain(S, d'_{th}(D, S)) \tag{4.8}$$

for any other distance threshold d'_{th}.

Definition 4.2.4. *(Shapelet) Given a time series dataset D, shapelet(D) is a subsequence that, with its corresponding optimal split point, satisfies*

$$Gain(shapelet(D), d_{OSP}(D, shapelet(D))) \geq Gain(S, d_{OSP}(D, S)) \tag{4.9}$$

for any other subsequence S.

In this formulation, shapelets are defined as nodes in a decision tree (Figure 4.2) that give maximum information gain w.r.t. the optimal splitting rule, based on discrepancies between time series and the shapelets. For a formal definition of decision trees, see Hastie, Trevorr [HTFF09], Section 9.2.

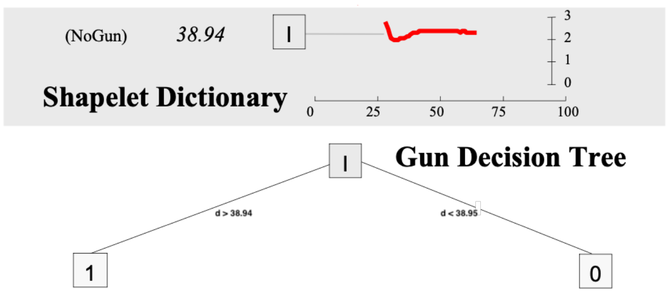


Figure 4.2: (top) The extracted shapelet from GunPoint dataset (Section 6.3), with the threshold d_{th}. (bottom) The associated decision tree. [YK09]

Given a dataset of N series with length Q, fixed a length $L < Q$ there are $N(Q - L + 1)$ shapelets candidates; it is reasonable to search among the subsequences since the time series objects in one class presumably contain some similar subsequences, which can be good shapelet candidates. Drawback of this method is its large computational complexity $\mathcal{O}(N^2Q^3)$.

Subsequently, in [LDHB12] Lines et al. propose to disconnect the process of finding shapelets from the classification algorithm by proposing the shapelet transformation of definition 4.1.5: in the transformed space, various classifiers can be applied to achieve better accuracy.

4.3 Unsupervised Shapelets

In 2012, a method for extracting unsupervised shapelets for clustering purposes was proposed by Zakaria et al. in [ZMK12]. Since the Information Gain cannot be computed if the class labels are not given, a new score, the **Gap** score, is defined to evaluate the clustering property of a candidate subsequence S. From now on we will refer to D as a dataset containing N time series with length Q and S as a subsequence contained in the data with length L fixed.

Suppose all the discrepancies from S and each time series in the train set have been computed and sorted in an ordeline. Define a threshold dt and take all the time series that have discrepancy smaller than dt to form the subset A and the others the subset B:

$$A = \{T \in D,\ d(T, S) < dt)\}; \quad B = A^C \tag{4.10}$$

The Gap score of S w.r.t. the threshold dt is

$$Gap(dt) = \mu_B - \sigma_B - (\mu_A + \sigma_A), \tag{4.11}$$

where

$$\mu_A = mean(\{d(T,S)\}_{T \in A}), \tag{4.12}$$

$$\sigma_A = std(\{d(T,S)\}_{T \in A}) = \sqrt{\frac{1}{|A|}\sum_{T \in A}(d(T,S) - \mu_A)^2}. \tag{4.13}$$

The optimal Gap, which we will refer to simply as Gap, is the maximum Gap considering all the possible thresholds. Intuitively, the Gap is big if the candidate separates very well two complementary subsets A and B of the dataset w.r.t. the discrepancy values.

An **unsupervised shapelet** (u-shapelet) is defined as a subsequence that maximizes the Gap score; once a u-shapelet is found, a part of the subset A (the time series with 'small' discrepancy to the u-shapelet) is removed from the dataset and the searching continues iteratively in the reduced dataset until either it becomes empty or a predefined number K of u-shapelets is discovered.

Computing Gap scores is computationally expensive: for each candidate (there are $N \cdot (Q - L + 1)$ of them) N discrepancies must be computed, which takes $\mathcal{O}(N \cdot Q)$ time, plus the Gap values have to be computed for N different thresholds, which takes $\mathcal{O}(N^2)$ time, and can be sorted in $\mathcal{O}(N \cdot log(N))$ time. In practical applications, the computational time of the brute force search is infeasible.

4.4 SpeedUp Techniques

4.4.1 Subsequence Distance Early Abandon

Obtaining the distance between all candidates shapelets and their nearest matching subsequence of each of the objects in the dataset is one the most expensive calculations in the brute force algorithm. An immediate saving of computational time can be achieved by caching the distance computed and using the 'Early Abandon' trick [YK09].

Algorithm 1 l^2 discrepancy early abandon

```
Require: $T$ time series, $S = \{s_i\}_{i=1}^{L}$ subsequence
Ensure: min_dist = $d(T, S)$
    1. Initialize a variable min_dist, the current minimum distance from the time
    series $T$ to the subsequence $S$, equal to $+\infty$.
    2. For each subsequence of $T$ with length $L$, $S'$, accumulate the distances between $S$ and $S'$, one data point at a time.
    3. If the cumulative sum is larger than or equal to the minimum distance known
    so far, abandon the distance calculation, since it does not contribute to the discrepancy value.
    4. If the abandon never happens, update min_dist to the current value.
```

Although the early abandon search is still $\mathcal{O}(Q)$, it can been seen [YK09] that this simple trick reduces the time required by a large, constant factor.

4.4.2 Other Algorithms

Many algorithms were proposed to speed up the research of the optimal shapelets; all of them start from the consideration that there is no need to find the best shapelets, since most of the times there are ”good enough” subsequences, which have same discriminating power. We will name only some of them, applied to the unsupervised case.

Scalable U-Shapelets (SUSh), proposed by Ulanova et al. in [UBK15], is a hash- based algorithm that allows fast u-shapelet discovery. First, all the subsequences with fixed length L are extracted from the dataset, then they are converted into Symbolic Aggregate approXimation (SAX), introduced in [LKLC03], which allows the transformation of a real-valued time series of length L to a discrete string of length $L' << L$, using d discrete symbols (Figure 4.3). Instead of computing discrepancies, collisions are verified: a time series T and a sub- sequence S are said to collide if T has a subsequence thet shares the same SAX representation with S. Candidate subsequences are then filtered and reordered based on the number of collisions: the expensive gap scores are computed only for the most promising candidates, which seem to separate well the data in terms of SAX collisions. Since computing SAX representation and counting the collisions is much less expensive than computing euclidean distances, the algorithm provides a saving in computational time.

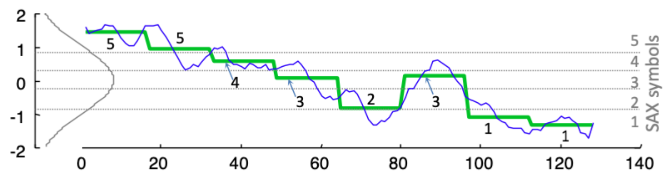


Figure 4.3: SAX representation with dimensionality 8, using 5 symbols [UBK15].

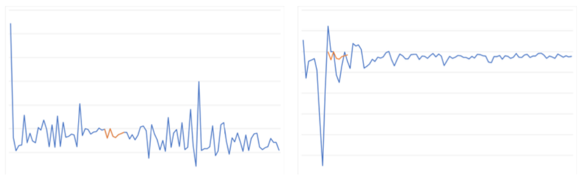
Another speed-up technique, the **Random Local Search** (RLS) algorithm, takes into consideration shapelets of different lengths; since if the time series in the dataset are well aligned, candidates with nearby starting position and length are likely to have similar scores, RLS proposes to calculate the expensive scores only for a random subset of the total number of subsequences. In this subset, the best scoring are taken and scores are computed iteratively for their neighbors, defined w.r.t. position and length. The algorithm stops when no more better candidate shapelets are found in the neighborhoods. In Section 5.3.3 we will describe in detail a version of RLS applied to anomaly detection.

4.5 Multivariate Shapelets

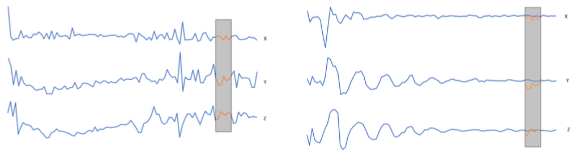
When dealing with multivariate shapelets, according to Bostrom, Bagnall [BB17] there are two possible ways of defining shapelets, depending on which hypothesis are made about the time series data:

- **Independent Shapelets** are single channel shapelets from any dimension. The motivation for this methods is that in some multivariate datasets the class defining feature may occur in only one channel, and it could even be independent of channel.

- **Dependent Shapelets** are multi-dimensional shapelets, that are then compared to a multivariate series using Frobenius norm as in Definition 4.1.3, maintaining the phase across channels.



(a) Univariate shapelet extracted from a single channel (left), and matched by sliding along the same channel (right).



(b) Multivariate shapelet extracted (left), and matched mantaining phase across channels (right).

Figure 4.4: Different methods for defining a shapelet in multivariate time series ([BB17]).

In Definition 4.1.3 we defined the multivariate l^2 distance for time series with C channels, implicitly considering the channels dependent from each other. In this thesis we always assumed channel-dependence and considered shapelets according to Definition 4.1.3.

4.6 Shapelets Learning

In their original formulation, shapelets are always subsequences of time series contained in the training set. In [GSWST14], the authors extend the concept of shapelets: they become parameters and are learnt jointly with a classifier model in the shapelets' feature space. This approach gives more flexibility to the shapelets, in order to be maximally representative of the different classes. A general framework for the supervised shapelets learning problem can be expressed as the following empirical risk minimization:

$$\mathbf{S}, f = \arg\min_{\mathbf{S},f} \left[I_T[f] + \lambda R(f) \right], \quad I_T[f] = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{4.14}$$

where $f \in H$ hypothesis space of functions, R regularization function, T is a dataset of N time series, **S** a set of K shapelets and $x_i = \sigma_{\mathbf{S}}(T_i)$ is the shapelet transform vector. The domain of f is $[0, +\infty[^K$, since it takes l^2 discrepancy values in input.

Note that searching methods start from the same equation as Eq. (4.14) but the shapelets **S** are constrained to be in the discrete space of the subsequences long L in the dataset T. Since the hypothesis space is larger, a better accuracy is expected from learning methods.

In [GSWST14], Grabocka at al. were the first to introduce the learning approach for classification; a linear logistic regression model is proposed, optimized through a regularized cross-entropy loss (in case of binary classification):

$$\min_{\mathbf{S},w,w_0} I_T + \lambda \left\| w \right\|; \;\; I_T = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y_i}); \tag{4.15}$$

where T, **S**, $\{x_i\}_{i=1}^{N}$ as before,

$$\hat{y_i} = \sigma(\langle x_i, w \rangle + w_0) \tag{4.16}$$

is the logistic regression output, with $\sigma(y) = (1 + e^{-y})^{-1}$ sigmoid function, and

$$L(y_i, \hat{y_i}) = -y_i log(\hat{y_i}) - (1 - y_i) log(1 \hat{-} y_i) \tag{4.17}$$

is the cross entropy loss. Gradient descent (GD) (Section 4.7) is proposed as optimization algorithm. The overall complexity of the algorithm is $\mathcal{O}(I \cdot N \cdot Q^2)$ where I is the maximum number of iterations of GD.

4.6.1 Interpretability

The main advantage of shapelets' methods stands in their interpretability: the use of a particular feature in the classification corresponds to the importance of the similarity to the correspondig shapelet. This may reveal some shape that is characteristic of a particular class, and can discover patterns in the data. As pointed out in [KML20], the predictive power of the discrepancy between a shapelet and a time series not necessarily needs to correlate with a similarity between the two. When learning shapelets, interpretability is sacrificed towards better accuracy and computational time, since the shapelets are no more real subsequences.

In order to learn interpretable shapelets, Kidger et al. propose to add a shapelet penalty regularization term to Eq. (4.14):

$$\min_{\mathbf{S},w,w_0} \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda_1 \|w\| + \lambda_2 \sum_{k=1}^{K} \min_{n=1,\ldots,N} d(T_n, S_k). \tag{4.18}$$

Taking a minimum over n asks that every shapelet should be similar to a single training sample, as in the original approach of finding shapelets by searching through the dataset instead of training differentiably.

In [WEF$^+$20] shapelets are defined as weights of a 1D Convolutional Layer of a Neural Network (see Section 4.6.2); to enhance their interpretability a Generative Adversarial Network architecture is proposed: the sequence learned are given in input to a discriminator which is trained to distinguish them from real subsequences. In such a way, shapelets that do not resemble real subsequences are penalized.

4.6.2 Connection with CNN

Definition 4.6.1. *A* ***discrete one dimensional signal*** *is a sequence of real numbers indexed by integers in* $\mathbb{Z}$

$$y = [\ldots, y_{-2}, y_{-1}, y_0, y_1, y_2, \ldots] \tag{4.19}$$

where y_k represents the signal amplitude measured at time t_k.

A signal has ***compact support*** *if there is $N \geq 1$ such that $y_k = 0$ for any $|k| > N$.*

A signal has ***finite energy*** *if $\sum_{k=-\infty}^{+\infty} y_k^2 < +\infty$.*

A signal can be processed by filtering. This involves a cross-correlation operation between the signal and a kernel or filter.

Definition 4.6.2. *A **kernel** (filter) is a compact support sequence of weights*

$$w = [..., w_{-2}, w_{-1}, w_0, w_1, w_2, ...] \tag{4.20}$$

Definition 4.6.3. *The **discrete convolution** between a finite energy signal y and a kernel w is the signal*

$$r := y * w = [..., r_{-2}, r_{-1}, r_0, r_1, r_2, ...] \tag{4.21}$$

defined by:

$$r_j = \sum_{k=-\infty}^{+\infty} y_{j-k} w_k. \tag{4.22}$$

*The above infinite sum makes sense since w has only a finite number of nonzero elements. The lag by which the kernel slides with respect to the signal is called **stride**, which in the standard definition is equal to 1.*

*Let $w^{-1} := [..., w_2, w_1, w_0, w_{-1}, w_{-2}, ...]$ be the time-reversed filter; the **cross-correlation** between y and w is the signal $c = y * w^{-1}$ obtained by the discrete convolution between y and w^{-1}.*

*The **cross-correlation coefficient** between y and w is the signal ρ defined as*

$$\rho_j = \frac{c_j}{\sqrt{v_y v_w}} \tag{4.23}$$

where $v_y := \sum_{k=-\infty}^{+\infty} y_k^2$ is the cross-correlation of y with itself at "lag" 0. Not that $\rho_j \leq 1 \ \forall j$. If v and w are finite sequences, the cross-correlation coefficient is defined as if they were infinite sequences, resulting from the original ones padded with zeros on both sides.

Convolutional neural networks (CNN) [Cal20] are "feedforward" neural networks (i.e., networks where the information flows from the input to the output) which contain Convolutional Layers: they enable linear computation with shared weights and sparse interactions, that is, most weights are equal to zero.

Given a compact support signal x, a **One Dimension Convolutional Layer** is the resulting of the composed operations on x:

$$Y = f(Wx + B) \tag{4.24}$$

where W is the "system of weights", B the bias term and f any differentiable activation function; W is the matrix of the cross-correlation operation. To give an example if the input signal is $X = [x_1, ..., x_6]^T$, the sliding kernel is $w = [w_1, w_2]$ and the bias is $b = [b, b, b, b, b]^T$ then the system of weights matrix is

$$W = \begin{pmatrix} w_1 & w_2 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 \end{pmatrix} \tag{4.25}$$

The concept of **pooling** will be useful to establish a connection between CNN and shapelets: pooling is a machine learning technique that provides a summary of the input, selecting some essential local features (such as maxima, minima, averages,...) and acts as an information contractor: in the discrete case it decreases the dimension of the input by a certain factor. The definition that follows considers the 1D case for a continuous input.

Definition 4.6.4. *Let $f : [a, b] \mapsto \mathbb{R}$ be a continuous function and consider the equidistant partition of the interval $[a, b]$*

$$a = x_0 < x_1 < ... < x_{n-1} < x_n = b. \tag{4.26}$$

The partition's size, $x_1 - x_0$, is called stride. Denote by $M_i = max_{[x_{i-1}, x_i]} f(x)$. Consider the simple function

$$S_n(x) = \sum_{i=1}^{n} M_i \mathbb{1}_{[x_{i-1}, x_i]}(x). \tag{4.27}$$

where $\mathbb{1}_{[x_{i-1}, x_i]}$ is the indicator function of the interval $[x_{i-1}, x_i]$. The ***max-pooling*** *is the process of approximating the function $f(x)$ by the step function S_n; analogously* ***min-pooling*** *replaces the max operation with the min.*

In [WEF+20] shapelets are defined as the learnable convolutions filters in the convolutional layer of a neural network. The shapelet transform is the result of the convolution layer with identity activation function and a max-pooling layer, where the maximum is taken w.r.t. the time index of the signal resulting from the cross-correlation. This approach simply uses a different measure of similarity between a shapelet and a subsequence: given an input time series, the value with maximum correlation (instead of minimum euclidean distance) to any subsequence is chosen.

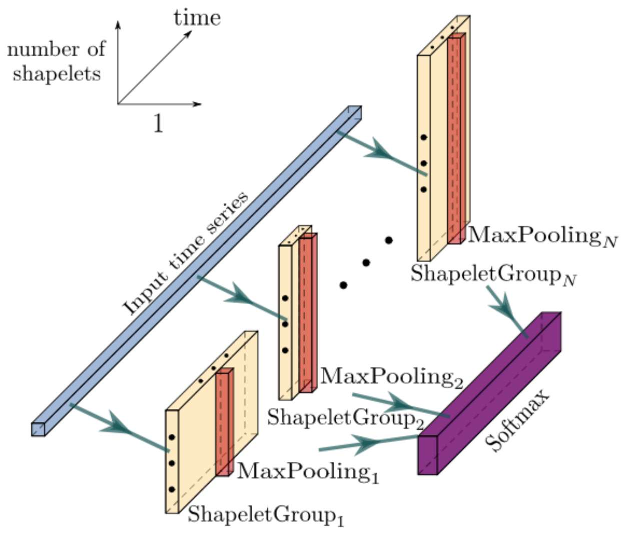


Figure 4.5: LS model as a neural network architecture. [WEF+20]

In figure 4.5, Grabocka's algorithm presented in section 4.6 is displayed in terms of a CNN architecture with a convolution, a max-pooling and a linear layer.

To sum up, we can say that shapelets are closely related to convolutional neural networks but their main advantage is to provide interpretability of the results.

4.6.3 ADSL

The *Anomaly Detection algorithm with Shapelet-based Feature Learning* (ADSL) is introduced by Beggel et. al in [BKS+19] and it is based on the modified Support Vector Data Description (SVDD) algorithm with center in the origin. K shapelets with length L are learned jointly with the hypersphere in feature space with radius R that classifies an instance x as anomalous if $\|x\|^2 > R^2$. A dataset suitable for anomaly detection problem is assumed , with a small proportion α of anomalous train instances w.r.t. the number of normal train instances. The optimization problem that follows is:

$$argmin_{\mathbf{S},R}\Big[R^2 + C\sum_{i=1}^{N}\max(0,\, l(x_i) - R^2) + \sum_{i=1}^{N} l(x_i)\Big] \tag{4.28}$$

where

$$x_i = \sigma_{\mathbf{S}}(T_i) = (d(T_i, S_1), \ldots, d(T_i, S_K)) \tag{4.29}$$

$$l(x_i) = \|x_i\|_2^2 = \sum_{k=1}^{K} d(T_i, S_k)^2, \;\; i = 1, ..., N \tag{4.30}$$

$$C = \frac{1}{\nu \cdot N} \tag{4.31}$$

ν is the expected proportion of anomalies w.r.t. N, the total number of training time series. The goal is to learn shapelets representing the normal class; if so, shapelet features of normal time series are expected to be small: the last regularization term encourages the (normal) data points to be pulled towards the origin in the transformed space during learning. Note that

$$\sum_{i=1}^{N} l(x_i) = \sum_{k=1}^{K}(\sum_{i=1}^{N} d(T_i, S_k)^2) \tag{4.32}$$

so the loss is low if the shapelets have small discrepancy with (almost) all the time series in the dataset.

In order to minimize the loss $L = L_{R,\mathbf{S}}$ in Equation (4.28) w.r.t. the SVDD radius R and the set of representative shapelets **S**, a block-coordinate algorithm is proposed in [BKS$^+$19]: the optimization is carried on with alternating updates, either update the radius R of the decision boundary is udated by sololving the linear constrained optimization problem in Equations 3.18 and , or the set of shapelets **S** using Gradient Descent techniques, explained in Section 4.7.

4.6.4 Shapelets similarity loss

Learning shapelets different from each other is a highly desirable characteristic of an algorithm, since if two shapelets are similar they are redundant and the effective number of learnt shapelets is less than the desired number K. Since in original ADSL algorithm's loss $L = L_{\mathbf{S},R}$ function there is no term to penalize similar shapelets, we introduce a penalty term based on l^2 discrepancy as measure of similarity, since it considers the different shifts of two shapelets. Given a shapelet S with length L, define S^0 the sequences with length $2L$ obtained by padding S with zeros in both the directions. Taking inspiration from [ZWY+16], we introduce the following discrepancy-based loss penalty term:

$$L_{total} = L + \lambda \frac{2}{K(K-1)} \sum_{i=1}^{K} \sum_{j<i} \exp\Big(- \frac{d(S_j^0, S_i)^2}{\sigma^2} \Big), \quad \lambda, \sigma > 0 \quad (4.33)$$

where $d(S_j^0, S_i)$ is the discrepancy between the padded j-th shapelets the i-th shapelet, $\frac{K(K-1)}{2}$ is the number of combinations with no repetitions of shapelets, λ and σ are regularization hyperparameters. We set $\sigma = 1$ by default in all the experiments in Section 6. By adding this term, the goal is to penalize similar shapelets in euclidean distance, at any lag.

4.7 Optimization

4.7.1 Gradient Descent Algorithm

The learning process consists of tuning the model parameters (which include the shapelets parameters) until a certain cost function is minimized.

One of the most largely used algorithms for finding minima is the **gradient descent**, which goes through the level sets of the cost functions into the direction of maximum cost decrease. In order to explain the Gradient Descent (GD) algorithm we start with some preliminary propositions; details can be found in [Cal20].

Definition 4.7.1. *Given a real valued function f defined on an open set*

$D \subset \mathbb{R}^n$ with $n \geq 2$, define the family of ***level sets***

$$S_c = f^{-1}(\{c\}) = \{x \in D,\ f(x) = c\}. \tag{4.34}$$

Given $c \in \mathbb{R}$, if f is differentiable with nonzero gradient, $\nabla f(x) \neq 0,\ \forall x \in S_c$, then S_c becomes an $(n-1)$-dimensional hypersurface in $\mathbb{R}^n$.

Proposition 4.7.1. *The gradient ∇f is perpendicular to S_c.*

Assume now the function has a local minimum at $x^* \in D$, i.e. $f(x^*) < f(x)\ \forall x \in V$, with V neighborhood of x^*. Then there is an $\varepsilon > 0$ such that $S_c \subset V\ \forall c \in [f(x^*), f(x^*) + \varepsilon[$. For $c = f(x^*)$ the hypersurface degenerates to a point, $S_c = \{x^*\}$. For small enough ε the family $\{S_c\}_{c \in [f(x^*),\, f(x^*)+\varepsilon[}$ is nested, i.e., if $c_1 < c_2$ then $S_{c_1} \subset Int(S_{c_2})$.

Recall that a function f is called Lipschitz continuous if there is a constant $K > 0$ such that $|f(x) - f(y)| \leq K|x - y|$, for all x and y in the domain of f.

Theorem 4.7.1. *In the previous settings, assume ∇f is Lipschitz continuous; for any point x_0 close enough to x^* there is a differentiable curve $\gamma : [0, \delta] \mapsto \mathbb{R}^n$ such that*

- *$\gamma(0) = x_0$;*
- *$\gamma(\delta) = x^*$;*
- *γ' is normal to $S_{f(\gamma(s))}$, $\forall s \in [0, \gamma[$.*

All points that can be joined to x^* by a curve γ satisfying the aforementioned properties form the ***basin of attraction*** *of x^*.*

The theorem states existence of a curve of *steepest descent*, γ, from x_0 to x^*, which intersects normally the level sets family S_c; in computer implementations, we need to approximate the curve γ by a polygonal line formed by a sequence $x_0,\ x_1, ...,\ x_m$.

The GD algorithm computes such a line using the following iteration:

1. Choose an initial point x_0 in the basin of attraction of the global minimum x^*.

2. Construct the sequence $(x_n)_{n\in\mathbb{N}}$ using the iteration

$$x_{n+1} = x_n - \eta\frac{\nabla f(x_n)}{\|\nabla f(x_n)\|}. \tag{4.35}$$

The constant η is called the **learning rate**. Note that the line from x_n to x_{n+1} is normal to $S_{f(x_n)}$. From the previous relation, the change in the function after each step is proportional to the magnitude of the gradient as well as to the learning rate. By Taylor's expansion of f centered in x_n, the update guarantees a negative change of the objective function given by

$$f(x_{n+1}) - f(x_n) \simeq -\eta\,\|\nabla f(x_n)\| < 0 \tag{4.36}$$

provided η is small enough. Indeed let $v \in \mathbb{R}^n$ be unit vector, the first order Taylor linear approximation is:

$$f(x_n + \eta v) - f(x_n) = \eta\langle\nabla f(x_n), v\rangle + o(\eta^2). \tag{4.37}$$

If η is small we can neglect the quadratic term; moreover the Cauchy inequality

$$-\|\nabla f(x_n)\|\;\|v\| \leq \langle\nabla f(x_n), v\rangle \leq \|\nabla f(x_n)\|\;\|v\| \tag{4.38}$$

is reached on the left side only if v and the gradient are negative proportional, i.e. if $v = -\nabla f(x_n)/\|\nabla f(x_n)\|$. This shows that the gradient direction ensures the largest negative change in the function.

We have two remarks regarding the size of the step η, which is a trade-off between the margin of error and time effectiveness of a running application:

- if η is large, the algorithm stops too early, before reaching a good approximation of the minimum;

- if η is too small, the stopping order m is large and it might not be time effective in the case of a computer implementation.

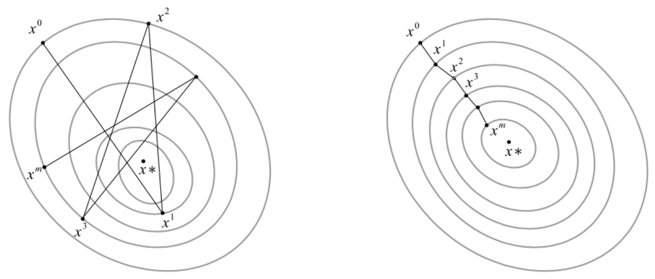


Figure 4.6: the polygonal line in the case η i s too large (left), or too small (right) [Cal20].

The drawback of this construction is that the sequence $(x_n)_{n\in\mathbb{N}}$ does not converge since $\|x_{n+1} - x_n\| = \eta$, so it is easy to miss the minimum x^*. To overcome this problem have been proposed modifications of GD with **adjustable** learning rate, in the sense that it becomes smaller as the function changes slower (when the gradient is small).

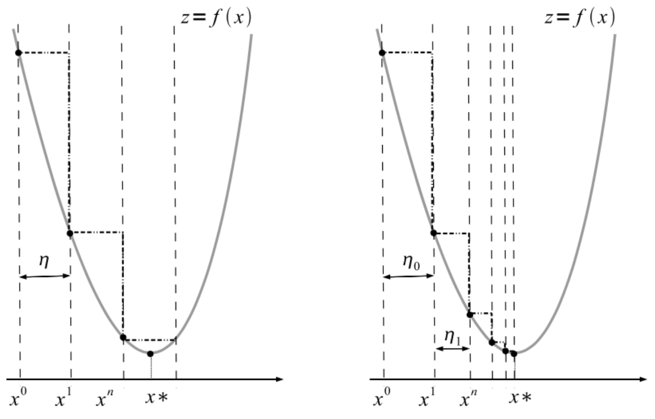


Figure 4.7: Constant learning rate (left), vs adjustable (right) [Cal20]

4.7.2 Batch training

When gradient descent method is used, each input time series T_i produces a gradient direction $\nabla L(T_i)$ (computed w.r.t. the parameters of the model). Citing the comparison made by Calin in [Cal20]: "a similar situation is when you are in a unknown city and ask for directions to people who may or may not give you the best direction; the ideal way to deal with this problem would be to ask all the people in the city and average the directions!". Similarly, averaging the gradients computed using different inputs helps to find the best direction towards the optimal parameters (note that this corresponds to differentiating the expected risk I_D, i.e. the average loss w.r.t. the empirical measure). If considering N input time series, the average direction is

$$\frac{1}{N}\sum_{i=1}^{N}\nabla L(T_i) \tag{4.39}$$

which by Central Limit Theorem has a variance N times smaller than a single raw direction. Since using full input data to compute the gradient of the empirical risk function can be time expensive, usually a trade-off between accuracy and computational time is made by sampling a random mini-batch, i.e. a batch of smaller size, from the dataset and computing an estimation of the gradient from it. This procedure is called (mini-batch) **stochastic gradient descent**.

Chapter 5

Methods for Shapelets Extraction

In this chapter we will present in details the algorithms we used in our experiments for extracting shapelets in an unsupervised way for anomaly detection. In describing such algorithms we will use pseudo-code conventions following Cormen's book [CLRS22].

5.1 Anomaly Detection Dataset

5.1.1 Dataset construction

Following [Agg13], [BKS$^+$19], [YN18] we made the assumption that anomaly detection is a highly imbalanced clustering task, i.e. we can assume that the majority of our training data consists of data from the normal class ($+1$), while a small (unknown) percentage of them is an anomaly (-1).

Following the approach suggested in [BKS$^+$19], we chose some datasets from the University of California, Riverside, (UCR) Time Series Classification Archive, originally collected in [DBK$^+$19] for classification purposes, and adapted to our problem in the following way:

- first, the original train and test set are merged together;
- the normal class label is chosen to be either the visually most distinguishable from the others either the one with the highest number of time series;

- a certain percentage of normal time series is randomly selected into the training set;
- if N_1 is the number of normal time series in the training set, a number $\alpha \cdot N_1$ of anomalous time series is selected randomly from the other classes into the train set. We will refer to α as **anomaly ratio** while the proportion of anomalies w.r.t. the total number of time series N will be denoted by ν.
- A proportion of the remaining time series is randomly selected in the validation set;
- finally, the remaining series will form the test set.

Since we want our methods to work in an unsupervised fashion, the correct proportion of anomalies α will not be used to as a known parameter of our models. In Section 6.3.4 we will discuss its impact on the results.

5.1.2 Scaling

In order to make them comparable, the time series in the dataset should be scaled; as suggested in [TFV$^+$20], a good procedure is to use the Min-Max normalization, which transforms a time series $T = \{t_i\}_{i=1}^{Q}$, $t_i \in \mathbb{R}$ as follows:

$$\bar{t_i} = \frac{t_i - m_T}{M_T - m_T}, \tag{5.1}$$

where $m_T = \min_j t_j$, $M_T = \max_j t_j$. If the time series is multivariate with C channels, i.e. $t_i \in \mathbb{R}^C$, the scaling is done channel-wise. Since shapelets-based algorithm look for patterns in the shape of the data, normalization is an important step and should be done carefully, depending on the type of data. In the experiments in Chapter 6, we will always use MinMax normalization.

5.2 Searching Methods

In this section we will see two searching algorithms for extracting optimal shapelets from a dataset, the BruteForce Extractor (BF) and the Random Local Search (RLS) Extractor algorithms.

5.2.1 Brute Force algorithm

Let $D = \{T_i\}_{i=1}^{N}$ be a dataset of N time series, K and L integers representing number and length of the shapelets to be extracted and $reverse$ a boolean whose meaning will be explained later. Sometimes we will refer to K^* and L^* as the proportion of K and L w.r.t. the length of the time series Q: $K = K^*Q$ and $L = L^*Q$.

Algorithm 2 BruteForce Extractor

```
Require: $D$, $K$, $L$, $reverse$ (boolean)
Ensure: shapelets
  candidates $\leftarrow$ ExtractCandidates($D$, $L$)        // Extract all subsequences long $L$
  scores $\leftarrow$ ComputeScores($D$, candidates)
  candidates $\leftarrow$ Sort(candidates, scores, $reverse$)
  // The first one is selected as a shapelet
  shapelets $\leftarrow$ [candidates[0]]        // Delete the first element in the order
  candidates.delete(1)
  while shapelets.length $\neq K$ do
    S $\leftarrow$ candidates[1]
    if Filter($S$, shapelets) then
      candidates.delete(1)
      pass
    else
      // If Filter is not satisfied, add the candidate to the shapelets array
      shapelets.append($S$)
      candidates.delete(1)
    end if
  end while
```

The procedure ExtractCandidates(D, L) stores all the subsequences of length L from the dataset D, together with their positions. The score of each candidate S is computed by the function ComputeScores in the following way:

$$score(S) = \sum_{i=1}^{N} d(T_i, S)^2, \tag{5.2}$$

where d is the l^2 discrepancy. The candidates are then sorted by the function Sort with respect to their scores in two possible ways:

- in **normal** order, from lowest to highest score ($reverse = False$). This enables to extract the shapelets which have least sum of discrepancies to each time series in the dataset, i.e. are the "closest" in similarity to the majority of them.

- in **reverse** order, from highest to lowest score ($reverse = True$). In such a way the shapelets extracted are the subsequences "most dissimilar" to the majority of the time series.

5.2.2 Candidates' filtering

If no constraints are imposed to the candidates to be selected as final shapelets, there is a risk that the extracted shapelets will be too similar to each other. The extraction of different shapelets is necessary in order to encode more information about the dataset in the transformed space; indeed suppose we extracted two shapelets which are very similar to each other, then the time series in the dataset will be projected on the main diagonal of the $\mathbb{R}^2$ plane, as it happens in Figure 5.1, since the discrepancy to each of the two shapelets will be nearly the same.

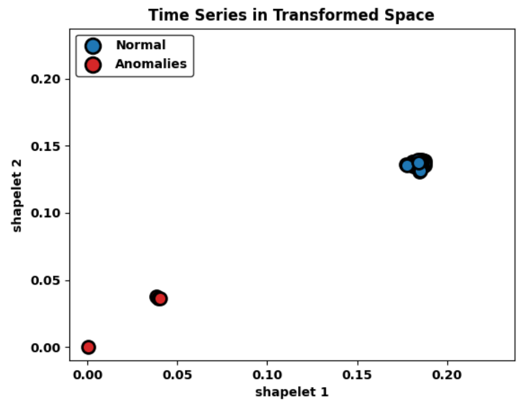


Figure 5.1: Trace Dataset (Section 6.1) transformed w.r.t. shapelets too similar to each other: the variance is only along the main diagonal.

For this, some constraints need to be introduced at the moment of selection of the shapelets; the function Filter in Algorithm 2 represents any filter explained in this Section and returns a boolean, $True$ if the filter condition is satisfied.

Definition 5.2.1. *(Distance filtering) Consider a set of discovered shapelets $S_1, ..., S_k$, $k > 0$ and a candidate shapelet S, all with the same length L.*

Eliminate S as a candidate if

$$\|S_i - S\|_2 \leq M \text{ for some } i = 1, ..., k. \tag{5.3}$$

where $M = 0.1 \cdot median(A)$, A set of the euclidean distances from S to a random subset of the subsequences with length L in the dataset.

The size of the subset A must be taken large enough in order to be representative of the entire set of subsequences; we set $|A| = 3000$ as the default value. Note that if shapelets with different lengths need to be extracted (from example in a range from L_{min} to L_{max}), distance filtering can be adapted using l^2 discrepancy instead of the euclidean distance.

The distance constraint allows to select shapelets with sufficient high euclidean distance to each other; its main disadvantage is that, if two shapelets are very similar to each other but are shifted from eah other, their euclidean distance will be large while their discrepancy values to a time series will be nearly identical and they will provide almost the same information in the transform domain. See Section 6.1.2 for an illustrative example. We introduced other contraints to overcome this issue.

Definition 5.2.2. *(Position filtering) Consider a set of discovered shapelets $S_1, ..., S_k$, $k > 0$ and a candidate shapelet S. Denote by p the starting position of S as a subsequence of the time series in the dataset from which it was extracted and p_i the starting position of the i-th shapelet. Given a threshold $T \in \mathbb{N}$, eliminate S as a candidate if*

$$|p - p_i| \leq T \text{ for some } i = 1, ..., k. \tag{5.4}$$

The position constraint is based on the hypothesis that subsequences with nearby starting positions are very likely to be very similar but shifted; such a constraint can be heavy since many candidates are discarded based only on their starting position. Moreover if the time series are not well aligned, a high scoring subsequence can be eliminated even if different from the already discovered shapelets. Therefore we introduce another measure of similarity between subsequences, which takes into account their shift.

Definition 5.2.3. *(Correlation filtering) Consider a set of discovered shapelets $S_1, ..., S_k$, $k > 0$ and a candidate shapelet S. Denote by ρ^i the cross-correlation coefficient (see Definition 4.6.3) between the candidate S and the i-th shapelet. Given a threshold $T \in]0, 1]$, eliminate S as a candidate if*

$$max_l \; \rho^i_l \geq T \text{ for some } i = 1, ..., k. \tag{5.5}$$

The correlation constraint has higher computational cost than the other two: the most efficient way to compute the correlation of two sequences long L is through the FastFourierTransform (FFT) algorithm which takes $\mathcal{O}(L \cdot log(L))$. Details can be found in [Wal17].

5.2.3 BF Complexity Analysis

For each subsequence, accounting to a total of $N \cdot (Q - L + 1)$, BF algorithm computes N times the discrepancy operation, one for each time series, which costs $\mathcal{O}(Q)$. The total complexity of the algorithm is therefore $\mathcal{O}(N^2 Q(Q - L + 1)) = \mathcal{O}(N^2 Q^2)$. The cost of the modified SVDD ($\mathcal{O}(N^{2+\varepsilon})$, see [CLS21]) or OCSVM ($\mathcal{O}(N^3)$) must be added to compute the boundary. The cost of the filtering procedure can be neglected.

5.2.4 RLS Extractor

In this Section we propose a variant of the RLS algorithm introduced in 4.4.2 adapted to the problem of unsupervised anomaly detection. Since RLS provides a great time saving, shapelets with varying lengths can be searched through the dataset. The algorithm selects randomly r subsequences, computes their scores and searches iteratively in the neighborhood of the best scoring ones.

Definition 5.2.4. *Let S be a candidate subsequence with length L and starting position j, S' be another subsequence with length L' and starting position i and $step > 0$ be an integer; S' is in the* ***neighborhood*** *of S if*

$$|j - i| \leq \varepsilon_0, \quad and \quad |L - L'| \leq \varepsilon_1 \cdot step; \tag{5.6}$$

ε_1 is therefore the maximum length difference using $step$ as a unit measure.

The parameters of Algorithm 3 are the following:

- D: dataset of N time series;
- K: number of shapelets to extract;
- L_{min}, $step$, n_{step}: define the lengths of the extracted subsequences;

$$L_{min};\ L_{min} + step;\ L_{min} + 2 \cdot step; ....;\ L_{min} + n_{step} \cdot step = L_{max}; \tag{5.7}$$

- r: integer, number of candidates randomly selected at the beginning;

- m: integer, number of best candidates to take in order to perform LocalSearch in their neighborhood;
- $\varepsilon = (\varepsilon_0, \varepsilon_1)$: tuple in $\mathbb{N} \times \mathbb{N}$ that define the neighborhood w.r.t. position, length as in equation 5.7;
- $\beta \in [0, 1]$: percentage of neighbors to discard, i.e. not to compute scores, each time a neighborhood is found (see Algorithm 4);
- $maxiter$: maximum number of neighborhood search iterations for each of the m best candidates (see Algorithm 4);
- $reverse$: boolean, whether to take the shapelets with max (True) or min (False) score.

Algorithm 3 RLS Extractor

```
Require: $D$, $K$, $L_{min}$, $step$, $n_{step}$, $r$, $m$, $\varepsilon$, $\beta$, $maxiter$, $reverse$
Ensure: shapelets
  // Extract all subsequences with lengths in 5.7
  total_candidates ← ExtractCandidates($D$, $L_{min}$, $step$, $n_{step}$)
  // Select randomly $r$ of them to compute scores
  random_candidates ← GetRandom(total_candidates, $r$)
  scores ← ComputeScores($D$, random_candidates)
  random_candidates ← Sort(random_candidates, scores, $reverse$)
  top_candidates ← GetTopCandidates(random_candidates, $m$, scores, $reverse$)
  neighbors ← LocalSearch(total_candidates, top_candidates, $\varepsilon$ $\beta$, $maxiter$)
  shapelets ← GetTopCandidates(random_candidates, $K$, scores, $reverse$)
```

The functions ExtractCandidates, ComputeScores and Sort work as described in BF Algorithm 2. GetRandom randomly takes r subsequences from total_candidates. The procedure GetTopCandidates selects the best scoring candidates as it is described in more detail in Algorithm 2. The shapelets' filtering process is as described in section 5.2.2; we omitted the thresholds parameters for position and correlation filtering to keep the notation tidy.

Algorithm 4 LocalSearch

```
Require: top_candidates, total_candidates, $\varepsilon$, $\beta$, $maxiter$
Ensure: neighbors
  neighbors ← []
  for $S$ in top_candidates do
    $iter \leftarrow 0$
    while $iter < maxiter$ do
      Γ ← GetNeighborhood($S$, total_candidates, $\beta$)
      scores ← ComputeScores(D, Γ)
      neighbors.append(Γ)
      if max(scores) < score(S) then
        break
      else
        // If a better scoring subsequence is found, search iteratively in its neigh-
        borhood
        S ← FindMaxScore(Γ, scores)
        iter ← iter + 1
      end if
    end while
  end for
```

The function GetNeighborhood finds all the subsequences that satisfy 5.6 and discards a percentage β of them. When a better scoring subsequence is found, the algorithm iteratively searches in its neighborhood until a maximum number of iterations $maxiter$ is reached.

The LocalSearch procedure returns the previous found top_candidates, together with their neighbors with computed scores; the function GetTop-Candidates is applied one more time to this new set of scored sequences, in order to find the optimal shapelets satisfying the desired constraints.

5.2.5 RLS Complexity Analysis

Define $M = maxiter$ and keep the notation of the previous Section. As seen in 5.2.3, the score computation costs $\mathcal{O}(N \cdot Q)$ for each candidate. In RLS are initially computed scores for the r randomly selected subsequences. Following, for the m best ones are computed the scores in the neighborhood, iteratively until at most M neighborhoods are explored. From the Definition 5.2.4 of neighborhood, each of the m best random candidates has at most $N \cdot (2\varepsilon_0 + 1) \cdot (2\varepsilon_1 + 1)$ neighbors; scores are calculated only for a fraction $(1 - \beta)$ of them. The total complexity is therefore:

$$\mathcal{O}\big(r \cdot N \cdot Q + m \cdot M \cdot (1-\beta) \cdot N \cdot (2\varepsilon_0 + 1)(2\varepsilon_1 + 1) \cdot N \cdot Q\big) = \mathcal{O}\big(M \cdot N^2 \cdot Q\big). \tag{5.8}$$

5.3 Learning

5.3.1 ADSL

We implemented Beggel's et al. ([BKS+19]) ADSL algorithm introduced in Chapter 4 using python's framework *Pytorch* [PGM+19].

The parameters of the algorithm are:

- D, dataset of N time series, K, L number and length of the desired shapelets respectively
- n_{epochs} number of **epochs**, which specifies the number of complete passes of the entire training dataset passing through learning process;
- $batch\_size$ is the number of time series used for an update of the gradient. In experiments we set $batch\_size$ equal to N by default;
- γ, learning rate of Gradient Descent;
- ν expected proportion of anomalies w.r.t. the total number N of time series;
- λ penalty parameter for the shapelet similarity loss.

With SVDD(X, ν) we will refer to the modified SVDD algorithm with center in the origin, trained on the transformed dataset X with cost parameter $C = 1/(\nu \cdot N)$.

Algorithm 5 ADSL

```
Require: $D$, $K$, $L$, $n_{epochs}$, $batch\_size$, $\gamma$, $\nu$, $\lambda$
Ensure: $\mathbf{S}$ set of shapelets.
  $\mathbf{S} \leftarrow$ InitializeShapelets($D$, $K$, $L$)
  for $epoch$ in $1, ..., n_{epochs}$ do
    $X \leftarrow \sigma_{\mathbf{S}}(D)$
    $R \leftarrow$ SVDD($X$, $\nu$).radius
    $D_{batch} \leftarrow$ GetBatch($D$, $batch\_size$)
    loss $\leftarrow$ ComputeLoss($D_{batch}$, $\mathbf{S}$, $R$, $\lambda$)
    $\mathbf{S} \leftarrow$ Update($\mathbf{S}$, $\gamma$)
  end for
```

The procedure GetBatch takes a random batch of data from D with size $batch\_size$. The procedure ComputeLoss computes the loss in Equation 4.33; Update performs a iteration of gradient descent as described in Section 4.7; InitializeShapelets is explained in the next Section.

5.3.2 ADSL Complexity Analysis

To calculate the loss in Equation (4.28), in each epoch (for a total of n_{epochs}), ADSL computes $\mathcal{O}(N \cdot Q)$ discrepancies for each of the K shapelets; the update of the radius R costs $\mathcal{O}(N^{2+\varepsilon})$ ([CLS21]). The cost for computing the shapelets' similarity penalty term in Equation (4.33) can be neglected. Note also that ADSL does not require to store the large number of candidate subsequences, while the search-based algorithms do.

5.3.3 Initialization and K-means algorithm

We adopted Gradient Descent to optimize the non convex loss in Equation (4.33) w.r.t. the shapelets' parameters **S**; since Gradient Descent optimization does not guarantee the discovery of global optima, it requires a good initialization of the parameters. Indeed, if the initialization starts the learning around a region where the global optimum is located, then the gradient can update the parameters to the exact location of the optimum. In [GSWST14], Grabocka et al. propose to use the K-means centroids of all segments as initial values for the shapelets , since centroids represent typical patterns of the data.

We now introduce the K-means algorithm, which is a fast method for finding clusters and cluster centers in a set of M unlabeled time series long L. Let $K > 0$ be a fixed number of clusters and ϕ a cluster assignment function $\phi(T_i) = k$, $k = 1, ..., K$. Define the **total cluster variance** as

$$W(\phi) = \frac{1}{2}\sum_{k=1}^{K} \sum_{\phi(T_i)=k} \sum_{\phi(T_{i'})=k} \|T_i - T_{i'}\|_2^2 = \sum_{k=1}^{K} M_k \sum_{\phi(T_i)=k} \left\|T_i - \bar{T}_k\right\|_2^2 , \tag{5.9}$$

where $\bar{T}^k = \{\bar{t}_1^k, ..., \bar{t}_L^k\}$ is the mean vector associated to the k-th cluster, also called **centroid**, and $M_k = \sum_{i=1}^{M} \mathbb{1}_k(\phi(T_i))$, with $\mathbb{1}_k$ indicator function of the set $\{k\}$. The goal is to find a clustering function ϕ^* that minimizes $W(\phi)$:

$$\phi^* = \arg\min_{\phi} W(\phi); \tag{5.10}$$

ϕ^* assigns the M observations to the K clusters in such a way that within each cluster the average dissimilarity of the observations from the cluster mean, as defined by the points in that cluster, is minimized. Noting that for any set of observations S

$$\bar{x}_S = \arg\min_{m} \sum_{x \in S} \|x - m\|^2 , \tag{5.11}$$

we can obtain ϕ^* by solving the enlarged optimization problem:

$$\arg\min_{\phi,\{m_k\}_{i=1}^{K}} \sum_{k=1}^{K} M_k \sum_{\phi(T_i)=k} \|T_i - m_k\|_2^2 . \tag{5.12}$$

In ADSL algorithm 5, shapelets need to be initialized as sequences of length L. K-means is therefore applied to a sample of subsequences with length L in the dataset.

Algorithm 6 Time Series K-means

```
Require: $K$ number of clusters, $D$ dataset of $N$ time series, $L$ subsequence length,
    $size$ sample size
Ensure: $K$ centroids
    1. Extract a sample $size$ of subsequences long $L$ from $D$ and initialize randomly
    a cluster assignment $\phi$.
    2. For a given cluster assignment $\phi$, the total cluster variance 5.12 is minimized
    with respect to $\{m_1, ..., m_K\}$ yielding the means of the currently assigned clus-
    ters 5.11.
    3. Given a current set of means $\{m_1, ..., m_K\}$, 5.12 is minimized by assigning
    each time series subsequence to the closest (current) cluster mean:
        $\phi(T_i) = argmin_{k=1,...,K} \|T_i - m_k\|$    (5.13)
    4. Steps 2. and 3. are iterated until the assignments do not change or a maximum
    number of iterations is reached.
```

Each of steps 2. and 3. reduces the value of the criterion 5.12, so that convergence is assured, but the result may represent a suboptimal local minimum.

Chapter 6

Experiments

In this Chapter we will expose some results obtained by applying the shapelets-based algorithms presented in Chapter 5. In the shapelets-transform space, either a OCSVM or a modified SVDD are always applied to find the decision boundary for anomaly detection. In order not to sacrifice the interpretability provided by shapelets, we limit ourselves to the use of linear kernels. If not specified, the true anomaly proportion ν will be used to set the cost parameter $C = 1/(N \cdot \nu)$, which controls the tightness of the boundary, as explained in Section 3.4.3. Since in real applications ν is not known, we will discuss its tuning in Section 6.3.4.

The code for the replicability of these experiments is available at the GitHub repository ”Shapelets” [git].

6.1 Trace Dataset

6.1.1 Description

Trace is 4-class dataset from the application domain of the process industry (Figure 6.1); it is a synthetic dataset designed to simulate instrumentation failures in nuclear power plants and contains 200 instances, 50 for each class, with length of $Q = 275$ data points.

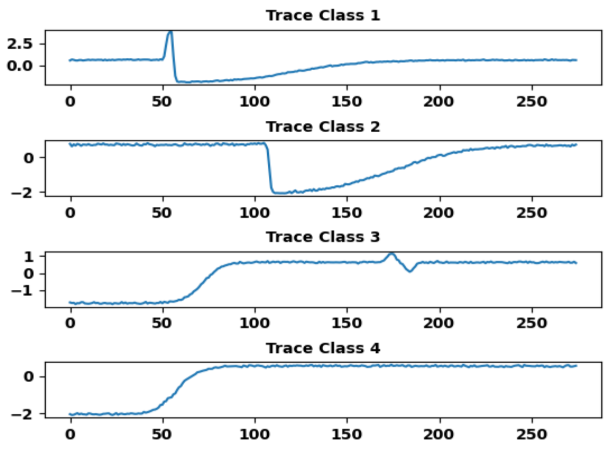


Figure 6.1: A sample time series from each class of Trace.

6.1.2 Shapelets searching with different filtering methods

In this simple experiment, we will show how the different filtering methods presented in Section 5.2.2 behave. From Trace dataset, we selected class 4 as normal class and class 3 as anomaly (the others are discarded) with anomaly ratio $\alpha = 0.1$; the training set (Figure 6.2) is made of $N = 44$ time series, 40 normal and 4 anomalies. A MinMax scaler is applied to each time series to normalize the dataset. We set $K = 0.02 \cdot Q = 6$ and $L = 0.2 \cdot Q = 55$ and extracted the shapelets using BF algorithm setting $reverse = True$, in order to obtain shapelets representing anomalies.

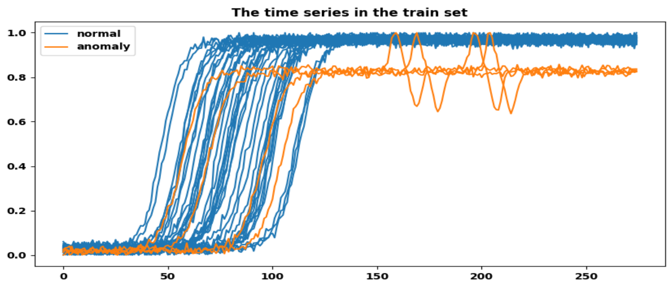
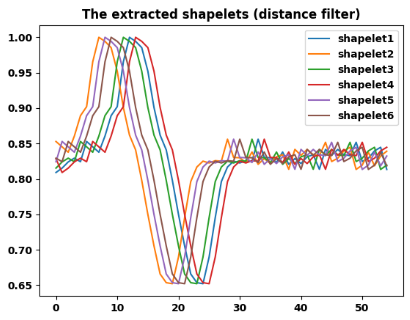
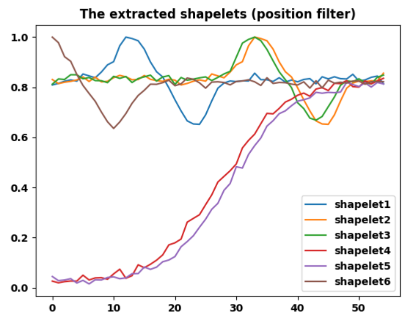


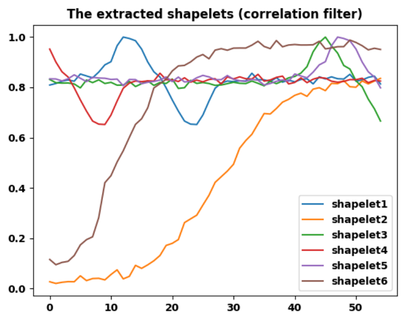
Figure 6.2: Train set of time series in class 3 (normal) and 4 (anomaly).



(a) Only distance filter



(b) Position and distance filters.



(c) Correlation and distance filters.

Figure 6.3: Shapelets extracted with different filtering methods.

In figure 6.3 are plotted the shapelets extracted with different filtering methods. In Figure 6.3a shapelets are taken by applying only a distance constraint (Definition 5.2.1) calculated over a sample of 3000 subsequences: they are the same sequence, but shifted by few positions; they are too similar to each other and consequently they don't provide more information than a single shapelet, as explained in Section 5.2.2. In Figure 6.3b shapelets are filtered using the distance constraint and a positional constraint (Definition 5.2.2) with threshold of 20 time steps. Finally, in Figure 6.3c they are filtered by the distance and the correlation contraint (Definition 5.2.3) with threshold of 0.8. Despite being the most expensive, correlation filtering ensures the most diverse shapelets.

We selected only the first shapelet, which is the same independenly on the filtering method used; in the transformed space a OCSVM is used for learning a boundary for anomalies, reaching a AUC score of 1.0 in the test set.

In Figure 6.4 the discrepancy plots from the extracted shapelet and two test time series are shown: the distance-based extraction of the shapelet makes the final decision of the algorithm interpretable and visually intuitive!

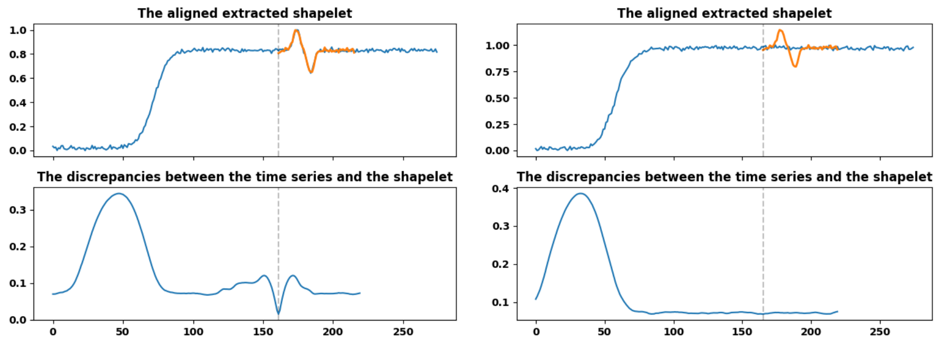


Figure 6.4: Discrepancy plots of the extracted shapelet from an anomalous test series (left) and a normal one (right).

6.2 Coffee Dataset

6.2.1 Description

Coffee is a dataset from the UEA-UCR Archive [DBK$^+$19]. It is a two class problem (Figure 6.5) to distinguish between Robusta and Aribica coffee beans' spectrographs, i.e. their electromagnetic spectra. Further information can be found in the original paper [BKW96]. The entire dataset Coffee is composed by $N = 56$ time series, 29 from Robusta class and 27 from Arabica, with length $Q = 286$.

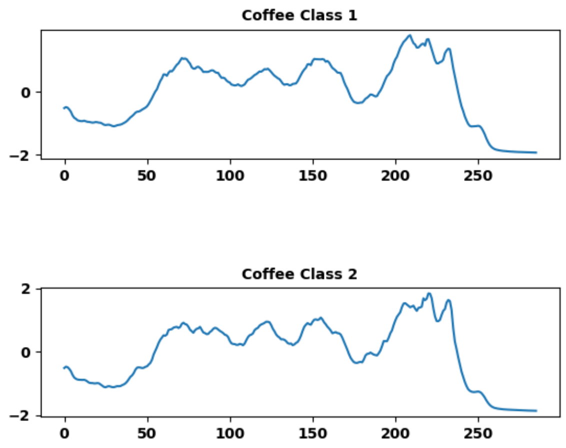


Figure 6.5: A sample time series from each class of Coffee.

6.2.2 Comparison between BF and RLS shapelets

We constructed the training and test datasets as explained in Section 5.1, setting the normal class as 1, Robusta beans, using a proportion $\alpha = 5\%$ of anomalies from class 2, Arabica. The training set (Figure 6.6) is composed by 24 time series, containing only 1 anomaly, while the test set is composed by 32 series, containing 26 anomalies. A MinMax scaler was applied to normalize the time series.

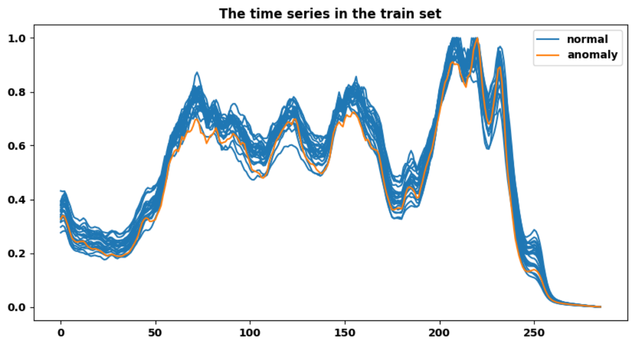


Figure 6.6: Training set.

Using RLS algorithm, we extracted $K = 0.02 \cdot Q = 6$ shapelets, with length $L = L_{min} = L_{max} = 0.2 \cdot Q = 57$, in normal score order ($reverse = False$), filtered with a correlation threshold of 0.8 and with a distance contraint calculated over a sample of 3000 subsequences. The other parameters were set as follows: $r = 500$, $m = 10$, $\varepsilon = (2, 1)$, $\beta = 0.7$ and $maxiter = 4$. We report the average results obtained over 10 runs of RLS algorithm. The candidates whose scores were calculated are on average 1262, with a standard deviation of 92, out of $N \cdot (Q - L + 1) = 24 \cdot (286 - 57 + 1) = 5520$ total number of subsequences, with a significant saving in computational time. RLS works well in this case, since it is based on a search by position and the training time series in Figure 6.6 are well aligned.

In Figure 6.7 are compared the shapelets extracted in a run of RLS to the ones obtained through the exhaustive search of BF algorithm:

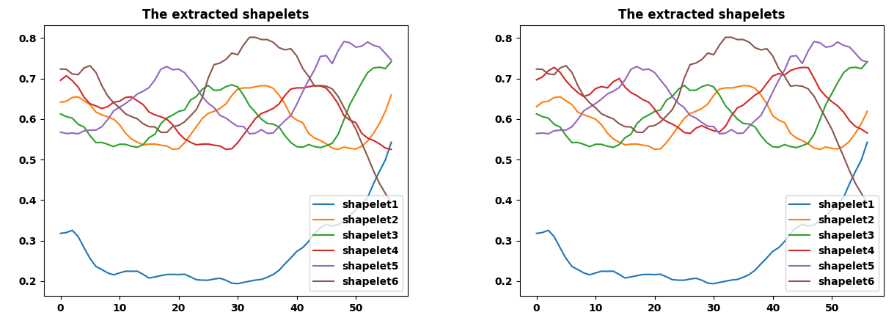


Figure 6.7: Shapelets extracted with BF (left) and in one run of RLS (right) algorithms.

A modified SVDD with center in the origin was trained in the transformed space, obtaining similar AUC score (Table 6.1) on the test set for both algorithms, proving that in many cases it is not necessary to extract the optimal shapelets; a sub-optimal result, in the sense that it is obtained searching only in a subset of all possible candidates, can have a similar impact on the final results.

	BF	RLS (mean)	RLS (std)
AUC	0.923	**0.929**	0.01
$F1$	**0.772**	0.73	0.03

Table 6.1: Results of SVDD predictions.

Let **S** be the set of shapelets extracted by BF; we now examine the values of the shapelet transform $\sigma_{\mathbf{S}} \in \mathbb{R}^6$ in order to find which shapelets contribute the most to the decision of the SVDD. Figure 6.8 displays the difference between the mean discrepancy values of the predicted anomalous and normal time series in the test set w.r.t. **S**, i.e.

$$mean(\{\sigma_{\mathbf{S}}(T_i);\ \ \hat{y}_i = -1\}) - mean(\{\sigma_{\mathbf{S}}(T_j);\ \ \hat{y}_j = 1\}) \tag{6.1}$$

where $\hat{y}_i$ is the predicted label of T_i, i-th test time series.

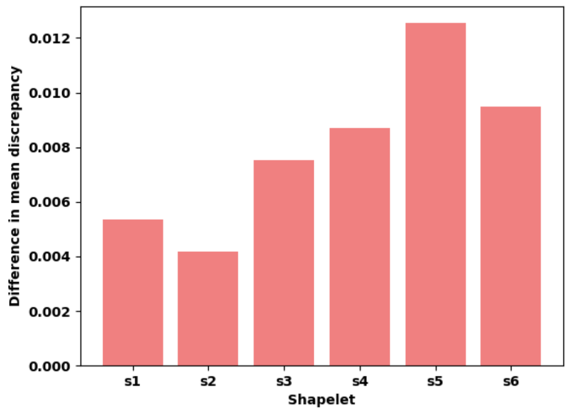


Figure 6.8: Mean discrepancy values (test set) from each of BF shapelets.

We note that the mean discrepancy difference is significantly higher for the 5-th shapelet (Figure 6.8), which is present also in RLS shapelets in Figure 6.7. When shapelets are extracted in normal order and labeles are available, large values in mean discrepancy difference can help to find the shapelets that contribute the most in the decision of the SVDD, leading to more interpretable results (Figure 6.9).

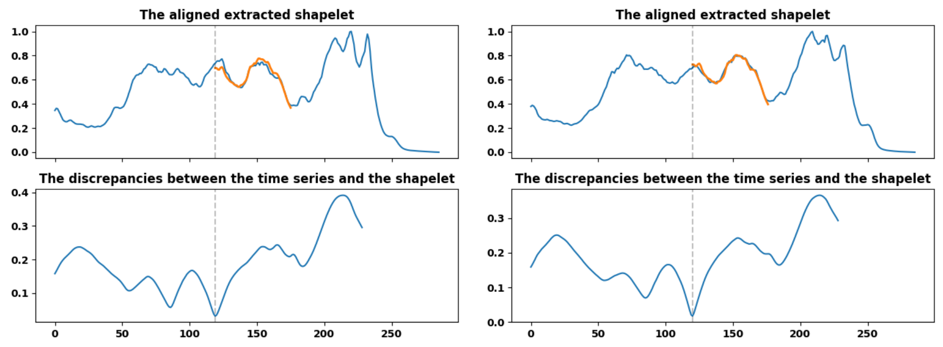


Figure 6.9: Discrepancy plots of shapelet5 from an anomalous test series (left) and a normal one (right).

This example shows that the best scoring shapelets are not always the ones that matter in the decision of the SVDD; indeed the score can be low for subsequences that appear both in normal and anomalous series, while the ultimate goal is to find shapelets that distinguish the two classes. Extracting many shapelets can be useful in such a case; in the next Section we will discuss how the number of extracted shapelets affects the results.

6.2.3 Number and length of shapelets

Table 6.2 shows the AUC scores of the BF algorithm with varying parameters K and L, while keeping the others as in previous Section.

L/K	2	4	6	8	10	15
22	0.192	0.397	0.333	0.660	0.827	0.819
42	0.884	0.903	0.923	0.929	0.923	0.933
57	0.884	0.904	0.923	0.923	0.923	0.903
62	0.872	0.929	**0.942**	0.923	0.936	**0.942**

Table 6.2: Results of SVDD predictions.

We can conclude that generally, the AUC score is improved by taking more and longer shapelets, while trivially interpretability becomes more and more difficult.

6.3 GunPoint Dataset

6.3.1 Description

Gunpoint is a dataset from the UEA-UCR Archive [DBK$^+$19]; it is divided into two classes (Figure 6.11), Gun-Draw (class 1) and Point (class 2). They represent the X-axis coordinate of an actor's hand's cantroid while drawing a replicate gun (Class 1) or just pointing their fingers (Class 2). See [DBK$^+$19] for a more detailed description. The entire dataset is composed by 200 time series, 100 from class 1 and 100 from class 2, with length $Q = 150$.

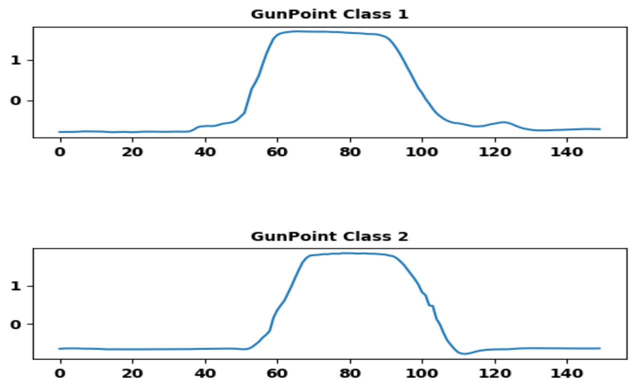


Figure 6.10: A sample time series in each class of GunPoint.

Training (Figure 6.11), validation and test set are constructed as explained in Section 5.1, using a ratio of anomalies $\alpha = 0.05$, a training proportion of 0.8 and a validation proportion of 0.3 (Table 6.3). A MinMax Scaler is applied to each time series.

	Train	Validation	Test
Class 1	80	6	14
Class -1	4	29	67

Table 6.3: Statistics about ECG dataset partition into train, validation and test set.

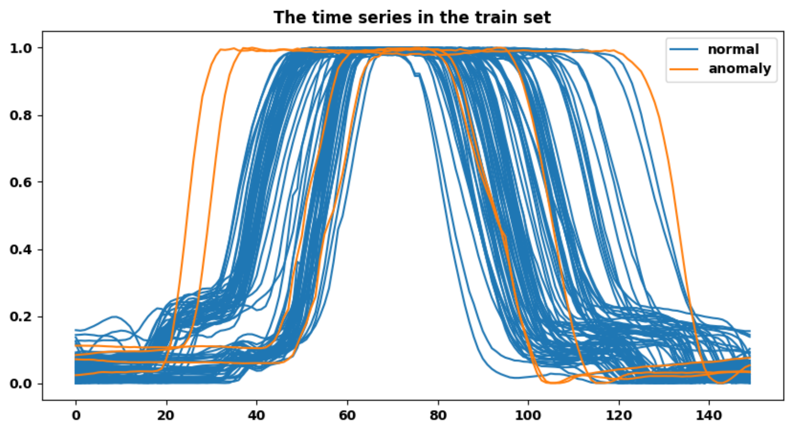
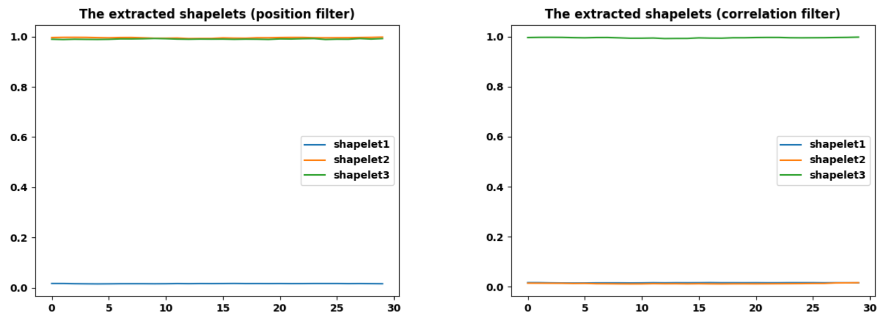


Figure 6.11: Training set of time series in class 1 (blue) and 2 (orange).

6.3.2 BF algorithm

Figure 6.12 shows the shapelets in output using BF algorithm, setting $K = 0.02 \cdot Q = 3$, $L = 0.2 \cdot Q = 30$ and $reverse = False$.



(a) Shapelets filtered by distance and position threshold=10.

(b) Shapelets filtered by distance and correlation threshold=0.8.

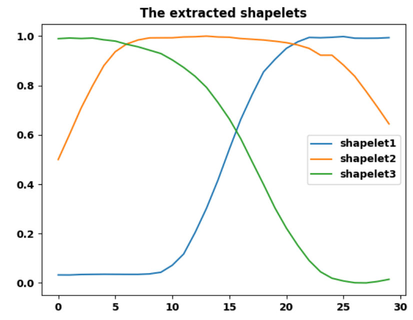
Figure 6.12: Shapelets extracted by BF algorithm.

The BF algorithm is not able to extract heterogeneous shapelets; the flat subsequences are the best scoring ones and it is hard to filter them for many reasons:

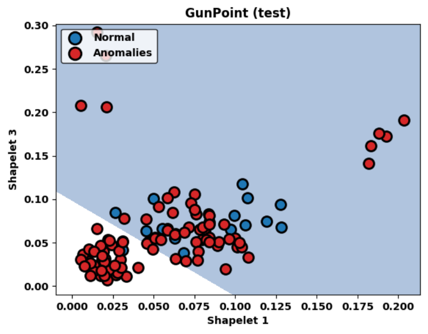
- the flat shapelets are not correlated;
- the time series are not aligned, therefore the position constraint does not help to diversify the shapelets extracted. The initial positions of the shapelets in Figure 6.12b are 0, 57, 90, 70, 10, 120, reflecting the misalignment of the time series;
- the distance constraint is based on a threshold calculated as the median of the distances from a sample of subsequences; since the flat subsequences are the most, such a threshold is too low.

The shapelets have no discriminating power, leading to poor performance of the SVDD in the transformed space ($AUC = 0.454$).

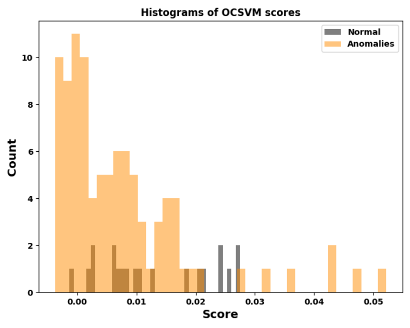
On the contrary, the shapelets extracted in reverse order, which aim to have low discrepancies with anomalies, give better results ($AUC = 0.71$, Figure 6.13):



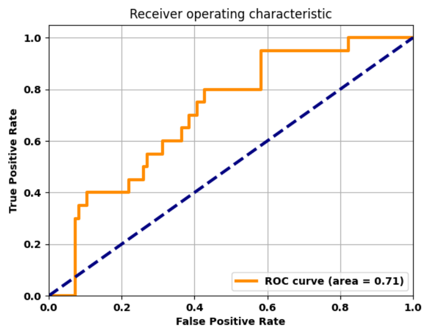
(a) Reverse order shapelets, filtered by distance and correlation threshold=0.8



(b) OCSVM boundary in transformed space w.r.t. shapelets 1 and 3.



(c) Distribution of OCSVM scores, i.e. signed distance to the hyperplane.



(d) ROC curve and AUC score.

Figure 6.13: Results for BF algorthm.

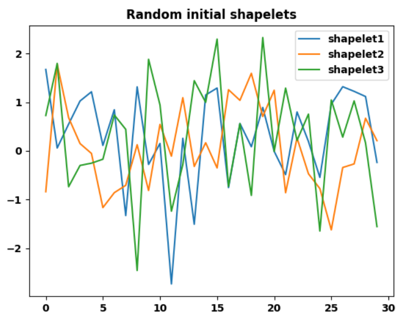
To make the boundary visually clearer, we trained a OCSVM using only shapelets 1 and 3 and plotted the test data and the learned description of normal data (light blue region) in Figure 6.13b. We chose shapelet 3 instead of shapelet 2 because the difference in (mean) discrepancy (Equation (6.1)) between the anomalous and normal class in the test set was larger for that shapelet. We see from the results that the shapelets are very close in discrepancy to many anomalous observations, but there is not a clear separation between the two classes.

6.3.3 ADSL algorithm

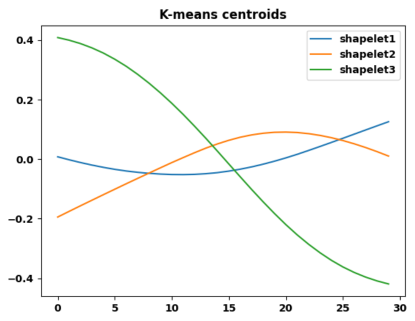
We trained the ADSL algorithm on the same dataset. The initial values of the shapelets' parameters are initialized in two ways (see Section 5.3.3):

- **randomly**, each value is taken independently as sample from the Standard Gaussian distribution $N(0, 1)$, Figure 6.14a;

- as centroids obtained from K**-means** algorithm, centered in order to have mean 0, Figure 6.14b.



(a) Random initialization.



(b) K-means clusters' centers.

Figure 6.14: ADSL shapelets' initialization.

With random initialized shapelets, we set the following parameters: learning rate $\gamma = 10^{-1}$, $n_{epochs} = 150$, $\lambda = 0.5$. With K-means initialization: $n_{epochs} = 15$, $\gamma = 10^{-1}$ (first 5 epochs) and $\gamma = 10^{-2}$ (last 5 epochs), $\lambda = 0.5$.

We repeated the random initialization for 10 times and report the mean AUC score obtained on the validation set, which is 0.856 with a standard deviation 0.085. The AUC for K-means initialized shapelets is 0.978. In Figure 6.15 are plotted the shapelets learned after a sample run with random initialization and after K-means initialization.

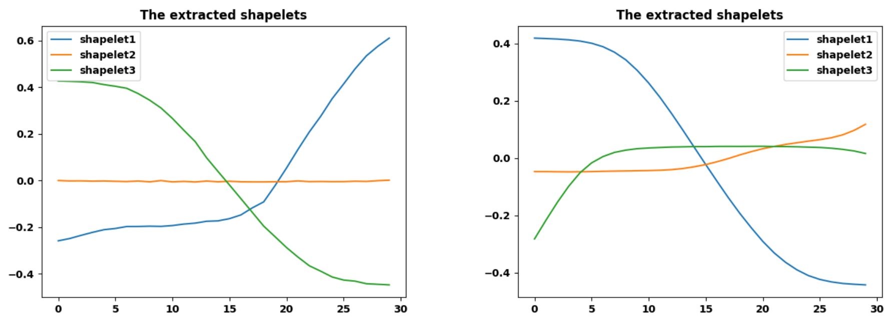


Figure 6.15: Shapelets after training with random (left) and K-means (right) initializations.

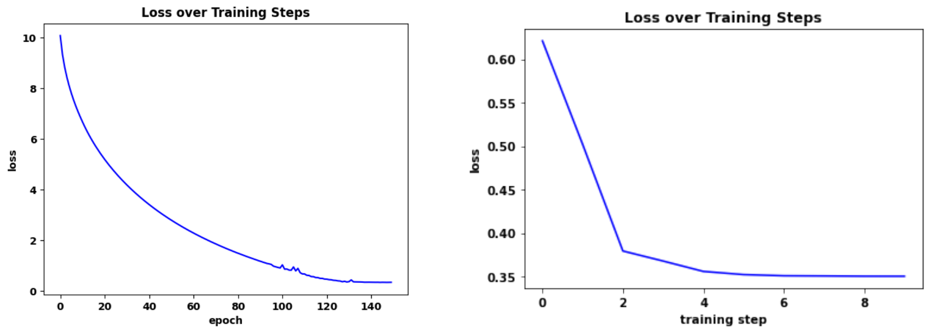


Figure 6.16: Loss decay with training steps; random (left) and K-means (right) initializations

.

Clearly initialization via K-means clusters' centers provides better initial parameters since:

- it ensures earlier convergence in a less number of epochs (Figure 6.16);
- cluster's centers represent patterns in the training data, which belong mainly to the normal class.

6.3.4 Hyperparameter ν

In order to train both the OCSVM and the SVDD we set by default the cost parameter according to the actual proportion of anomalies, $C = 1/(\nu N)$. By ν-property (Theorem 3.4.2), ν is an upper bound for the fraction of outliers identified by the algorithm. In real world situations, the true parameter ν is often not known and only an estimate of it can be given; moreover the ν-property suggests that ν should be overestimated. In this section we will discuss the influence of the parameter ν in the ADSL algorithm. Note that in BF algorithm it is used only to learn the decision boundary so the AUC score does not change with varying ν, because it already takes into account different decision thresholds. In ADSL, a SVDD is used to calculate the radius of the hypersphere boundary after each epoch, and its value affect the loss function 4.28; therfore we expect (small) changes in AUC.

Table 6.4 shows the $F1$ and AUC scores of ADSL algorithm with varying ν values. Shapelets are initialized using K-means clusters' centers and the other parameters are set as in the previous Section.

ν	0.01	0.03	0.05	0.1	0.15	0.20	0.25	0.5
F1	0.0	0.474	0.868	0.972	0.972	**0.983**	0.951	0.935
AUC	0.960	0.960	0.960	0.960	0.960	0.960	0.960	0.960

Table 6.4: Results of SVDD predictions on validation set.

As expected, overestimating the parameter ν results in better $F1$ score, while the AUC does not change. Since the number of train series N is small, a vary large value $\nu = 0.20$ gives the best results on the validation set. We tested the model using an SVDD trained in the three dimensional shapelet transform space with $\nu = 0.2$ on the test set, obtaining a $F1$ score of 0.971 and a AUC score of 0.978 (Figure 6.17). Figure 6.18 shows the resulting confusion matrix.

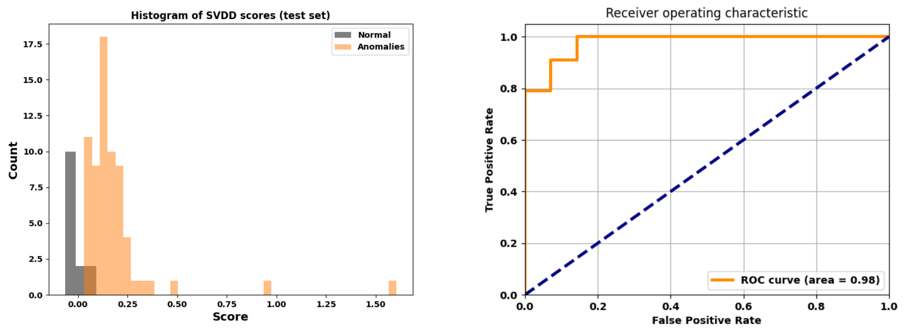


Figure 6.17: Histogram of test SVDD scores (left) and ROC curve (right).

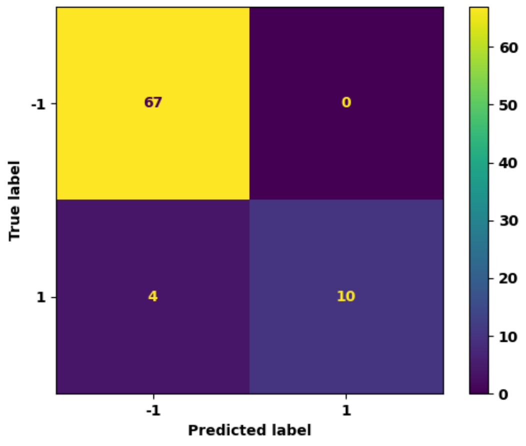
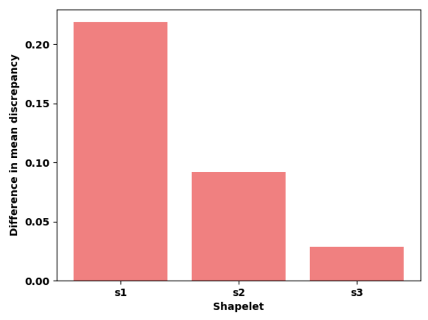
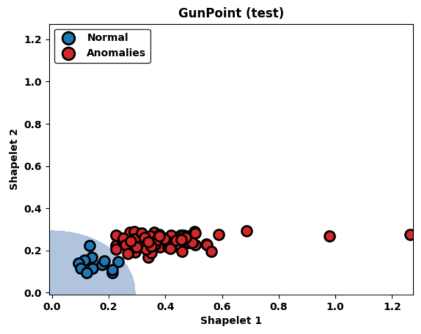


Figure 6.18: Confusion matrix $\nu = 0.2$.

Figure 6.19a plots the difference in mean discrepancy (Equation (6.1)) for each shapelet. For visual purposes, we selected the two most meaningful shapelets according to this score (the 1st and the 2nd) and we projected train and test data into 2 dimensions. A SVDD is trained again in such a space (Figure 6.19b).



(a) Mean discrepancy difference.



(b) SVDD boundary in 2D.

Figure 6.19: Mean discrepancy difference histograms (left) and SVDD boundary using the two most representative shapelets (right).

6.3.5 Comparison between learned and extracted shapelets

The results of the algorithms on the test set are reported in Table 6.5:

	BF $(reverse = F)$	BF $(reverse = T)$	ADSL (random)	ADSL (K-means)
AUC	0.454	0.708	0.856	**0.978**

Table 6.5: Test results for the different algorithms

This is an extreme case: while the BF algorithm with $reverse = False$ is unable to find heterogeneous shapelets and becomes stuck in a single flat shape, ADSL is able to learn smooth shapelets that resemble the characteristic shapes of the training data, which are mostly from the normal class. The learning approach permits to find sequences that capture average shapes in the data, without being forced to be any real subsequence but at the same time being close to them. This happens because the loss function in Equation (4.28) is discrepancy-based: in order to deal with the unsupervised nature of the problem, it penalizes large discrepancies from the training time series. On the contrary, in a supervised setting, the loss as in Equation (4.28), penalizes errors in prediction: the consequent learned shapelets can be far from real subsequences in order to minimize the prediction-based loss. See [KML20] for a more datailed discussion about learned shapelets' interpretability.

Eventually it has to be noted the similarity between the shapelets learned by ADSL in Figure 6.15 and the shapelets extracted by BF in reverse order in Figure 6.13a: even if they have different goals (BF aims to learn shapelets close, in discrepancy, to anomalies) the two algorithms capture the same 'shapes' that make the two classes different. ADSL performs better, thanks to the more flexibility of the learning approach.

6.4 Multivariate Dataset: PenDigits

6.4.1 Description

PenDigits is a dataset from the University of East Anglia (UEA) archive [BDL+18]; it is a multivariate dataset, each time series having $Q = 8$ points in $\mathbb{R}^2$, representing the coordinates of a pen drawing a digit from 0 to 9, traced across a digital screen (Figure 6.20). The entire dataset is composed by 10992 time series, more or less 1000 for each class.

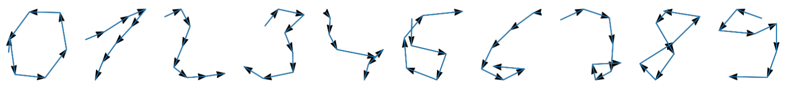


Figure 6.20: A sample time series for each class of PenDigits. The arrows indicate the time direction.

6.4.2 Distinguish between a 5 and a 6

We construct the anomaly detection dataset from PenDigits by taking only class 5 (normal) and 6 (anomaly). A ratio of anomalies $\alpha = 5\%$ is set in the training set: the resulting anomaly detection dataset is composed by 886 train time series, containing 42 anomalies, and 1225 test, containing 1014 anomalies. The dataset is then normalized using MinMax scaling. We trained ADSL algorithm with K-mean initialization for 200 epochs, setting length and number of shapelets $L = 4$ and $K = 1$, the learning rate $\eta = 0.01$ and the SVDD cost parameter C according to the actual proportion of anomalies. The resulting SVDD prediction of the test set gives the following scores: $AUC = 0.993$, $F1 = 0.949$.

In Figure 6.21 we plot two test time series and the extracted shapelet; the time dependence between the two-dimensional points is represented by arrows. As it is described by Kidger et al. in [KML20], who employ a supervised algorithm, the learned shapelet capture the difference between a 5 and a 6, which stands in their direction.

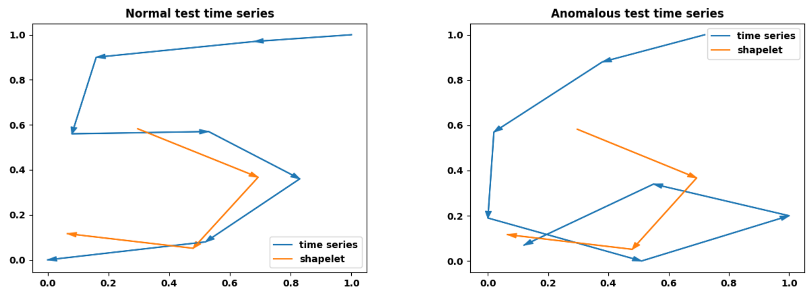


Figure 6.21: The shapelet learned plotted against a normal (left) and an anomalous (right) test series.

In order to demonstrate the robustness of the ADSL algorithm to the presence of anomalies in the training set, we costructed the training set using two extreme values for α, $\alpha = 0$, without anomalies, and $\alpha = 0.15$, a large ratio of anomalies. The true value for the proportion of anomalies ν is used to set the cost parameter C. The results are in Table 6.6: it works best when no anomalies are present in the training set, but it performs similarly even when it is contaminated by several anomalies.

α	0	0.05	0.15
$F1$	**0.967**	0.949	0.966
AUC	**0.998**	0.993	0.997

Table 6.6: Results of SVDD test predictions for different values of α.

6.5 ECG Dataset

6.5.1 Description

The ECG dataset [ecg] is made of complete ECG of patients with $Q = 140$ data points, which can be normal (class 1, 2919 time series) or anomalous (class -1, 2079 time series); a sample from each class is illustrated in Figure 6.22. It is used as a benchmark dataset in various notebooks for testing anomaly detection algorithms; in particular Autoencoder-based algorithms (see [not]) perform well on ECG reaching an AUC score 0.949.

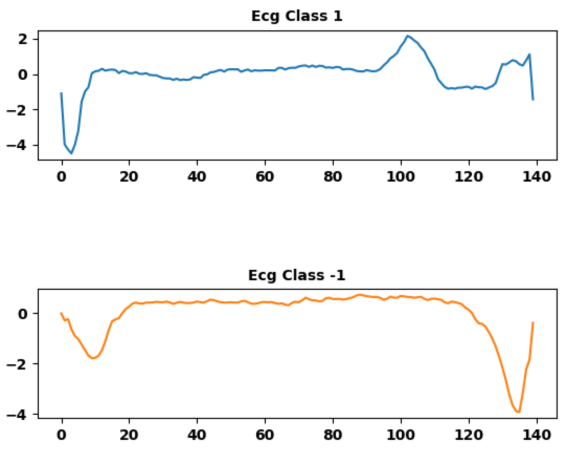
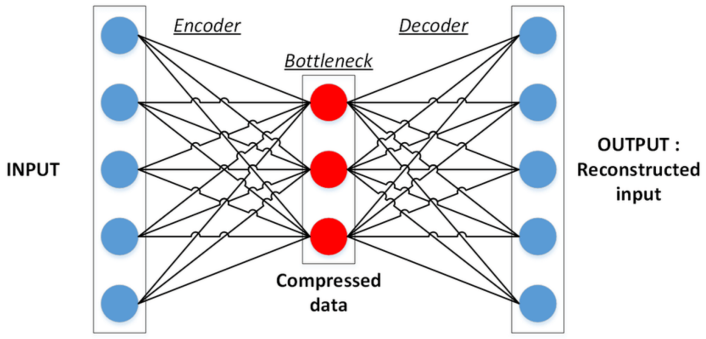
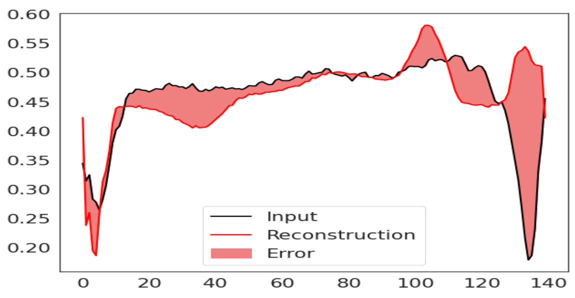


Figure 6.22: A sample from each class of ECG dataset.

An Autoencoder [LCLL14] is obtained by composing two neural networks ([Cal20]), an encoder and a decoder (Figure 6.23a). The final output has the same dimension as the input, while the ”bottleneck”, i.e. the output of the encoder, called *latent space*, has lower dimension. Such a model is trained in order to get an output that resembles the original input. In order to detect anomalies, an autoencoder is trained only on normal data; when an anomalous time series is given in input, the reconstruction error, that is the error between the original and the output signal, should be large: a threshold is computed in order to differentiate between normal and anomalous instances, based on the reconstruction errors (Figure 6.23b) of the training data.



(a) Autoencoder architecture [aut].



(b) Reconstruction error [not].

Figure 6.23: Autoencoder-based anomaly detection.

6.5.2 ADSL algorithm with similarity loss

Training, validation and test sets (Table 6.7) are constructed as in Section 5.1, using 70% of the total time series from class 1 and an anomaly ratio $\alpha = 0.01$; the proportion of validation time series is set equal to 0.3.

	Train	Validation	Test
Class 1	2043	280	596
Class -1	20	600	1459

Table 6.7: Statistics about ECG dataset partition into train, validation and test set.

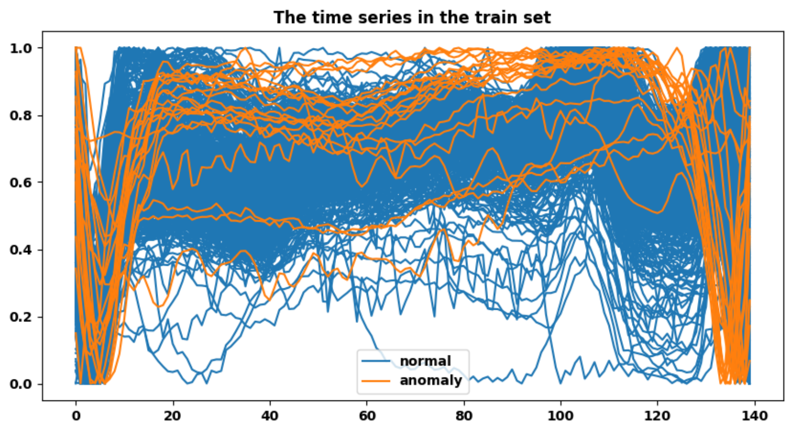


Figure 6.24: ECG training set.

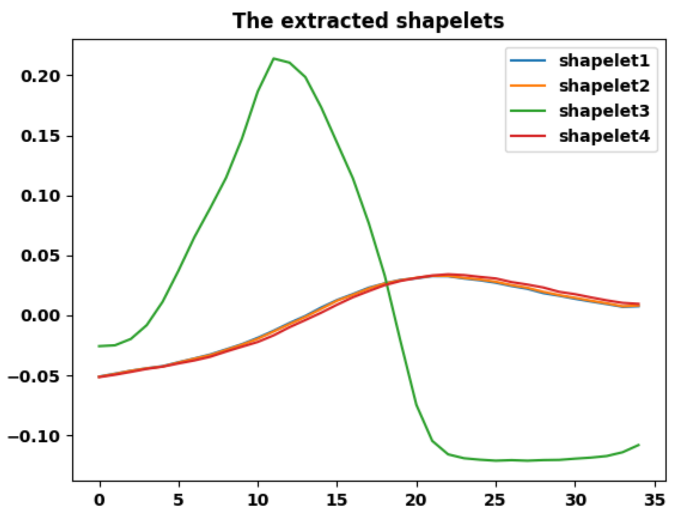
We set ADSL parameters (Section 5.3) as follows: $K = 4$, learning rate $\gamma = 0.01$, similarity penalty parameter $\lambda = 2$; three different values for $L = L^* \cdot Q$ were evaluated on the validation set by setting $L^* = 0.2,\ 0.25,\ 0.3$. In Section 6.3.4 we found empirically that overestimating the expected number of anomalies ν in the training set (Figure 6.24) gives better $F1$ score test values; therefore we set $\nu = 0.05$ by default, instead of the true proportion 0.01. Shapelets' parameters are initialized using K-means clusters' centers. The results obtained are shown in Table 6.9

L^*	0.2	0.25	0.3
AUC	0.977	**0.982**	0.978
$F1$	0.966	**0.970**	0.965

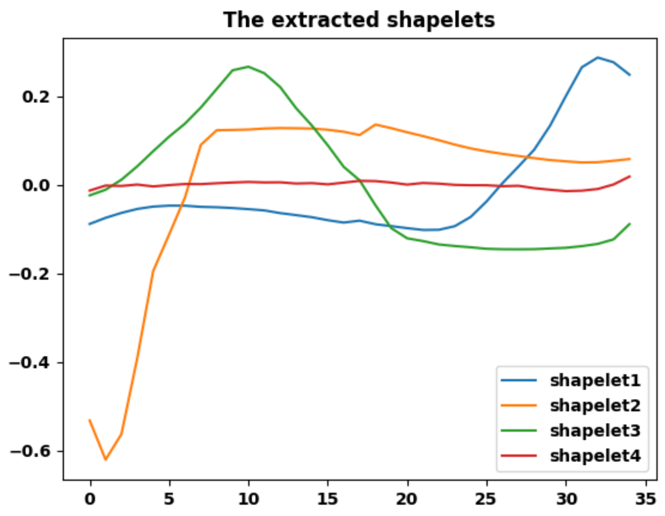
Table 6.8: Scores on validation set for different lengths.

We decided to keep $L = 35$ and $K = 4$ and tested the learned SVDD decision boundary on the test set, obtaining an AUC score of 0.978 and $F1$ score of 0.971.

We also compared the shapelets learned with $\lambda = 2$ and $\lambda = 0$, that is without the similarity penalty in Equation (4.33): In Figure 6.25 are plotted the learned shapelets for $L = 0.25 \cdot Q = 35$ in both the cases.



(a) Shapelets with no similarity penalty.



(b) Similarity penalty parameter λ=2.

Figure 6.25: Learned shapelets without (a) and with similarity loss (b).

Similarity loss helps the learning process towards different shapelets and avoids the collapse into a single one, with the drawback that the convergence needs much more epochs: we trained ADSL algortithm with $n_{epochs} = 25$ for $\lambda = 0$ and $n_{epochs} = 200$ for $\lambda = 2$. This is a cost for avoiding the mono-modality of Figure 6.25a.

Figure 6.26 shows the difference in mean discrepancy (Equation (6.1)) from each shapelet in Figure 6.25b between the test time series predicted in class -1 and 1:

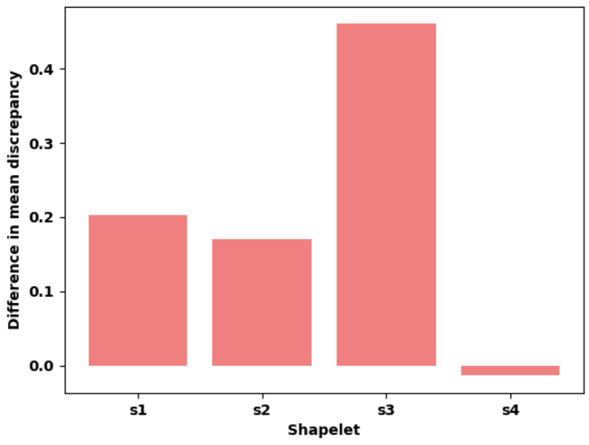


Figure 6.26: Difference between the mean discrepancy of anomalous and normal test series to each shapelet.

We find that the most meaningful shapelet in the decision on the SVDD is the third, as it is visually clear in Figure 6.27.

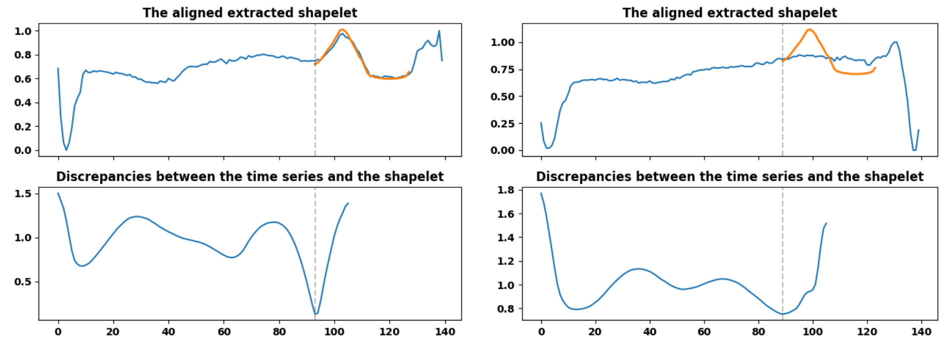


Figure 6.27: 3rd shapelet aligned with a normal (left) and an anomalous (right) test series.

6.5.3 Other Datasets

To conclude, we present the results obtained applying RLS and ADSL algorithms to other datasets, both univariate and multivariate (BasicMotions, Epilepsy, HandMovementDirection). The full description of such datasets can be found in [] and []. The datasets are constructed as described in Section 5.1, taking 80% of the series from the class chosen as normal into the training set, setting $\alpha = 0.5$, with no validation set. The cost parameter C of the SVDD is set using $\nu = 0.1$, overestimating the proportion of anomalies. The time series are normalizes using MinMax scaler. We set the length $L = L^* \cdot Q$ and number $K = K^* \cdot Q$ of shapelets in proportion to the length Q of the time series with $L^* = 0.2$ and $K^* = 0.02$. For RLS, the parameters are set as $r = 500$, $m = K$, $\varepsilon = (2, 1)$, correlation threshold $T = 0.8$, $reverse = False$ and $maxiter = 2$. For ADSL we set the learning rate to 0.1 and trained until convergence of the loss function.

Name	Normal Class	RLS($AUC\backslash F1$)	ADSL($AUC\backslash F1$)
BasicMotions	”Running”	$1\backslash 0.99$	$1\ \backslash 1$
BeetleFly	1	$0.78\backslash 0.78$	$0.95\ \backslash 0.86$
ECG200	1	$0.44\backslash 0.19$	$0.58\ \backslash 0.30$
Epilepsy	”Epilepsy”	$0.86\backslash 0.61$	$0.87\ \backslash 0.70$
HandMovementDirection	”Forward”	$0.37\backslash 0.21$	$0.56\ \backslash 0.35$
Meat	1	$0.78\backslash 0.934$	$0.968\ \backslash 0.99$
ToeSegmentation1	0	$0.84\backslash 0.934$	$0.987\ \backslash 0.980$

Table 6.9: AUC and $F1$ scores for RLS and ADSL algorithms on various datasets.

Chapter 7

Conclusions

In this thesis an overview of shapelets methods is given, with a particular focus on the anomaly detection task. For this purpose, we explained two algorithms for anomaly detection, OCSVM and SVDD, which are conceptually derived from the Support Vector Machine (SVM) algorithm for classification. We introduced SVM as a regularized Empirical Risk Minimization problem in the more general context of Kernel Methods in Machine Learning, giving insights into the most famous results. In the second experimental part of the thesis, three shapelet-based algorithms for anomaly detection are proposed and analyzed, namely BruteForce extractor (BF), Random Local Search (RLS) extractor and Anomaly Detection algorithm with Shapelet-based Feature Learning (ADSL). A shapelets' similarity penalization term based on discrepancies is introduced in addition to ADSL loss; furthermore ADSL has been extended to multivariate time series.
These algorithms have been tested on several benchmark datasets, both univariate and multivariate. The results reveal the main advantages of shapelets:

- the proposed algorithms work in an unsupervised fashion, with the additional hypothesis that the training set is possibly contaminated with a small percentage of anomalies;

- they provide interpretable and visually intuitive results;

- they achieve performance comparable to more complex methods such as autoencoders;

- the possibility of learning shapelets instead of searching through the subsequences gives more flexibility to shapelets' algorithms. According to the experimental results, the discrepancy-based loss of the learning algorithm ADSL seems to preserve interpretability;
- despite the lower computational cost of ADSL, searching methods give the possibility to get shapelets that represent anomalies, while by construction ADSL can only learn shapelets characterizing normality.

Further research can be conducted to develop an algorithm that learns the most distant shapelets from the time series, maintaining them close to a real subsequence using the regularization term in Equation (4.18).

Bibliography

[ADV20] Martin Andersen, Joachim Dahl, and Lieven Vandenberghe. Cvxopt: Convex optimization. *Astrophysics Source Code Library*, pages ascl–2008, 2020.

[Agg13] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[aut] https://lilianweng.github.io.

[BB17] Aaron Bostrom and Anthony Bagnall. A shapelet transform for multivariate time series classification. *arXiv preprint arXiv:1712.06428*, 2017.

[BDL$^+$18] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[BKS$^+$19] Laura Beggel, Bernhard X Kausler, Martin Schiegg, Michael Pfeiffer, and Bernd Bischl. Time series anomaly detection based on shapelet learning. *Computational Statistics*, 34(3):945–976, 2019.

[BKW96] Romain Briandet, E Katherine Kemsley, and Reginald H Wilson. Discrimination of arabica and robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics. *Journal of agricultural and food chemistry*, 44(1):170–174, 1996.

[Cal20] Ovidiu Calin. *Deep learning architectures*. Springer, 2020.

[CLRS22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[CLS21] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

[DBK$^+$19] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[DTS$^+$08] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[ecg] https://www.kaggle.com/datasets/devavratatripathy/ecg-dataset.

[EPP00] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1–50, 2000.

[git] https://github.com/ludosgit/shapelets.

[GSWST14] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.

[Haw80] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[HTFF09] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[Jol05] Ian Jolliffe. Principal component analysis. *Encyclopedia of statistics in behavioral science*, 2005.

[Kab08] Sergei Igorevich Kabanikhin. Definitions and examples of inverse and ill-posed problems. 2008.

[KML20] Patrick Kidger, James Morrill, and Terry Lyons. Generalised interpretable shapelets for irregular time series. *arXiv preprint arXiv:2005.13948*, 2020.

[LCLL14] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. *Autoencoder for words*. *Neurocomputing*, 139:84–96, 2014.

[LDHB12] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297, 2012.

[LKLC03] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, 2003.

[Mol20] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[not] https://www.kaggle.com/code/ohseokkim/dectecting-anomaly-using-autoencoder/notebook.

[ORA16] Luca Oneto, Sandro Ridella, and Davide Anguita. Tikhonov, ivanov and morozov regularization for support vector machine learning. *Machine Learning*, 103(1):103–136, 2016.

[PGM$^+$19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[PR09] Tommaso Poggio and Lorenzo Rosasco. *Statistical Leraning Theory and Applications, MIT course slides*. https://www.mit.edu/ 9.520/spring09/. 2009.

[SC08] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[SPST$^+$01] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[SSB$^+$02] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[TD04] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[TFV$^+$20] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, et al. Tslearn, a machine learning toolkit for time series data. *J. Mach. Learn. Res.*, 21(118):1–6, 2020.

[UBK15] Liudmila Ulanova, Nurjahan Begum, and Eamonn Keogh. Scalable clustering of time series with u-shapelets. In *Proceedings of the 2015 SIAM international conference on data mining*, pages 900–908. SIAM, 2015.

[Vap99] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[Wal17] James S Walker. *Fast fourier transforms*. CRC press, 2017.

[WEF$^+$20] Yichang Wang, Rémi Emonet, Elisa Fromont, Simon Malinowski, and Romain Tavenard. Adversarial regularization for explainable-by-design time series classification. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1079–1087. IEEE, 2020.

[YK09] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, 2009.

[YN18] Akihiro Yamaguchi and Takeichiro Nishikawa. One-class learning time-series shapelets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2365–2372. IEEE, 2018.

[ZMK12] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–794. IEEE, 2012.

[ZWY$^+$16] Qin Zhang, Jia Wu, Hong Yang, Yingjie Tian, and Chengqi Zhang. Unsupervised feature learning from time series. In *IJCAI*, pages 2322–2328. New York, USA, 2016.