Karlsruher Institut für Technologie

Institut für Technik der
Informationsverarbeitung (ITIV)

Institutsleitung
Prof. Dr.-Ing. Dr. h. c. J. Becker (Sprecher)
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

Bachelorarbeit Nr. ID-3229

von Herrn cand. inform. Daniel **Schneider**

Evaluation von Diskretisierungsansätzen zur Repräsentation von Zeitreihen im Kontext der Mustererkennung; Evaluation of Discretization Approaches for Representing Time Series Data in the Context of Pattern Recognition

Beginn:	15.12.2022
Abgabe:	17.04.2023
Betreuer:	M. Sc. Hanno Stage FZI M. Sc. Patrick Petersen FZI
Korreferent:	

Prof. Dr.-Ing. Eric Sax

FZI - Forschungszentrum Informatik
Embedded Systems and Sensors Engineering

Evaluation of Discretization Approaches for Representing Time Series Data in the Context of Pattern Recognition

Bachelor's Thesis

presented by

Daniel Schneider

Date:	17.04.2023
Advisor:	Prof. Dr.-Ing. Eric Sax
	Prof. Dr. Ralf Reussner
Preceptor:	M.Sc. Patrick Petersen
	M.Sc. Hanno Stage

Declaration

I hereby declare that I wrote my Bachelor's Thesis on my own and that I have followed the regulations relating to good scientific practice of the Karlsruhe Institute of Technology (KIT) in its latest form. I did not use any unacknowledged sources or means and I marked all references I used literally or by content.

Karlsruhe, 17.04.2023 D.Schneider

Abstract

Data-driven systems in areas such as autonomous driving or predictive maintenance are increasingly based on the processing of time series data that are collected by the increasing amount of sensors. For finding patterns in these data, pattern recognition algorithms are applied. However, most of these algorithms do not use the original time series as input data, but a more suitable representation of them. Such a representation can be obtained by discretizing the time series in a preprocessing step based on a time series discretization approach. For an efficient and effective pattern finding, such a time series discretization approach shall provide a compressed representation of the original time series, while preserving its features.

In this thesis, five time series discretization approaches are evaluated. First, for each time series discretization approach, its goodness with respect to a feature-preserving discretized representation of the original time series is measured. Second, the applicability of these approaches with respect to motif discovery, which is the task of finding recurrently occurring patterns in a time series, is evaluated. For this evaluation, each time series discretization approach is used in a preprocessing step to obtain discretized time series representations that represent the input data of three motif discovery algorithms. Moreover, for two of these three algorithms, the results based on applying a time series discretization approach in a preprocessing step are compared to the results when inputting the original time series without discretization.

The experimental results indicate that two of the five evaluated time series discretization approaches are superior with respect to a feature-preserving discretized representation. Moreover, the results for the employed motif discovery algorithms indicate that four time series discretization approaches perform similar, while the remaining approach performs worse. Overall, the results imply that those four approaches are suitable as a preprocessing step for the employed motif discovery algorithms.

Contents

List of Figures

List of Tables

List of Acronyms

PAA	Piecewise Aggregate Approximation
SAX	Symbolic Aggregate Approximation
eSAX	Extended Symbolic Aggregate Approximation
1d-SAX	1d-Symbolic Aggregate Approximation
aSAX	Adaptive Symbolic Aggregate Approximation
SSE	Sum of Squared Errors
MAE	Mean Absolute Error
MSE	Mean Squared Error
MMDL	Mean Motif Density per Label
RWLR	Random Walk Labeling Ratio
FRSR	Found Recurrent Subsequences Ratio
TP	True Positives
FN	False Negatives
FP	False Positives
KL	Kullback-Leibler divergence
SKL	symmetric Kullback-Leibler divergence

List of Algorithms

1 Introduction

1.1 Motivation

The influence of time series data on data-based decisions has increased enormously in the past. Due to the advancing digitization, this trend will continue in the future [1]. Data-driven systems in areas such as autonomous driving or predictive maintenance are increasingly based on the processing of time series data. For example, a fully autonomous vehicle is estimated to generate 19 terabytes of data per hour that are generated by its sensors [2]. As a result of this increase in time series data processing, the demand for optimized databases for collecting and analyzing this type of data has also risen sharply in recent years [3].
The advantage of analyzing time series data is the ability to identify changes and patterns that develop over time [4]. To exploit this advantage, it is necessary to develop effective and efficient algorithms for processing and analyzing time series data. A prerequisite for such algorithms is meaningful input data, which can be obtained by a suitable representation of the time series [4]. Based on the state of the art of science, the discretization of time series represents a prominent approach in this respect [4]. This motivates the evaluation of different discretization approaches for the representation of time series in the context of pattern recognition.

1.2 Problem Statement

As most algorithms for time series pattern recognition do not use the original time series as input data, a preprocessing step is applied in which the time series is converted into a more suitable representation [4]. The first objective of such a preprocessing step is to compress the time series data in order to reduce the time and memory requirements of the pattern recognition algorithms [4]. The other objective is to transform the time series into a representation that reflects its characteristic features [4].
Thus, the difficulty when using discretization as such a preprocessing step is to preserve and extract the characteristic features of the time series as accurately as possible, while removing less informative data such as noise. By preserving

the characteristic features, the results of pattern recognition algorithms based on the corresponding representation can be reliably transferable to the original time series. Hence, discretization approaches for time series have to be found that fulfill two properties. First, for effectively applying pattern recognition algorithms, they shall provide a feature-preserving representation of the original time series. Second, for efficiently applying pattern recognition algorithms, they shall provide a compressed representation of the original time series.

1.3 Aim of the Thesis

In this thesis, existing discretization approaches for the representation of time series shall be evaluated. For this purpose, suitable metrics shall be used to quantify and evaluate the goodness of these approaches with respect to a feature-preserving representation of the original time series.
Furthermore, the applicability of the examined discretization approaches as a preprocessing step for time series pattern recognition algorithms shall be evaluated. The focus of time series pattern recognition shall be the identification of recurrently occurring patterns in a time series, which is called Motif Discovery [4].

1.4 Structure of the Thesis

Chapter 2 introduces fundamental concepts that are used in this thesis. In Chapter 3, the evaluated discretization approaches for time series are presented. Chapter 4 presents the pattern recognition algorithms with respect to motif discovery. These algorithms are used to evaluate the applicability of the examined time series discretization approaches as a preprocessing step. Chapter 5 contains the results of the experimental evaluation of the examined time series discretization approaches. In Chapter 6, a summary of the findings of this thesis along with an outlook is presented.

2 Fundamentals

2.1 Mathematical Concepts

2.1.1 Minkowski Distance

Let $X = (x_1, ..., x_N)$, $Y = (y_1, ..., y_N) \in \mathbb{R}^N$ $(N \geq 1)$ be two points. The Minkowski distance for these two points is defined for $p \geq 1$ as [5]:

$$D_p(X, Y) = \left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{1/p} \tag{2.1}$$

In this thesis, two instantiations of the Minkowski distance are used. The first one results from setting $p = 1$ and is called the Manhattan distance [5]:

$$D_1(X, Y) = \sum_{i=1}^{N} |x_i - y_i| \tag{2.2}$$

The second one results from setting $p = 2$ and is called the Euclidean distance [5]:

$$D_2(X, Y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{2.3}$$

2.1.2 Statistical Concepts

Random Variable

$X : S \to \mathbb{R}$ is called a random variable, when S is the set of all possible outcomes of a random event [6]. As an example, consider tossing a coin, then it could be $X : \{heads, tails\} \to \{-1, 1\}$. When $Pr \geq 0$ is a probability measure defined on S, probabilities $Pr(X = x) = Pr(\{s \in S \mid X(s) = x\})$ can be determined [6].
X is called a discrete random variable if $Pr(X = x) > 0$ holds for at most as

many different $x \in R' \subset \mathbb{R}$, where R' is a countable set [6]. A set is countable if there exists a bijection to $\mathbb{N}$. In this case, the probability mass function of X is defined as $f_d : \mathbb{R} \to [0, 1]$ with $f_d(x) = Pr(X = x)$ [6]. Note that it is $f_d(x) = 0$ if there is no $s \in S$ with $X(s) = x$. The probability mass function of X defines the discrete probability distribution of X [6].
X is called a continuous random variable if there exists a function $f_c : \mathbb{R} \to [0, +\infty)$ with $f_c \geq 0$, such that for every interval $I \subseteq \mathbb{R}$, the probability of $X = x$ for $x \in I$ is the integral of f_c over I [6]. The function f_c is called the probability density function of X. In this case, it is for example for the bounded and closed interval $[a, b] \subseteq \mathbb{R}$ [6]:

$$Pr(a \leq X \leq b) = \int_a^b f_c(x)\ dx \tag{2.4}$$

Discrete Uniform Distribution

Let X be a discrete random variable and $a, b \in \mathbb{N}$ with $a \leq b$. If $Pr(X = x)$ is the same for each $x \in \{a, ..., b\}$, then X is uniformly distributed for $\{a, ..., b\}$ [6]. In this case, the discrete probability distribution of X is defined by the following probability mass function of X [6]:

$$f_d(x) = \begin{cases} \frac{1}{b-a+1}, & x \in \{a, ..., b\} \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

Gaussian Distribution

Let X be a continuous random variable. X follows the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu \in \mathbb{R}$ and variance σ^2 ($\sigma > 0$) if for each $x \in \mathbb{R}$ it is [6]:

$$f_c(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.6}$$

It is written as $X \sim \mathcal{N}(\mu, \sigma^2)$. For $\mu = 0$ and $\sigma = 1$, the resulting Gaussian distribution $\mathcal{N}(0, 1)$ is called the standard normal distribution [6].

Quantile Function

Let X be a (discrete or continuous) random variable. The cumulative distribution function of X is defined for $x \in (-\infty, +\infty)$ as [6]:

$$F(x) = Pr(X \leq x) \tag{2.7}$$

Define the function $Q : (0, 1) \to \mathbb{R}$ with $Q(p) = x$, where x is the smallest value that fulfills $F(x) \geq p$. The function Q is called the quantile function of X and $Q(p)$ is called the p quantile of X, which is equivalent to the $100 \cdot p$ percentile of X [6].

Standardization

Let $X = (x_1, ..., x_N) \in \mathbb{R}^N$ $(N \geq 1)$ be a point. Applying standardization on X involves transforming each x_i $(1 \leq i \leq N)$ to $\hat{x}_i$ [7]:

$$\hat{x}_i = \frac{x_i - \mu_X}{\sigma_X}, \tag{2.8}$$

where μ_X is the mean of all x_i, and σ_X is the standard variation of all x_i. The resulting standardized point is then represented by $\hat{X} = (\hat{x}_1, ..., \hat{x}_N)$. Further, it is $\mu_{\hat{X}} = 0$ and $\sigma_{\hat{X}} = 1$.
Applying standardization on multiple points X improves the comparability of these points as they are transformed to the same scale and unit [7]. After standardization, each x_i is measured in terms of the number of standard deviations it deviates from the mean μ_X.

2.1.3 Kullback-Leibler Divergence

Let $P = \{p_1, ..., p_k\}, Q = \{q_1, ..., q_k\}$ $(k \geq 1)$ be two discrete probability distributions, meaning that $0 \leq p_i, q_i \leq 1$ $(1 \leq i \leq k)$ are probabilities and the sum of all p_i and q_i, respectively, equals one. The Kullback-Leibler divergence (KL) for comparing P and Q is defined as [8]:

$$KL(P, Q) = \sum_{i=1}^{k} p_i \cdot log_2(\frac{p_i}{q_i}) \tag{2.9}$$

The KL is only defined for $p_i, q_i > 0$ due to the logarithm in Equation 2.9. It measures the difference between P and Q [9]. The larger the value of the KL, the more different are P and Q, and vice versa. Moreover, the KL is non-negative and it is $KL = 0$ if and only if $p_i = q_i$ [9]. The KL is not symmetric, which means that it is $KL(P, Q) \neq KL(Q, P)$ in general. However, taking the mean of both directions results in a symmetric version of the KL, which is called the symmetric Kullback-Leibler divergence (SKL) [8]:

$$SKL(P, Q) = \frac{1}{2} \cdot (KL(P, Q) + KL(Q, P)) \tag{2.10}$$

2.2 Machine Learning Algorithms

2.2.1 Linear Regression

Let $X_1, ..., X_k$ $(k \geq 1)$ and Y be random variables. Then, the regression function of Y on $X_1, ..., X_k$ is given by [6]:

$$E(Y \mid x_1, ..., x_k) = w_0 + w_1 \cdot x_1 + ... + w_k \cdot x_k, \tag{2.11}$$

where $E(Y \mid x_1, ..., x_k)$ is the conditional expectation of Y for given values $x_1, ..., x_k$ of $X_1, ..., X_k$. The regression function assumes a linear relationship between the expected value of Y and the $x_1, ..., x_k$, which is controlled by the regression coefficients $w_0, ..., w_k$ [6]. These regression coefficients are unknown parameters whose values need to be estimated based on given data points.
For the simple linear regression of Y on a single random variable X, the regression function reduces to $E(Y \mid x) = w_0 + w_1 \cdot x$, where w_0 is the unknown intercept, w_1 is the unknown slope, and x is a given value of X [6]. For the estimation of the values $\hat{w}_0$ and $\hat{w}_1$ for the unknown parameters w_0 and w_1, respectively, the objective is to minimize the sum of squared errors based on $n \geq 1$ given data points $(x_1, y_1), ..., (x_n, y_n)$ [6]:

$$\min_{w_0, w_1} Q(w_0, w_1) = \sum_{i}^{n} [y_i - E(Y|x_i)]^2 \tag{2.12}$$

Let $\overline{x}$ and $\overline{y}$ be the mean of all x_i and y_i $(1 \leq i \leq n)$, respectively. Then, the values that minimize the sum of squared errors are obtained by [6]:

$$\hat{w}_1 = \frac{\sum_i^n (y_i - \overline{y})(x_i - \overline{x})}{\sum_i^n (x_i - \overline{x})^2} \tag{2.13}$$

$$\hat{w}_0 = \overline{y} - \hat{w}_1 \cdot \overline{x} \tag{2.14}$$

2.2.2 k-Means Clustering

Given a set of observations $X = \{x_1, ..., x_N\} \subset \mathbb{R}^d$ $(d, N \geq 1)$, the k-means clustering algorithm partitions X into a given number of $1 \leq k \leq N$ disjoint sets $C_1, ..., C_k$, which are called clusters [10]. These clusters shall minimize the Sum of Squared Errors (SSE) [10]:

$$\min_{C_1, ..., C_k} \sum_{i}^{k} \sum_{x \in C_i} D_2(x, \mu_i)^2, \tag{2.15}$$

where D_2 is the Euclidean distance and $\mu_i \in \mathbb{R}^d$ $(1 \leq i \leq k)$ is the mean of all points $x \in C_i \subseteq X$, which is also called the cluster center of C_i. For finding such minimizing clusters, the following procedure is applied [10]:

1. randomly select k points from X as initial cluster centers
2. repeat until $C_1, ..., C_k$ stop changing:
   2.1 assign each $x \in X$ to the closest cluster C_i with respect to the Euclidean distance between x and μ_i
   2.2 for each cluster C_i compute its new cluster center as the mean of all assigned points $x \in C_i$

2.3 Time Series

In this thesis, a time series X shall be defined as an ordered sequence of $N \geq 1$ real-valued observations $X = x_1, ..., x_N$, that shall be observed at uniformly spaced points in time $T = t_1, ..., t_N$ [4]. N shall be called the length of the time series. Furthermore, a subsequence S of a time series X of length N shall be defined as a time series of length $1 \leq M \leq N$ that consists of contiguous observations from X, such that $S = x_k, x_{k+1}, ..., x_{k+M-1}$ with $1 \leq k \leq N - M + 1$ [4].

3 Time Series Discretization

One property of time series is that they often consist of a number of points that is too large to define data mining algorithms that work directly on the original time series, since their processing time would be intractable [4]. As an example, consider that a day has 86,400 seconds. A sensor measurement every second would thus result in 86,400 time series points per day. This would add up to over 31 million time series points when measuring every day in a year.
Therefore, algorithms are needed that transform time series into a representation with a reduced number of points while preserving their essential characteristics like extrema, anomalies and repeated patterns. Taking such a representation as the input for data mining algorithms would not only reduce the processing time, since it should meet all of the following requirements [4]:

1. reduction of the number of time series points
2. extraction of the essential characteristics of the time series
3. computation of the representation with a lower processing time compared to the data mining algorithms
4. reconstruction of the original time series from the representation with high quality based on reconstruction error measures
5. implicit noise removal or insensitivity to noise

A representation that tries to meet these requirements is obtained by discretizing the original time series [11]. The basic idea of such a discretization involves dividing the amplitude of the original time series into intervals based on breakpoints that are set at computed values along the amplitude. The discretized representation is then obtained by assigning each time series point to its respective interval and representing all points within an interval with the discrete value that is assigned to the interval (see Figure 3.1).

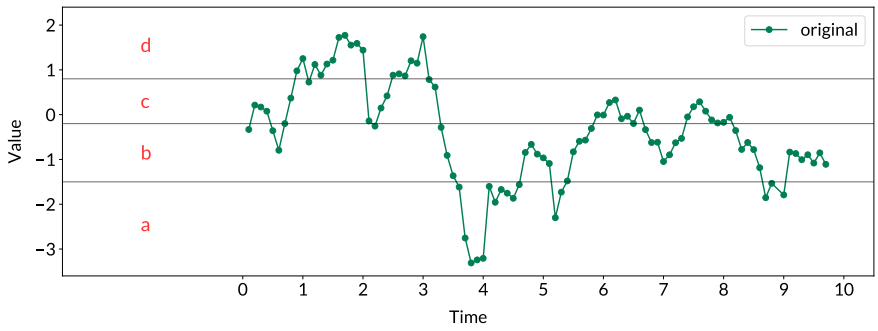


Figure 3.1: As an example, the amplitude of the plotted original time series is divided into four discretization intervals based on three breakpoints. The plotted discretization intervals are assigned the Latin letters a, b, c, and d as discrete values. For instance, all points of the original time series that are located in the a-discretization interval are assigned a to obtain the discretized representation of the original time series. The resulting discretized representation is then obtained by concatenating these discrete values in order of time the corrsponding points occur in the original time series [11]. For the first five points this is bcccb.

3.1 Piecewise Aggregate Approximation

The algorithms for time series discretization explained in this chapter all depend on the Piecewise Aggregate Approximation (PAA) as a preprocessing step. The PAA first standardizes the original time series before it transforms it to an intermediate representation [12]. This intermediate representation is then used by the time series discretization algorithms to obtain the discretized representation of the original time series.

Main Procedure

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. Further, consider X as a point in an N-dimensional space. Then, the main idea of the PAA is to project X onto a lower-dimensional space of dimensionality $1 \leq n \leq N$ [12].
First, a non-overlapping sliding window of length $1 \leq w \leq N$ is used to partition X into subsequences of equal length [12]. Assuming that N is divisible by w, this results in $n = N \cdot w^{-1}$ subsequences of length w. The final step is to compute the mean of the w points of each subsequence to get an approximation of the subsequence (see Figure 3.2) [13].
Let $\overline{x}_i$ $(1 \leq i \leq n)$ be the mean of the w points of the i-th extracted subsequence from X. Then, $\overline{X} = \overline{x}_1, ..., \overline{x}_n$ is the resulting time series in the lower-dimensional

space of dimensionality n, where $\overline{x}_i$ is computed by [12]:

$$\overline{x}_i = \frac{n}{N} \sum_{j=\frac{N}{n}(i-1)+1}^{\frac{N}{n}i} x_j \tag{3.1}$$

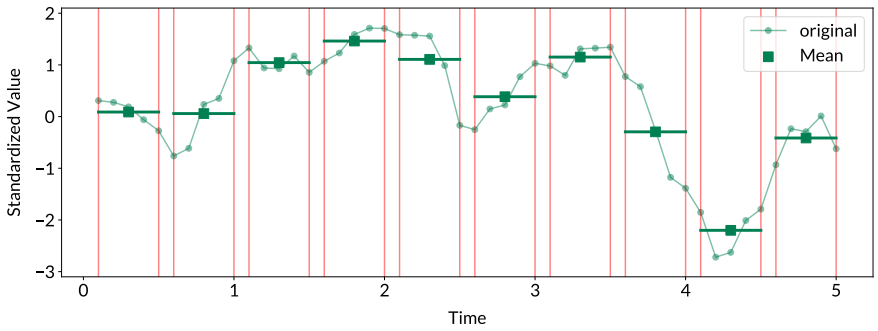


Figure 3.2: The original time series is partitioned by a non-overlapping sliding window into subsequences of equal length. For each subsequence the mean of its points is computed. These means (green squares) are the points of the time series that is the result of the PAA [12].

Strategies for Time Series of Indivisable Length

The assumption that N is divisible by w is not a requirement of the PAA, but it simplifies notation and understanding [12].
One strategy when N is not divisible by w is to append additional points with a value of zero to X until N is divisible by w [13]. This is a straightforward strategy. But, on the other hand, the mean computed for the last extracted subsequence will be distorted. The extent of this distortion will be greater the closer the number of additionally appended points is to w.
Another strategy is to truncate the points of X until N is divisible by w. This is also a straightforward strategy, but the time series is modified and information, in the form of time series points, is lost.
The strategy that is used for the evaluation in this thesis (see Chapter 5) tries to find a compromise between the strategies mentioned above. First, the window length w is applied on the time series. Then, depending on the number of points contained in the last extracted subsequence, it is decided to truncate these points or to compute the mean of these points. When this number of points is greater than $\frac{w}{2}$, the mean of these points is computed, otherwise these points along with the last extracted subsequence are truncated. Due to this threshold of $\frac{w}{2}$, this strategy

is straightforward, does not distort the mean computed for the last extracted subsequence, and limits the loss of information.

Role of the Window Length as Input Parameter

Since the PAA is a downsampling approach, $\overline{X}$ is an approximation of X which incurs a loss of information [13]. On the other hand, this approximation also incurs a gain in free memory. From $n \leq N$ it follows that storing $\overline{X}$ needs at most as much memory as storing X. These two observations imply a trade-off between the loss of information and the gain in free memory incurred when applying the PAA on X [14].
The decisive parameter that affects this trade-off is the window length w [14]. Qualitatively, as w increases more information is lost, but less memory for storing $\overline{X}$ is required, because less means $\overline{x}_i$ are computed to approximate X. On the other hand, as w decreases less information is lost, but more memory for storing $\overline{X}$ is required, because more means $\overline{x}_i$ are computed to approximate X. When assigning each point of an extracted subsequence the corresponding mean of the subsequence, this trade-off can be visualized (see Figure 3.3) [14].
Note that $\overline{X}$ is the mean of X for the extreme case of $w = N$ and $\overline{X} = X$ for the extreme case of $w = 1$, meaning that each point of X is its own mean [12].

Time Complexity

For the PAA, $\frac{N}{w}$ subsequences need to be extracted from X while each subsequence contains w points. Further, the computation of the mean is linear in the number of points. Hence, the PAA has a time complexity of $\mathcal{O}(\frac{N}{w} \cdot w) = \mathcal{O}(N)$ [12].

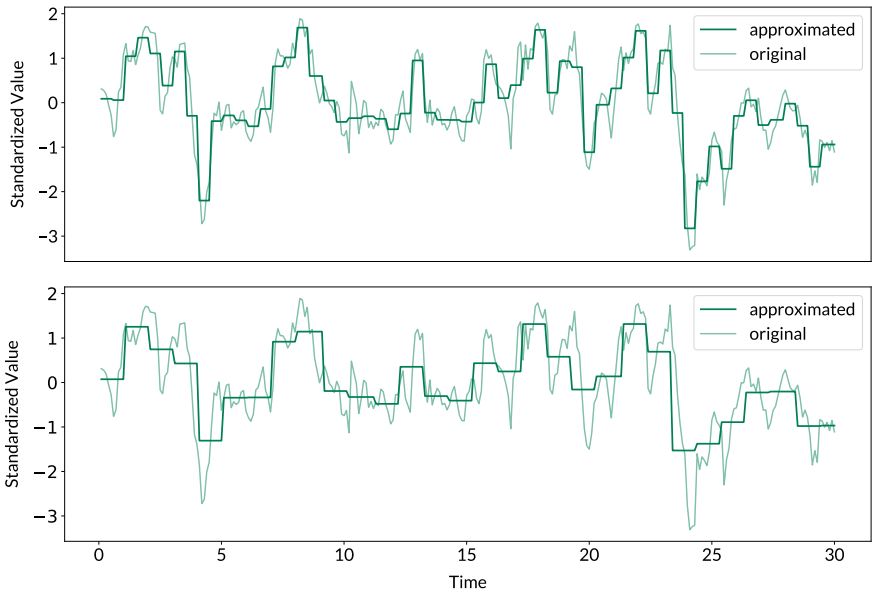


Figure 3.3: The original time series in these two plots consists of 300 points. In the plot above a window length of $w = 5$ (i.e. five points) is used to extract the subsequences, while in the plot below a window length of $w = 10$ (i.e. ten points) is used. The approximation in the plot above captures more fine-grained characteristics of the original time series like local extrema, while the approximation in the plot below is more coarse-grained. However, the approximation in the plot above is created with 60 points (i.e means) compared to 30 points (i.e. means) that are used for the approximation in the plot below.

3.2 Fixed-breakpoints Discretization

The following three time series discretization algorithms all assume that the original time series that shall be discretized follow the standard normal distribution $\mathcal{N}(0, 1)$ after applying standardization [14]. Based on this assumption, the breakpoints that define the discretization intervals along the amplitude of the time series are computed by the quantiles of the standard normal distribution [14]. Thus, given a fixed number of discretization intervals to be used, all time series are discretized based on the same breakpoints. Therefore, the breakpoints are not adaptive to the time series data at hand, which is the reason the following three discretization algorithms are classified into fixed-breakpoints discretization.

3.2.1 Symbolic Aggregate Approximation

The Symbolic Aggregate Approximation (SAX) discretization algorithm applies amplitude discretization to the PAA representation to obtain the discretized SAX representation of the original time series [14]. Due to the fixed breakpoints discussed above, every PAA representation is discretized based on the same discretization intervals.

Main Procedure

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. Assume that X follows the standard normal distribution $X \sim \mathcal{N}(0, 1)$. Further, define $P := \{\frac{j}{a} \mid j \in \{1, ..., a-1\}\}$ for a given natural number $a \geq 2$. Applying the quantile function of the standard normal distribution to the probabilities in P results in the a-quantiles β_j $(1 \leq j \leq a-1)$ of the standard normal distribution. The ascending sorted list of these a-quantiles $B := (\beta_0, \beta_1, ..., \beta_{a-1}, \beta_a)$ with $\beta_0 = -\infty$ and $\beta_a = +\infty$ are called breakpoints and a is called alphabet size [15]. Define $A := \{\alpha_j \mid j \in \{1, ..., a\}\}$ as the corresponding alphabet with a symbols (e.g. letters from the Latin alphabet).
Discretizing X now first requires its transformation into its PAA representation $\overline{X} = \overline{x}_1, ..., \overline{x}_n$ with $1 \leq n \leq N$. Then, the discretized SAX representation $\hat{X} = \hat{x}_1, ..., \hat{x}_n$ of X is obtained by mapping the means $\overline{x}_i$ to alphabet symbols α_j [15]:

$$\hat{x}_i = \alpha_j \iff \beta_{j-1} \leq \overline{x}_i < \beta_j \qquad (1 \leq i \leq n,\ 1 \leq j \leq a) \tag{3.2}$$

Hence, all means $\overline{x}_i$ of the PAA representation of X that are smaller than the breakpoint β_1 are mapped to the alphabet symbol α_1, all means that are greater than or equal to the breakpoint β_1 and smaller than the breakpoint β_2 are mapped to the alphabet symbol α_2 and so on (see Figure 3.4) [15]. Therefore, the discretization intervals along the amplitude are defined by $[\beta_{j-1}, \beta_j)$ $(1 \leq j \leq a)$. Further, note that the breakpoints β_j split the standard normal distribution into a areas, each containing $\frac{1}{a}$ of the total area under the curve, given their definition as a-quantiles of the standard normal distribution [15].

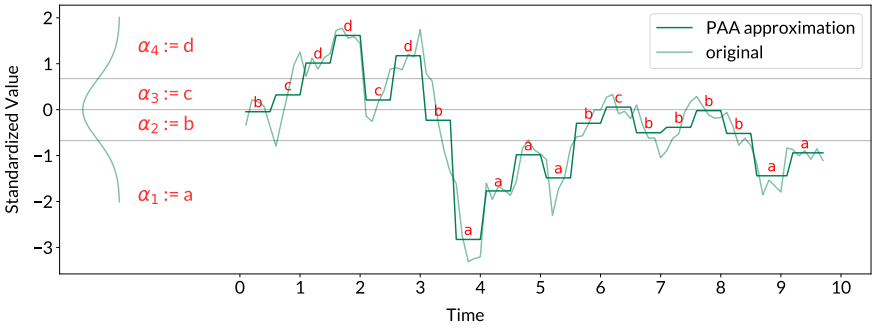


Figure 3.4: The original time series is discretized based on its PAA representation with an alphabet size of $a = 4$. It is assumed that the original time series follows the standard normal distribution $\mathcal{N}(0, 1)$. Based on this assumption the discretization intervals along its amplitude are determined [15]. Using lowercase letters from the Latin alphabet for the alphabet symbols, the resulting discretized representation of the original time series is: `bcddcdbaaaabcbbbbaa`.

Role of the Alphabet Size as Input Parameter

From the definition of the breakpoints as a-quantiles of the standard normal distribution, it follows that the alphabet size a is equal to the number of discretization intervals. Furthermore, if the alphabet size a increases the distance $\beta_j - \beta_{j-1}$ between two adjacent breakpoints decreases, because each discretization interval $[\beta_{j-1}, \beta_j)$ covers $\frac{1}{a}$ of the area under the curve of the standard normal distribution [15]. Therefore, the range of continuous values that each alphabet symbol covers becomes narrower when the alphabet size a increases.
Assuming that the means $\overline{x}_i$ $(1 \leq i \leq n)$ of the PAA representation $\overline{X}$ follow the standard normal distribution, each discretization interval contains $\frac{1}{a}$ of the total number of means $\overline{x}_i$. Thus, the number of means in each discretization interval decreases when the alphabet size a increases.
Hence, the alphabet size a controls the granularity of the discretization of the means $\overline{x}_i$ [14]. Meaning that the greater the alphabet size, the more likely it is that each different mean is assigned a different alphabet symbol when discretizing. In other words, it is more likely that each different mean is assigned its own unique alphabet symbol.
This effect can be even more extreme when considering a small window length used in the PAA. For example, consider a window length of $w = 1$. Then it is $\overline{X} = X$, meaning that each point in X is its own mean. Further, suppose an alphabet size of $\lim_{a \to +\infty} a$. Assuming that X does not contain any duplicate values, it follows that each point x_i $(1 \leq i \leq N)$ of X is assigned its own unique alphabet symbol when discretizing.
All in all, increasing the alphabet size results in a more granular discretization of the means of the PAA representation and vice versa (see Figure 3.5). However,

increasing the alphabet size also results in increased memory requirements for storing the SAX representation $\hat{X}$, because the more alphabet symbols used the greater the memory requirement per symbol. Therefore, a trade-off dependent on the alphabet size exists between the granularity of the discretization and the memory requirements of the SAX representation of the original time series [14].

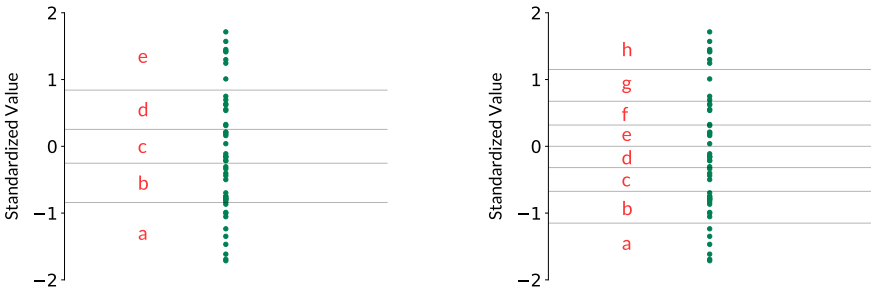


Figure 3.5: In the left plot, the breakpoints for an alphabet size of $a = 5$ are shown. In the right plot, the breakpoints for an alphabet size of $a = 8$ are shown. For instance, the range of continuous values the alphabet symbol c covers is narrower for $a = 5$ compared to $a = 8$. Thus, the discretization shown in the right plot is more granular by using more alphabet symbols.

Distance Measures

One usage of distance measures in the context of time series is to measure the similarity between a given time series and each time series stored in a database in order to retrieve similar time series from it [4]. There are two reasons why such a procedure is more time efficient with SAX representations than with the corresponding original time series.
The first reason is that time series databases storing original time series are more likely to need to be stored on disk while those storing SAX representations are more likely to be held in main memory due to less memory requirements [14]. Therefore, accessing SAX representations is more likely to be faster than accessing original representations.
The second reason is that the time efficiency of distance computations between two time series is dependent on the number of points these time series have. Therefore, distance computations between SAX representations are faster than between the corresponding original time series due to less number of points respectively alphabet symbols [14].
For these two reasons, the following time efficient procedure can be applied based on SAX representations [14, 16].

Suppose two time series databases $DB := \{X \mid X \textit{ time series of length } N \geq 1\}$ and $DB^{SAX} := \{\hat{X} \mid X \in DB\}$ where each time series X is stored in its SAX representation $\hat{X}$ of length $1 \leq n \leq N$. Further, assume that DB is stored on disk while DB^{SAX} is held in main memory due to less memory needed for DB^{SAX} than for DB.
Now given a time series X^* of length N, the set of time series

$$DB^* := \{X \mid X \in DB,\ D_2(X^*, X) \leq r\}$$

shall be found, where D_2 is the Euclidean distance between two original time series and $r \geq 0$ is a given real number distance. The Euclidean distance is assumed to be the true distance between two original time series and is one possible distance measure for this procedure [4].
The first step for finding DB^* is to discretize X^* based on the SAX in order to be able to compare its SAX representation $\hat{X}^*$ with the SAX representations in DB^{SAX}.
The next step is to define a distance measure $MINDIST$ that measures the distance between any two SAX representations and lower-bounds the Euclidean distance for the corresponding original time series (see Figure 3.6):

$$MINDIST(\hat{X}^*, \hat{X}) \leq D_2(X^*, X) \tag{3.3}$$

With $MINDIST$, the SAX representations in DB^{SAX} can be filtered by discarding those that definitely do not fullfill $D_2(X^*, X) \leq r$. Then, the resulting filtered database is $DB^{SAX'} := \{\hat{X} \mid\ X \in DB,\ MINDIST(\hat{X}^*, \hat{X}) \leq r\}$. This filtering property of $MINDIST$ follows from Equation 3.3, because each $\hat{X} \in DB^{SAX}$ with $MINDIST(\hat{X}^*, \hat{X}) > r$ can be neglected, since it is:

$$MINDIST(\hat{X}^*, \hat{X}) > r \implies D_2(X^*, X) > r \tag{3.4}$$

Equation 3.3 also guarantees no false dismissals during this filtering process. Hence, there is no $\hat{X} \in DB^{SAX}$ that is filtered out, but fulfills $D_2(X^*, X) \leq r$. This property of $MINDIST$ is known as the Lower Bounding Lemma [16].
However, after this filtering process there can be still some $\hat{X} \in DB^{SAX'}$ where $D_2(X^*, X) > r$ holds, because $MINDIST$ underestimates the Euclidean distance according to Equation 3.3.
Hence, the last step to find DB^* is to filter these false positives out of $DB^{SAX'}$. This is done by retrieving the original time series X of all $\hat{X} \in DB^{SAX'}$ from DB and computing $D_2(X^*, X)$. DB^* then results from keeping each $X \in DB$ that fulfills $D_2(X^*, X) \leq r$.
Note that only during this last step, the original time series X are retrieved from DB that is stored on disk. Further, not every $X \in DB$ is retrieved, but only those that are in the filtered $DB^{SAX'}$. Before this last step, the distances are computed on the SAX representations $\hat{X}$ based on $MINDIST$. These SAX representations

are held in main memory and are represented by less number of alphabet symbols compared to the number of points of the original time series.
The question that now remains is what distance measure $MINDIST$ that lower-bounds the Euclidean distance shall be used. One important characteristic of the SAX is that it provides such a lower bounding distance measure. It is defined as [14]:

$$MINDIST(\hat{X}, \hat{Y}) := \sqrt{\frac{N}{n}} \sqrt{\sum_{i=1}^{n} (dist(\hat{x}_i, \hat{y}_i))^2} \tag{3.5}$$

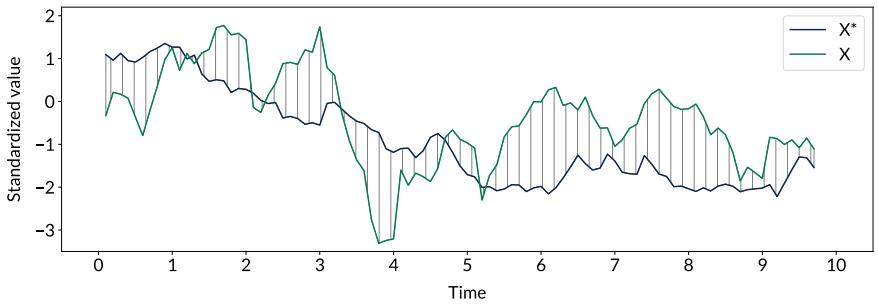
where $\hat{X} = \hat{x}_1, ..., \hat{x}_n$ and $\hat{Y} = \hat{y}_1, ..., \hat{y}_n$ are the SAX representations with n alphabet symbols of the original time series X and Y of length N, respectively. The subfunction $dist$ is defined for $1 \leq i, j \leq a$ as:

$$dist(\alpha_i, \alpha_j) := \begin{cases} 0, & \text{if } |i - j| \leq 1 \\ \beta_{\max\{i,j\}-1} - \beta_{\min\{i,j\}}, & \text{otherwise} \end{cases} \tag{3.6}$$

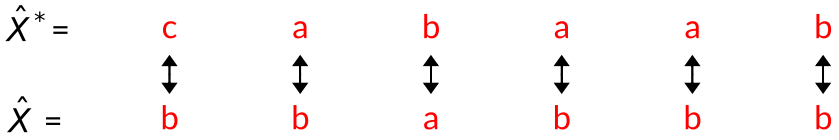
where α_i, α_j are alphabet symbols and β_i, β_j are breakpoints.

Time Complexity

The first step of the SAX involves the computation of the breakpoints. A static lookup table that contains the a-quantiles of the standard normal distribution can be used to retrieve the ascending sorted breakpoints for a given alphabet size a [14]. Thus, the time complexity for computing the breakpoints for a given alphabet size is $\mathcal{O}(1)$. Note that the breakpoints $\beta_0 = -\infty$ and $\beta_a = +\infty$ are not actually needed. Therefore, a alphabet symbols can be represented by $a - 1$ breakpoints, since it is $|B| = a + 1$.
The final step is to map each of the n means of the PAA representation to one alphabet symbol. Based on the retrieved ascending sorted breakpoints, a binary search can be applied for the mapping of each mean. Thus, the time complexity for this step is $\mathcal{O}(n \cdot log_2(a - 1)) = \mathcal{O}(n)$ for a given alphabet size a.
Therefore, the time complexity of the discretization process with the SAX is $\mathcal{O}(n)$. Taken the time complexity of $\mathcal{O}(N)$ of the PAA into account, the total time complexity of the SAX is $\mathcal{O}(n) + \mathcal{O}(N) = \mathcal{O}(N)$, because it is $N \geq n$ [17].



(a) The Euclidean distance is indicated by connecting each pair of points of the original time series X^* and X that belong to the same point in time [14].



(b) The $MINDIST$ of the corresponding SAX representations $\hat{X}^*$ and $\hat{X}$ is indicated by connecting the pairwise symbols whose distance shall be measured [14].

Figure 3.6: A visual intuition for the $MINDIST$ and the Euclidean distance. The $MINDIST$ shall lower-bound the Euclidean distance: $MINDIST(\hat{X}^*, \hat{X}) \leq D_2(X^*, X)$ [14].

3.2.2 Extended Symbolic Aggregate Approximation

Similar to the SAX, the Extended Symbolic Aggregate Approximation (eSAX) discretization algorithm applies amplitude discretization to the PAA representation in order to obtain the discretized eSAX representation of the original time series [18]. Moreover, the eSAX uses the same fixed breakpoints as the SAX for computing the discretization intervals along the amplitude.
However, the eSAX does not use the same PAA version as the SAX, but modifies it in order to extract two additional characteristic values from each subsequence that is extracted by the sliding window [18]. With these two additional characteristic values, the trend of the points of a subsequence shall be captured.

Main Procedure

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$ that follows the standard normal distribution $X \sim \mathcal{N}(0, 1)$. The first step of discretizing X based on the eSAX is to apply a modified version of the PAA on X [18]. In addition to computing the mean, the minimum as well as the maximum point of the points in each sliding window are retrieved. Hence, the resulting PAA representation of this modified version can be represented by $X' = \{min_1, \overline{x}_1, max_1\}, ..., \{min_n, \overline{x}_n, max_n\}$, where $1 \leq n \leq N$ and $\{min_i, \overline{x}_i, max_i\}$ $(1 \leq i \leq n)$ is the minimum point, mean, and maximum point of the points of the i-th extracted subsequence by the sliding window, respectively.
In the next step of the eSAX, these computed means along with the minima and maxima are then discretized analogous to the SAX discretization based on Equation 3.2 [18]. The resulting time series after this step can then be represented by $X'' = \{\hat{min}_1, \hat{x}_1, \hat{max}_1\}, ..., \{\hat{min}_n, \hat{x}_n, \hat{max}_n\}$, where $\{\hat{min}_i, \hat{x}_i, \hat{max}_i\}$ are the alphabet symbols for the minimum point, mean, and maximum point, respectively. Thus, the eSAX accounts for extreme points within a subsequence that are middled out in the SAX discretization (see Figure 3.7) [18].
However, X'' is not the final discretized eSAX representation, because the positions of $\hat{min}_i$ and $\hat{max}_i$ in the eSAX representation shall correspond to the positions of the corresponding min_i and max_i within X with respect to the point in time they occur [18].
Therefore, the last step of the eSAX involves sorting each $\{\hat{min}_i, \hat{x}_i, \hat{max}_i\}$ according to the points in time of the corresponding $\{min_i, \overline{x}_i, max_i\}$ within X [18]. Thus, the final discretized eSAX representation can be represented by $\hat{X} = sort(\{\hat{min}_1, \hat{x}_1, \hat{max}_1\}), ..., sort(\{\hat{min}_n, \hat{x}_n, \hat{max}_n\})$ (see Figure 3.8), where the function *sort* is described by [18]:

1. Compute the points $t(min_i)$, $t(\overline{x}_i)$, and $t(max_i)$ in time within X. The point in time of $\overline{x}_i$ is computed by

$$t(\overline{x}_i) = \frac{t(s_i^1) + t(s_i^w)}{2},$$

where $t(s_i^1)$ and $t(s_i^w)$ is the starting and ending point in time of the i-th extracted subsequence within X based on a window length of $w \geq 1$.
2. Sort $\hat{min}_i$, $\hat{max}_i$, and $\hat{x}_i$ in ascending order based on the corresponding computed points in time in 1.
3. Return the sorted values.

Since min_i and max_i can be arbitrarily located in the i-th extracted subsequence of X, $sort(\{\hat{min}_i, \hat{x}_i, \hat{max}_i\})$ can have $3! = 6$ different return values, namely every permutation of $\{\hat{min}_i, \hat{x}_i, \hat{max}_i\}$ [18].

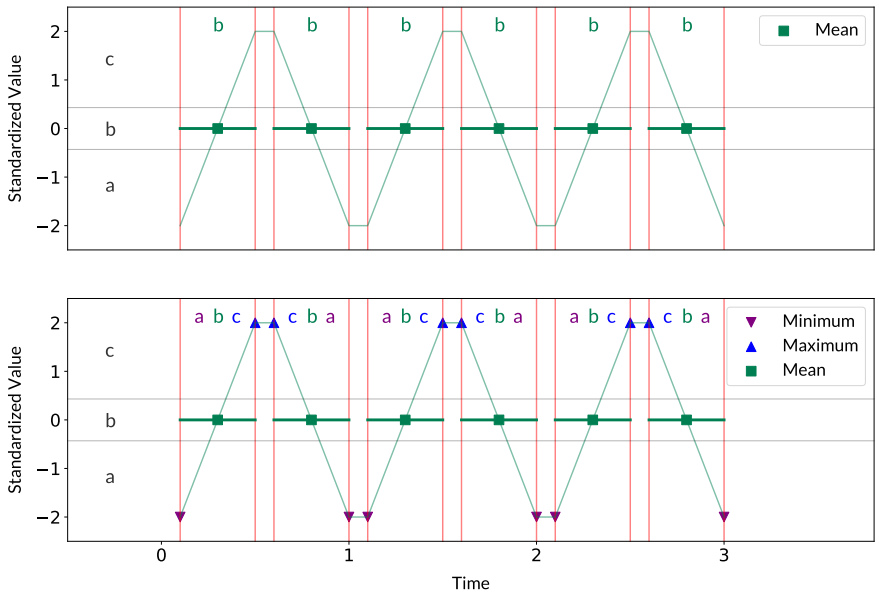


Figure 3.7: Based on the alphabet symbols a, b, c and a window length of $w = 5$, the SAX representation in the above plot is b b b b b b for the original time series. On the other hand, the eSAX representation in the below plot does not middle out the extreme points within a subsequence and captures the actual pattern of the original time series [18]: abc cba abc cba abc cba.

Time Complexity

In the modified version of the PAA that is used for the eSAX, the minimum and maximum for each extracted subsequence can be computed along with the mean in linear time. Therefore, the time complexity of the modified version of the PAA remains $\mathcal{O}(N)$.
Since for the eSAX three values need to be discretized for each of the n extracted subsequences, the time complexity for the discretization process is $\mathcal{O}(3n \cdot log_2(a-1))$ compared to $\mathcal{O}(n \cdot log_2(a-1))$ for the SAX, where $a \geq 2$ is the fixed alphabet size used for discretizing.
As a last step, the minimum, maximum, and mean needs to be sorted for each of the n extracted subsequences. This can be done in $\mathcal{O}(3 \cdot log_2(3) \cdot n)$.
Therefore, the total time complexity of the eSAX with the time complexity of the PAA is $\mathcal{O}(N)+\mathcal{O}(3n \cdot log_2(a-1))+\mathcal{O}(3 \cdot log_2(3) \cdot n) = \mathcal{O}(N)+\mathcal{O}(n)+\mathcal{O}(n) = \mathcal{O}(N)$, because it is $N \geq n$.

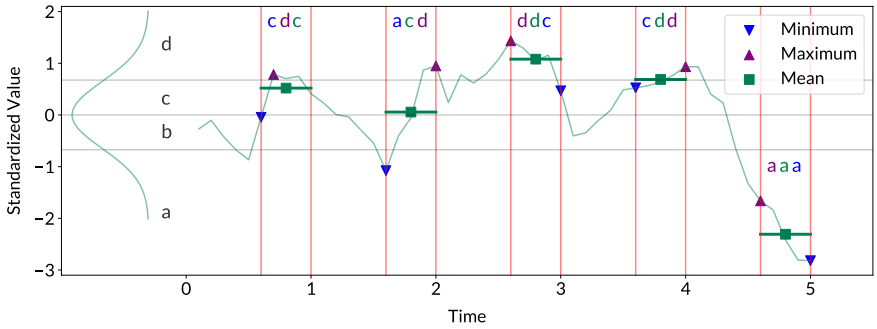


Figure 3.8: For the eSAX, the minimum point, mean, and maximum point per extracted subsequence are discretized analogous to the SAX discretization based on Equation 3.2 [18]. Thus, with the alphabet symbols a, b, c, d and a window length of $w = 5$, the resulting discretized eSAX representation for this plot is: ... cdc ... acd ... ddc ... cdd ... aaa. Note that for visual clarity only every second extracted subsequence is shown in this plot.

3.2.3 1d-Symbolic Aggregate Approximation

Similar to the eSAX, the 1d-Symbolic Aggregate Approximation (1d-SAX) discretization algorithm uses a modified version of the PAA in order to compute one additional characteristic value of each subsequence along with the mean [19]. This additional value shall also capture information about the trend of the points of a subsequence.
But, in comparison to the eSAX and the SAX, this additional value is not discretized based on the same assumption that is made for the discretization of the minimum, mean, and maximum of the points of a subsequence [19].

Main Procedure

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$ that follows the standard normal distribution $X \sim \mathcal{N}(0, 1)$. The first step of discretizing X based on the 1d-SAX is to apply a modified version of the PAA on X [19]. In addition to computing the mean, the trend of the points of each subsequence extracted by the sliding window is computed. This trend is computed based on the slope of the linear regression of the points of the respective subsequence. Let $S_i = s_i^1, ..., s_i^w$ be the i-th $(1 \leq i \leq n)$ extracted subsequence of $1 \leq n \leq N$ extracted subsequences from X with a window length of $w \geq 1$. Further, let $T_i = t(s_i^1), ..., t(s_i^w)$ be the corresponding points in time within X. For the 1d-SAX, only the slope $\tilde{s}_i$ of the linear regression of the points of the i-th extracted subsequence is needed. This can be computed based on the closed form of the ordinary least squares

estimation [19]:

$$\tilde{s}_i = \frac{\sum_{j=1}^{w}(t(s_i^j) - \overline{T}_i)(s_i^j - \overline{S}_i)}{\sum_{j=1}^{w}(t(s_i^j) - \overline{T}_i)^2} \overset{\overline{S}_i=0}{=} \frac{\sum_{j=1}^{w}(t(s_i^j) - \overline{T}_i)s_i^j}{\sum_{j=1}^{w}(t(s_i^j) - \overline{T}_i)^2}, \tag{3.7}$$

where $\overline{S}_i$ and $\overline{T}_i$ represent the means of S_i and T_i, respectively. Note that $\overline{S}_i = 0$, because the linear regression is computed within a sliding window. Therefore, the mean of the points of the respective extracted subsequence by the sliding window is used as the x-axis and all these points are normalized with respect to this x-axis (see Figure 3.9).
Thus, the resulting PAA representation of this modified version can be represented by $X' = (\overline{x}_1, \tilde{s}_1), ..., (\overline{x}_n, \tilde{s}_n)$, where $(\overline{x}_i, \tilde{s}_i)$ is the mean and the slope of the i-th extracted subsequence, respectively.
In the next step of the 1d-SAX, these computed means along with the slopes are then discretized [19]. The discretization of the means and the slopes is done separately. While the means are discretized analogous to the SAX discretization based on Equation 3.2, the slopes are discretized based on a different assumption. It is assumed that the slope values follow a Gaussian distribution with mean 0 variance σ_w^2, where σ_w^2 is a decreasing function of the window length w [19]. The discretization of the slope values is then based on the quantiles of this Gaussian distribution $\mathcal{N}(0, \sigma_w^2)$. Analogous to the SAX discretization, these quantiles are then used as breakpoints for discretizing the slope values based on Equation 3.2 [19]. Note that due to the separate discretization of means and slopes, two possibly different alphabets for discretization can be used [19]. For example, the means can be discretized based on six alphabet symbols, while the discretization of the slopes is based on four alphabet symbols.
The question that now remains is what function σ_w^2 should be used to describe the variance of the Gaussian distribution that is used for discretizing the slopes. Based on the literature, it is recommended to use $\sigma_w^2 = \frac{0.03}{w}$ [19]. Empirical analyses with time series that follow Gaussian distributions found that with this function the Gaussian distribution $\mathcal{N}(0, \sigma_w^2)$ best describes the distribution of the computed slopes. Note that for any window length $w \geq 1$, the values of such a Gaussian distribution are more concentrated around the mean 0 compared to the standard normal distribution, because it is $\sigma_w^2 < 1$ (see Figure 3.10).
Applying the described discretizations for the PAA representation X', results in the discretized 1d-SAX representation of the original time series X that can be represented by $\hat{X} = (\hat{x}_1, \hat{s}_1), ..., (\hat{x}_n, \hat{s}_n)$, where $(\hat{x}_i, \hat{s}_i)$ represents the separately discretized mean and slope of the i-th extracted subsequence, respectively.
Note that for the evaluation in this thesis (see Chapter 5), this representation is used as the 1d-SAX representation of the original time series. However, this representation deviates from the one proposed in the literature [19]. In the literature, the separately discretized means and slopes are represented by bit strings and then interleaved. Consider, for example, $\hat{x}_i := \tilde{0}\tilde{1}$ and $\hat{s}_i := 10$.

Then, the interleaving would result in one bit string that represents $\hat{x}_i$ and $\hat{s}_i$ together: $\tilde{0}1\tilde{1}0$. This representation is not used in order to be consistent with the representations of the other evaluated discretization algorithms to have a standardized representation across these algorithms where a discretized value is represented by one alphabet symbol.

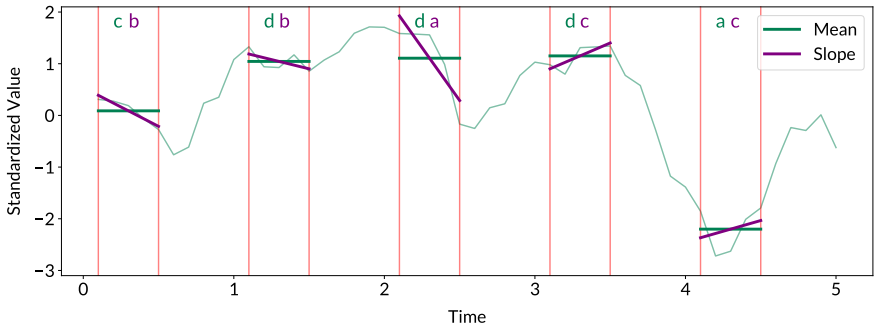


Figure 3.9: For the 1d-SAX, the slope within a subsequence is computed with a linear regression [19]. Both, the mean and slope per extracted subsequence are discretized based on Equation 3.2. The discretized 1d-SAX representation for the original time series in this plot could be `cb ... db ... da ... dc ... ac ...` . Compared to the discretized SAX representation `c ... d ... d ... d ... a ...`, the 1d-SAX representation captures information about the trend of the points of a subsequence. Note that for visual clarity only every second extracted subsequence is shown in this plot.

Time Complexity

For the modified version of the PAA, the slope of the linear regression for an extracted subsequence can be computed along with the mean. Using the closed form of the ordinary least squares estimation described by Equation 3.7, the slope can be computed in $\mathcal{O}(w) = \mathcal{O}(1)$ for a fixed window length w. Therefore, the time complexity of the modified version of the PAA remains $\mathcal{O}(N)$.
Since for the 1d-SAX two values need to be discretized for each of the n extracted subsequences, the time complexity for the discretization process is $\mathcal{O}(2n \cdot log_2(a-1))$ compared to $\mathcal{O}(n \cdot log_2(a-1))$ for the SAX, where $a \geq 2$ is the fixed alphabet size used for discretizing.
Therefore, the total time complexity of the 1d-SAX with the time complexity of the PAA is $\mathcal{O}(N) + \mathcal{O}(2n \cdot log_2(a-1)) = \mathcal{O}(N) + \mathcal{O}(n) = \mathcal{O}(N)$, because it is $N \geq n$.

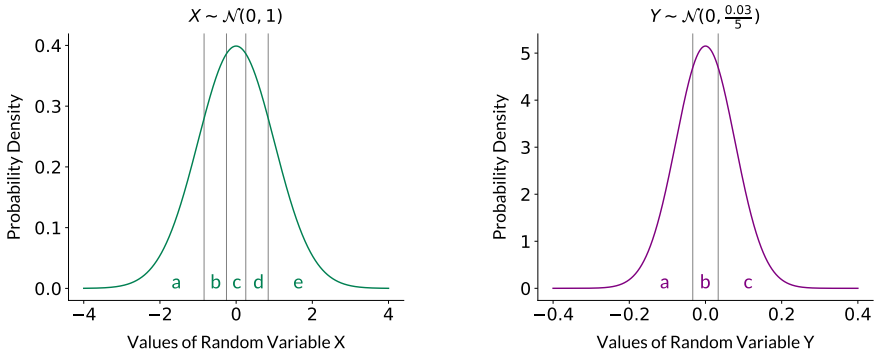


Figure 3.10: The means of the extracted subsequences are discretized based on breakpoints that are the quantiles of the standard normal distribution as on the left plot [19]. Compared to that, the slopes of the extracted subsequences are discretized based on breakpoints that are the quantiles of a Gaussian distribution $\mathcal{N}(0, \frac{0.03}{w})$ with window size w [19]. The right plot shows such a Gaussian distribution for $w = 5$. Further, the discretization granularity controlled by the alphabet size $a \geq 2$ can be different. Based on this figure, the means are discretized with $a = 5$ alphabet symbols, while the slopes are discretized with $a = 3$ alphabet symbols.

3.2.4 Criticism - Assumption of Standard Normal Distribution

For the SAX, eSAX, and 1d-SAX, there are two problems in connection with the assumption that the original time series follows the standard normal distribution $\mathcal{N}(0, 1)$ after applying standardization.

Considerations about Time Series Standardization

The first problem relates to the statement from the literature that a time series follows the standard normal distribution after applying standardization, regardless of the distribution of the corresponding non-standardized time series [14]. The argumentation for this statement is based on an illustration of normal probability plots for eight selected different time series with a length of 128 points each. But, examining standardized time series from the UCR Time Series Classification Archive [20] by testing them for the standard normal distribution based on a statistical test contradicts this statement (see Table 3.1). Thus, this statement is not true. However, given that a non-standardized time series follows a Gaussian distribution, it is proofed that the corresponding standardized time series follows the standard normal distribution [6].

Training Dataset	Number	Length	Fraction
SmoothSubspace	150	15	0.23
SyntheticControl	300	60	0.56
SonyAIBORobotSurface1	20	70	0.60
FordB	3636	500	0.64
ItalyPowerDemand	67	24	0.66
FordA	3601	500	0.66
FaceAll	560	131	0.73
Crop	7200	46	0.74
FacesUCR	200	131	0.76
Plane	105	144	0.84
SonyAIBORobotSurface2	27	65	0.85
SwedishLeaf	500	128	0.88
OSULeaf	200	427	0.96

Table 3.1: For each training dataset from the UCR Time Series Classification Archive [20] that contains equal-length time series of ≤ 5000 points, the standardized time series are tested to follow the standard normal distribution with the Shapiro-Wilk test. Based on this test, this table shows the fraction of standardized time series in each training dataset for that the null hypothesis to follow the standard normal distribution can be rejected, based on a p-value of < 0.05. The 100 remaining examined training datasets where this fraction is ≥ 0.99 are omitted for brevity. All in all, based on the employed Shapiro-Wilk test, 51,745 out of 61,953 examined time series do not follow the standard normal distribution after applying standardization.

Effect of the PAA

Remember that for a window length of $w > 1$, not the original standardized time series points are discretized in the SAX, eSAX, and 1d-SAX. Instead, features like the mean of the points of subsequences that are extracted by the (modified) PAA are discretized. This implies the second problem, that the computation of the breakpoints is only based on the assumption that the original standardized time series follows the standard normal distribution. However, the distribution of the features extracted by the (modified) PAA should also be considered (see Figure 3.11) [21].

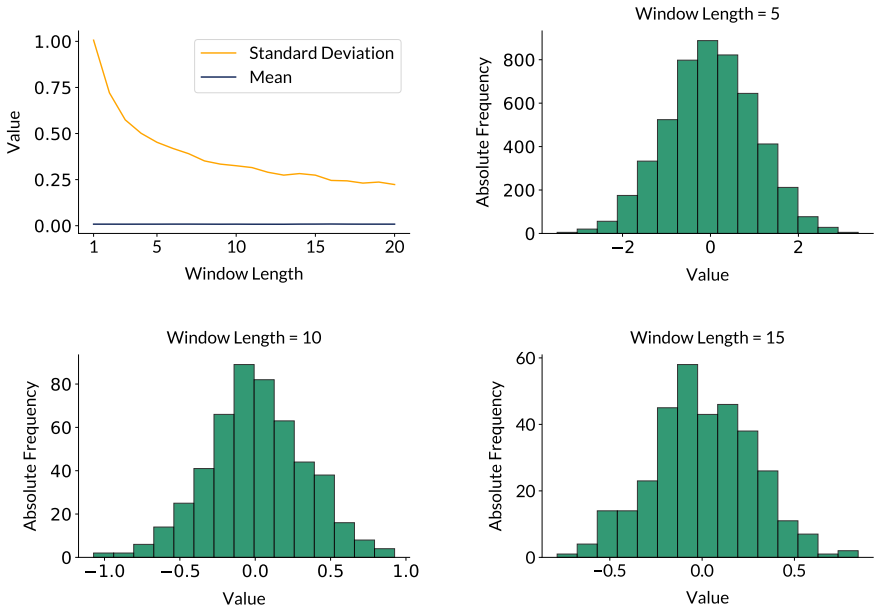


Figure 3.11: For the creation of this figure, the PAA is applied with different window lengths on 5,000 time series points drawn from the standard normal distribution $\mathcal{N}(0, 1)$ [21]. The corresponding distributions of the means extracted by the PAA are examined. The three plotted histograms along with the summary statistics in the upper left plot show that these extracted means approximately follow a Gaussian distribution with a mean of zero and a standard deviation < 1 [21]. Moreover, the standard deviation decreases with an increasing window length. Thus, these means extracted by the PAA do not follow the standard normal distribution.

Impact on Discretization

Since it is the extracted features by the (modified) PAA that are discretized, their distributions directly effect the distribution of the alphabet symbols in the discretized time series [21]. For example, given the distribution of the means extracted by the PAA from a standardized time series that follows the standard normal distribution, the corresponding discretized time series will not contain each alphabet symbol with the same probability (see Figure 3.12) [21]. This contradicts the statement from the literature that the computed breakpoints for discretizing the extracted means in the SAX, eSAX, and 1d-SAX imply equi-probable alphabet symbols in the corresponding discretized time series [14]. The assumption that would need to hold for this statement is not that the points of the original standardized time series follow the standard normal distribution, but the means extracted by the PAA [21].

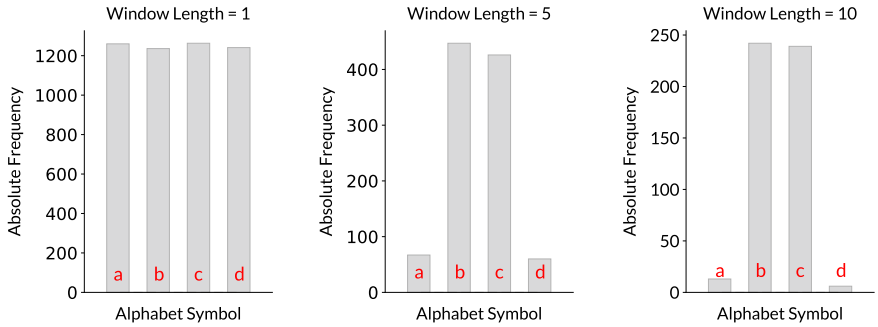


Figure 3.12: For the creation of this figure, the PAA is applied with window lengths of $w = 1$, $w = 5$, and $w = 10$ on 5,000 time series points drawn from the standard normal distribution $\mathcal{N}(0, 1)$ [21]. The means extracted by the PAA are discretized based on the breakpoints computed by the SAX, eSAX, and 1d-SAX for an alphabet size of $a = 4$. The resulting distributions of the alphabet symbols are shown in the three plotted histograms. For a window length of $w = 1$, the alphabet symbols are equi-probable distributed, since the examined time series follows the standard normal distribution. However, for an increasing window length, the alphabet symbols are more concentrated on the two middle alphabet symbols [21]. This also reflects the decreasing standard deviation of the distribution of the corresponding extracted means for an increasing window length.

3.3 Adaptive-breakpoints Discretization

Compared to the fixed-breakpoints discretization algorithms, the following two time series discretization algorithms do not assume that the original standardized time series that shall be discretized follow the standard normal distribution. Even more, they do not make any assumption about the distribution the time series follow [22, 8]. Therefore, they do not compute the breakpoints that define the discretization intervals based on the quantiles of a distribution, but compute them intrinsically based on the time series data at hand [22, 8]. Thus, given a fixed number of discretization intervals to be used, the time series are discretized based on individual breakpoints that are adapted to their data. This is the reason why the following two discretization algorithms are classified into adaptive-breakpoints discretization.

3.3.1 Adaptive Symbolic Aggregate Approximation

Similar to the fixed-breakpoints discretization algorithms, the Adaptive Symbolic Aggregate Approximation (aSAX) discretization algorithm applies amplitude

discretization to the PAA representation to obtain the discretized aSAX representation of the original time series [22]. However, for computing the breakpoints that define the discretization intervals, it employs the k-means clustering algorithm on the points of the time series that shall be discretized [22]. Hence, it takes the time series data into account for discretization and computes adapted breakpoints, respectively.

Main Procedure

Let $DB := \{X \mid X = x_1, ..., x_N\}$ be a time series database containing standardized time series of length $N \geq 1$. Further, define $P := \{\frac{j}{a} \mid j \in \{1, ..., a-1\}\}$ for $a \geq 2$. Applying the quantile function of the standard normal distribution to the probabilities in P results in the a-quantiles β_j $(1 \leq j \leq a-1)$ of the standard normal distribution. The ascending sorted list of these a-quantiles $B := (\beta_0, \beta_1, ..., \beta_{a-1}, \beta_a)$ with $\beta_0 = -\infty$ and $\beta_a = +\infty$ are then the breakpoints used for discretization in the SAX based on Equation 3.2 and an alphabet size of a [15].
The aSAX now employs the k-means clustering algorithm in one dimension to adapt these breakpoints B, such that the adapted breakpoints B^* reflect the distributions of the time series in DB (see Figure 3.13) [22]. For this, a part of the time series in DB are transformed into their PAA representations and taken as a training set [22]. Let $DB_{Tr} := \{\overline{X} \mid X \in DB, \overline{X} = \overline{x}_1, ..., \overline{x}_n\}$ be such a training set, where $\overline{x}_i$ $(1 \leq i \leq n)$ is the mean of the i-th extracted subsequence based on the PAA with $1 \leq n \leq N$ extracted subsequences from each $X \in DB$ that is used for the training set. Then, the employed k-means clustering algorithm shown in Algorithm 3.1 uses B as initial breakpoints and clusters all means $\overline{x}_i$ of all PAA representations in DB_{Tr} into $k = a$ clusters [22]. Based on the computed cluster centers, it computes and eventually returns the adapted breakpoints B^* (see Figure 3.14).
These adapted breakpoints are then used for transforming each time series $X \in DB$ into its discretized aSAX representation [22]. First, each time series that is not in DB_{Tr} is also transformed into its PAA representation. Then, the discretization for each time series is performed analogous to the SAX based on its PAA representation and Equation 3.2, but instead of using B as breakpoints, the adapted breakpoints B^* are used.

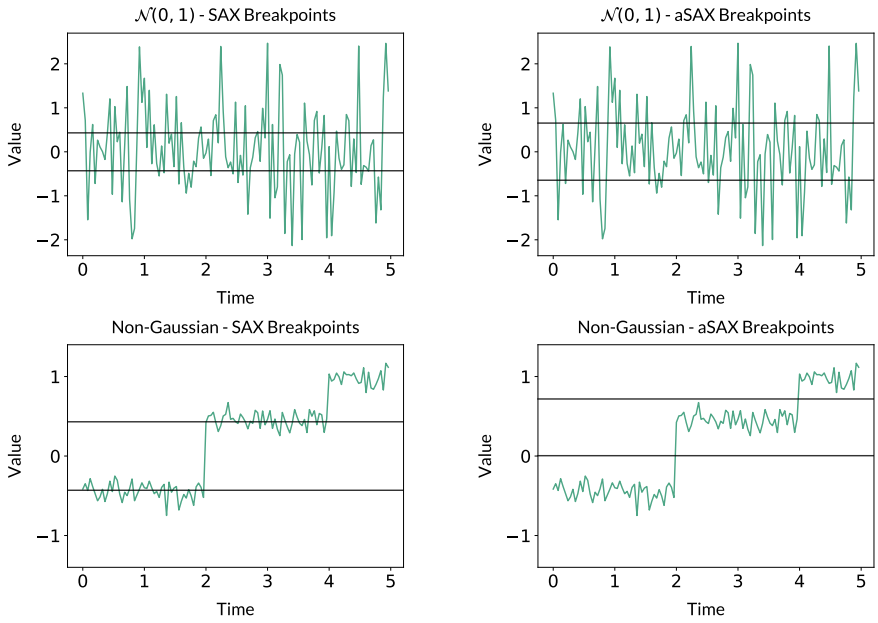


Figure 3.13: In the above two plots, 125 time series points drawn from the standard normal distribution $\mathcal{N}(0, 1)$ are shown. In the below two plots, the plotted time series is composed of 50 + 50 + 25 points drawn from $\mathcal{N}(-0.45, 0.1)$, $\mathcal{N}(0.45, 0.1)$, and $\mathcal{N}(1, 0.1)$, respectively. Hence, this time series follows a non-Gaussian distribution. On the left side of this figure, the breakpoints based on the SAX are plotted, while on the right side, the breakpoints based on the aSAX are plotted. The breakpoints of the aSAX adapt to the non-Gaussian distribution by separating the points drawn from the three different distributions. On the other hand, the breakpoints of the SAX are not able to separate these points.

Input: k // number of clusters that shall be found
DB_{Tr} // means of PAA representations
B // initial breakpoints
$\gamma > 0$ // threshold for relative training error
Output: B^* // adapted breakpoints

1 $\Delta \leftarrow \infty$ // training error of previous clustering
2 $B^* \leftarrow B$
3 $C \leftarrow ()$ // cluster centers
4 **for** $i \leftarrow 0$ **to** $k-1$ **do**
5 $b_i \leftarrow [\beta_i, \beta_{i+1})$ // $\beta_i, \beta_{i+1} \in B^*$
6 $n_i \leftarrow \sum_{\overline{x} \in b_i} 1$ // number of all $\overline{x} \in DB_{Tr}$ in cluster b_i
7 $s_i \leftarrow \sum_{\overline{x} \in b_i} \overline{x}$ // sum of all $\overline{x} \in DB_{Tr}$ in cluster b_i
8 $c_i \leftarrow \frac{1}{n_i} s_i$ // cluster center of the ith cluster
9 $C(i) \leftarrow c_i$
10 **end**
11 **for** $i \leftarrow 1$ **to** $k-1$ **do**
12 $\beta_i \leftarrow (c_{i-1} + c_i)$ / 2 // new breakpoint, $c_{i-1}, c_i \in C$
13 $B^*(i) \leftarrow \beta_i$
14 **end**
15 $\Delta' \leftarrow 0$ // training error of current clustering
16 **for** $i \leftarrow 1$ **to** k **do**
17 $b_i \leftarrow [\beta_i, \beta_{i+1})$ // $\beta_i, \beta_{i+1} \in B^*$
18 $e_i \leftarrow \sum_{\overline{x} \in b_i} (\overline{x} - c_i)^2$ // training error in cluster b_i, $c_i \in C$
19 $\Delta' \leftarrow \Delta' + e_i$
20 **end**
21 **if** $\frac{\Delta - \Delta'}{\Delta} < \gamma$ **then**
22 return B^*
23 **else**
24 $\Delta \leftarrow \Delta'$
25 goto line 4 // adapt clustering for lower training error
26 **end**

Algorithm 3.1: For the aSAX, the k-means clustering algorithm clusters the means of the PAA representations of the given training time series [22]. Based on the found clustering, it computes adapted breakpoints that are used to transform time series into their respective aSAX representation.

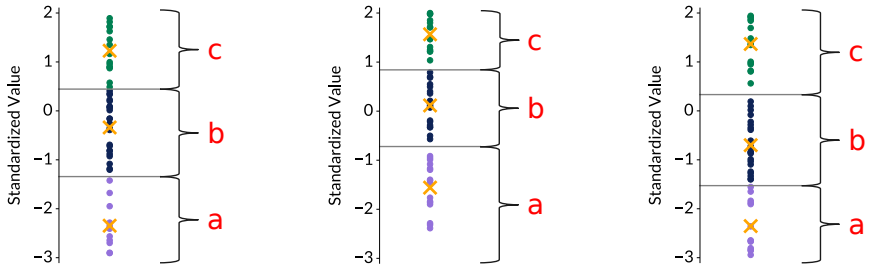


Figure 3.14: The adapted breakpoints that are computed by the k-means clustering algorithm are the midpoints of each two adjacent cluster centers (yellow crosses) [22]. These breakpoints adapt based on the distribution of the time series that shall be discretized. Points with the same color belong to the same cluster.

Modifications

Instead of computing adapted breakpoints B^* for a whole time series database DB, the aSAX can also be used for computing individual adapted breakpoints for each time series $X \in DB$. This can be done by transforming each $X \in DB$ into its PAA representation $\overline{X}$ and applying the k-means clustering algorithm in Algorithm 3.1 for each PAA representation $\overline{X}$ individually. With this modification, each time series $X \in DB$ has individually adapted breakpoints that solely reflect its own distribution. However, the time and memory requirements for computing and storing these individual breakpoints are higher compared to computing adapted breakpoints for a whole time series database.
A further modification is to use a part of the time series $DB' \subset DB$ as a training set without transforming them into their PAA representations [22]. So, instead of clustering the means of their PAA representations, the original points of the time series in the training set would be clustered for computing the adapted breakpoints B^*. While the k-means clustering algorithm in Algorithm 3.1 does not need to be modified, except for using DB' instead of DB_{Tr} as input, the recommended approach in the literature is to use DB_{Tr} as the training set [22]. The advantage of using DB_{Tr} instead of DB' is that the k-means clustering algorithm needs to cluster less points for $n < N$. Thus, its time requirements are lower.

Unsupervised Discretization

Since the aSAX computes the breakpoints used for discretization based on the k-means clustering algorithm, it can also be used for unsupervised discretization [23]. This means that the alphabet size a is no longer an input parameter of the aSAX, but can be determined intrinsically. In this case, the aSAX only requires a

single input parameter which is the window length $1 \leq w \leq N$ that is used for the PAA.
The question that now remains is how to intrinsically determine an appropriate alphabet size for the aSAX. One possibility is to apply a decision criterion for the k-means clustering algorithm shown in Algorithm 3.1 [23]. Let $A^* := (a_{min}, ..., a_{max})$ be an ascending sorted list of different alphabet sizes, where $a_{min} \geq 2$ is the minimum and $a_{max} \geq a_{min}$ is the maximum alphabet size. Then, Algorithm 3.1 can be run for each alphabet size in A^* while computing the value of the decision criterion for the corresponding resulting clustering. This way, the best alphabet size $a^* \in A^*$ is the one that was used for the resulting clustering that achieved the best value based on the used decision criterion. Thus, the breakpoints that were computed for the alphabet size a^* can then be used for discretizing the respective time series.
It still remains open what decision criterion should be used. One possibility is to use the SSE that is computed with the for-loop starting at line 16 in Algorithm 3.1. Using the SSE, the elbow method can be applied for finding a^* by plotting the value of the SSE that was achieved for each alphabet size $a \in A^*$ the k-means clustering algorithm was run (see Figure 3.15) [24].

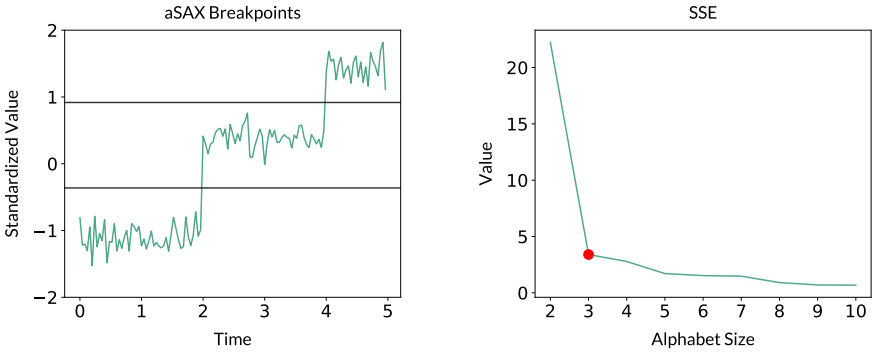


Figure 3.15: The time series in the left plot is composed of 50 + 50 + 25 points drawn from $\mathcal{N}(-0.4, 0.1)$, $\mathcal{N}(0.4, 0.1)$, and $\mathcal{N}(1, 0.1)$, respectively, with a subsequent standardization of the composed time series. Then, the k-means clustering algorithm shown in Algorithm 3.1 is run for different alphabet sizes $a \in (2, ..., 10)$ to cluster the points of the plotted time series. The corresponding values of the SSE are shown in the right plot. Using the elbow method, the plotted time series should be discretized based on an alphabet size of $a^* = 3$. The corresponding breakpoints for discretization in the left plot are adapted to the three distributions the time series points were drawn from.

Time Complexity

The first step of the aSAX without any modifications is to compute the initial parameters for the k-means clustering algorithm in Algorithm 3.1. The computation of DB_{Tr} involves transforming a fixed number $0 < r \leq |DB|$ of the time series in DB into their PAA representations. This can be done in $\mathcal{O}(r \cdot N) = \mathcal{O}(N)$, since the time complexity of the PAA for one time series is $\mathcal{O}(N)$. Further, as explained for the SAX, the time complexity of computing the initial breakpoints B is $\mathcal{O}(1)$. Hence, the total time complexity of computing the initial parameters for the k-means clustering algorithm is $\mathcal{O}(N)$.
The next step is the computation of the adapted breakpoints B^* based on the k-means clustering algorithm. This computation has a total time complexity of $\mathcal{O}(iters \cdot k \cdot n) = \mathcal{O}(iters \cdot n)$, where $k = a$ is the given number of clusters that shall be computed and $iters \geq 1$ is the number of iterations until the algorithm converges. This total time complexity follows, because for a single iteration, the first and third for-loop of Algorithm 3.1 both have a time complexity of $\mathcal{O}(k \cdot n) = \mathcal{O}(n)$, while the second for-loop has a time complexity of $\mathcal{O}(k) = \mathcal{O}(1)$.
The final discretization step of the aSAX is analogous to the SAX except for using the computed adapted breakpoints B^* instead of B. Therefore, the time complexity for this step is $\mathcal{O}(n)$ as described for the SAX.
Thus, the total time complexity of the aSAX without any modifications is $\mathcal{O}(N) + \mathcal{O}(iters \cdot n) + \mathcal{O}(n) = \mathcal{O}(N + iters \cdot n)$.

3.3.2 Persist

Similar to the aSAX, the Persist discretization algorithm takes the time series data into account for discretization and computes adapted breakpoints, respectively. However, these adapted breakpoints are not computed based on the k-means clustering algorithm, but on a decision criterion called persistence score [8].
Moreover, in its unmodified version, the Persist applies amplitude discretization to the original standardized time series instead of the PAA representation to obtain the discretized Persist representation [8]. But, modifications to use the PAA representation are possible as described below.

Main Procedure

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. The Persist uses a decision criterion, namely the persistence score described below, to compute

the best breakpoints based on this decision criterion for discretizing X [8].
The first step is to determine an ascending sorted list of $l \geq 1$ candidate breakpoints $B := (\beta_j \mid j \in \{1, ..., l\})$ from which the Persist selects the best [8]. In the literature, equal frequency binning of the points of X with a default value of 100 bins (i.e. $l = 99$) is proposed to obtain the percentiles of the time series points as candidate breakpoints [8]. Taking these percentiles as candidate breakpoints results in a finer sampling in regions with relatively many points compared to a coarser sampling in regions with relatively few points.
Now, let $A := \{\alpha_j \mid j \in \{1, ..., a\}\}$ be the alphabet used for discretizing X based on $a \geq 2$ alphabet symbols (e.g. letters from the Latin alphabet). Then, the second step is to define a discretization function $f : X \times (2^B \setminus \varnothing) \rightarrow A$ that discretizes all time series points of X given a sorted list of $1 \leq k \leq l$ breakpoints from B [8]. Such a discretization function f could be defined, for example, based on the SAX discretization described by Equation 3.2, where the original time series points of X instead of the means of its PAA representation are used.
Based on these two initial steps, the Persist algorithm shown in Algorithm 3.2 builds up the ascending sorted list of best breakpoints iteratively by selecting one of the candidate breakpoints in B in each of the $a - 1$ iterations [8]. In each iteration, it adds all available candidate breakpoints in B individually to the current list of best breakpoints. Based on this extended list of breakpoints, it computes the persistence score by applying the discretization function f on X. Then, the candidate breakpoint with the highest persistence score is added to the current list of best breakpoints and the next iteration starts.

Input: $X = x_1, ..., x_N$ // standardized time series of length $N \geq 1$
$B = (\beta_j \mid j \in \{1, ..., l\})$ // $l \geq 1$ candidate breakpoints
f // discretization function
a // alphabet size $a \geq 2$
Output: B^* // ascending sorted breakpoints

1 $B^* \leftarrow ()$
2 **for** $i \leftarrow 1$ **to** $a - 1$ **do**
3 $P \leftarrow ()$ // persistence scores
4 **foreach** $\beta \in B$ **do**
5 $\hat{X} \leftarrow f(X, B^* \cup (\beta))$ // discretized time series
6 $P \leftarrow P \cup (Persistence(\hat{X}))$
7 **end**
8 $j^* \leftarrow argmax(P)$
9 $B^* \leftarrow B^* \cup (\beta_{j^*})$ // breakpoint with highest persistence score
10 $B \leftarrow B \setminus (\beta_{j^*})$
11 **end**
12 return B^*

Algorithm 3.2: The Persist computes the best breakpoints for discretizing X based on a decision criterion called persistence score that is described below [8]. Note that selecting breakpoints that are near to each other can be avoided by additionally excluding $r \geq 1$ of the available adjacent breakpoints of each β_{j^*} in line 10 of this algorithm [8].

Persistence Score

Define $A := \{\alpha_j \mid j \in \{1, ..., a\}\}$ as an alphabet with $a \geq 2$ alphabet symbols (e.g. letters from the Latin alphabet). Assume that $\hat{X} = \hat{x}_1, ..., \hat{x}_n$ is a discretized time series of length $n \geq 1$, where $\hat{x}_i \in A$ $(1 \leq i \leq n)$. Let $p(\alpha_j)$ $(1 \leq j \leq a)$ be the marginal probability of the alphabet symbol α_j in $\hat{X}$. Further, the $a \times a$ matrix containing the transition probabilities with respect to the alphabet symbols in $\hat{X}$ shall be represented by $M(j, k) = p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1} = \alpha_k)$ [8]. In this matrix, the self-transition probabilities $p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1} = \alpha_j)$ are the values on the main diagonal $M(j, j)$.
Now, the first case to consider is when the time series does not have any temporal structure [8]. Then, the alphabet symbols α_j of $\hat{X}$ can be interpreted as they were generated by a random variable based on the marginal probabilites $p(\alpha_j)$. Thus, the probability of observing a symbol at a point in time in $\hat{X}$ does not depend on the previous symbol. Therefore, the transition probabilities become $M(j, k) = p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1} = \alpha_k) = p(\alpha_j)$ the marginal probabilities.

The other case to consider is a temporal structure described by a first order Markov model [8]. In this model, the probability of observing an alphabet symbol α_j at a point in time in $\hat{X}$ depends on the previous symbol. Hence, this probability is described by the transition probability $p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1})$.
The comparison of these two cases is used to measure the persistence of the time series $\hat{X}$ [8]. Without any temporal structure in this time series, the transition probabilities of the Markov model $p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1})$ are expected to not deviate too much from the corresponding marginal probabilities $p(\alpha_j)$, since there is no dependence on the previous symbol.
On the other hand, when the time series has a temporal structure, there are two cases [8]. First, larger self-transition probabilities $p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1} = \alpha_j)$ than corresponding marginal probabilities $p(\alpha_j)$ indicate a persisting behavior of the time series. Second, lower self-transition probabilities than corresponding marginal probabilities at a point in time indicate that the next alphabet symbol in the time series is more likely than unlikely to be different compared to the current symbol. Note that the Persist does not assume the time series $\hat{X}$ to be generated by a first order Markov model [8]. Only the self-transition probabilities based on this model are used as an indicator for a persisting behavior of the time series.
Thus, the self-transition probabilities $p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1} = \alpha_j)$ as well as the corresponding marginal probabilities $p(\alpha_j)$ are of importance for measuring the persistence of the time series $\hat{X}$ [8]. Based on these probabilities, two discrete probability distributions for each alphabet symbol α_j are created [8]. The first distribution for alphabet symbol α_j is based on the self-transition probability $M(j,j) = p(\hat{x}_i = \alpha_j \mid \hat{x}_{i-1} = \alpha_j)$ and the complementary probability $M(j,j)^c := 1 - M(j,j)$. Hence, define this distribution as $P_j := \{M(j,j), M(j,j)^c\}$. The second distribution for alphabet symbol α_j is based on the marginal probability $p(\alpha_j)$ and the complementary probability $p(\alpha_j)^c := 1 - p(\alpha_j)$. Hence, define this distribution as $Q_j := \{p(\alpha_j), p(\alpha_j)^c\}$.
Based on the two discrete probability distributions P_j and Q_j, the persistence score for each alphabet symbol α_j is computed by [8]:

$$Persistence(\alpha_j) := sgn(M(j,j) - p(\alpha_j)) \cdot SKL(P_j, Q_j), \tag{3.8}$$

where SKL is the symmetric Kullback-Leibler divergence (see Subsection 2.1.3) and sgn is the sign function. Alphabet symbols α_j with larger self-transition probabilities $M(j,j)$ than corresponding marginal probabilities $p(\alpha_j)$ are assigned a positive persistence score, while alphabet symbols with lower self-transition probabilities than corresponding marginal probabilities are assigned a negative persistence score, since in these cases it is $SKL > 0$. Further, it is $Persistence(\alpha_j) = 0$ if and only if the corresponding discrete probability distributions P_j and Q_j are equal [8] (see Figure 3.16).
The final persistence score for $\hat{X}$ used in the Persist is then defined by the mean

persistence score across all alphabet symbols [8]:

$$Persistence(\hat{X}) := \frac{1}{a}\sum_{j=1}^{a} Persistence(\alpha_j) \tag{3.9}$$

Remember that the higher the persistence score of a discretized time series the longer the time intervals containing only one alphabet symbol. Thus, for a discretized time series to increase the likelihood of observing the same alphabet symbol at any point in time as for the previous point in time, almost all alphabet symbols need to have high persistence scores, since the mean persistence score is computed [8].

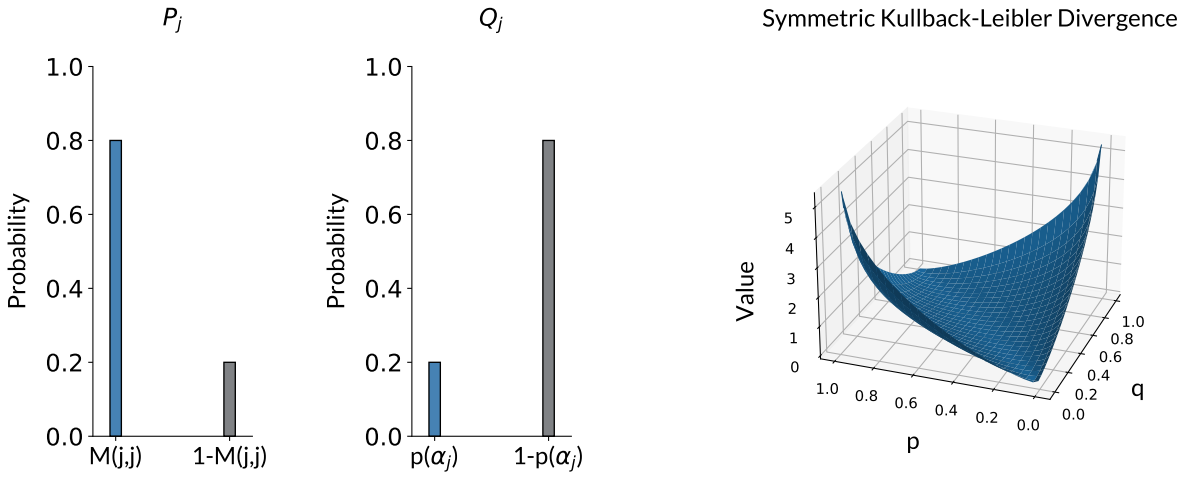


Figure 3.16: On the left, exemplary discrete probability distributions P_j and Q_j for alphabet symbol α_j are plotted. On the right, the value of the SKL of two discrete probability distributions $P = \{p, 1-p\}$ and $Q = \{q, 1-q\}$ for any $0 < p, q < 1$ is plotted [8]. This value increases the greater the distance $|p - q|$ becomes and is zero if and only if $p = q$ holds.

Considerations for Implementing the Persistence Score

Maximum likelihood estimators for the marginal probabilities and the transition probabilities in the matrix M can be obtained by counting the number of alphabet symbols in $\hat{X}$ [8]. For example, for the marginal probability $p(\alpha_j)$ this involves counting the occurrences of alphabet symbol α_j in $\hat{X}$. For the transition probability $M(j,k) = p(\alpha_j \mid \alpha_k)$ this involves counting the number of times α_j follows α_k in $\hat{X}$ (see Figure 3.17).
However, due to the computation of the (symmetric) KL, a problem arises if there

is any α_j where $M(j,j) = p(\alpha_j \mid \alpha_j) = 0$ holds, because the Kullback-Leibler divergence is not defined if any probability used is zero [8]. $M(j,j) = 0$ can happen if α_j never follows itself in $\hat{X}$. Note that for the marginal probabilities involved in the computation of the Kullback-Leibler divergence, $0 < p(\alpha_j) < 1$ holds for any α_j in the Persist, because every discretization interval contains at least one original time series point [8].
For cases with $M(j,j) = 0$, a small value $\epsilon > 0$ (e.g. n^{-1}) can be added to $M(j,j)$ and subtracted from the complementary probability $M(j,j)^c = 1 - M(j,j)$ [8]. As long as the smallest observed probability > 0 used in the computation of the Kullback-Leibler divergence is larger than ϵ, the persistence score can still be used as the decision criterion in the Persist [8].

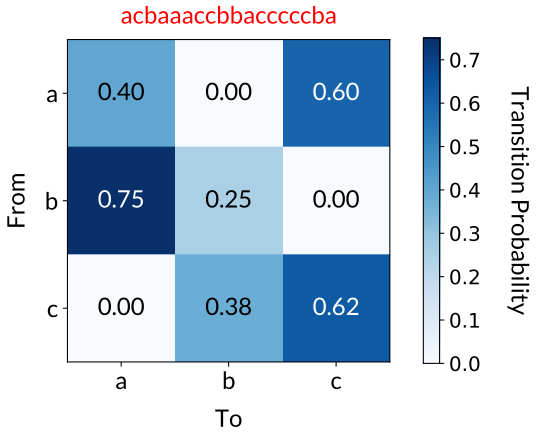


Figure 3.17: This is the transition probability matrix for the discretized time series $\hat{X} =$ acbaaaccbbacccccba. The self-transition probabilities are the values on the main diagonal [8]. For example, M(a,a) $= p$(a|a) $= 0.40$ is the result of dividing two occurrences in $\hat{X}$ where the alphabet symbol a follows the alphabet symbol a (itself) by five occurences where the alphabet symbol a is followed by any alphabet symbol (including a).

Modifications

In the literature, the Persist is presented to discretize the original points of the standardized time series X, rather than the means $\overline{x}_i$ $(1 \leq i \leq n)$ of the corresponding PAA representation $\overline{X} = \overline{x}_1, ..., \overline{x}_n$, where $1 \leq n \leq N$ [8]. While this version of the Persist can also be interpreted as the discretization of the PAA representation based on a window length of $w = 1$, a modified version based on a window length of $w > 1$ could be applied.
With such a modified version, the input for Algorithm 3.2 would be the PAA

representation $\overline{X}$ instead of the original standardized time series X. Moreover, in line 5, $\overline{X}$ instead of X would be discretized. Applying this modified version, the Persist would be more time-efficient, because from the window length of $w > 1$ it follows $n < N$.
Moreover, the PAA representation $\overline{X}$ could also be used for computing the candidate breakpoints B used for initializing Algorithm 3.2. Instead of computing the percentiles of the points of X, the percentiles of the means of $\overline{X}$ could be computed and used as candidate breakpoints. This would further increase the time-efficiency of the Persist.

Unsupervised Discretization

Similar to the aSAX, the Persist can also be used for unsupervised discretization [8]. In the first place, this means that the alphabet size a is no longer an input parameter of the Persist, but can be determined intrinsically. Secondly, when using the unmodified version of the Persist, the need for the window length as an input parameter is eliminated as well, since the PAA representation is not needed for both finding candidate breakpoints and discretization. Thus, with these two considerations, it is possible to discretize time series based on the Persist without providing any input parameter.
In comparison to the aSAX, the Persist uses the persistence score instead of a clustering evaluation metric like the SSE as a decision criterion for determining the alphabet size intrinsically [8]. Let $A^* := (a_{min}, ..., a_{max})$ be an ascending sorted list of different alphabet sizes, where $a_{min} \geq 2$ is the minimum and $a_{max} \geq a_{min}$ is the maximum alphabet size. Then, Algorithm 3.2 can be run for each alphabet size in A^* while computing the persistence score for the corresponding resulting discretization. This way, the best alphabet size $a^* \in A^*$ is the one that was used for the resulting discretization that achieved the highest persistence score [8]. Thus, the breakpoints that were computed for the alphabet size a^* can then be used for discretizing the respective time series (see Figure 3.18).

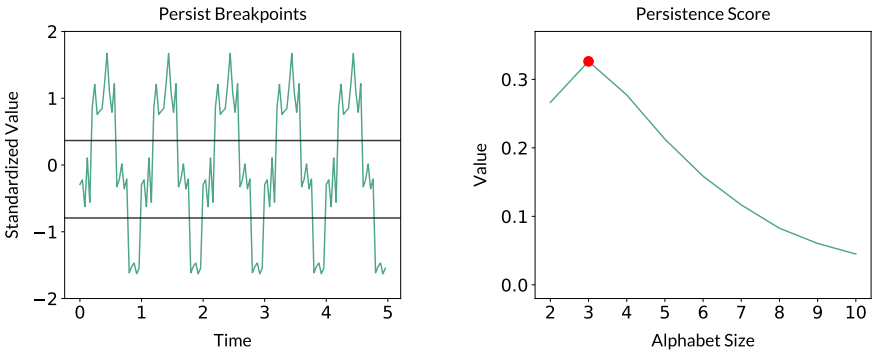


Figure 3.18: The time series in the left plot is a pattern that is repeated 5 times. The pattern is composed of 5 + 10 + 5 + 5 points drawn from $\mathcal{N}(0, 0.3)$, $\mathcal{N}(2, 0.4)$, $\mathcal{N}(0, 0.3)$, and $\mathcal{N}(-2, 0.2)$, respectively, with a subsequent standardization of the composed time series. Then, the Persist shown in Algorithm 3.2 is run for different alphabet sizes $a \in (2, ..., 10)$. The corresponding values of the persistence score are shown in the right plot. Based on the persistence score, the plotted time series should be discretized using an alphabet size of $a^* = 3$. The corresponding breakpoints for discretization in the left plot are adapted to the three different distributions the time series points were drawn from.

Time Complexity

For the initialization of the Persist without any modifications, the candidate breakpoints B need to be computed as the percentiles of the points of the time series X. Sorting these points can be done in $\mathcal{O}(N \cdot log_2(N))$. The subsequent retrieval of each of the fixed number of percentiles needs constant time. Therefore, the time complexity for this initialization step is $\mathcal{O}(N \cdot log_2(N))$.
Starting at line 4 in Algorithm 3.2, all available candidate breakpoints in B are iterated. The first step in such an iteration involves adding the current candidate breakpoint to the sorted list of current best breakpoints B^*. This can be done with a binary search in $\mathcal{O}(log_2(a-1)) = \mathcal{O}(1)$ for a fixed alphabet size $a \geq 2$, as it is $|B^*| \leq a - 1$.
The next step in such an iteration is to discretize X based on the SAX discretization described by Equation 3.2 and the previously extended list of current best breakpoints. As described for the SAX, the time complexity of this discretization step is $\mathcal{O}(N \cdot log_2(a-1)) = \mathcal{O}(N)$.
Based on the resulting discretized time series $\hat{X}$, the next step is to compute the persistence score. As described above, this first involves counting the number of alphabet symbols in $\hat{X}$ to compute the marginal and transition probabilities. This can be done by iterating in $\mathcal{O}(N)$. Then for each alphabet symbol in $\hat{X}$, the persistence score based on the SKL of the two described discrete probability distributions P and Q needs to be computed. The total time complexity for this

computation is $\mathcal{O}(a \cdot 2) = \mathcal{O}(1)$ as both probability distribution have two possible outcomes and $|B^*| \leq a - 1$. Taking the mean of the persistence scores of the alphabet symbols results in the persistence score of $\hat{X}$ and has a time complexity of $\mathcal{O}(a) = \mathcal{O}(1)$. The persistence score of $\hat{X}$ can then be appended to the previously computed persistence scores in constant time. All in all, the computation of the persistence score of $\hat{X}$ is dominated by counting the number of alphabet symbols in $\hat{X}$, which results in a time complexity of $\mathcal{O}(N)$.
Thus, the time complexity of an iteration for a candidate breakpoint in B is $\mathcal{O}(1) + \mathcal{O}(N) + \mathcal{O}(N) = \mathcal{O}(N)$. Therefore, iterating over a fixed number of $|B|$ candidate breakpoints has a time complexity of $\mathcal{O}(|B| \cdot N) = \mathcal{O}(N)$
Starting at line 8 in Algorithm 3.2, the last step is to find the candidate breakpoint that results in the highest persistence score. This can be done by iterating over the persistence scores that correspond to the available candidate breakpoints. Hence, the time complexity is $\mathcal{O}(|B|) = \mathcal{O}(1)$.
All in all, iteratively finding the best $a-1$ breakpoints according to the persistence score with Algorithm 3.2, results in a total time complexity of $\mathcal{O}((a-1) \cdot N) = \mathcal{O}(N)$ for the Persist without any modifications and the initialization step [8]. Taking the initialization step into account, results in a total time complexity of $\mathcal{O}(N \cdot log_2(N)) + \mathcal{O}(N) = \mathcal{O}(N \cdot log_2(N))$ for the Persist without any modifications.

4 Motif Discovery

Motif discovery is the task of finding all recurrently occurring subsequences in a given time series [4]. More formally, let $X = x_1, ..., x_N$ be a time series of length $N \geq 1$. Then, the task of motif discovery is to find all subsequences $S = s_1, ..., s_M$ of length $1 \leq M < N$ that recurrently occur in X (see Figure 4.1) [4].
The determination of recurrently occurring subsequences in X involves a similarity measurement between subsequences based on a given similarity measure D (e.g. the Euclidean distance) and a given similarity distance $r \geq 0$ [25]. Two subsequences S_1 and S_2 of X are then called matching subsequences if and only if $D(S_1, S_2) \leq r$ [25]. Moreover, trivially matching subsequences of X are defined as matching subsequences whose starting points in time are not further apart than a given $k \geq 0$ points in time [25]. Based on matching subsequences, recurrently occurring subsequences in X can be found as explained for the following motif discovery algorithms. Further note that the recurrent occurrence of a subsequence in a given time series is relative to the given similarity measure and the given similarity distance.

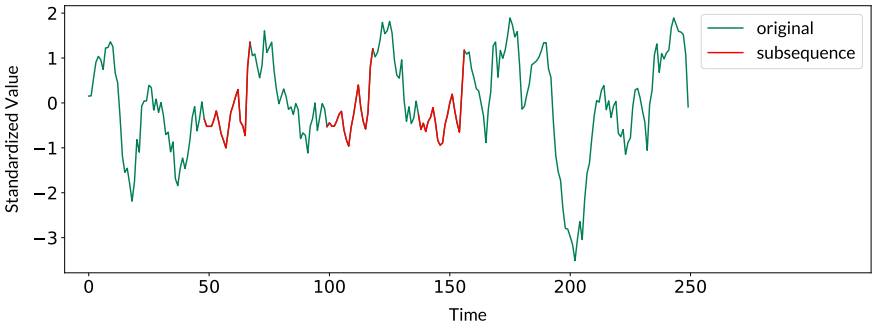
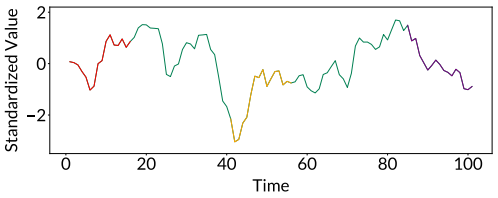


Figure 4.1: The three plotted subsequences of the original standardized time series show a similar pattern. Hence, this pattern represents a recurrently occurring subsequence in the original standardized time series [25].

4.1 Random Projection

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. Then, the first step of the Random Projection motif discovery algorithm is to extract subsequences of X with an overlapping sliding window of a given length $1 \leq l \leq N$ (see Subfigure 4.2a) [26]. These extracted subsequences are then individually discretized based on one of the time series discretization algorithms described in Chapter 3. The selected time series discretization algorithm is employed with the same parameters for each extracted subsequence. This also includes that the adaptive breakpoints used for discretization in the aSAX and Persist are not computed individually for each extracted subsequence, but for the whole standardized time series X. Thus, the discretized subsequences are comparable with each other.
Based on these discretized subsequences, a matrix $\hat{M}_{p \times q}$ $(p, q \geq 1)$ is then created (see Subfigure 4.2b) [26]. In each row, this matrix contains one of the discretized subsequences, while in each column it contains one of the alphabet symbols of the respective discretized subsequence. As each subsequence of X is extracted and discretized based on the same parameters, they all consist of the same number of alphabet symbols. Moreover, the row-wise order of the discretized subsequences within $\hat{M}_{p \times q}$ corresponds to their order of extraction. Such that the discretized subsequence that was extracted first is contained in the first row and the discretized subsequence that was extracted last is contained in the last row.
Based on the constructed matrix $\hat{M}_{p \times q}$, the random projection procedure is employed [26]. First, a given number of $0 \leq s < q$ columns are randomly selected. These randomly selected columns act as a mask as they are hidden and modify $\hat{M}_{p \times q}$ to $\hat{M}_{p \times q'}$ with $q' := q - s$ (see Subfigure 4.2c). Second, a collision matrix $C_{p \times p}$ is created that contains a row and a column for each of the p discretized subsequences (see Subfigure 4.2d). In cell (i, j) $(1 \leq i, j \leq p)$, this collision matrix contains the number of collisions of the remaining alphabet symbols of the discretized subsequences i and j in $\hat{M}_{p \times q'}$, when performing the random selection of s columns $iters \geq 1$ times on $\hat{M}_{p \times q}$. The discretized subsequences i and j collide if their alphabet symbols are equal for each column of $\hat{M}_{p \times q'}$. Note that $C_{p \times p}$ is a symmetric matrix. Further note that for the 1d-SAX and eSAX, s is the number of two respectively three adjacent columns as they use two respectively three alphabet symbols for discretizing a subsequence extracted by the PAA.
Based on the collision matrix $C_{p \times p}$, matching subsequences are likely to have a relatively high number of collisions and a relatively high number of collisions indicates matching subsequences [26]. Therefore, the next step is to use the collision matrix $C_{p \times p}$ as a filter that indicates which extracted subsequences of X shall be examined for finding matching subsequences and which not [26]. First, the extracted subsequences that correspond to the largest number of collisions in $C_{p \times p}$ are retrieved. Let S_1^* and S_2^* be those subsequences. Then, the Euclidean distance

$D_2(S_1^*, S_2^*)$ is computed and compared to a given similarity distance $r \geq 0$. If $D_2(S_1^*, S_2^*) \leq r$ holds, S_1^* and S_2^* are matching subsequences and form a so called tentative motif. This tentative motif is expanded to a motif by adding each other extracted subsequence of X that is a matching subsequence to S_1^* or S_2^*.
To find those other extracted subsequences while avoiding costly memory accesses, the collision matrix $C_{p \times p}$ is used as a filter again [26]. Only those extracted subsequences of X that correspond to discretized subsequences whose number of collisions with the discretized subsequences corresponding to S_1^* or S_2^* is above a given threshold $min\_collisions \geq 1$ are retrieved. Let S_3 be such an extracted subsequence, then it is included into the tentative motif formed by S_1^* and S_2^* if and only if it is a matching subsequence to S_1^* or S_2^* (i.e. $D_2(S_1^*, S_3) \leq r$ or $D_2(S_2^*, S_3) \leq r$). The final motif is then the one that contains S_1^* and S_2^* along with all of their matching subsequences. This motif building procedure is done iteratively, while all extracted subsequences that are already included in a motif cannot be selected for future motifs [26]. In each iteration, the two extracted subsequences corresponding to the highest number of collisions in the current iteration are retrieved until all remaining numbers of collisions are below *min_collisions* or no more extracted subsequences are available to be included in a motif.
Note that for the SAX, even less extracted subsequences may need to be retrieved by applying the procedure based on the $MINDIST$ described in Subsection 3.2.1 for discretized subsequences as an additional filter [26].



(a) Three exemplary subsequences of the standardized time series X that are extracted by an overlapping sliding window.

0	b	a	b	c
...	...	...	...	...
40	a	a	b	b
...	...	...	...	...
85	b	b	b	a

(b) The matrix $\hat{M}_{p \times q}$ with $p = 86$ extracted subsequences that were discretized into $q = 4$ alphabet symbols.

0	b		b	
...	...		...	
40	a		b	
...	...		...	
85	b		b	

(c) For instance, the second and fourth column are randomly selected to be hidden in the first iteration. Then, the first and last discretized subsequence collide.

	0	...	40	...	85
0					
...	...				
40	0	...			
...	...	...	...		
85	1	...	0	...	

(d) The symmetric collision matrix $C_{p \times p}$ with $p = 86$ after the first of $iters$ iterations. It contains the collision of the first and last discretized subsequence.

Figure 4.2: Steps and data structures that are employed by the Random Projection motif discovery algorithm [26].

4.2 Matrix Profile

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. Then, the first step of the Matrix Profile motif discovery algorithm is to discretize X based on one of the time series discretization algorithms described in Chapter 3. Let $\hat{X}$ be the resulting discretized time series corresponding to X. The second step is to encode the alphabet symbols that represent $\hat{X}$ based on the alphabet size $a \geq 2$ used for discretization. For this, the encoding $e(\alpha_j) = j - 1$ $(1 \leq j \leq a)$ is applied on the alphabet symbols α_j that represent $\hat{X}$. Note that for the 1d-SAX, the discretized means and discretized slopes need to be encoded based on the size of the respective alphabet. Let $\hat{X}^e$ be the encoded discretized time series that results from applying the encoding on each alphabet symbol that represents $\hat{X}$.
In the next step, the Matrix Profile procedure is applied on $\hat{X}^e$ [27]. With this procedure, the pairwise distances between any two subsequences of $\hat{X}^e$ based on a given length $l \geq 1$ of the subsequences and a given Minkowski distance D are computed. Compared to other procedures that compute such pairwise distances (e.g. brute force), the Matrix Profile procedure provides a more efficient computation [27]. However, it only works on numerical data, which is the reason $\hat{X}$ needs to be encoded.

Based on the computed pairwise distances between any two subsequences of $\hat{X}^e$, the pairwise mutually nearest neighbors are computed. Two subsequences S_1 and S_2 of $\hat{X}^e$ are pairwise mutually nearest neighbors if both have their smallest distance to the respective other according to the computed pairwise distances. All pairwise mutually nearest neighbors are then sorted in an ascending order with respect to the pairwise distance.
Based on this sorting, the idea of the Random Projection motif discovery algorithm described above is applied [26]. Matching subsequences of X are likely to have a relatively small pairwise distance. Therefore, the next step is to use this sorting as a filter that indicates which subsequences of $\hat{X}^e$ shall be examined for finding matching subsequences of X and which not. First, the two subsequences of $\hat{X}^e$ that represent the pairwise mutually nearest neighbors with the smallest pairwise distance are retrieved. Let S_1^* and S_2^* be those subsequences. If $D(S_1^*, S_2^*) \leq r$ holds for a given similarity distance $r \geq 0$, then S_1^* and S_2^* are declared a tentative motif. This tentative motif is expanded to a motif by adding each other subsequence S_3 with length l of $\hat{X}^e$ that fulfills $D(S_1^*, S_3) \leq r$ or $D(S_2^*, S_3) \leq r$. Such a subsequence S_3 is found by computing the pairwise distance between any other subsequence with length l of $\hat{X}^e$ and S_1^* respectively S_2^* based on the Matrix Profile procedure [27].
The final motif is then the one that contains S_1^* and S_2^* along with all other subsequences S_3 with length l of $\hat{X}^e$ that fulfill $D(S_1^*, S_3) \leq r$ or $D(S_2^*, S_3) \leq r$ (see Figure 4.3) [26]. Based on such a final motif, the corresponding subsequences of X can be retrieved. This motif building procedure is done iteratively, while all subsequences of $\hat{X}^e$ that are already included in a motif cannot be selected for future motifs. In each iteration, the two subsequences of $\hat{X}^e$ that represent the pairwise mutually nearest neighbors with the smallest pairwise distance in the current iteration are retrieved until no more subsequences of $\hat{X}^e$ are available to be included in a motif.
Note that for the 1d-SAX and eSAX, a subsequence of X extracted by the PAA transforms into two respectively three encoded alphabet symbols in $\hat{X}^e$. Therefore, assuming that for $\hat{X}^e$ the points in time start at zero, subsequences of $\hat{X}^e$ that are used for pairwise distance computations should start at a point in time that is divisible by two respectively three. Moreover, l should also be divisable by two respectively three. These two actions ensure comparability between the subsequences of $\hat{X}^e$ when computing pairwise distances. Otherwise, they would not correspond to the subsequences of X extracted by the PAA for the 1d-SAX and eSAX.

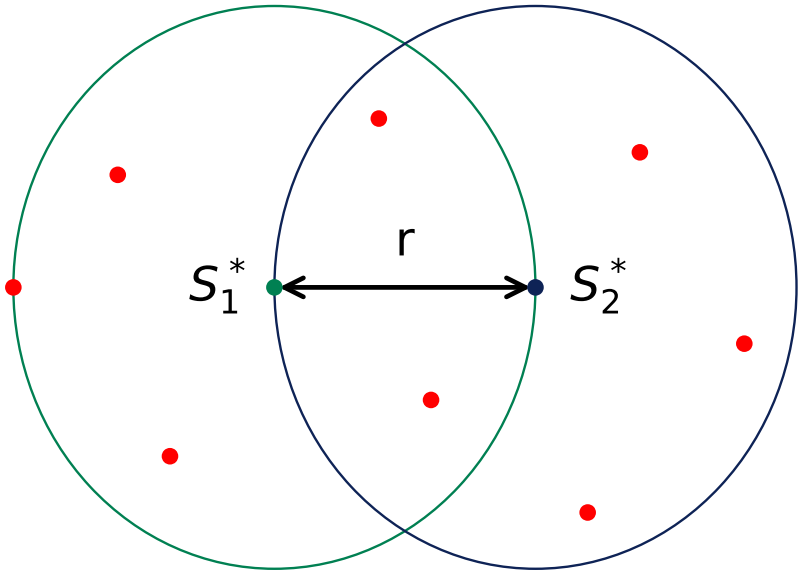


Figure 4.3: A visual intuition for a motif that is discovered by the Matrix Profile motif discovery algorithm [26]. In this plot, $D(S_1^*, S_2^*) = r$ is indicated as an example. Moreover, the red dots indicate subsequences S_3 of $\hat{X}^e$ that fulfill $D(S_1^*, S_3) \leq r$ or $D(S_2^*, S_3) \leq r$ and are therefore included in the motif.

4.3 Brute Force

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. The first two steps of the Brute Force motif discovery algorithm are the same as for the Matrix Profile motif discovery algorithm described above. Therefore, let $\hat{X}^e$ be the encoded discretized time series that results from applying the encoding on each alphabet symbol that represents the discretized time series $\hat{X}$.
The next step is to extract subsequences of $\hat{X}^e$ with an overlapping sliding window of a given length $l \geq 1$ [25]. Analogous to the reasoning for the Matrix Profile motif discovery algorithm described above, extracted subsequences of $\hat{X}^e$ should start at a point in time that is divisable by two respectively three when $\hat{X}$ was discretized based on the 1d-SAX respectively eSAX, assuming that for $\hat{X}^e$ the points in time start at zero. Also, l should be divisible by two respectively three. The extracted subsequences of $\hat{X}^e$ are then used as the input for Algorithm 4.1 [25]. For each of those, this algorithm collects all extracted subsequences of $\hat{X}^e$ that fulfill the match()-predicate in line 6 of Algorithm 4.1. It then returns the largest collection of such subsequences, which is called a motif. Algorithm 4.1 is invoked iteratively as long as a motif that contains at least two subsequences is found, since all subsequences that are already included in a motif cannot be

selected for future motifs. When Algorithm 4.1 does not find a motif anymore, the subsequences of X that correspond to the found motifs can be retrieved.
The match()-predicate in line 6 of Algorithm 4.1 iteratively compares the currently selected subsequence with all inputted subsequences [25]. Let S_1 and S_2 be two subsequences of $\hat{X}^e$ that shall be compared. Then, the match()-predicate first checks if $D(S_1, S_2) \leq r$ holds, where D is a given Minkowski distance and $r \geq 0$ is a given similarity distance. If this first check is passed, it then compares each pair of corresponding encoded alphabet symbols in S_1 and S_2 at the same point in time. For each of these pairs, it checks if their absolute difference is not too large based on a given threshold $abs\_diff \geq 0$. For the last check, the match()-predicate checks if $H(S_1, S_2) \leq h$ holds, where H is the Hamming distance and $h \geq 0$ is a given threshold.

```
   Input: subsequences                    // extracted subsequences of X̂^e
          thresholds for match()-predicate in line 6
   Output: largest_motif
 1 largest_motif ← ()
 2 largest_count ← 0
 3 for curr_subsequence in subsequences do
 4 |   curr_motif ← ()
 5 |   for subsequence in subsequences do
 6 |   |   if match(curr_subsequence, subsequence) then
 7 |   |   |   curr_motif.add(subsequence)
 8 |   |   end
 9 |   end
10 |   if curr_motif.size() > largest_count then
11 |   |   largest_motif ← curr_motif
12 |   |   largest_count ← curr_motif.size()
13 |   end
14 end
15 return largest_motif
```

Algorithm 4.1: This brute force algorithm fixes one of the inputted subsequences in each iteration and employs a nested loop over all inputted subsequences [25]. It finds the largest collection from the inputted subsequences that fulfill the match()-predicate in line 6 with respect to the corresponding fixed subsequence.

5 Evaluation

5.1 Reconstruction Error

Since the discretized time series resulted from the time series discretization algorithms are represented by alphabet symbols, they need to be transformed into a numerical representation in order to be comparable with the corresponding original standardized time series. This transformation is called numerical reconstruction [28]. After performing the numerical reconstruction, the reconstruction error measures the goodness of approximation of the reconstructed numerical representation of the discretized time series with respect to the corresponding original standardized time series [28]. Since the time series discretization algorithms presented in this thesis do not inherently define a numerical reconstruction, the first task is to define such. Subsequently, the second task is to define the measurement of the reconstruction error (i.e. the goodness of approximation).

General Numerical Reconstruction

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. Assume that $\hat{X} = \hat{x}_1, ..., \hat{x}_n$ is the corresponding discretized time series resulted from one of the time series discretization algorithms applied on X with a length of $1 \leq n \leq N$. Suppose that $A := \{\alpha_j \mid j \in \{1, ..., a\}\}$ is the corresponding alphabet with $a \geq 2$ alphabet symbols such that $\hat{x}_i \in A$ $(1 \leq i \leq n)$. Then, the first step of reconstructing X from $\hat{X}$ is to determine a function $f : A \to \mathbb{R}$ that maps each alphabet symbol $\alpha_j \in A$ to a numerical value [29]. Applying f on each $\hat{x}_i$ results in an intermediate time series $\tilde{X}' = \tilde{x}_1, ..., \tilde{x}_n$. However, this is not the final result of the numerical reconstruction if $n < N$, because the reconstructed numerical time series needs to have a length of N to be comparable with X. Therefore, each $\tilde{x}_i$ $(1 \leq i \leq n)$ is concatenated w times with itself to obtain $\tilde{x}_i^1, ..., \tilde{x}_i^w$ as a reconstructed numerical subsequence for the ith subsequence that was extracted by the PAA while discretizing X into $\hat{X}$ based on a window length of $w \geq 1$ [29]. Thus, the final reconstructed numerical time series is $\tilde{X} = \tilde{x}_1^1, ..., \tilde{x}_1^w, ..., \tilde{x}_n^1, ..., \tilde{x}_n^w$ with a length of $w \cdot n = N$ (see Figure 5.1).

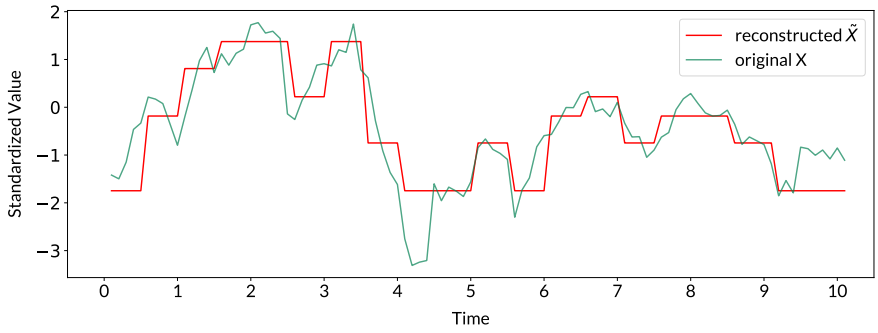
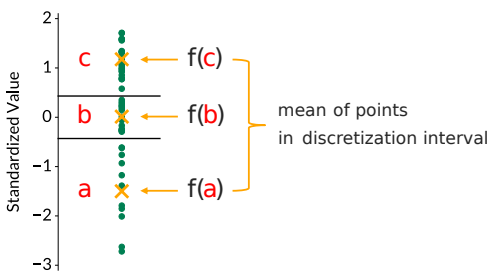


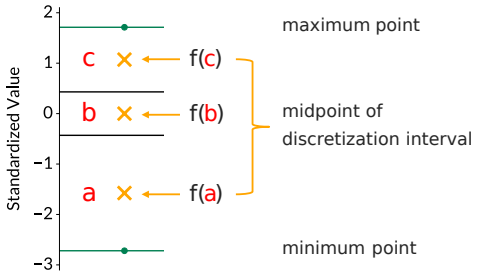
Figure 5.1: Based on the corresponding discretized time series resulted from a time series discretization algorithm, the original standardized time series is reconstructed. Since this numerical reconstruction is only an approximation of the original standardized time series, the reconstruction error measures the goodness of this approximation [28].

Strategies for Determining $f : A \to \mathbb{R}$

Let $B := (\beta_0, \beta_1, ..., \beta_{a-1}, \beta_a)$ be the ascending sorted list of breakpoints computed by a time series discretization algorithm with $\beta_0 = -\infty$ and $\beta_a = +\infty$. Then define $d_j := [\beta_{j-1}, \beta_j)$ $(1 \leq j \leq a)$ as the corresponding discretization intervals.
The first strategy for determining the function f is based on the points of the original standardized time series X. Based on Equation 3.2, each point x_i $(1 \leq i \leq n)$ is assigned to the discretization interval d_j it is located in. But, instead of discretizing these points, the mean $\overline{d}_j$ of the points in each d_j is computed. Then, f is determined by mapping each alphabet symbol α_j to the mean of its corresponding discretization interval: $f(\alpha_j) = \overline{d}_j$ $(1 \leq j \leq a)$ [29].
The second strategy for determining the function f is a variant of the first strategy (see Subfigure 5.2a). Instead of computing the mean of the points in each d_j, the median is computed. Thus, these medians $f(\alpha_j)$ are more robust to outliers within the corresponding discretization intervals d_j compared to the means from the first strategy.
Instead of considering the points of the original standardized time series, the third strategy considers the discretization intervals itself. For this strategy, the function f is determined by mapping each alphabet symbol α_j to the midpoint of its corresponding discretization interval such that $f(\alpha_j) = (\beta_{j-1} + \beta_j)/2$ [11]. However, there is one pitfall for the computation of the midpoints $f(\alpha_1)$ and $f(\alpha_a)$. Since $\beta_0 = -\infty$ and $\beta_a = +\infty$, these midpoints are not defined. One possibility to cope with this pitfall is to set the values of the minimum and maximum point of the original standardized time series X as artificial breakpoints β_0 and β_a, respectively. This way, the midpoints $f(\alpha_1)$ and $f(\alpha_a)$ are defined and can be computed like the other midpoints (see Subfigure 5.2b).



(a) For the first, the points of the original standardized time series (green) are assigned to the respective discretization interval (a, b, c) they are located in. Within each discretization interval, the mean of the points is computed. This value represents the numerical value (f(a), f(b), f(c)) for the alphabet symbol that is assigned to the corresponding discretization interval [29]. The procedure for the second strategy is analogous for the median.



(b) For the third strategy, the midpoint of each discretization interval (a, b, c) is computed. This value represents the numerical value (f(a), f(b), f(c)) for the alphabet symbol that is assigned to the corresponding discretization interval [11]. Since the first and last discretization interval are not lower respectively upper bounded, the values of the minimum and maximum point of the original standardized time series are set as artificial breakpoints for computing the respective midpoint.

Figure 5.2: Strategies for determining numerical values for alphabet symbols to perform the numerical reconstruction.

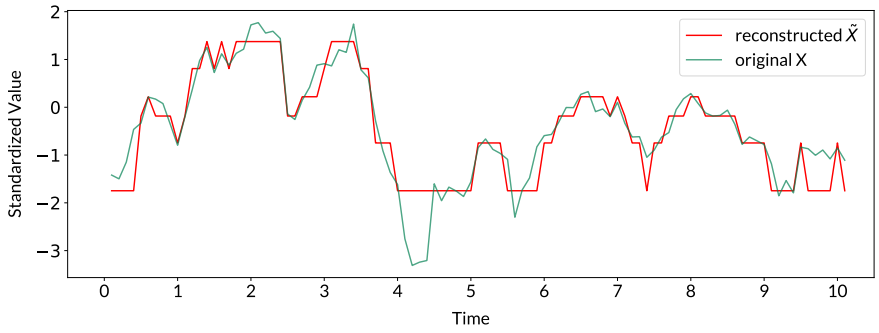
Pecularities for Multiple Alphabet Symbols per Subsequence

While the explained numerical reconstruction along with the determination of the function f can be applied without modifications for the time series discretization algorithms that use one alphabet symbol for representing an extracted subsequence by the PAA, there are pecularities for the eSAX and 1d-SAX.
Let $\hat{X} = sort(\{\hat{min}_1, \hat{x}_1, \hat{max}_1\}), ..., sort(\{\hat{min}_n, \hat{x}_n, \hat{max}_n\})$ be the eSAX representation of X. Recall that $\hat{X}$ contains three alphabet symbols per extracted subsequence. For the numerical reconstruction of X from $\hat{X}$, the determination and application of the function f does not need to be modified. Thus, the intermediate time series $\tilde{X}' = sort(\{\tilde{min}_1, \tilde{x}_1, \tilde{max}_1\}), ..., sort(\{\tilde{min}_n, \tilde{x}_n, \tilde{max}_n\})$ is obtained by applying f on each alphabet symbol of $\hat{X}$. However, to obtain the final reconstructed numerical time series $\tilde{X}$ (see Subfigure 5.3a), each $\tilde{x}_i$ $(1 \leq i \leq n)$ is first concatenated w times with itself to obtain $\tilde{x}_i^1, ..., \tilde{x}_i^w$, as explained previously. But in addition, those points in time that correspond to the points in time of the minimum and maximum point in the i-th extracted subsequence by the PAA, are overwritten with $\tilde{min}_i$ and $\tilde{max}_i$, respectively. As an example, consider the concatenation $\tilde{x}_i^1, \tilde{x}_i^2, \tilde{x}_i^3, \tilde{x}_i^4, \tilde{x}_i^5$ of $\tilde{x}_i$ based on a window length of $w = 5$. For obtaining $\tilde{X}$, this concatenation would further be transformed to $\tilde{x}_i^1, \tilde{max}_i, \tilde{x}_i^3, \tilde{x}_i^4, \tilde{min}_i$, assuming the maximum and minimum point in the i-th extracted subsequence are at

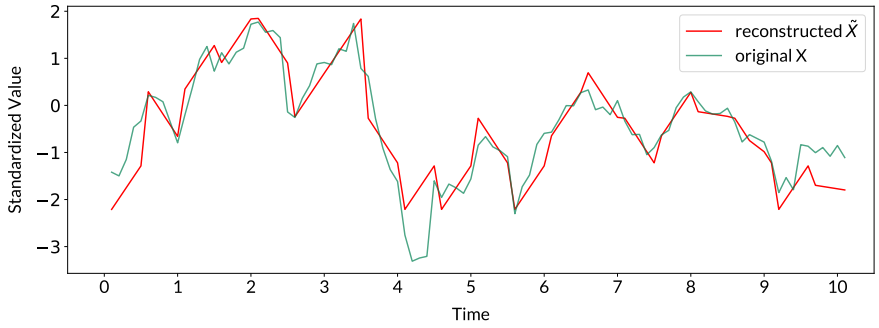
positions 2 and 5, respectively. Note that for $w = 2$, the numerical reconstruction of an extracted subsequence is only based on the minimum and maximum point of that subsequence while ignoring its mean. Further, for $w = 1$, the mean, minimum and maximum point of a subsequence are equal.
For the 1d-SAX, the determination and application of the function f needs to be modified. Recall that the 1d-SAX representation $\hat{X} = (\hat{x}_1, \hat{s}_1), ..., (\hat{x}_n, \hat{s}_n)$ of X contains two alphabet symbols per extracted subsequence by the PAA. While for the other time series discretization algorithms one function f is sufficient, for the 1d-SAX two different functions of this kind are needed. The reason is that the values of the means and slopes of the points of the extracted subsequences by the PAA are separateley discretized based on different discretization intervals [19]. Therefore, the strategies for determining f need to be adapted to these different discretization intervals. Let $f^m : A^m \to \mathbb{R}$ and $f^s : A^s \to \mathbb{R}$ be the two different functions to be determined based on the alphabets A^m and A^s that are used for discretizing the means and slopes, respectively. While f^m can be determined based on any of the three strategies described above, the first and second strategy need to be modified for determining f^s. For these two strategies, instead of using the points of the original standardized time series, the computed slope values of the extracted subsequences by the PAA that shall be discretized need to be used. Further, note that for the third strategy, the respective discretization intervals for the means or the slopes need to be used depending if f^m or f^s shall be determined. After determining f^m and f^s, the intermediate time series $\tilde{X}' = (\tilde{x}_1, \tilde{s}_1), ..., (\tilde{x}_n, \tilde{s}_n)$ is then obtained by separately applying f^m and f^s on each $\hat{x}_i$ and $\hat{s}_i$ $(1 \leq i \leq n)$, respectively [19].
Finally, let $t_i^1, ..., t_i^w$ be the points in time of the i-th extracted subsequence by the PAA based on a window length of w. Then, the numerical reconstruction for this subsequence is obtained based on the formula [19]:

$$\tilde{x}_i^j = \tilde{x}_i + \tilde{s}_i \cdot (t_i^j - t_i^{mid}) \qquad (1 \leq j \leq w),$$

where $t_i^{mid} = (t_i^1 + t_i^w)/2$. Applying this formula for each extracted subsequence by the PAA, results in the final reconstructed numerical time series $\tilde{X} = \tilde{x}_1^1, ..., \tilde{x}_1^w, ..., \tilde{x}_n^1, ..., \tilde{x}_n^w$ of length $w \cdot n = N$ (see Subfigure 5.3b) [19].



(a) The reconstructed numerical time series $\tilde{X}$ based on the discretization of the original standardized time series X with the eSAX.



(b) The reconstructed numerical time series $\tilde{X}$ based on the discretization of the original standardized time series X with the 1d-SAX.

Figure 5.3: The reconstructed numerical time series $\tilde{X}$ are plotted against the corresponding original standardized time series X. Performing the numerical reconstruction on the discretized time series $\hat{X}$ results in the reconstructed numerical time series $\tilde{X}$ [28].

Measuring the Reconstruction Error

The reconstruction error of the reconstructed numerical time series $\tilde{X} = \tilde{x}_1, ..., \tilde{x}_N$ with respect to the original standardized time series $X = x_1, ..., x_N$ is measured based on the deviation of the pairwise points of X and $\tilde{X}$ at the same point in time (see Figure 5.4). Thus, for each point of $\tilde{X}$, the goodness of its reconstruction with respect to the corresponding point of X is measured.
One possible measure that measures the average goodness of these point deviations is the Mean Absolute Error (MAE) [29]:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |x_i - \tilde{x}_i|.$$

Another possible measure for measuring the average goodness of these point deviations is the Mean Squared Error (MSE) [30]:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \tilde{x}_i)^2$$

While the point deviations are equally weighted in the MAE, the MSE gives relatively larger weights to relatively greater point deviations due to the squaring operation. But, both measures should be minimized for maximizing the goodness of approximation of the reconstructed numerical time series $\tilde{X}$ with respect to the original standardized time series X.

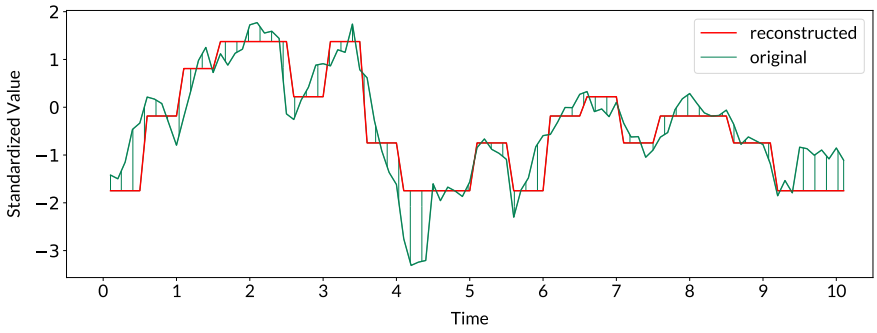


Figure 5.4: The measurement of the reconstruction error is indicated by connecting the pairwise points of the original standardized time series X and the reconstructed numerical time series $\tilde{X}$ at the same point in time. The deviation of these pairwise points is measured for measuring the reconstruction error [29, 30].

Datasets Used for Experimental Evaluation

The dataset labeled `Gaussian` contains 100 time series with 1,000 points each. These points were drawn from the standard normal distribution $\mathcal{N}(0, 1)$.
The dataset labeled `Non-Gaussian` contains 500 time series with 1,000 points each. For each time series, five different values $\mu_i \in \{-4, -3, ..., 3, 4\}$ $(1 \leq i \leq 5)$ and five different values $\sigma_i \in \{0.1, 0.2, ..., 0.8, 0.9\}$ $(1 \leq i \leq 5)$ were randomly drawn. Then, a random number of points, but at least 20, were successively drawn from each Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i)$ $(1 \leq i \leq 5)$, such that the resulting concatenated time series consists of 250 points in total. This time series was concatenated four times with itself to obtain the final time series with 1,000 points (see Figure 5.5).
For the dataset labeled `UCR`, 100 time series were randomly drawn from each of the training datasets named FordB, FordA, FaceAll, FacesUCR, Plane, SwedishLeaf, and OSULeaf (see Table 3.1) from the UCR Time Series Classification Archive

[20]. Thus, this dataset contains 700 time series in total.
The dataset labeled `Motif` contains 500 synthetic time series with 5,000 points each. These time series contain recurrently occurring subsequences of fixed length that are randomly placed between random walks. In total, the dataset is composed of 100 time series per recurrently ocurring subsequences of length 50, 100, 150, 200, and 250 points.
All time series were standardized before performing the experimental evaluation.

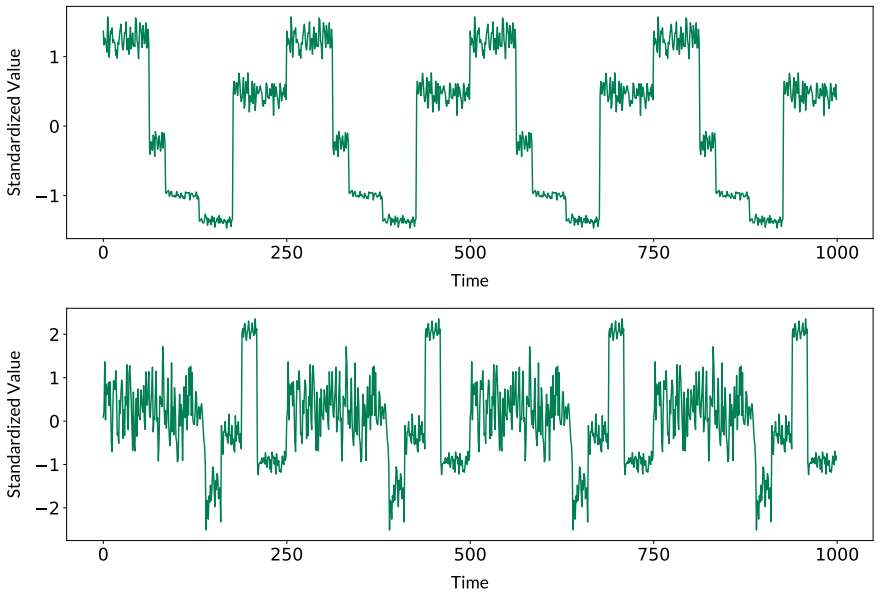


Figure 5.5: Two exemplary time series from the dataset labeled `Non-Gaussian`. Every 250 points, the time series are repeated based on the respective pattern that is composed of the points that were successively drawn from five different Gaussian distributions.

Configurations Used for Experimental Evaluation

The time series discretization algorithms were experimentally evaluated based on their respective main procedure presented in Chapter 3. However, for the aSAX and Persist presented modifications were applied. The aSAX was evaluated based on the modification of computing individual adapted breakpoints for each time series instead of discretizing all time series of a dataset based on the same adapted breakpoints (see Subsection 26). This modification was applied to obtain a more accurate measurement of the reconstruction error for each time series. The Persist was evaluated based on the modification of discretizing the points of the PAA

representation (i.e. the extracted means) instead of the points of the corresponding original standardized time series (see Subsection 3.3.2). This modification was applied to increase the comparability of the Persist with respect to the other time series discretization algorithms.
Furthermore, for an uniform numerical reconstruction, the function $f : A \to \mathbb{R}$ was determined based on the first strategy presented (see Subsection 5.1). The results of the experimental evaluation are qualitatively consistent for the other two strategies.

Experimental Results for Different Window Lengths

The time series discretization algorithms presented in Chapter 3 were experimentally evaluated with respect to the reconstruction error based on different window lengths used for the PAA (see Table 5.1). For a concise presentation of the corresponding experimental results, the reconstruction error was measured based on the MAE, since the results are qualitatively consistent when measuring the reconstruction error based on the MSE. The mean of the MAE across all time series of the respective dataset is presented as experimental result. Moreover, for discretizing the features of subsequences extracted by the PAA, an alphabet size of $a = 5$ was used for computing the breakpoints except for the extracted slopes in the 1d-SAX. For these, an alphabet size of $a = 3$ was used, since in the literature it is recommended to use a smaller alphabet size for the extracted slopes compared to that for the extracted means [19]. An experimental evaluation for other alphabet sizes was conducted as well (see Table 5.2).

	MAE				
$a = 5$	$w = 2$	$w = 5$	$w = 10$	$w = 20$	$w = 40$
Gaussian					
SAX	0.59	0.73	0.77	0.79	0.80
eSAX	**0.23**	**0.39**	**0.54**	**0.65**	**0.73**
1d-SAX	0.37	0.66	0.74	0.77	0.79
aSAX	0.59	0.73	0.78	0.80	0.82
Persist	0.65	0.78	0.82	0.84	0.85
Non-Gaussian					
SAX	0.21	0.25	0.27	0.35	0.47
eSAX	**0.17**	**0.19**	**0.23**	**0.31**	0.43
1d-SAX	0.18	0.23	0.26	0.32	**0.40**
aSAX	0.19	0.22	0.25	0.34	0.47
Persist	0.19	0.22	0.25	0.34	0.47
UCR					
SAX	0.26	0.36	0.50	0.66	0.73
eSAX	**0.22**	**0.26**	0.39	0.57	0.68
1d-SAX	0.23	0.29	**0.38**	**0.52**	**0.64**
aSAX	0.24	0.35	0.50	0.67	0.79
Persist	0.29	0.39	0.53	0.68	0.75
Motif					
SAX	0.21	0.24	0.26	0.31	0.39
eSAX	**0.20**	**0.21**	**0.23**	0.28	0.36
1d-SAX	**0.20**	0.22	**0.23**	**0.26**	**0.32**
aSAX	**0.20**	0.22	0.25	0.30	0.38
Persist	0.23	0.25	0.28	0.32	0.40

Table 5.1: This table contains experimental results of the five time series discretization algorithms with respect to the reconstruction error across the four datasets. The reconstruction error is measured based on the MAE. In this table, the mean of the MAE across all time series of the respective dataset is presented. The time series discretization algorithms were run based on an alphabet size of $a = 5$. For the PAA, the window lengths of $w = 2$, $w = 5$, $w = 10$, $w = 20$, and $w = 40$ were used. For each configuration, the smallest reconstruction error across all time series discretization algorithms is marked in bold.

For all evaluated time series discretization algorithms, the overall experimental results based on different window lengths used for the PAA show that the reconstruction error increases with an increasing window length. This finding is in line

with the considerations in Subsection 3.1 about the goodness of approximation of a time series by its PAA representation.
For the `Gaussian` dataset, the eSAX results in the lowest reconstruction error followed by the 1d-SAX for all evaluated window lengths. While the other evaluated time series discretization algorithms only use the means of the subsequences extracted by the PAA, the eSAX uses the minimum and maximum points and the 1d-SAX uses the slopes as additional information for discretization. This indicates why these two result in the two lowest reconstruction errors for all evaluated window lengths. Moreover, for all evaluated window lengths, the SAX and aSAX result in similar reconstruction errors and the Persist results in the largest reconstruction error. Another finding is that the respective reconstruction error converges for all time series discretization algorithms as the window length increases. Based on the considerations in Subsection 3.2.4, the reason for this convergence is the distributions of the means and slopes of the subsequences extracted by the PAA. Since the time series of the `Gaussian` dataset follow the standard normal distribution $\mathcal{N}(0, 1)$, the distributions of the extracted means and slopes have a mean of zero and decreasing standard deviations with an increasing window length. Therefore, as the window length increases, the extracted means and slopes start to concentrate in the respective discretization intervals near the value of zero. Hence, the numerical reconstruction, and therefore the reconstruction error, becomes stable with an increasing window length. The convergence of the reconstruction error also applies for the eSAX. However, it is slower due to the arbitrarily located minimum and maximum points across discretization intervals, that slow down the described concentration effect towards the discretization intervals near the value of zero. But, it still converges, because the number of subsequences extracted by the PAA decreases with an increasing window length, and therefore, the number of extracted minimum and maximum points decreases as well.
The first finding for the `Non-Gaussian` dataset is that the aSAX and Persist are competitive to the eSAX and 1d-SAX up to the evaluated window length of $w = 20$. However, this is also in line with the considerations in Chapter 3, as the aSAX and Persist are classified as adaptive-breakpoints discretization algorithms. In contrast to the other evaluated time series discretization algorithms, these two adapt their breakpoints to the distributions of the time series of the `Non-Gaussian` dataset. Another finding is that the eSAX results in the lowest reconstruction error up to the evaluated window length of $w = 20$, while the 1d-SAX results in the lowest reconstruction error for $w = 40$. On the one hand, the minimum and maximum point of a subsequence extracted by the PAA become less significant for the numerical reconstruction of the subsequence as the window length (i.e. the number of points of the subsequence) increases. Therefore, with respect to the reconstruction error, the benefit of additionally using the minimum and maximum point for discretization, as for the eSAX, decreases with an increasing window length. On the other hand, with the slope of a subsequence, the 1d-SAX uses a

global property of the subsequence that is computed based on all points of the subsequence in contrast to considering only two extreme points as for the eSAX. Therefore, with respect to the reconstruction error based on a window length of $w = 40$, the experimental results indicate that additionally using the slope as a global property of a subsequence is more advantageous compared to additionally using only two extreme points of a subsequence.
For the `UCR` dataset, this change in leadership already occurs at the evaluated window length of $w = 10$. While the eSAX results in the lowest reconstruction error up to the evaluated window length of $w = 5$, the 1d-SAX results in the lowest reconstruction error from the evaluated window length of $w = 10$ up to the largest evaluated window length of $w = 40$. This finding can be explained with the same argumentation as for the `Non-Gaussian` dataset. Another finding is that the aSAX results in the largest reconstruction error for a window length of $w = 40$, while for the smaller evaluated window lengths it is similar to the SAX and lower than the Persist. The reason for this finding is that 400 out of 700 time series in the `UCR` dataset contain about 130 to 140 points. Applying a window length of $w = 40$ then implies the clustering of $130/40 \approx 3$ points (i.e. means) of the PAA representation for computing the individual adapted breakpoints for the corresponding time series based on the aSAX. Therefore, the quality of these computed breakpoints is degraded for discretization. Thus, the aSAX should only be applied for PAA representations that consist of an appropriate number of points (i.e. means). Based on the experimental results, the PAA representations resulted from a window length of $w = 20$ seem to have an appropriate number of points, since the aSAX results in similar reconstruction errors as the SAX and Persist. Another solution is to apply the described modification of clustering the points of the corresponding original standardized time series instead of the points of the PAA representation (see Subsection 26).
As discussed before, the experimental results for the `Motif` dataset also show that the 1d-SAX results in a lower reconstruction error for larger evaluated window lengths starting from $w = 20$ compared to the eSAX.

Experimental Results for Different Alphabet Sizes

The time series discretization algorithms presented in Chapter 3 were experimentally evaluated with respect to the reconstruction error based on different alphabet sizes $a \in (3, 6, 12, 24)$ that were used for computing the breakpoints for discretization (see Table 5.2). For the 1d-SAX, the corresponding alphabet sizes $a \in (2, 3, 6, 12)$ were used for discretizing the slopes of the subsequences extracted by the PAA. In the literature, it is recommended to use a smaller alphabet size for the extracted slopes compared to that for the extracted means [19]. Furthermore, for the subsequences extracted by the PAA, the window lengths $w = 5$ and $w = 20$

were used for the overall experimental evaluation. For a concise presentation of the corresponding experimental results, the reconstruction error was measured based on the MAE, since the results are qualitatively consistent when measuring the reconstruction error based on the MSE. The mean of the MAE across all time series of the respective dataset is presented as experimental result.

	MAE ($w = 5$)				MAE ($w = 20$)			
	$a = 3$	$a = 6$	$a = 12$	$a = 24$	$a = 3$	$a = 6$	$a = 12$	a = 24
Gaussian								
SAX	0.76	0.72	0.72	0.71	0.80	0.78	0.78	0.78
eSAX	**0.47**	**0.37**	**0.32**	**0.29**	**0.69**	**0.65**	**0.62**	**0.61**
1d-SAX	0.70	0.66	0.64	0.64	0.79	0.77	0.76	0.76
aSAX	0.77	0.73	0.72	0.72	0.86	0.79	0.78	0.78
Persist	0.81	0.79	0.73	0.71	0.83	0.84	0.80	0.78
Non-Gaussian								
SAX	0.31	0.23	0.21	0.20	0.40	0.34	0.32	0.32
eSAX	**0.28**	**0.17**	**0.12**	**0.09**	**0.37**	**0.29**	**0.27**	0.26
1d-SAX	0.31	0.22	0.18	0.17	0.39	0.30	**0.27**	**0.25**
aSAX	**0.28**	0.21	0.20	0.20	0.38	0.33	0.32	0.32
Persist	0.30	0.23	0.22	0.21	0.40	0.37	0.36	0.35
UCR								
SAX	0.44	0.34	0.30	0.29	0.68	0.65	0.64	0.64
eSAX	**0.38**	**0.23**	**0.17**	**0.14**	0.62	0.56	0.54	0.53
1d-SAX	0.39	0.26	0.20	0.16	**0.57**	**0.51**	**0.47**	**0.46**
aSAX	0.43	0.33	0.30	0.29	0.71	0.67	0.66	0.66
Persist	0.61	0.38	0.31	0.29	0.74	0.67	0.64	0.64
Motif								
SAX	0.33	0.21	0.16	0.15	0.38	0.30	0.27	0.26
eSAX	**0.32**	**0.18**	**0.11**	**0.08**	0.36	0.26	0.23	0.21
1d-SAX	**0.32**	0.19	0.13	0.11	**0.35**	**0.25**	**0.20**	**0.18**
aSAX	**0.32**	0.20	0.16	0.14	0.37	0.29	0.26	0.26
Persist	0.40	0.22	0.16	0.14	0.45	0.30	0.26	0.25

Table 5.2: This table contains experimental results of the five time series discretization algorithms with respect to the reconstruction error across the four datasets. The reconstruction error is measured based on the MAE. In this table, the mean of the MAE across all time series of the respective dataset is presented. The time series discretization algorithms were run based on the alphabet sizes of $a = 3$, $a = 6$, $a = 12$, and $a = 24$. For the PAA, the window lengths of $w = 5$ and $w = 20$ were used. For each configuration, the smallest reconstruction error across all time series discretization algorithms is marked in bold.

The overall experimental results show for all evaluated time series discretization algorithms across all evaluated datasets that the reconstruction error decreases with an increasing alphabet size. This finding is in line with the considerations

in Subsection 3.2.1 about the granularity of the discretization dependent on the alphabet size. Furthermore, the findings for an alphabet size of $a = 5$ and different window lengths described in the last subsection (see Subsection 5.1) are also indicated by these experimental results for different alphabet sizes up to an alphabet size of $a = 12$. For the increase of the evaluated alphabet size from $a = 12$ to $a = 24$, these experimental results indicate three new findings that are consistent across all evaluated datasets.

The first finding is that the reconstruction error converges with an increasing alphabet size for all evaluated time series discretization algorithms. When increasing the alphabet size from $a = 12$ to $a = 24$, the reduction in the reconstruction error is less significant compared to the increase from $a = 3$ to $a = 6$ or $a = 6$ to $a = 12$. On the other hand, the reduction in the reconstruction error is most significant when increasing the alphabet size from $a = 3$ to $a = 6$. The reason for this convergence of the reconstruction error with an increasing alphabet size is based on the considerations in Subsection 3.2.1. With an increasing alphabet size, the granularity of the discretization increases as the number of discretization intervals increases. Thus, the numerical values $f(\alpha_j)$ $(1 \leq j \leq a)$ become a more accurate approximation of the points assigned to the respective discretization interval corresponding to the alphabet symbol α_j $(1 \leq j \leq a)$ overall. The experimental results indicate that the improvement of this approximation decreases with an increasing alphabet size, since the reduction of the reconstruction error decreases. This convergence of the reconstruction error is even faster for the window length of $w = 20$ compared to $w = 5$ as the reduction in the reconstruction error from $a = 6$ to $a = 12$ is less significant for $w = 20$.

Based on this finding, a recommendation for an appropriate alphabet size to handle the tradeoff described in Subsection 3.2.1 between the granularity of the discretization and the memory requirements of the discretized time series can be derived. As the most significant reduction of the reconstruction error is achieved when increasing the alphabet size from $a = 3$ to $a = 6$, an appropriate alphabet size for minimizing the reconstruction error (i.e. maximizing the granularity of the discretization) is lower bounded by $a > 3$. Moreover, as the least significant reduction of the reconstruction error is achieved when increasing the alphabet size from $a = 12$ to $a = 24$, an appropriate alphabet size for minimizing the memory requirements of the discretized time series, is upper bounded by $a \leq 12$. This upper bound can also be increased to $a = 16$, since representing a single alphabet symbol out of 12 or 16 alphabet symbols requires four bits in both cases. Thus, an appropriate alphabet size to handle the described tradeoff is bounded by $3 < a \leq 16$.

The second finding is that with an increasing alphabet size, the distance of the reconstruction errors between the eSAX and 1d-SAX one the one hand and the SAX, aSAX, and Persist on the other hand increases. This finding indicates that with an increasing alphabet size, the benefit of discretizing additional features like the minimum point, maximum point, and slope of subsequences increases with

respect to minimizing the reconstruction error.
The third finding is that the reconstruction error for the Persist converges to that of the SAX and aSAX with an increasing alphabet size. An explanation for this effect is that the flexibility of computing adapted breakpoints decreases as the number of breakpoints (i.e. the alphabet size) increases. Therefore, the difference in the breakpoints across different time series discretization algorithms decreases. Thus, the difference in the reconstruction errors across different time series discretization algorithms decreases as well when discretizing the same feature like the mean of subsequences.

5.2 Motif Discovery

Compared to the abundance of ground truth data for evaluating time series classification algorithms (e.g. UCR Time Series Classification Archive [20]), there is a lack of ground truth benchmark datasets for experimentally evaluating motif discovery algorithms. Therefore, it is necessary to construct synthetic time series data that contain recurrently occurring subsequences with corresponding labels to be able to experimentally evaluate motif discovery algorithms.

Datasets Used for Experimental Evaluation

For the experimental evaluation, two synthetic datasets were used. Each dataset contains 40 time series with varying length. For the construction of such a time series, 11 or 12 subsequences of fixed length were first created by applying a random walk with a random step size and a minimum and maximum value [31]. Then, a varying number of copies of each of these created subsequences were selected and noise was added to each copy. Lastly, these noisy copies were concatenated in a random order by applying a random walk of varying length between each two noisy copies to obtain the final time series [31].
The constructed datasets are labeled `Motif60` and `Motif120`, which contain time series that are constructed based on noisy copies of subsequences of length 60 and 120, respectively. All in all, `Motif60` contains time series of lengths between 6,400 and 8,200, while `Motif120` contains time series of lengths between 11,200 and 14,000.
For each time series, the noisy copies of the same subsequence represent a recurrently occurring subsequence that shall be discovered by a motif discovery algorithm. To conduct the experimental evaluation, all time series points that belong to a common recurrently occurring subsequence are given the same label

[31]. Moreover, those time series points that were created by the random walk are also labeled accordingly [31].
All time series were standardized before performing the experimental evaluation.

Measuring the Goodness of Motif Discovery

Let $X = x_1, ..., x_N$ be a standardized time series from one of the datasets used for experimental evaluation with a length of $N \geq 1$ that contains $R \geq 1$ recurrently occurring subsequences. Moreover, let $T = t_1, ..., t_N$ be the corresponding points in time. Define $I := \{1, ..., N\}$ as the set of indices corresponding to T. Further, suppose that $M \subset I$ is a motif that results from one of the motif discovery algorithms described in Subsection 4, which contains the starting points in time of the corresponding discovered subsequences of X [31]. As the length of the R recurrently occurring subsequences in X is known in advance, the respective motif discovery algorithm discovers subsequences of this length. Therefore, the starting point in time of a subsequence of X is sufficient to extract the whole corresponding subsequence from X.
Thus, the first step to perform experimental evaluation is to extract all subsequences from X that correspond to the starting points in time that are contained in M. Suppose that the set S^M contains these extracted subsequences. Then, each subsequence in S^M is assigned the label that most often occurs across its corresponding points. Based on this assignment, M is assigned the label that most often occurs accross the labeled subsequences in S^M. This labeling is done for each motif M that is discovered for X by the respective motif discovery algorithm. For the following, let $L(M)$ be the assigned label for motif M.
Now, define the True Positives (TP) as the number of all time series points across the subsequences in S^M that are labeled $L(M)$ [31]. Further, define the False Negatives (FN) as the number of all time series points of X that are not contained in a subsequence of S^M, but are labeled $L(M)$ [31]. Moreover, define the False Positives (FP) as the number of all time series points across the subsequences in S^M that are not labeled $L(M)$ [31]. Based on these definitions, two measures for measuring the goodness of motif discovery can be defined. Both are derived from the evaluation measures called Recall and Precision [32]:

$$\text{Motif Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.1}$$

$$\text{Motif Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2}$$

These two measures are lower bounded by zero and upper bounded by one [32]. They should be maximized for each discovered motif for maximizing the goodness

of motif discovery by maximizing the TP and minimizing the FN, respectively, FP (see Figure 5.6) [32].
The Motif Recall and Motif Precision evaluate the goodness of an individual motif that results from applying one of the motif discovery algorithms described in Subsection 4 on X. However, these motif discovery algorithms do not exclude the possibility of discovering multiple motifs M that are assigned the same label $L(M)$. Also, they do not exclude the possibility of discovering motifs M, where the majority of the subsequences in S^M were created by the random walk and labeled accordingly. Hence, such motifs are assigned the label corresponding to the random walk. Let $L(M)^{rw}$ be this label. Based on these two observations, the Mean Motif Density per Label (MMDL) measures the mean number of motifs that were discovered per assigned label $L(M) \neq L(M)^{rw}$:

$$\text{MMDL} = \frac{|\widetilde{X}^M|}{|\{L(M) \mid M \in \widetilde{X}^M\}|}, \tag{5.3}$$

where for $X^M := \{M \mid M \text{ discovered by motif discovery algorithm for } X\}$ it is $\widetilde{X}^M := \{M \in X^M \mid L(M) \neq L(M)^{rw}\}$. The MMDL is lower bounded by one, which results when each $M \in \widetilde{X}^M$ is assigned a different label.
Moreover, the Random Walk Labeling Ratio (RWLR) measures the ratio of the discovered motifs that were assigned the label $L(M)^{rw}$:

$$\text{RWLR} = \frac{|\{M \in X^M \mid L(M) = L(M)^{rw}\}|}{|X^M|} \tag{5.4}$$

The RWLR is lower bounded by zero and upper bounded by one, which results when no discovered motifs for X are assigned label $L(M)^{rw}$, respectively, when all discovered motifs for X are assigned label $L(M)^{rw}$.
The previous evaluation measures all consider the motifs resulted from the employed motif discovery algorithm. However, the motif discovery algorithms described in Subsection 4 do not guarantee to discover all recurrently occurring subsequences contained in X. Therefore, the Found Recurrent Subsequences Ratio (FRSR) measures the number of discovered recurrent subsequences of X compared to the total number of recurrent subsequences R contained in X:

$$\text{FRSR} = \frac{|\{L(M) \mid M \in \widetilde{X}^M\}|}{R} \tag{5.5}$$

The FRSR is lower bounded by zero and upper bounded by one. The lower bound results when there is no discovered motif for X that is assigned a label other than $L(M)^{rw}$. The upper bound results when for each recurrently occurring subsequence in X, there is at least one discovered motif assigned a label corresponding to the label of the respective recurrently occurring subsequence.

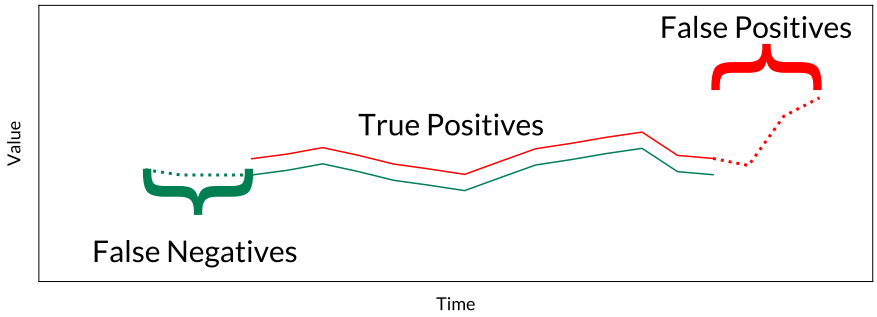


Figure 5.6: For instance, the red subsequence (including the dotted points at the end) is included in S^M for the discovered motif M. This motif is assigned the label corresponding to the recurrently occurring subsequence in X, where the green subsequence (including the dotted points at the beginning) is a noisy copy of. A perfect motif discovery algorithm would have discovered the green subsequence. However, the employed motif discovery algorithm discovered the red subsequence as the best approximation for the green subsequence. The TP, FN, and FP for this approximation are indicated for visual intuition [31]. Note that the vertical space between the two plotted subsequences is only for visual clarity.

Measures of Goodness Used for Experimental Evaluation

Across all time series of the respective dataset used for experimental evaluation, summary statistics were computed based on the measures of goodness defined above.
Given the discovered motifs of a motif discovery algorithm for a standardized time series X from the used dataset, the MMDL, RWLR, and FRSR were first computed based on these discovered motifs for X. This was done for all of the standardized time series of the used dataset. Then, the respective mean of the corresponding values for the MMDL, RWLR, and FRSR was computed and is reported in the experimental results below.
Moreover, the Motif Recall and Motif Precision was first computed based on each $M \in \widetilde{X}^M$. Again, this was done for all of the standardized time series X of the used dataset. Then, the respective mean of the corresponding values for the Motif Recall and Motif Precision was computed and is reported in the experimental results below. The motifs assigned the label $L(M)^{rw}$ were excluded for these computations, because including them would have distorted the results. For example, a motif assigned the label $L(M)^{rw}$ could have a relatively high Motif Recall and Motif Precision. However, as motif discovery is the task of finding recurrently occurring subsequences in a given time series, it is not desired to find motifs that correspond to time series points that were created by the random walk.

Configurations Used for Experimental Evaluation

The time series discretization algorithms described in Chapter 3 were experimentally evaluated for each motif discovery algorithm described in Chapter 4. All time series discretization algorithms, except the Persist, were evaluated based on their respective main procedure described in Chapter 3. The Persist was evaluated based on the modification of discretizing the points of the PAA representation (i.e. the extracted means) instead of the points of the corresponding original standardized time series (see Subsection 3.3.2). This modification was applied to increase the comparability of the Persist with respect to the other time series discretization algorithms.
Moreover, modified versions of the Matrix Profile and Brute Force motif discovery algorithm were additionally employed. In these versions, the original standardized time series were inputted without any modification (i.e. discretization) applied. These modified versions were additionally employed to benchmark the performance of the time series discretization algorithms. In the following experimental results, these modified versions are labeled as Raw. For the Random Projection motif discovery algorithm, such a modified version cannot be applied as this algorithm inherently depends on time series discretization.
Further, the employed time series discretization algorithms were evaluated based on a window length of $w = 5$ and $w = 10$ used for the PAA. The other parameter values used for the evaluation are shown below when discussing the experimental results for the respective motif discovery algorithm.

Parameter Tuning

Given a time series discretization algorithm and a motif discovery algorithm, their parameters need to be tuned to achieve the best performance with respect to the measures of goodness defined above. This parameter tuning involves finding an appropriate alphabet size for the time series discretization algorithm and appropriate values for the parameters of the motif discovery algorithm.
For this evaluation, parameter tuning was performed by first selecting the four time series with the smallest length from the respective dataset used. Based on those four time series, the parameter values were experimentally determined by running the respective motif discovery algorithm for the respective time series discretization algorithm multiple times for different parameter values. Note that for the modified versions labeled as Raw, only the respective motif discovery algorithm had to be run. Moreover, for the modified version of the Brute Force motif discovery algorithm labeled as Raw, the four selected time series were additionally shortened to 3,500 points due to the long execution time of this algorithm compared to the

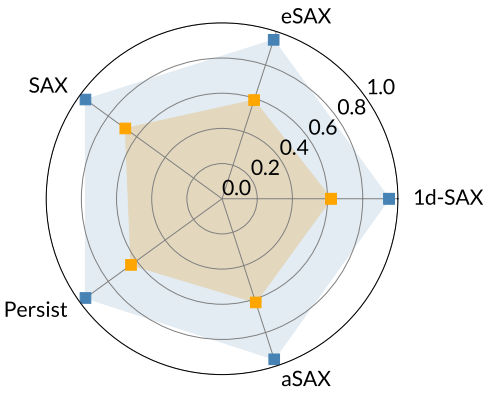
other two employed motif discovery algorithms.
For the corresponding optimization of the measures of goodness, an emphasis was put on the Motif Recall and Motif Precision. Based on the respective parameters of the employed motif discovery algorithms, a value near the upper bound of one is not as challenging to achieve for the Motif Precision as for the Motif Recall. Therefore, the objective for parameter tuning was to achieve a similar and relatively high Motif Precision for each time series discretization algorithm, while optimizing its corresponding Motif Recall. This approach was applied to increase the comparability between the time series discretization algorithms. Since for all time series discretization algorithms a similar Motif Precision is obtained, the performance of the time series discretization algorithms can be compared based on the Motif Recall. The actual parameter values that were determined based on this approach are shown in the following experimental results for the respective motif discovery algorithm. The parameters for the respective motif discovery algorithm are explained in Chapter 4.
However, one parameter was added for all employed motif discovery algorithms. As described in Chapter 4, when performing motif discovery, there can be trivially matching subsequences of X. To account for such trivially matching subsequences, the parameter k was added for all employed motif discovery algorithms. This parameter is explained in Chapter 4 as well and its value was also experimentally determined within the approach described above. Based on the parameter k, all trivially matching subsequences of a subsequence S of X were excluded from being included in a motif, once S was included in a motif for the respective motif discovery algorithm [26]. This procedure improved the performance of all employed motif discovery algorithms with respect to the measures of goodness, which is the reason the parameter k was added.

Experimental Results for the Random Projection Algorithm

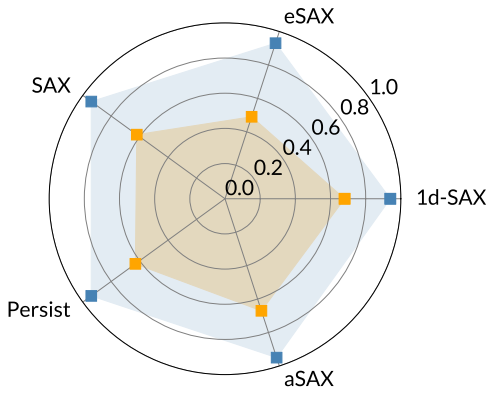
Each of the time series discretization algorithms described in Chapter 3 was experimentally evaluated with respect to the Random Projection motif discovery algorithm described in Section 4.1 (see Figure 5.7 and Table 5.4). For this evaluation, the values for the involved parameters were experimentally determined based on the parameter tuning approach explained above (see Table 5.3).
The results for the computed summary statistics of the RWLR and FRSR are consistent, as the RWLR is 0.00 and the FRSR is 0.99 or 1.00 for all evaluated configurations and time series discretization algorithms.

	a	s	iters	r	min_collisions	k
w = 5						
SAX	26/26	10/22	12/12	1/2.2	0/4	60/100
eSAX	12/12	11/22	12/32	1/2.2	1/4	50/100
1d-SAX	26/26	11/22	14/18	1/2.2	1/4	50/100
aSAX	26/26	10/22	12/12	1/2.2	0/4	60/100
Persist	26/26	10/22	12/12	1/2.2	0/4	60/100
w = 10						
SAX	23/26	4/10	14/15	1/2.2	0/4	60/100
eSAX	8/12	5/10	12/36	1/2.2	1/4	50/80
1d-SAX	26/20	5/10	14/18	1/2.2	1/3	60/100
aSAX	23/26	4/10	14/15	1/2.2	0/4	60/100
Persist	23/26	4/10	14/15	1/2.2	0/4	60/100

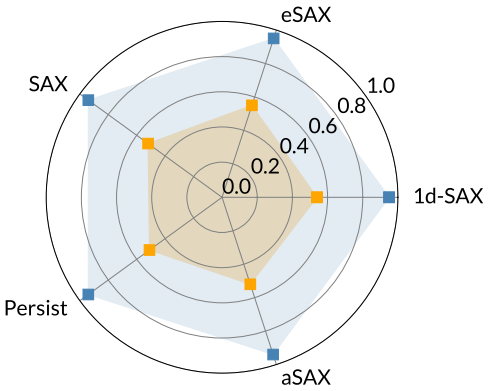
Table 5.3: This table contains the parameter values used for the evaluation. In the first column, $w = 5$ and $w = 10$ indicates the window length that was used for the PAA. For the cells of the other columns, x/y represents the respective parameter values x and y that were used for the datasets `Motif60` and `Motif120`, respectively. Parameter a in the second column represents the alphabet size used for discretization. For the 1d-SAX, the alphabet size used for discretizing the means is presented in the table. The alphabet size used for discretizing the slopes was $a = 1/2$ and $a = 3/3$ for $w = 5$ and $w = 10$, respectively. The remaining columns contain the values used for the parameters of the Random Projection motif discovery algorithm as described in Section 4.1. Note that the length l of the extracted subsequences by this algorithm is given by the respective fixed length of the recurrently occurring subsequences of the time series contained in `Motif60` and `Motif120`.



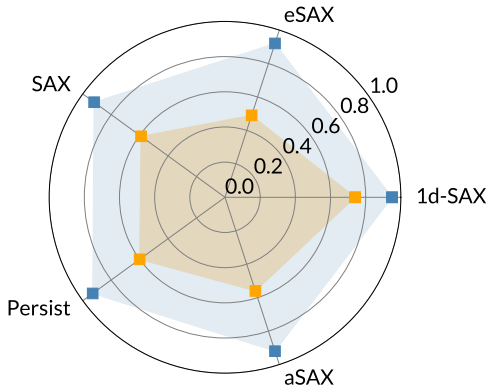
(a) $\mathbf{w = 5}$, `Motif60:` For all time series discretization algorithms, the Motif Precision is 0.95 or 0.96. With a Motif Recall of 0.68, the SAX performs best, while the eSAX performs worst with a Motif Recall of 0.59.



(b) $\mathbf{w = 5}$, `Motif120:` The 1d-SAX performs best for both measures of goodness with a Motif Precision of 0.94 and a Motif Recall of 0.68. The eSAX performs worst for both measures of goodness with a Motif Precision of 0.93 and a Motif Recall of 0.49.



(c) $\mathbf{w = 10}$, `Motif60:` For all time series discretization algorithms, the Motif Precision is 0.94 or 0.95. With a Motif Recall of 0.55, the eSAX performs best, while the Persist performs worst with a Motif Recall of 0.51. Compared to Subfigure 5.7a, the overall performance is worse, which results from the increase in the window length used for the PAA from $w = 5$ to $w = 10$.



(d) $\mathbf{w = 10}$, `Motif120:` The 1d-SAX performs best for both measures of goodness with a Motif Precision of 0.95 and a Motif Recall of 0.74. The eSAX performs worst for both measures of goodness with a Motif Precision of 0.92 (also for the SAX and aSAX) and a Motif Recall of 0.49.

Figure 5.7: This figure presents the results for the computed summary statistics of the Motif Precision and Motif Recall for each evaluated time series discretization algorithm with respect to the Random Projection motif discovery algorithm. The window length w used for the PAA and the used dataset is indicated for each subfigure.

	w = 5		**w = 10**	
	`Motif60`	`Motif120`	`Motif60`	`Motif120`
SAX	1.17	1.50	1.33	1.57
eSAX	1.24	1.80	1.32	1.81
1d-SAX	1.21	1.36	1.34	1.25
aSAX	1.23	1.39	1.36	1.64
Persist	1.23	1.47	1.38	1.56

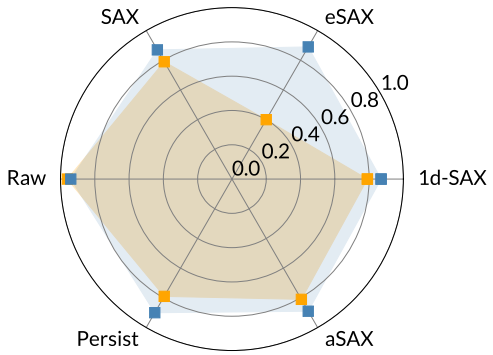
Table 5.4: This table contains the computed summary statistics of the MMDL for each evaluated time series discretization algorithm with respect to the Random Projection motif discovery algorithm. The window length w used for the PAA and the used dataset are indicated by the columns. Comparing the values of the MMDL with the values of the Motif Recall presented in Figure 5.7, indicates a negative correlation between these two measures of goodness. Meaning that a time series discretization algorithm with a relatively high value for the Motif Recall seems likely to show a relatively low value for the MMDL, and vice versa.

Experimental Results for the Matrix Profile Algorithm

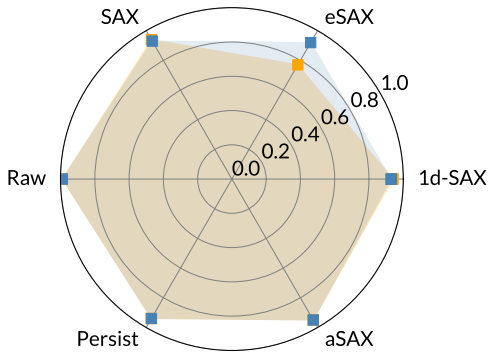
Each of the time series discretization algorithms described in Chapter 3 was experimentally evaluated with respect to the Matrix Profile motif discovery algorithm described in Section 4.2 (see Figure 5.8 and Table 5.6). For this evaluation, the values for the involved parameters were experimentally determined based on the parameter tuning approach explained above (see Table 5.5).
The results for the computed summary statistics of the RWLR are consistent as the RWLR is between 0.00 and 0.03 (inclusive) for all evaluated configurations and time series discretization algorithms. Moreover, the results for the computed summary statistics of the FRSR are consistent except for one configuration. Except for this configuration, the FRSR is between 0.96 and 1.00 (inclusive) for all evaluated configurations and time series discretization algorithms. However, in the case of the `Motif60` dataset with a window length of $w = 10$ used for the PAA, the FRSR is between 0.55 and 0.66 (inclusive) for all time series discretization algorithms. While the eSAX performs best with a value of 0.66, the SAX and 1d-SAX perform worst with a value of 0.55.
The reason for this underpeformance is that a subsequence of length 60 is represented by too few encoded alphabet symbols compared to a window length of $w = 5$. Therefore, the overall performance of the Matrix Profile motif discovery algorithm is worse.

	a	**r**	**k**
w = 5			
SAX	18/18	9/23	55/100
eSAX	9/12	12/55	50/100
1d-SAX	18/18	9/23	55/100
aSAX	18/18	9/23	55/100
Persist	18/18	9/23	55/100
w = 10			
SAX	26/16	7/9	120/100
eSAX	9/9	8/20	100/100
1d-SAX	26/16	7/9	120/100
aSAX	26/16	7/9	120/100
Persist	26/16	7/9	120/100
Raw	-/-	13/20	60/120

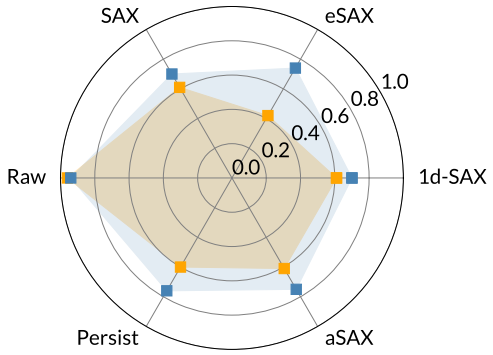
Table 5.5: This table contains the parameter values used for the evaluation. In the first column, $w = 5$ and $w = 10$ indicates the window length that was used for the PAA. For the cells of the other columns, x/y represents the respective parameter values x and y that were used for the datasets `Motif60` and `Motif120`, respectively. Parameter a in the second column represents the alphabet size used for discretization. For the 1d-SAX, the alphabet size used for discretizing the means is presented in the table. The alphabet size used for discretizing the slopes was $a = 1/1$ for $w = 5$ and $w = 10$. The remaining columns contain the values used for the parameters of the Matrix Profile motif discovery algorithm as described in Section 4.2. Moreover, as Minkowski distance D, the Manhattan distance was used for all configurations. Further, note that the length l of the examined subsequences in the Matrix Profile motif discovery algorithm is determined by the respective fixed length of the recurrently occurring subsequences of the time series contained in `Motif60` and `Motif120` and the used window length w.



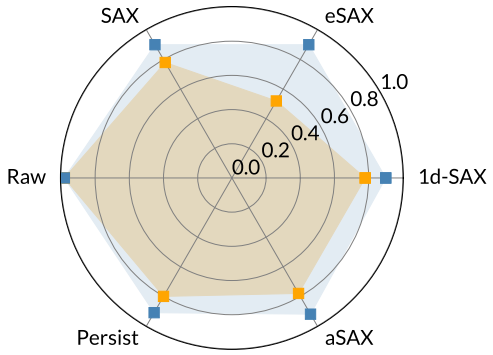
(a) $\mathbf{w = 5}$, `Motif60:` For all time series discretization algorithms, the Motif Precision is between 0.87 and 0.90 (inclusive). Moreover, the Motif Recall is between 0.79 and 0.81 (inclusive) except for the eSAX, which obtained a Motif Recall of 0.40. Furthermore, all time series discretization algorithms are worse than the Raw version that obtained a Motif Precision of 0.94 and a Motif Recall of 0.96.



(b) $\mathbf{w = 5}$, `Motif120:` The eSAX performs worst for both measures of goodness with a Motif Precision of 0.92 and a Motif Recall of 0.77. For all other time series discretization algorithms, the Motif Precision is between 0.93 and 0.95 (inclusive) and the Motif Recall is between 0.94 and 0.95 (inclusive). The Raw version performs best with a Motif Recall and Motif Precision of 0.99.



(c) $\mathbf{w = 10}$, `Motif60:` Compared to Subfigure 5.8a, all time series discretization algorithm perform worse. The reason is that a subsequence of length 60 is represented by fewer encoded alphabet symbols, since a window length of $w = 10$ is used for the PAA.



(d) $\mathbf{w = 10}$, `Motif120:` For all time series discretization algorithms, the Motif Precision is between 0.90 and 0.92 (inclusive). The eSAX performs worst based on a Motif Recall of 0.52, while for the other time series discretization algorithms the Motif Recall is between 0.78 and 0.80.

Motif Precision Motif Recall

Figure 5.8: This figure presents the results for the computed summary statistics of the Motif Precision and Motif Recall for each evaluated time series discretization algorithm and the Raw version with respect to the Matrix Profile motif discovery algorithm. The window length w used for the PAA and the used dataset is indicated for each subfigure. Note that the Raw version does not depend on the window length w.

	w = 5		w = 10	
	Motif60	Motif120	Motif60	Motif120
SAX	1.19	1.04	1.05	1.22
eSAX	1.71	1.30	1.25	1.74
1d-SAX	1.19	1.04	1.05	1.22
aSAX	1.18	1.02	1.05	1.25
Persist	1.21	1.04	1.07	1.20
Raw	1.02	1.00	1.02	1.00

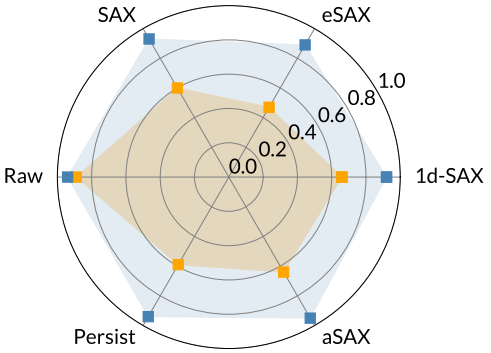
Table 5.6: This table contains the computed summary statistics of the MMDL for each evaluated time series discretization algorithm and the Raw version with respect to the Matrix Profile motif discovery algorithm. The window length w used for the PAA and the used dataset are indicated by the columns. Note that the Raw version does not depend on the window length w. As for the evaluation based on the Random Projection motif discovery algorithm, the comparison of the values of the MMDL with the values of the Motif Recall presented in Figure 5.8, also indicates a negative correlation between these two measures of goodness. For example, for all evaluated configurations, the Raw version obtained the highest Motif Recall and the lowest MMDL across all evaluated algorithms. Moreover, the eSAX obtained the lowest Motif Recall and the highest MMDL.

Experimental Results for the Brute Force Algorithm

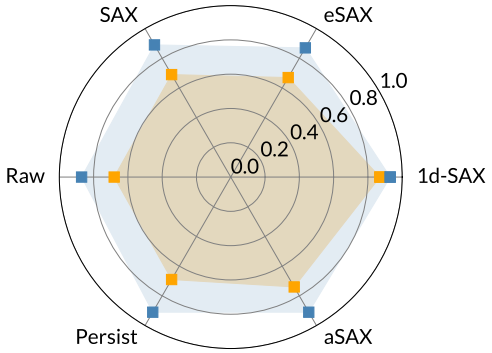
Each of the time series discretization algorithms described in Chapter 3 was experimentally evaluated with respect to the Brute Force motif discovery algorithm described in Section 4.3 (see Figure 5.9 and Table 5.8). For this evaluation, the values for the involved parameters were experimentally determined based on the parameter tuning approach explained above (see Table 5.7).
The results for the computed summary statistics of the RWLR and FRSR are consistent, as the RWLR is between 0.00 and 0.02 (inclusive) and the FRSR is between 0.95 and 1.00 (inclusive) for all evaluated configurations and time series discretization algorithms.

	a	**r**	**h**	**abs_diff**	**k**
w = 5					
SAX	19/19	10/15	10/15	1/2	50/60
eSAX	12/10	19/50	19/50	5/2	50/70
1d-SAX	19/19	14/31	14/31	2/2	50/70
aSAX	19/19	10/15	10/15	1/2	50/60
Persist	19/19	10/15	10/15	1/2	50/60
w = 10					
SAX	23/16	5/7	5/7	1/2	50/80
eSAX	14/9	12/25	12/25	5/2	50/100
1d-SAX	23/20	7/16	7/16	2/2	50/100
aSAX	23/16	5/7	5/7	1/2	50/80
Persist	23/16	5/7	5/7	1/2	50/80
Raw	-/-	8/15	-/-	0.5/0.5	30/40

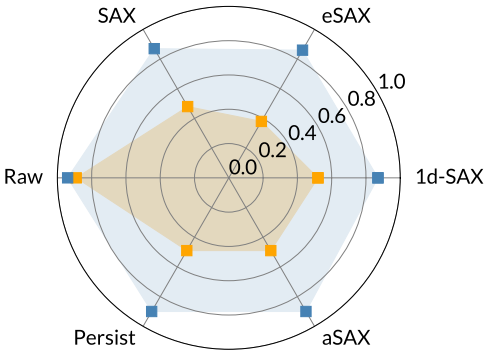
Table 5.7: This table contains the parameter values used for the evaluation. In the first column, $w = 5$ and $w = 10$ indicates the window length that was used for the PAA. For the cells of the other columns, x/y represents the respective parameter values x and y that were used for the datasets `Motif60` and `Motif120`, respectively. Parameter a in the second column represents the alphabet size used for discretization. For the 1d-SAX, the alphabet size used for discretizing the means is presented in the table. The alphabet size used for discretizing the slopes was $a = 3/2$ for $w = 5$ and $w = 10$. The remaining columns contain the values used for the parameters of the Brute Force motif discovery algorithm as described in Section 4.3. Moreover, as Minkowski distance D, the Manhattan distance was used for all configurations. Further, note that the length l of the examined subsequences in the Brute Force motif discovery algorithm is determined by the respective fixed length of the recurrently occurring subsequences of the time series contained in `Motif60` and `Motif120` and the used window length w.



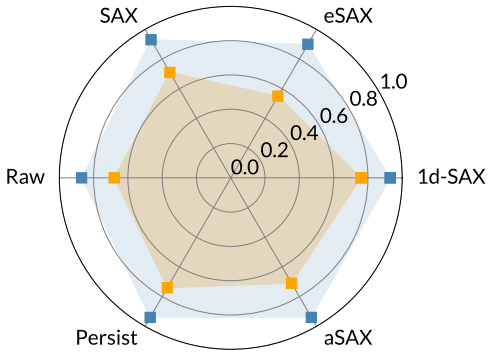
(a) $\mathbf{w = 5}$, `Motif60`: For the Motif Recall, all time series discretization algorithms perform worse than the Raw version that obtained 0.89. The Motif Precision for all evaluated algorithms, except the eSAX, is between 0.92 and 0.95 (inclusive). With a Motif Recall of 0.66, the 1d-SAX performs best across the time series discretization algorithms. The eSAX performs worst with a Motif Precision of 0.89 and a Motif Recall of 0.47.



(b) $\mathbf{w = 5}$, `Motif120`: For all evaluated algorithms, the 1d-SAX performs best for both measures of goodness with a Motif Precision of 0.93 and a Motif Recall of 0.87. For both measures of goodness, the eSAX performs worst across all evaluated algorithms with a Motif Precision of 0.87 and a Motif Recall of 0.67. However, the Raw version also obtained a Motif Precision of 0.87 and a Motif Recall of 0.68.



(c) $\mathbf{w = 10}$, `Motif60`: The Motif Precision for all time series discretization algorithms, except the eSAX, is between 0.87 and 0.90 (inclusive). For the Motif Recall, the 1d-SAX performs best across the time series discretization algorithms with a value of 0.52. The eSAX performs worst with a Motif Precision of 0.86 and a Motif Recall of 0.38.



(d) $\mathbf{w = 10}$, `Motif120`: For all time series discretization algorithms, except the eSAX, the Motif Precision is between 0.93 and 0.94 (inclusive). With a Motif Recall of 0.76, the 1d-SAX performs best for all evaluated algorithms. With a Motif Precision of 0.90 and a Motif Recall of 0.55, the eSAX performs worst for all time series discretization algorithms.

Motif Precision Motif Recall

Figure 5.9: This figure presents the results for the computed summary statistics of the Motif Precision and Motif Recall for each evaluated time series discretization algorithm and the Raw version with respect to the Brute Force motif discovery algorithm. The window length w used for the PAA and the used dataset is indicated for each subfigure. Note that the Raw version does not depend on the window length w.

	w = 5		**w = 10**	
	`Motif60`	`Motif120`	`Motif60`	`Motif120`
SAX	1.48	1.43	1.65	1.26
eSAX	1.57	1.36	1.72	1.49
1d-SAX	1.36	1.11	1.55	1.17
aSAX	1.42	1.33	1.67	1.30
Persist	1.53	1.41	1.70	1.23
Raw	1.12	1.65	1.12	1.65

Table 5.8: This table contains the computed summary statistics of the MMDL for each evaluated time series discretization algorithm and the Raw version with respect to the Brute Force motif discovery algorithm. The window length w used for the PAA and the used dataset are indicated by the columns. Note that the Raw version does not depend on the window length w. As for the evaluation based on the previous two motif discovery algorithm, the comparison of the values of the MMDL with the values of the Motif Recall presented in Figure 5.9, also indicates a negative correlation between these two measures of goodness.

5.3 Memory Requirements

Let $X = x_1, ..., x_N$ be a standardized time series of length $N \geq 1$. Further, suppose that $1 \leq n \leq N$ is the number of subsequences the PAA extracted from X for applying one of the time series discretization algorithms described in Chapter 3. Let $\hat{X}$ be the resulting discretized time series corresponding to X. Note that for the SAX, aSAX, and Persist, n is the length of $\hat{X}$. For the 1d-SAX and eSAX, $\hat{X}$ has a length of $2n$ and $3n$ as they discretize two and three features per extracted subsequence, respectively. Now, define the Compression Factor as [29]:

$$\text{Compression Factor} = \frac{n}{N} \tag{5.6}$$

Further, assume that storing one (real-valued) time series point of X requires 64 bits and storing one alphabet symbol of $\hat{X}$ requires $\lceil \log_2(a) \rceil$ bits, where $a \geq 2$ is the alphabet size used for discretization [29].
Remember that for the 1d-SAX, two separate alphabets are used [19]. One for discretizing the means and one for discretizing the slopes of the subsequences extracted by the PAA. Therefore, storing the respective alphabet symbols may require a different amount of bits when the two alphabets have different sizes.
Based on the above assumptions, let $B(X) = 64 \cdot N$ be the total amount of bits

required for storing X and let $B(\hat{X})$ be the total amount of bits required for storing $\hat{X}$. Then, define the Compression Ratio as [29]:

$$\text{Compression Ratio} = \frac{B(\hat{X})}{B(X)} \cdot 100 \tag{5.7}$$

Note that due to the different amount of discretized features per extracted subsequence by the PAA, $B(\hat{X})$ is different for the time series discretization algorithms described in Chapter 3. For the SAX, aSAX, and Persist, it is $B(\hat{X}) = \lceil \log_2(a) \rceil \cdot n$ [29]. Further, for the 1d-SAX it is $B(\hat{X}) = \lceil \log_2(a_m) \rceil \cdot n \ + \lceil \log_2(a_s) \rceil \cdot n$, where $a_m \geq 2$ is the alphabet size used for discretizing the means and $a_s \geq 2$ is the alphabet size used for discretizing the slopes of the subsequences extracted by the PAA. Lastly, it is $\lceil \log_2(a) \rceil \cdot 3n$ for the eSAX.

Analysis for Different Window Lengths

Assuming that N is divisible by n, it follows from the definition that the Compression Factor is the reciprocal of the window length used for the PAA in the first step of the time series discretization algorithms described in Chapter 3. Therefore, given an alphabet size used for discretizing, the Compression Ratio can be analyzed for discretizing X based on different window lengths used for the PAA and the different time series discretization algorithms (see Figure 5.10). Note that the Persist was analyzed based on the modification of discretizing the points of the PAA representation (i.e. the extracted means) instead of the points of the corresponding original standardized time series (see Subsection 3.3.2). This was done to increase the comparability of the Persist with respect to the other time series discretization algorithms, as the Persist is employed with this modification throughout the evaluation in this thesis.
From the theoretical analysis above, it follows that the Compression Ratio corresponding to the eSAX is three times larger than that corresponding to the SAX, aSAX, and Persist, and one and a half times larger than that corresponding to the 1d-SAX, when it is $a_m = a = a_s$. Moreover, the Compression Ratio corresponding to the 1d-SAX is two times larger than that corresponding to the SAX, aSAX, and Persist, when it is $a_m = a = a_s$.

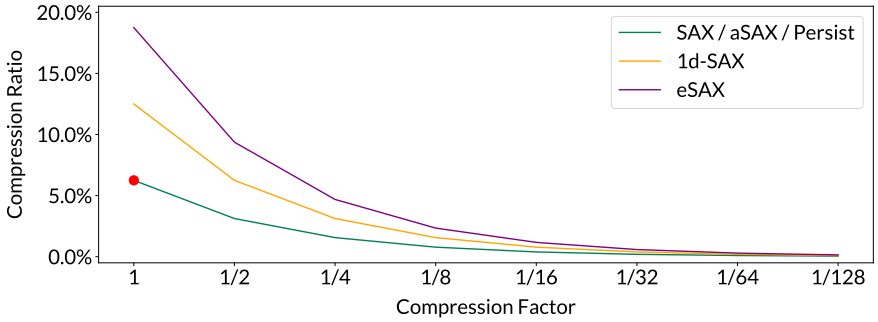


Figure 5.10: For this plot, the Compression Factor (i.e. the window length used for the PAA) is varied for a given alphabet size of $a = 16 = a_m = a_s$. The plot reflects the theoretical analysis above. Furthermore, it shows for example that for a Compression Factor of 1, the Compression Ratio is around 6-7% for the SAX/aSAX/Persist (red dot). This means that storing all alphabet symbols of $\hat{X}$ requires around 6-7% of the bits that are required for storing all points of X. This analysis holds regardless of the actual length N of X.

Analysis for Different Alphabet Sizes

From the theoretical analysis above, it also follows that the Compression Ratio is dependent on the alphabet size a. Therefore, given a Compression Factor, the Compression Ratio can be analyzed for different alphabet sizes (see Figure 5.11). Note that the Persist was analyzed based on the modification as described for the analysis for different window lengths above.

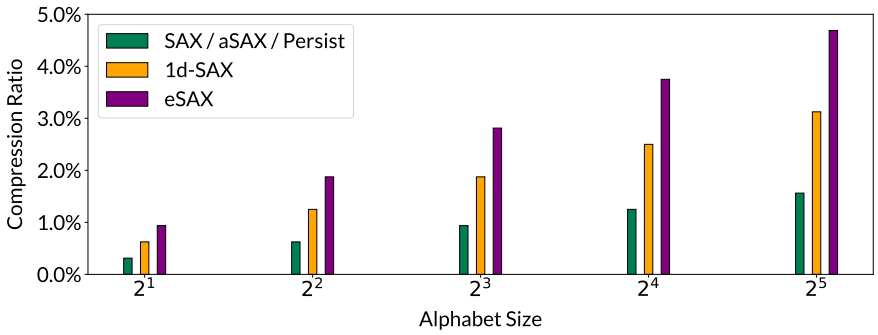


Figure 5.11: For this plot, the alphabet size $a = a_m = a_s$ used for discretization is varied for a given Compression Factor of $1/5$ (i.e. a window length of 5 used for the PAA). For each alphabet size $a \in (2^{p-1}, 2^p]$ $(p \geq 2)$, the Compression Ratio equals that corresponding to 2^p, since all of the a alphabet symbols can be represented by p bits. As soon as the alphabet size is greater than 2^p, the Compression Ratio increases to that corresponding to 2^{p+1}.

6 Summary & Outlook

6.1 Summary

In this thesis, the time series discretization approaches called SAX, eSAX, 1d-SAX, aSAX, and Persist were evaluated.
The reconstruction error was used as a metric to quantify and evaluate the goodness of these approaches with respect to a feature-preserving discretized representation of the corresponding original time series. For the reconstruction error, the deviation between the original time series and a numerical reconstruction of the corresponding discretized time series is measured. Qualitatively, the results are as follows:

eSAX & 1d-SAX:
The eSAX and 1d-SAX perform best across all evaluated time series discretization approaches and configurations. Compared to the other evaluated time series discretization approaches, the eSAX and 1d-SAX benefit from using more information for the discretization of time series.

SAX & aSAX: The results for the SAX and aSAX are similar across all evaluated configurations except for one. For time series with a specific structure, the aSAX performs better than the SAX. The reason is that the aSAX adapts its discretization process to the respective time series at hand, while the SAX does not adapt its discretization process.

Persist: The Persist performs worst across all evaluated time series discretization approaches and configurations, except for the configuration for that the aSAX performs better than the SAX as explained above. For this configuration, the Persist performs similar to the aSAX, as it adapts its discretization process to the time series data at hand, as well.

In addition to measuring the reconstruction error, three motif discovery algorithms were applied to evaluate the applicability of the examined time series discretization approaches as a preprocessing step for those motif discovery algorithms. Qualitatively, the SAX, aSAX, and Persist perform similar across all evaluated

configurations, while the eSAX performs worst. Moreover, the results indicate that the 1d-SAX slightly outperforms all other evaluated time series discretization approaches for one of the three applied motif discovery algorithms. For the other two, the 1d-SAX performs similar to the SAX, aSAX, and Persist.
For comparison, two of the three applied motif discovery algorithms were also executed with the original time series as input (i.e. without any preprocessing step). Overall, these results indicate that the precision with which recurrently occurring patterns in time series are detected only decreases slightly, if at all, when using the time series discretization algorithms for preprocessing. However, the number of recurrently occurring patterns detected in time series decreases more and more often.

6.2 Outlook

In this thesis, a time series is assumed to be univariate, meaning that for each observed point in time, there is only one observation [4]. However, time series can also be multivariate, meaning that for each observed point in time there are multiple, but the same number of observations, where each observation belongs to a different univariate time series [4]. Therefore, a possible extension of the evaluation in this thesis, could comprise time series discretization approaches that are applicable for multivariate time series.
Moreover, in this thesis, it is assumed that the observations of a time series are observed at uniformly spaced points in time. This assumption could be relaxed by evaluating time series discretization approaches that are applicable for time series with observations that are observed at irregularly spaced points in time.
Lastly, this thesis focuses on motif discovery with respect to time series pattern recognition. This scope could be extended by incorporating other tasks of time series pattern recognition like anomaly detection [4].

Bibliography

[1] Ashok Reddy. "Accelerate Speed Of Data And AI-Driven Business Value With Time Series Technology". In: *Forbes* (2023). url: \url{https://www.forbes.com/sites/forbestechcouncil/2023/01/13/accelerate-speed-of-data-and-ai-driven-business-value-with-time-series-technology/}.

[2] Stephan Heinrich. *Flash Memory in the emerging age of autonomy*. Flash Memory Summit, Santa Clara (CA, USA), 2017. https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2017/20170808_FT12_Heinrich.pdf.

[3] *DB-Engines*. Popularity Changes per Category - Complete trend. https://db-engines.com/en/ranking_categories. 2023.

[4] Philippe Esling and Carlos Agon. "Time-series data mining". In: *ACM Computing Surveys (CSUR)* 45.1 (2012), pp. 1–34.

[5] Leigh Metcalf and William Casey. *Cybersecurity and applied mathematics*. Syngress, 2016.

[6] Morris H DeGroot and Mark J Schervish. *Probability and statistics*. Pearson Education, 2012.

[7] Peter Bruce, Andrew Bruce, and Peter Gedeck. *Practical statistics for data scientists: 50+ essential concepts using R and Python*. O'Reilly Media, 2020.

[8] Fabian Mörchen and Alfred Ultsch. "Optimizing time series discretization for knowledge discovery". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 660–665.

[9] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[10] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[11] R Azulay et al. "Temporal Discretization of medical time series-A comparative study". In: *IDAMAP 2007 workshop*. 2007.

[12] Eamonn Keogh et al. "Dimensionality reduction for fast similarity search in large time series databases". In: *Knowledge and information Systems* 3 (2001), pp. 263–286.

[13] Byoung-Kee Yi and Christos Faloutsos. "Fast time sequence indexing for arbitrary Lp norms". In: (2000).

[14] Jessica Lin et al. "Experiencing SAX: a novel symbolic representation of time series". In: *Data Mining and knowledge discovery* 15 (2007), pp. 107–144.

[15] Jessica Lin et al. “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*. 2003, pp. 2–11.

[16] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. “Fast subsequence matching in time-series databases”. In: *ACM Sigmod Record* 23.2 (1994), pp. 419–429.

[17] Patrick Schäfer and Mikael Högqvist. “SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets”. In: *Proceedings of the 15th international conference on extending database technology*. 2012, pp. 516–527.

[18] Battuguldur Lkhagva, Yu Suzuki, and Kyoji Kawagoe. “Extended SAX: Extension of symbolic aggregate approximation for financial time series data representation”. In: *DEWS2006 4A-i8* 7 (2006).

[19] Simon Malinowski et al. “1d-sax: A novel symbolic representation for time series”. In: *Advances in Intelligent Data Analysis XII: 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings 12*. Springer. 2013, pp. 273–284.

[20] Hoang Anh Dau et al. *The UCR Time Series Classification Archive*. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`. 2018.

[21] Matthew Butler and Dimitar Kazakov. “Sax discretization does not guarantee equiprobable symbols”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.4 (2014), pp. 1162–1166.

[22] Ninh D Pham, Quang Loc Le, and Tran Khanh Dang. “Two novel adaptive symbolic representations for similarity search in time series databases”. In: *2010 12th International Asia-Pacific Web Conference*. IEEE. 2010, pp. 181–187.

[23] Rajashree Dash, Rajib Lochan Paramguru, and Rasmita Dash. “Comparative analysis of supervised and unsupervised discretization techniques”. In: *International Journal of Advances in Science and Technology* 2.3 (2011), pp. 29–37.

[24] Ayushi Verma, Neetu Sardana, and Sangeeta Lal. “Developer Recommendation for Stack Exchange Software Engineering Q&A Website based on K-Means clustering and Developer Social Network Metric”. In: *Procedia Computer Science* 167 (2020), pp. 1665–1674.

[25] JLEKS Lonardi and Pranav Patel. “Finding motifs in time series”. In: *Proc. of the 2nd Workshop on Temporal Data Mining*. 2002, pp. 53–68.

[26] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. “Probabilistic discovery of time series motifs”. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 493–498.

[27] Chin-Chia Michael Yeh et al. “Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets”. In: *2016 IEEE 16th international conference on data mining (ICDM)*. Ieee. 2016, pp. 1317–1322.

[28] Bernard Hugueney. "Adaptive segmentation-based symbolic representations of time series for better modeling and lower bounding distance measures". In: *Knowledge Discovery in Databases: PKDD 2006* (2006), pp. 545–552.

[29] Anita Sant'Anna and Nicholas Wickström. "Symbolization of time-series: An evaluation of sax, persist, and aca". In: *2011 4th international congress on image and signal processing*. Vol. 4. IEEE. 2011, pp. 2223–2228.

[30] Kiburm Song, Minho Ryu, and Kichun Lee. "Transitional sax representation for knowledge discovery for time series". In: *Applied Sciences* 10.19 (2020), p. 6980.

[31] Fabian Kai Dietrich Noering. *Unsupervised Pattern Discovery in Automotive Time Series: Pattern-based Construction of Representative Driving Cycles*. Vol. 159. Springer Nature, 2022.

[32] David MW Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: *arXiv preprint arXiv:2010.16061* (2020).