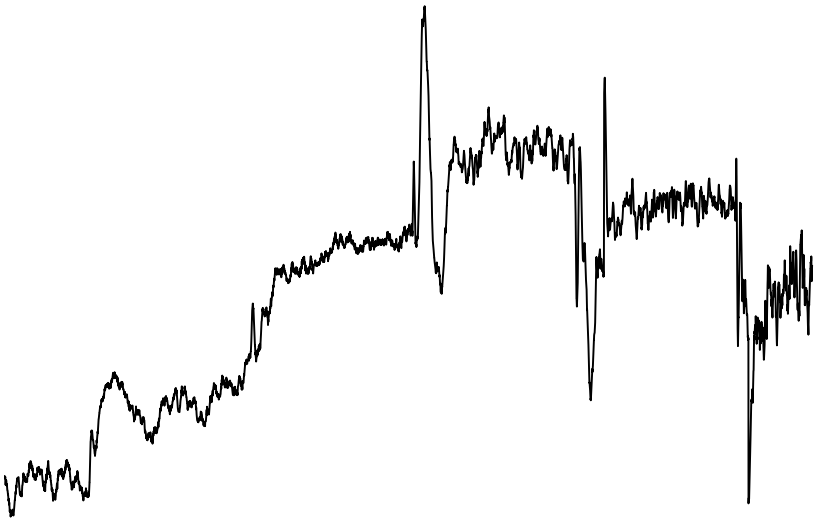
CHALMERS



A Study on Time Series Data Mining Techniques for Automotive Applications

Master's Thesis in Engineering Mathematics and Computational Science

EMIL STAF

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

Master's Thesis: EX018/2016

A Study on Time Series Data Mining Techniques for Automotive Applications

EMIL STAF

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

A Study on Time Series Data Mining Techniques for Automotive Applications
EMIL STAF

Supervisors
Krister Johansson, Volvo Cars
Lia Silva-Lopez, Combine Control Systems AB

Examiner
Tomas McKelvey, Professor, Vice Head of Department, Signals and Systems

Master's Thesis: EX018/2016
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Printed by Reproservice (Chalmers printing services)
Gothenburg, Sweden 2016

A Study on Time Series Data Mining Techniques for Automotive Applications

EMIL STAF
emil.staf@gmail.com
Department of Signals and Systems
Chalmers University of Technology

Abstract

A company from the automotive sector has shown interest in looking for cause-effect relations in some systems with multiple input and output signals. The available data from such systems is in the form of time series. This thesis aims to study and develop univariate time series data mining tools, which will be used for clustering, classification and anomaly detection in time series subsequences.

A popular choice for performing data analysis over time series subsequences is the use of Motif-detection. Motifs are defined as frequently occurring patterns of subsequences. A reason why Motif-detection is popular is because the concept is fairly simple to understand and implement. Therefore one of the data mining methods explored in this thesis is Motif based.

We found some drawbacks using a Motif based data mining technique. A standard Motif based technique returning optimal Motifs has shown not to be scaleable with an increasing number of subsequences. This result originate from the vital part to compare every subsequence to all the other subsequences and thus its time complexity is quadratic in the number of subsequences. Moreover, no good solution to update the Motifs, as new time series data arrive, has been found.

Considering such drawbacks, we explored Mixture Model based data mining techniques. Basing ourselves on the concepts of an existing state of the art online kernel density estimation technique, namely oKDE, we derive an even lighter technique suitable for finding patterns in time series subsequences. The technique we propose, Compressed Mixture Model, is suitable for real time applications because of the possibility to incrementally update the model. Our technique is linear in the number of subsequences and thus overcomes the drawbacks of using a Motif based technique, which is not scalable in the number of subsequences.

Methods for classification and anomaly detection of subsequences using cluster models obtained from the two clustering methods mentioned are proposed. The classification and anomaly detection methods take into account of the locations, shapes as well as the frequencies of clusters.

The Compressed Mixture Model is concluded superior to the clustering method using Motifs in many ways. Especially, the Compressed Mixture Model allows incremental learning which in turn enable online clustering, classification and anomaly detection.

Keywords: Machine Learning, Data Mining, Clustering, Classification, Anomaly Detection, Density Modeling, Time Series Analysis

Sammanfattning

Ett företag inom fordonsindustrin har visat intresse av att analysera samband mellan orsak och verkan samband i system med flera in- och utsignaler. Den tillgängliga datan för analys från de tänkta systemen är i form av tidsserier. Målet med det här examensarbetet är att utveckla ett verktyg för tidsserieanalys, av endimensionell tidsseriedata, med egenskaper som tillåter klustring, klassificering samt upptäckning av onormalt beteende hos delsekvenser i tidsserier.

Ett populärt val vid analys av delsekvenser i tidsserier är att använda sig av motiv. Motiv definieras som frekventa mönster hos delsekvenser i en tidsserie. Att motiv är ett populärt val har framför allt två orsaker. Konceptet är tämligen enkelt att förstå samt att implementera, varför ett av de två föreslagna verktygen för tidsserieanalys är baserat på motiv.

Under arbetets gång upptäckte vi ett antal nackdelar med att använda en teknik baserad på motiv. En teknik med avseende på att hitta optimala motiv, samt är skalbar i antalet delsekvenser, hittades inte. Detta grundar sig i det kritiska momentet att jämföra alla delsekvenser med varandra, vilket resulterar i kvadratisk tidskomplexitet med avseende på antalet delsekvenser. Dessutom har ingen bra lösning för att uppdatera en motivbaserad model hittats.

Med tanke på dessa nackdelar utforskade vi en probabilistisk teknik för att utföra tidsserieanalys. Genom att utgå från en toppmodern teknik för att online estimera en täthetsfunktion baserad på sannolikhetskärnor, oKDE, härleder vi en lättviktsversion lämpad för att hitta mönster hos delsekvenser i tidsseriedata. Tekniken som vi föreslår, Compressed Mixture Model, är lämplig för online-applikationer i och med möjligheten att inkrementellt uppdatera modellen. Tekniken är linjär i antalet delsekvenser och överkommer därmed nackdelarna av att använda en motivbaserad modell.

Förslag av metoder ges för klassificering samt upptäckning av onormalt beteende, genom att använda de två föreslagna metoderna för klustring av delsekvenser i tidsserier. Metoderna tar hänsyn till både position, form samt frekvens hos klustren.

Vi drar slutsatsen att den föreslagna tekniken, Compressed Mixture Model, är överlägsen den motivbaserade tekniken för klustring i flera avseenden. I synnerhet tillåter sig den probabilistiska tekniken, Compressed Mixture Model, att utföra klustring, klassificering samt upptäckning av avvikelser på kontinuerligt strömmande data.

Acknowledgements

First of all, I am very grateful for the opportunity to carry out my Master's thesis at Combine. I extend my gratitude towards Gustaf Gulliksson who introduced me to Combine. It has proven rewarding in many ways, especially the opportunity of being surrounded by such intelligent and equal-minded people.

I would like to extend my deepest gratitude to my two industrial supervisors, Lia Silva-Lopez and Krister Johansson, for their enthusiasm, commitment and never ending flood of ideas. The initial development of CMM by Krister proved immensely helpful.

Last but not least, I would like to thank my family and girlfriend Karin for their support, understanding and for always being there in times of need.

Emil Staf, Göteborg June 16, 2016

Contents

1

Introduction

BUILDING cleaner and safer cars involves many challenges for data analytics. Until relatively recent years, a significant amount of such challenges have been tackled by humans. Global economic pressures together with more demanding regulatory frameworks create the need for more thorough testing. More thorough testing increases the rate at which data is being generated, at a point in which the traditional ways of analyzing data is no longer suitable. This has created a motivation for the automotive industry to look into computing techniques for analyzing time series data.

A company from the automotive sector has shown interest in looking for cause-effect relations in some systems with multiple input and output signals (see Figure 1.1). The available data from such systems is in the form of time series. More specifically the company would like to use such

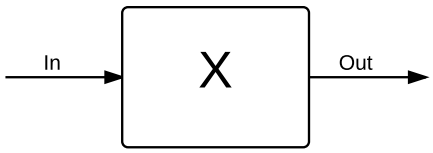


Figure 1.1: System X.

data mining tool for compressing many Gigabytes of time series data into a much small number of unique subsequence scenarios (see Figure 1.2). The content of this thesis focus on solving the problem of learning unique patterns of in-/output signals of a system X. When all the unique subsequence scenarios have been extracted and stored, the plan is to use them for system development purposes (see Figure 1.3). Before the patterns can be used for system development purposes, an engineer will have to manually look through the unique output patterns extracted for the signals of interest in system X. For each existing output pattern the engineer label it as either OK or NOT OK. The engineer should have a vast knowledge of the system and the signals at hand to be trusted with such an important task. Such development processes might involve simulating system behavior by feeding various input patterns. For such a process an engineer will have to manually analyze the simulated system output signals. Also, parameter tuning of control system parameters being part of system X is achievable having access to use the unique patterns. If a method to reconstruct the (controllable) input signals of system X, verification in

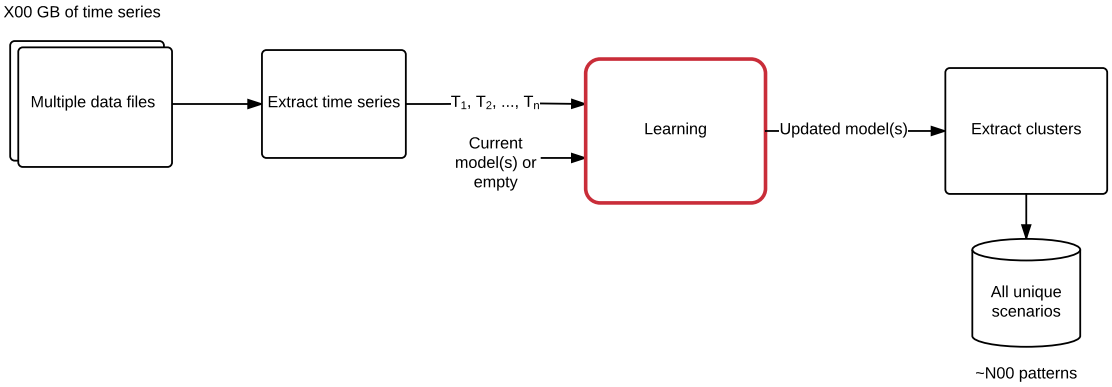


Figure 1.2: Pattern learning phase. A huge amount (X00 GB) of time series data is compressed into a small number of unique subsequence scenarios (∼N00 scenarios per time series).

form of a Design of Experiments (DoE) would be possible to perform. At last, regression of new data producing reports of the system output is made possible by having a simple model of each signal (see Figure 1.4).

This thesis aims to study and develop univariate time series data mining tools, which will be used for clustering, classification and anomaly detection in time series subsequences.

Esling and Agon [3] summarize three fundamental aspects to consider when working with any time-series data mining system: *data representation*, *similarity measures*, and *indexing methods*. A good time series representation technique should derive the notion of shape by reducing the dimensionality of data while retaining its essential characteristics. A proper similarity measure should establish a notion of similarity between any pair of time series. Such notion of similarity is usually based on some perceptual criteria. Finally, indexing methods facilitate scaling the algorithm in the amount of data points.

For our work, we will be concerned with data representation methods and similarity measures. Two data representation techniques will be studied namely the Piecewise Aggregate Approximation (PAA) introduced by Keogh et al. [9] and the Symbolic Aggregate approXimation (SAX) introduced by Lin et al. [13]. We argue that PAA, the predecessor to SAX, serves as the better choice of data representation method for the application at hand. Reasons for such statement will be explained and summarized in section 4.1.

Regarding similarity measures, we will use a measure that Keogh et al. [9] presented for PAA and a measure that Lin et al. [13] presented for SAX. In section 2.3 we will explain the reasons why those measures are suitable for our study. Indexing methods are out of the scope of this work. Yet, shall our tool be taken into production, a suitable indexing method will indeed affect performance in a positive way. Considering that, a by-product of this study is concerned with some thoughts regarding indexing methods. Those thoughts are discussed in section 4.2.4.

A method is considered to be meaningless if the output is independent of the input. Conversely, a meaningful method adapts to the input data and produces results accordingly. The former notion was coined by Keogh and Lin [8]. They also showed that traditional clustering methods applied on time series subsequences produce meaningless results. Then, to avoid meaningless results and instead find meaningful clusters, the same authors suggested the concept of Motifs in time series. Time series Motifs are defined as frequently occurring patterns of subsequences. One of the methods for clustering time series subsequences in this thesis relies upon

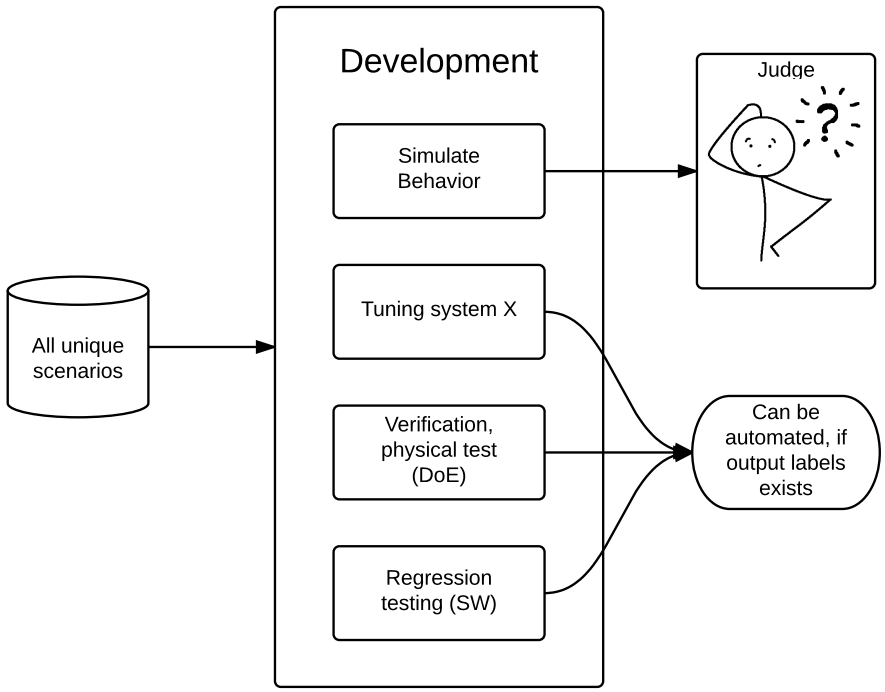


Figure 1.3: Pattern usage phase. The unique subsequence scenarios extracted are used for development purposes. Having access to all the unique patterns it is possible to perform simulations of system behavior feeding various input patterns. An engineer will have to manually analyze the simulated system output signals. Also, parameter tuning of control system parameters being part of system X is in the scope of how to use the unique patterns. If a method to reconstruct (controllable) input signals based on the unique patterns or model found can be developed, verification in form of a Design of Experiments (DoE) is possible to perform. At last, regression of new data producing reports of the system output is made possible by having a simple model of each signal.

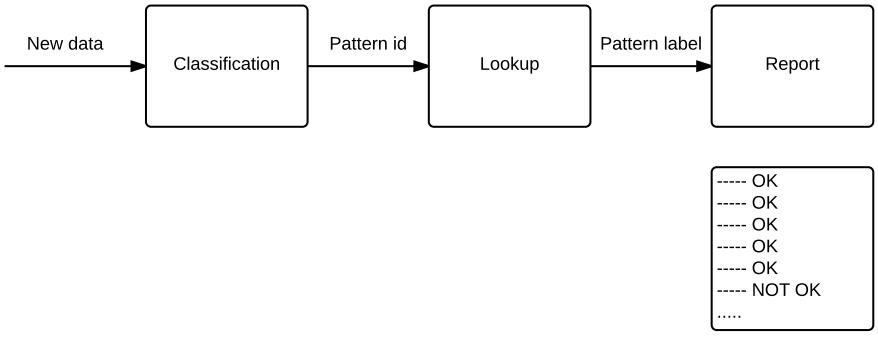


Figure 1.4: Regression of system response on new input data.

the concept of Motifs.

We also use a completely different technique for comparison. Based on the concepts of an existing state of the art online kernel density estimation technique proposed by Kristan et al. [10], namely oKDE, we derive an even lighter technique. The technique we propose, Com-

pressed Mixture Model (CMM), is suitable for real time applications because of the possibility to incrementally update the model. Additionally, our technique scales linear to the number of subsequences. CMM discovers and adapts the number of clusters as well as their location and shape in a feature space. It features online clustering, classification and anomaly detection. It also allows storage reduction in the form of compression. Classification and anomaly detection, using CMM, takes into account for the shapes and the frequencies of clusters.

The second chapter of this thesis introduces the reader to the underlying theory for the later proposed clustering, classification and anomaly detection techniques. In addition, the chapter covers the concepts of time series representation techniques, similarity measures, how to merge two Gaussian distributions and at last a part regarding density modeling. In the Method chapter, a detailed outline of two data mining methods is provided. One of the methods feature a threshold based solution of the data mining tasks, while the other method features a probabilistic solution. A comparison of the two methods as well as an evaluation of the highest potential method when applied on use case data is given in the Results & Discussion chapter. Moreover, the results will be discussed in parallel with the results shown. In the Conclusions chapter conclusions regarding the two methods are presented. Finally, in the Future Work chapter some proposals for improvements and additional features for the method with the higher potential are made.

2

Theory

This chapter is initialized by a short introduction to general time series concepts. Then, the two time series representation techniques *Piecewise Aggregate Approximation* (PAA) and *Symbolic Aggregate approXimation* (SAX) are presented. The two representation techniques is followed by a section introducing similarity measures for the two time series representation techniques. Also, similarity measures for probability distributions are presented in the same section. This is followed by a method to merge two Gaussian distributions, together with a method to estimate the information lost by merging. At last, theory covering kernel density estimation and mixture model is provided.

2.1 Time Series Introduction

Esling and Agon [3] defines a time series as an ordered sequence of real-valued variables.

Definition 2.1. A *time series* T is an ordered sequence of n real-valued variables

$$T = (t_1, \ldots, t_n), t_i \in \mathbb{R}.$$

Throughout this study time series measurements are assumed to be collected at a constant sampling rate. Thereby, the time series consists of uniformly spaced (in time) measurements from the underlying process. The time series considered in the thesis are real-valued, and thus the data mining methods are developed for such time series. Esling and Agon [3] continues defining subsequences of a time series.

Definition 2.2. Given a time series $T = (t_1, \ldots, t_n)$ of length n, a *subsequence* S of T is a time series of length $m \leq n$ consisting of contiguous time instants from T

$$S = (t_k, t_{k+1}, \ldots, t_{k+m-1})$$

with $1 \leq k \leq n - m + 1$. We denote the set of all subsequences of length m from T as $\mathbf{S}_T^m$.

The time series in Definition 2.1 is *univariate*, i.e. the resulting measurements from one process producing one dimensional values. Multivariate time series are several univariate time series that together span multiple dimensions for the same set of time stamps.

2.2 Time Series Representation Techniques

Time series are often approximated by dimensionality reducing representations to deal with one major issue for time series data mining, namely the *high dimensionality* of the data. Esling and Agon [3] provides a formal definition of a time series representation.

Definition 2.3. Given a time series $T = (t_1, \ldots, t_n)$ of length n, a *representation* of T is a model $\bar{T}$ of reduced dimensionality $\bar{d}$, ($\bar{d} << n$), such that $\bar{T}$ closely resembles T.

As mentioned earlier the two time series representation techniques PAA and SAX are used throughout this study. The SAX representation is a symbolical version of the PAA representation, both will be described in more detail in the upcoming sections.

According to Keogh and Kasetty [7] it is well understood that it is not meaningful to compare time series with different offsets and amplitude. This statement is taken into consideration by Lin et al. [13] in the paper introducing SAX, where each subsequence is normalized to zero mean and unit standard deviation before obtaining the discrete representation.

Normalizing subsequences removes information regarding subsequence *amplitude* and *offset*. The similarity between subsequences is thus reduced to only account for *shapes* in time series subsequences. A special case is when the standard deviation of a subsequence is very small (almost constant subsequence), then by normalization unwanted patterns might emerge. Thus, normalization is performed if the standard deviation is higher than a threshold value ϵ. Otherwise the subsequence is considered totally flat (all zeros).

Throughout this study the time series subsequences are normalized, zero mean and unit standard deviation, before finding a dimensionality reduced representation. The results obtained will have to be analyzed keeping the normalization in mind.

The time series representation techniques will be visualized using the time series in Figure 2.1.

2.2.1 Piecewise Aggregate Approximation

The time series representation technique PAA is introduced by Keogh et al. [9] and a brief summary of the technique follows in this section. For a more detailed description the reader is referred to Keogh et al. [9].

A time series subsequence $S = (s_1, \ldots, s_m)$ can be represented in a reduced N-dimensional space ($1 \leq N \leq m$) by a vector $\bar{S} = (\bar{s}_1, \ldots, \bar{s}_N)$. The element $\bar{s}_i$ is calculated according to the decimation scheme (2.1).

$$\bar{s}_i = \frac{N}{m} \left(\sum_{j=\frac{m}{N}(i-1)+1}^{\frac{m}{N}i} s_j \right) \tag{2.1}$$

It is assumed that N is a factor of m, which is not a requirement, but a simplification for the notation. The vector $\bar{S}$ is referred to as the PAA representation of S.

In simpler terms the PAA representation is obtained by dividing S into equi-sized *frames*, where the mean value of the data within each frame form a vector of means and becomes the PAA representation $\bar{S}$.

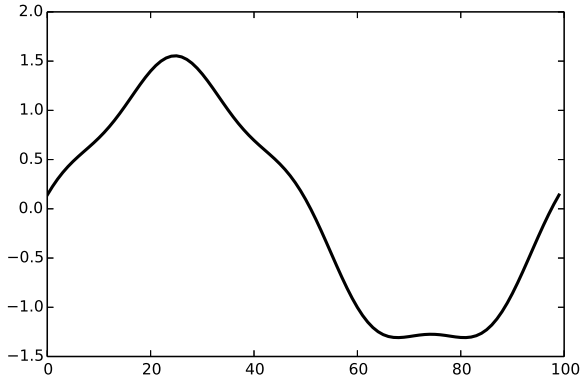


Figure 2.1: Normalized signal to visualize the two time series representation techniques, PAA and SAX.

Note: that the PAA representation in equation (2.1) can be obtained by filtering and down-sampling of a time series[1].

An example of how to reduce a subsequence S into a PAA representation $\bar{S}$ is graphically visualized in Figure 2.2.

According to Keogh et al. [9], the time complexity to generate PAA representations for all subsequences of length m (i.e. $\boldsymbol{S}_T^m$) from a time series of length n is $\mathcal{O}(|\boldsymbol{S}_T^m|m)$. The reasoning behind this statement is that for $|\boldsymbol{S}_T^m|$ subsequences (2.1) has to be calculated N times and (2.1) has a summation length of m/N. It is possible to reduce this complexity to $\mathcal{O}(|\boldsymbol{S}_T^m|N)$ according to Keogh et al. [9, p. 5], which will be valuable if $N << m$.

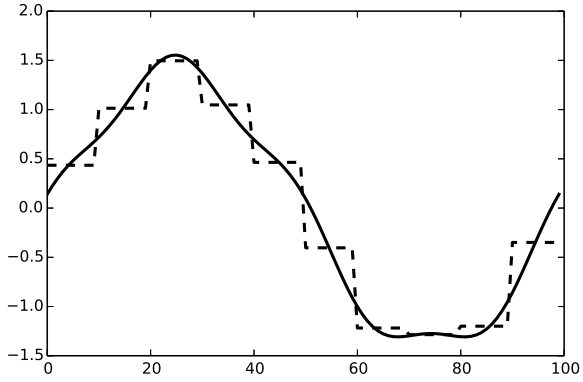


Figure 2.2: PAA representation $\bar{S}$ (dashed line) of the signal S (solid line) using a frame size of 10.

[1]The dimensionality reduced representation PAA could equally well be obtained by filtering the time series using a non-causal symmetrical FIR-filter, where all filter coefficients are the same (=N/m).

2.2.2 Symbolic Aggregate approXimation

Having transformed a normalized[2] subsequence S into a PAA representation $\bar{S}$, it is possible to apply one additional transformation to obtain a discrete representation. The symbolic time series representation technique SAX is introduced by Lin et al. [13] and inherits the dimensionality reduction obtained from the PAA representation. A brief summary of the time series representation technique follows. For a more detailed description of SAX the reader is referred to Lin et al. [13].

It is indicated by Lin et al. [13] that it is beneficial to have equiprobable symbols. Also, the authors state that normalized subsequences have a standard Gaussian distribution. They propose splitting space into α equiprobable parts by determining *breakpoints* using a standard Gaussian distribution in Definition 2.4.

Definition 2.4. *Breakpoints*: breakpoints are a sorted list of numbers $B = \beta_1, \ldots, \beta_{\alpha-1}$ such that the area under a $N(0,1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/\alpha$ (β_0 and β_α are defined as $-\infty$ and ∞, respectively).

From the breakpoints using an *alphabet* of size α, the SAX representation $\hat{S}$ is obtained from the PAA representation $\bar{S}$ by assigning a discrete symbol for each element in $\bar{S}$ based on the alphabet. Using the English alphabet, all elements $\bar{s}_i$ smaller than the first breakpoint are assigned the first symbol in the alphabet **a**, all elements greater than or equal to the first breakpoint but smaller than the second breakpoint are assigned the second symbol of the alphabet **b**, etc.

The concatenation of symbols that represents a subsequence is referred to as a *word* and the definition given by Lin et al. [13] follows.

Definition 2.5. *Word*: A subsequence S of length n can be represented by a *word* $\hat{S} = \{\hat{s}_1, \ldots, \hat{s}_N\}$ as follows. Let alpha_i denote the i^{th} element of the alphabet, i.e., $\text{alpha}_1 = \mathbf{a}$ and $\text{alpha}_2 = \mathbf{b}$. Then the mapping from a PAA approximation $\bar{S}$ to a word $\hat{S}$ is obtained as follows:

$$\hat{s}_i = \text{alpha}_\text{j}, \quad \text{iff} \quad \beta_{j-1} \leq \bar{s}_i < \beta_j \tag{2.2}$$

A word obtained using Definition 2.5 will be referred to as the SAX representation of a subsequence from here on. An example of how to reduce a subsequence S into a SAX representation $\hat{S}$ is graphically visualized in Figure 2.3.

Note: that in signal processing terms the procedure of obtaining the SAX representation is referred to as quantization.

One matter deserving to be enlightened is the fact that the SAX symbols have been proven not to be equiprobable by Butler and Kazakov [2]. More specifically, they show how the process of finding the PAA representation decreases the standard deviation of the reduced representation below one. This means that the symbols will be concentrated around the symbols in the middle of the alphabet. In this study the non-equiprobable symbols are regarded as a minor flaw of the SAX representation and no additional effort will be put into this matter.

2.3 Similarity measures

Esling and Agon [3] claims that nearly every task in time series data mining is dependent on a similarity measure between time series and gives a formal definition.

[2]Zero mean and unit standard deviation.

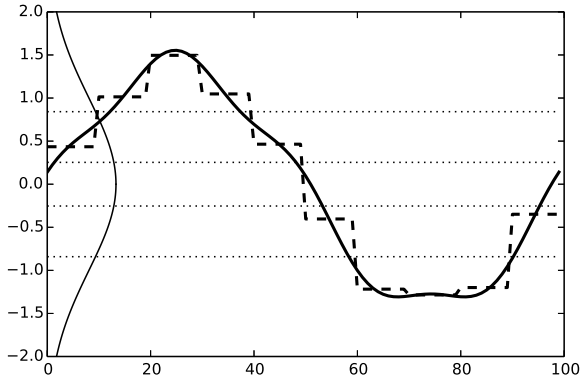


Figure 2.3: SAX representation $\hat{S}$ of the signal S (solid thick line) using an alphabet size of $\alpha = 5$ and number of frames $N = 10$ is **deeedbaaab** (using the English alphabet). The horizontal dotted lines divides space into equiprobable regions with respect to a standard Gaussian distribution and the SAX representation is obtained by studying in which region the PAA representation (dashed line) falls into for each frame.

Definition 2.6. The *similarity measure* $\mathcal{D}(T, U)$ between time series T and U is a function taking two time series as inputs and returning the *distance* d between these series.

Definition 2.6 is general enough to define similarity measures between dimensionality reduced time series representations by replacing T, U by either the PAA representations $\bar{T}, \bar{U}$ or the SAX representations $\hat{T}, \hat{U}$. The distance has to be non-negative, i.e. $\mathcal{D}(T, U) \geq 0$.

A measure is a metric if it in addition to the non-negative property satisfies the *symmetry* property $\mathcal{D}(T, U) = \mathcal{D}(U, T)$ and *subadditivity* (known as the *triangle inequality*) $\mathcal{D}(T, V) \leq \mathcal{D}(T, U) + \mathcal{D}(U, V)$ and also $\mathcal{D}(T, U) = 0 \Leftrightarrow T = U$.

If two subsequences are found sufficiently similar with respect to a threshold value δ they are said to be a *match*.

Definition 2.7. Given a query time series Q and a similarity measure $\mathcal{D}(Q, T)$, the two time series Q and T is a *match* of order δ if $\mathcal{D}(Q, T) \leq \delta$, for any choice of $\delta > 0$.

A commonly used and easy to implement similarity measure is the Euclidian distance, which is a metric by definition. If a similarity measure lower bounds the Euclidian distance between any two time series (or dimensionality reduced representations) it implies that a match in the original space will still be a match in the reduced space under this similarity measure. Faloutsos et al. [4] refer to this property as the *Lower Bounding Lemma* and it guarantees no false dismissals.

The Euclidian distance between two time series $Q = (q_1, \ldots, q_n)$ and $C = (c_1, \ldots, c_n)$ is given by

$$D(Q,C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}. \tag{2.3}$$

2.3.1 Similarity Measure for Piecewise Aggregate Approximation

The similarity measure between two PAA representations proposed by Keogh et al. [9] is a scaled euclidian distance in the reduced space.

$$DR(\bar{Q}, \bar{C}) = \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (\bar{q}_i - \bar{c}_i)^2} \tag{2.4}$$

Here n is the number of data points in the original time series and N is the dimension (number of frames) in the reduced space. This similarity measure satisfies the conditions for being a metric, since it is a scaled Euclidian distance in the reduced space. The similarity measure $DR(\bar{Q},\bar{C})$ has been proven to be a lower bound approximation of the Euclidian distance $D(Q,C)$, proof given by Keogh et al. [9, Appendix A].

2.3.2 Similarity Measure for Symbolic Aggregate approXimation

The similarity measure $MINDIST(\hat{Q},\hat{C})$ proposed by Lin et al. [13] lower bounds the PAA similarity measure $DR(\bar{Q},\bar{C})$ and thereby inherits the lower bounding property of the Euclidian distance $D(Q,C)$, proof given by Lin et al. [14].

$$MINDIST(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (dist(\hat{q}_i, \hat{c}_i))^2} \tag{2.5}$$

Here N is the word length (equal to the number of frames in PAA), and n is the number of data points in the original space. The function $dist(\hat{q}_i, \hat{c}_i)$ returns the smallest distance between the symbols $\hat{q}_i$ and $\hat{c}_i$. It can be implemented using a lookup table as illustrated in Table 2.1. The

	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Table 2.1: A lookup table used by $MINDIST$ for an alphabet size $\alpha = 4$. As the lookup table illustrates, the distance between two adjacent symbols is zero.

distance between two adjacent symbols has to be zero in order to guarantee the lower bounding property. The value in each cell of the lookup table (Table 2.1) is calculated with respect to the breakpoints according to (2.6).

$$cell_{r,c} = \begin{cases} 0, & \text{if } |r - c| \leq 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & \text{otherwise.} \end{cases} \tag{2.6}$$

Interesting to note is that while both the Euclidian distance $D(Q,C)$ and the PAA distance $DR(\bar{Q},\bar{C})$ are metrics, the SAX distance $MINDIST(\hat{Q},\hat{C})$ is not. It fails to satisfy the two properties of *subadditivity* and $\mathcal{D}(\hat{Q}, \hat{C}) = 0 \Leftrightarrow \hat{Q} = \hat{C}$, due to the fact that the distance between adjacent symbols is zero.

Example 2.8. For example the distance between the two words **aaa** and **aac** is greater than zero, but the distance between **aaa** and **aab** as well as the distance between **aab** and **aac** is zero. Thus, by taking the detour from **aaa** to **aab** and then going to **aac** from **aab** gives a total distance of zero.

There is an important trade-off to consider when choosing which of the two representation methods to work with. As a direct consequence of using $MINDIST$ for SAX distance measure, the resolution (distance between SAX representations) is traded against a smaller cardinality discrete representation. In applications where a discrete representation is necessary this is an easy trade, but for applications where the trade can be abstained PAA is to prefer.

2.3.3 Similarity Measures for Probability Densities

To measure the distance between two probability densities, information theory measures serves as great candidates. Two possible measures, the Kullback-Leibler divergence and the Hellinger distance, are presented below.

Kullback-Leibler divergence

Kullback-Leibler divergence is an information theory measure that measures the difference between two probability distributions P and Q. According to Burnham and Anderson [1, p. 51] the Kullback-Leibler divergence can be interpreted as either the information lost when the distribution Q is used to approximate the distribution P or the distance from Q to P. The Kullback-Leibler divergence in the case of continuous distribution functions is given by (2.7).

$$\bar{D}_{\mathrm{KL}}(P,Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \tag{2.7}$$

In the special case when comparing two Gaussian distributions, $P \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $Q \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, the Kullback-Leibler divergence have a closed form solution.

$$\bar{D}_{\mathrm{KL}}(P,Q) = \frac{1}{2}\left(\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathsf{T}}\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - k + \ln\left(\frac{\det \boldsymbol{\Sigma}_1}{\det \boldsymbol{\Sigma}_0}\right)\right) \tag{2.8}$$

Here k is the number of dimensions of the distributions.

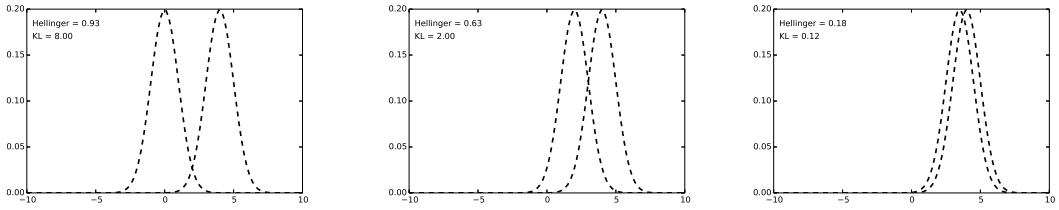
The Kullback-Leibler divergence is always non-negative, but lack the property of symmetry. The non-symmetry property can be taken care of with the following simple adjustment.

$$D_{\mathrm{KL}}(P,Q) = \frac{1}{2}(\bar{D}_{\mathrm{KL}}(P,Q) + \bar{D}_{\mathrm{KL}}(Q,P)) \tag{2.9}$$

The Kullback-Leibler divergence is in this study calculated using the symmetric formula in (2.9). Examples of Kullback-Leibler divergence values comparing two Gaussian distributions is presented in Figure 2.4.

Hellinger Distance

The Hellinger distance is an information theory distance and quantifies the similarity between two probability distributions. In (2.10) the squared Hellinger distance is stated in the case of continuous distribution functions P and Q.



(a) Distant locations **(b)** Midrange locations **(c)** Close locations

Figure 2.4: Kullback-Leibler divergence and Hellinger distance between two Gaussian distributions with distant locations (left) midrange locations (middle) and close locations (right). The standard deviation is equal for the two Gaussian distributions in all subplots.

$$D_{\mathrm{H}}^2(P,Q) = \frac{1}{2}\int \left(\sqrt{\frac{dP}{d\lambda}} - \sqrt{\frac{dQ}{d\lambda}}\right)^2 d\lambda \tag{2.10}$$

In the special case when $P \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) = \mathcal{N}_0$ and $Q \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) = \mathcal{N}_1$, the squared Hellinger distance have a closed form solution.

$$D_{\mathrm{H}}^2(P,Q) = 1 - \frac{|\boldsymbol{\Sigma}_0|^{1/4}|\boldsymbol{\Sigma}_1|^{1/4}}{\left|\frac{\boldsymbol{\Sigma}_0+\boldsymbol{\Sigma}_1}{2}\right|^{1/2}} \exp\left\{-\frac{1}{8}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathsf{T}} \left(\frac{\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1}{2}\right)^{-1} (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)\right\} \tag{2.11}$$

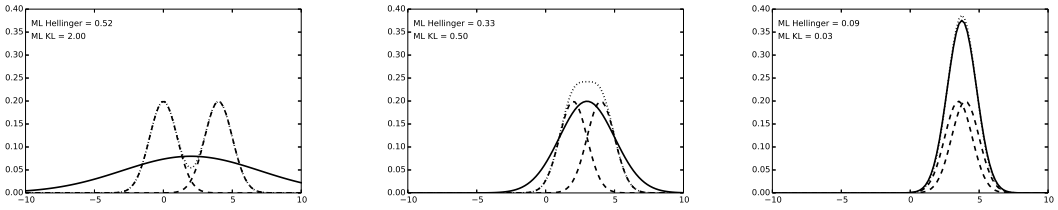
The Hellinger distance satisfies the property $0 \leq D_{\mathrm{H}}(P,Q) \leq 1$ and also the property of symmetry. Examples of Hellinger distance values comparing two Gaussian distributions is presented in Figure 2.4.

2.4 Merging Gaussian Distributions

Consider two multivariate Gaussian distributions $P_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $P_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ of weights w_1 and w_2 respectively. An aggregation method to merge the two probability distributions is suggested by Kristan et al. [10] using moment matching, and the resulting composite $Q \sim \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ with weight $\hat{w}$ is stated in (2.12).

$$\begin{aligned}
\hat{w} &= \sum_{i=1}^{2} w_i \\
\hat{\boldsymbol{\mu}} &= \hat{w}^{-1} \sum_{i=1}^{2} w_i \boldsymbol{\mu}_i \\
\hat{\boldsymbol{\Sigma}} &= \hat{w}^{-1} \sum_{i=1}^{2} w_i \left(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\mathsf{T}}\right) - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^{\mathsf{T}}
\end{aligned} \tag{2.12}$$

If the two distributions P_1 and P_2 are replaced by their composite Q according to (2.12) information will be lost. It is of interest in this study to bound the amount of information lost. This requires a measure of the information lost, let us refer to this quantity as the *merge loss*.



(a) Distant locations **(b)** Midrange locations **(c)** Close locations

Figure 2.5: Merge loss when merging the two Gaussian distributions (dashed lines) into their composite Gaussian (solid line). The dotted lines in the subplots are the weighted sum of the two individual Gaussian distributions in each subplot respectively. The merge loss describes the distance between the composite (solid line) and the weighted sum (dotted line). In this figure the two original Gaussian distributions have equal weights of 0.5 and both the composite and the weighted sum has weights of 1.

The merge loss can be assessed by calculating the information lost by replacing each individual of the original distributions by the composite and then averaging the results using the weights as in (2.13).

$$ML(P_1,P_2) = \hat{w}^{-1} \sum_{i=1}^{2} w_i D_{\mathrm{xx}}(P_i,Q) \tag{2.13}$$

In equation (2.13) D_{xx} can be replaced by either the Kullback-Leibler divergence D_{KL} (2.8) inserted in (2.9) or the Hellinger distance D_{H} (2.11).

Hellinger distance is preferable to Kullback-Leibler divergence. By using Hellinger distance the merge loss is a quantity with the property $0 \leq ML(P_1, P_2) \leq 1$, where 0 means no merge loss and 1 means total merge loss. Thus, the merge loss is easier to interpret using Hellinger distance than using Kullback-Leibler divergence with merge loss values ranging between 0 and ∞.

Examples how the composite looks like merging two Gaussian distributions is graphically visualized in Figure 2.5, using the same example distributions as in Figure 2.4.

2.5 Density Modeling

Density modeling is a powerful tool in order to characterize a process. Often the kind of tasks that involve decision making can benefit the most from density models. A density model enables possibilities of drawing random samples from a process without physically running it and finding out how likely an event is to occur.

2.5.1 Kernel Density Estimation

Kernel Density Estimation (KDE) or Parzen estimator is a non-parametric method utilizing multiple *local* kernels to estimate a *global* probability density function. By construction, a KDE has the ability of handling various data types choosing suitable kernels for the data at hand. KDE is non-parametric and exploratory in the sense that it does not assume any prior distribution of the data. According to Parzen [16] and Shalizi [17] the KDE for observed real-valued univariate

data x_i (the i$^{\text{th}}$ observation) is obtained by adding together all the local kernels as

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-x_i}{h}\right), \tag{2.14}$$

where K is a kernel function which integrates to 1 and h the *kernel width* or bandwidth. The factor $1/h$ ensures $\hat{f}_h(x)$ integrates to 1.

It can be found that the Integrated Squared Error (ISE), a common quantity to measure quality of the KDE, is given by (2.15).

$$\text{ISE} \approx \frac{h^4\sigma_K^4}{4}\int \left(f''(x)\right)^2\,dx + \frac{\int K^2(u)\,du}{nh} \tag{2.15}$$

In (2.15) $f(x)$ is the true underlying distribution to be estimated. The optimal bandwidth selection is found as the optima (minima) of ISE (2.15) by taking the derivative with respect to h and finding the stationary point h_{opt}.

$$h_{\text{opt}} = \left(\frac{\int K^2(u)\,du}{\sigma_K^4\int \left(f''(x)\right)^2 dx}\right)^{1/5} n^{-1/5} = \mathcal{O}(n^{-1/5}) \tag{2.16}$$

Here $\sigma_K^2 = \int K(u)u^2\,du$. For a more detailed derivation the reader is referred to Shalizi [17, p. 313-315]. By plugging in h_{opt} found from (2.16) into ISE (2.15) the best ISE goes to zero in the order of $\mathcal{O}(n^{-4/5})$. Parzen [16] derive at the same convergence rate with a different derivation approach.

For a Gaussian kernel one can show, using (2.16), that the optimal bandwidth for estimating a Gaussian distribution is $1.06\sigma n^{-1/5}$, where σ is the standard deviation of that Gaussian distribution. This is referred to as the *Gaussian reference rule* kernel width selection and can be used as a less time consuming alternative to *cross-validation*. Another technique discussed by Shalizi [17] is the *plug-in method*. The plug-in method starts with an initial bandwidth from e.g. the Gaussian reference rule and then finds a preliminary estimate of the density. This estimate can be plugged in into equation (2.16) to get a new bandwidth and it is possible (not necessary) to iterate multiple times.

Multivariate Kernel Density Estimate

The transition from univariate to multivariate data is accomplished by defining a multivariate kernel. A multivariate kernel could be defined in many ways. Two of the most common are to either use a multivariate distribution as the kernel for the vector $\boldsymbol{x}$ or to use a product of univariate kernels to form the multivariate kernel. Shalizi [17] defines a product of kernels.

$$K(\boldsymbol{x}-\boldsymbol{x}_i) = K_1(x^1 - x_i^1)K_2(x^2 - x_i^2)\cdot\ldots\cdot K_p(x^p - x_i^p) \tag{2.17}$$

Further, using the product of kernels proposed in equation (2.17) a bandwidth for each dimension is required and the multivariate KDE is given by (2.18).

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n\prod_{j=1}^{p}h_j}\sum_{i=1}^{n}\prod_{j=1}^{p}K_j\left(\frac{x^j - x_i^j}{h_j}\right). \tag{2.18}$$

Shalizi [17] that the ISE (Integrated Square Error) goes to zero like $\mathcal{O}(n^{-\frac{4}{(4+p)}})$, were p is the number of dimensions. This rate is asymptotically obtained using cross-validation to pick

bandwidth. From this result it is clear that as the number of dimensions increases the convergence rate decreases, which is recognized as the *curse of dimensionality*.

A consequence choosing a multivariate distribution as a kernel is that it is possible to incorporate correlations between dimensions for some choices of data types and distributions. By using products of univariate kernels as a multivariate kernel there are no dependencies between dimensions.

Mixed data types

Li and Racine [12] consider the problem of using kernel density to estimate an unknown distribution defined over a mixture of discrete and continuous variables. They define a multivariate kernel for categorical data in the simplified case when X is a k-dimensional binary variable, $X \in \{0,1\}^k$.

$$L_{ix} \equiv L(X_i,x,\lambda) = \prod_{t=1}^{k} l(X_{i,t},x_t) = (1-\lambda)^{k-d_{ix}} \lambda^{d_{ix}} \tag{2.19}$$

Here $d_{ix} = (X_i - x)^\mathsf{T}(X_i - x)$ equals the number of disagreement components between X_i and x and $X_{i,t}$ denote the t^{th} dimension of the i^{th} data point. The univariate categorical kernel $l(X_{i,t},x_t)$ is defined to take the value $l(X_{i,t}, x_t) = 1 - \lambda$ if $X_{i,t} = x_t$ and $l(X_{i,t}, x_t) = \lambda$ if $X_{i,t} \neq x_t$, where λ is a smoothing parameter.

They further define a multivariate kernel for continuous variables.

$$W_{h,iy} \equiv W_h(Y_i, y) = h^{-p} W\left(\frac{Y_i - y}{h}\right) = h^{-p} \prod_{t=1}^{p} w\left(\frac{Y_{i,t} - y_t}{h}\right) \tag{2.20}$$

Here $w(\cdot)$ is a univariate kernel function and $Y_{i,t}$ denotes the t^{th} dimension of the i^{th} data point. Having a multivariate kernel for categorical data as well as a multivariate kernel for continuous data Li and Racine [12] move on to propose to estimate the underlying mixed density function by $\hat{f}(x,y)$ given in equation (2.21). Let $Z = (X,Y)$, and $f(z) = f(x,y)$.

$$\hat{f}(z) = \frac{1}{n} \sum_{i=1}^{n} K_{h,iz} \tag{2.21}$$

Here $K_{h,iz} = L_{ix} W_{h,iy}$.

2.5.2 Mixture Model Intro

Shalizi [17] defines a **mixture model** f as a mixture of K component distributions $f_1, f_2, \ldots, f_K$

$$f(x) = \sum_{k=1}^{K} \lambda_k f_k(x; \theta_k), \tag{2.22}$$

where λ_k defines the *mixing weight* of component k, $\lambda_k > 0$ and $\sum_k \lambda_k = 1$. It is possible to generate data from $f(x)$ due to the fact that $f(x)$ is a probability density function for the mixture. The mathematical expressions for generating random samples is given in equation (2.23).

$$\begin{aligned} Z &\sim Mult(\lambda_1, \lambda_2, \ldots, \lambda_k), \\ X|Z &\sim f_Z(x). \end{aligned} \tag{2.23}$$

In plain english, first a component Z is randomly chosen based on the mixing weights and then a data point is generated based on the randomly chosen components density function f_Z.

3

Method

The goal of this study is to build a univariate time series analysis software tool covering the data mining tasks of clustering, classification and anomaly detection on subsequences of a time series. Two different approaches were studied: a Motif based and a Mixture Model based. Using Motifs we developed threshold based methods to solve the data mining tasks, while using the Mixture Model based method we developed probabilistic methods to solve the same set of data mining problems.

3.1 Clustering

According to Esling and Agon [3] clustering is the process of finding natural groups in a data set. A clustering method should tend to minimize the variation within a cluster while maximizing the variation between clusters.

In this study it is of interest to find clusters in the set of all subsequences $\mathbf{S}_T^m$ of a time series.

The concepts of Motifs and Components presented in the two upcoming sections will be used to represent clusters of time series subsequences.

3.1.1 Motifs

Keogh and Lin [8] make a surprising claim and provides convincing evidence that traditional clustering of time series subsequences is meaningless. They define an algorithm to be *meaningless* if the output produced is independent of the input.

Time series Motifs are overrepresented subsequences in a time series and the definition by Keogh and Lin [8] is stated in Definition 3.2.

An important observation made by Keogh and Lin [8] is that different subsequences among the set of all subsequences $\mathbf{S}_T^m$ have different numbers of trivial matches. A slightly modified definition of trivial matches from the one suggested by Keogh and Lin [8] is used in this study, found in Definition 3.1.

Definition 3.1. Given a subsequence S beginning at position p, a matching subsequence M beginning at q, and a distance R, define M to be a *trivial match* to S of order R, if either

$p = q$ or there does not exist a subsequence M' beginning at q' such that $\mathcal{D}(S, M') > R$, and either $q < q' < p$ or $p < q' < q$. In addition the two subsequences S and M can not share less than 50 percent of their data points to be considered trivial matches.

The difference between an uneventful and an eventful time series is that subsequences of the former tends to have a tremendous amount of trivial matches in contrast to the latter. The amount of trivial matches greatly influence the position of cluster centers and therefore not frequently occurring subsequences with many trivial matches would be preferred as cluster centers over frequently occurring subsequences with less trivial matches.

A workaround to find meaningful clusters by using the concept of Motifs is suggested. In this study the algorithm proposed to find Motifs is unsuitable for massive data sets, since it scales quadratic with the number of subsequences.

Definition 3.2. Given a time series T and a distance range R, the most significant motif in T (called 1-*Motif*) is the subsequence S_1 that has the highest count of non-trivial matches. Subsequently, the J^{th} most significant motif in T (called J-*Motif*) is the subsequence S_J that has the highest count of non-trivial matches, and satisfies $\mathcal{D}(S_J, S_i) > 2R$, for all $1 \leq i < J$.

Definition 3.2 ensures that no two Motifs can be matches. Also, no matching subsequence of a Motif is allowed to be a matching subsequence to any other Motif. The previously mentioned consequences is visualized in Figure 3.1. The concept of non-trivial matches was introduced

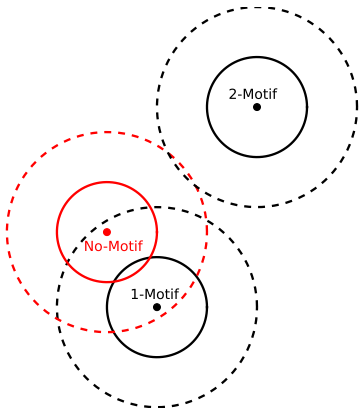


Figure 3.1: A simple graphical visualization in 2D showing the properties of Motifs. The solid circles have a radius of R from the Motif, while the dashed circle has a radius of $2R$. The point marked as No-Motif can not be considered a Motif since it is covered by the circle of radius $2R$ from the 1-Motif.

without a formal definition. Keogh and Lin [8] defines the set of non-trivial matches to a subsequence S_i as the set difference between all matching subsequences of S_i and its trivial matches. Defining non-trivial matches in such a way will not take into account for the trivial matches of subsequence S_i's non-trivial matches. In this study the following definition will be used.

Definition 3.3. Given a subsequence S_i its matching subsequences $\mathbf{M}$ of order R and its trivial matches $\mathbf{TM}$, consider the set $\mathbf{A} = \mathbf{M} \backslash \mathbf{TM}$. The set $\mathbf{A}$ holds S_i's non-trivial matches together with their trivial matches. Thus, S_i's *non-trivial matches* are found as the middle element among coherent subsequences in $\mathbf{A}$, where coherent subsequences are e.g. S_{j-2},$S_{j-1}, S_j, S_{j+1}, S_{j+2}$.

Finding Motifs

The algorithm to find Motifs in a univariate time series is presented in this section. The algorithm is inspired by the findings of Keogh and Lin [8] and is exhaustive in the sense that it finds all the Motifs of the input time series. A subsequence is considered a Motif only if it has at least one non-trivial match. The procedure of finding the Motifs in a time series is stated below.

1. Find all subsequences in T given a window size W, i.e. $\mathbf{S}_T^W$.
2. Find SAX representations for the subsequences $\hat{\mathbf{S}}_T^W$ given the set of parameters: word length w, alphabet size α and minimum standard deviation ϵ.
3. Calculate all pairwise distances between SAX representations $\mathfrak{D}$, using $MINDIST$.
4. Find and save the Motif $\mathcal{M}$ with the highest count of non-trivial matches according to $\mathfrak{D}$ with respect to a distance threshold R.
5. Remove the subsequences closer than $2R$ to $\mathcal{M}$.
6. Repeat 4-5 until no new Motifs can be found.

3.1.2 Compressed Mixture Model

In this section we outline an exploratory density model technique named Compressed Mixture Model (CMM), inspired by Kristan et al. [10], is outlined together with a novel proposition how to apply it on time series subsequences. It features online learning, compression (storage reduction), and exploration of not only the number of clusters but also their locations and shapes. As defined in section 2.5.2 a mixture model is a mixture of component densities. A CMM is a mixture model incrementally attained feeding subsequences incorporating possibilities of component merges. Thus a CMM is a probabilistic model of time series subsequences and the component densities found can be interpreted as cluster densities.

The core elements of the CMM is presented next.

Encoder

The encoder translates a subsequence into a *feature vector*. In this study the feature vector is the PAA representation of a normalized subsequence. A feature vector is then a real valued N-dimensional vector, where N is the number of frames in the PAA representation. The encoded subsequence will hence forth be referred to as the feature vector.

The PAA representation was chosen over the SAX representation for the feature vector because of the advantage of being able to use well-defined kernels as explained next.

Atomic Unit

An atomic unit is a kernel for the feature vector. The multivariate Gaussian kernel is used and an atomic unit is therefore a multivariate Gaussian kernel with a location defined by the feature vector and shape by a default covariance matrix Σ. Every component in a CMM is initially created from an atomic unit.

Three possible alternatives exist to define the covariance matrix Σ for the atomic unit. In increasing order of complexity: a spherical covariance matrix Σ (equal standard deviation for all feature dimensions); a diagonal covariance matrix Σ (non-equal standard deviations for all feature dimensions); and a full covariance matrix Σ incorporating feature dimension dependencies. In this study the first alternative is implemented and used, while the other two more complex alternatives could be considered in future studies.

Kernel Usually when dealing with multivariate data a kernel width for each dimension is needed. Using the PAA representation as the feature vector where the subsequence is normalized beforehand, the feature dimensions approximately ranges between ± 1 (unit standard deviation). Thus, one can argue that only a single kernel width for all feature dimension is needed.

The Gaussian reference rule $1.06\sigma n^{-1/5}$ suggests kernel widths depending on the number of data points according to Table 3.1, where unit standard deviation is assumed. A rule-of-thumb is

n	kernel width
1,000	0.25
10,000	0.16
100,000	0.1
1,000,000	0.06

Table 3.1: Kernel width suggested by the Gaussian reference rule for variables of unit standard deviation.

to choose the kernel density in the magnitude of 0.1 times the range for each feature dimension. Using PAA as the feature vector, each dimension ranges between ± 1 and thus a kernel width of 0.2 would be suitable. This is in accordance with the values in Table 3.1. Moreover, the two methods of cross-validation and plug-in method briefly mentioned earlier in section 2.5.1 are too computationally expensive for our application to be considered as an option for kernel width estimation.

Aggregation Method

If two components are to be merged the composite is found using the moment matching aggregation method to produce their composite as described in section 2.4. Important to note is that even though the default covariance matrix of an atomic unit is not accounting for feature dependencies, the method producing a composite is. This will result in an introduction of feature dimension dependencies during aggregation.

Component Distance

To assess distances between component densities in CMM, information theory measures serve the purpose well. Two well-known and commonly used probability measures are the Kullback-

Leibler Divergence 2.3.3 and the Hellinger Distance 2.3.3, and both of them are implemented in the CMM.

Merge Loss

The information lost when replacing two components C_1 and C_2 with their composite $\Psi(C_1,C_2)$ is an important part constructing a CMM. Replacing the two components should only be valid if the merge loss is smaller than a defined threshold value. The merge loss is assessed using the method described in section 2.4.

Definition 3.4. The operator $\Psi(C_1,C_2)$ returns the composite component of the two components C_1 and C_2.

CMM Update Procedure

A pseudo-code of the CMM update procedure is presented in Algorithm 3.1.

The algorithm parses through each feature vector x_i supplied in the list of feature vectors $\boldsymbol{X}$ (line 2). When a feature vector is supplied to the CMM an atomic unit c is created (line 3) using the feature vector x_i as the mean and a default covariance matrix as described by the Atomic Unit section in 3.1.2. Next, the distance between the atomic unit c and all components $c_k \in$ CMM is calculated (line 4) as described by the Component Distance section in 3.1.2 as well as the closest component K together with the distance δ is found (line 5). The composite $\Psi(c,K)$ is found (line 6) according to the Aggregation Method described in section 3.1.2 and the merge loss of replacing the two components c and K by their composite $\Psi(c,K)$ (line 7) according to the Merge Loss section in 3.1.2. The component distance is needed in addition to the merge loss because of the fact that the merge loss will decrease as a components weight grows and therefore huge components will tend to grow uncontrollably, "the snowball effect". By introducing also the component distance, the shape of the component density is taken into consideration which will maintain a controllable component growth. If the distance between c and K is smaller than the distance threshold $\bar{\delta}$ and if the merge loss is smaller than the merge loss threshold $\bar{\xi}$ then the merge is accepted and further action is required (lines 9-24). Otherwise the merge is rejected and the atomic unit is added as a component to the CMM (line 26) and the component distances between all pairs of components are updated and saved in Δ (line 27).

In the case of an accepted merge the following actions is needed. First, the composite $\Psi(c,K)$ is added to the CMM (line 9), then the component K is removed from the CMM (line 10) and the inter component distance Δ is updated (line 11). It is necessary to determine if any new merges were made possible by replacing component K with the composite $\Psi(c,K)$. As long as the two closest components in CMM c_a and c_b can be merged (lines 13-24), add their composite $\Psi(c_a,c_b)$ to CMM (line 18), remove them from CMM (line 19) and update the inter component distances Δ (line 20).

3.2 Classification and Anomaly Detection

Anomaly detection is a classification problem with two classes, normal and abnormal. In this study a subsequence is considered normal if it can be explained by a model of the subsequences generated from the same signal and abnormal otherwise. In the case a subsequence is classified as normal it belongs to a cluster in the model and will be labeled accordingly. Next the process of classification and anomaly detection is explained using Motifs and CMM respectively to model subsequences of a signal.

Algorithm 3.1: Compressed Mixture Model

Input: $\boldsymbol{X}$ - list of feature vectors, σ - kernel width, $\bar{\xi}$ - Merge loss threshold, $\bar{\delta}$ - Distance threshold

Output: CMM - Compressed Mixture Model (set of components)

Initialize: CMM (empty), Δ - inter component distances (empty)
for $x_i \in \boldsymbol{X}$ **do**
    Create a component (atomic unit) c of the feature vector x_i
    Calculate the distance between c and all $c_k \in$ CMM
    Let $K \in$ CMM be the closest component to c and δ their distance
    Find the composite $\Psi(c,K)$
    Calculate the merge loss ξ of replacing c and K with $\Psi(c,K)$
    if $\xi < \bar{\xi}$ and $\delta < \bar{\delta}$ **then**
        Add composite $\Psi(c,K)$ to CMM
        Remove K from CMM
        Update inter component distances Δ
        Set *continue* to TRUE
        while *continue* **do**
            Find the closest components $c_a, c_b \in$ CMM and their distance δ
            Find the composite $\Psi(c_a,c_b)$
            Calculate the merge loss ξ of replacing c_a and c_b with $\Psi(c_a,c_b)$
            if $\xi < \bar{\xi}$ and $\delta < \bar{\delta}$ **then**
                Add composite $\Psi(c_a,c_b)$ to CMM
                Remove c_a and c_b from CMM
                Update inter component distances Δ
            else
                Set *continue* to FALSE
            end
        end
    else
        Add c to CMM
        Update inter component distances Δ
    end
end
return CMM

3.2.1 Motifs

Assuming a model of subsequences from a signal have been acquired, where the model is the Motifs present in the time series. A new subsequence is assessed normal if it belongs to or can be explained by any of the Motifs in the model, or abnormal otherwise. A subsequence is defined to be explained by the model if the distance to the closest Motif is smaller than $2R$. Here R is

the same distance as used during training of the model. If the subsequence can be explained by the model it is considered normal and is labeled to belong to the closest Motif, otherwise it is abnormal.

3.2.2 Compressed Mixture Model

As mentioned earlier, a subsequence is considered normal if it can be explained by a model of subsequences generated from the same signal, otherwise abnormal.

Using a CMM as the model, a feature vector is considered abnormal if it can not be explained by the model or if it is unfrequent. In more details a feature vector is considered abnormal if the feature vectors atomic unit is:

1. not allowed to merge with a component in the CMM, or
2. allowed to merge with a component in the CMM, but the frequency of that component (number of subsequences added to that component during training) is smaller than a frequency threshold,

To choose a good value for the frequency threshold, it is important to consider the possibility of trivial-matches. A good threshold value will preferably be chosen in the range of the largest amount of possible trivial matches, i.e. the length of each subsequence m divided by 2. As the length of the subsequences outgrow the length of the patterns in the signal such a naive choice of threshold is no longer suitable, and the length of the smallest interesting pattern will then serve as a substitute for the subsequence length m when defining the threshold. The threshold will differ between signals due to their individual nature, some are rapidly changing while others are smooth and slowly changing.

By construction of the CMM every feature vector is considered equally important and the CMM is a model of both normal and abnormal behavior simultaneously. A component is considered abnormal as long as the frequency of the component is below the threshold value. Thus, abnormal components are regarded as temporarily abnormal until it is more frequently observed.

If a subsequence is considered normal it is labeled to belong to the closest component in the CMM based on component distance values.

4

Results & Discussion

This chapter starts of with a short discussion regarding the two time series representation techniques in the PAA vs SAX section. We will compare scalability and applicability of the Motif based and the Mixture Model based approach. Two datasets will be used for comparison: first a data set in which the most fundamental concepts can be explained with ease, and secondly a dataset from a real automotive application.

4.1 PAA vs SAX

In this study the SAX representation was used as a segmentation method for time series in the process of finding Motifs. Resolution was traded for a discrete representation method. The theory of finding Motifs does not rely on time series being represented using discrete representations. With this in mind, PAA would be just as suitable as the segmentation method in the Motif based approach. There exist other data mining applications in which a discrete time series is favorable. However, we decided to use SAX as the segmentation method in the Motif based approach for diversity. As we will soon see, the Motif based algorithm scales quadratic no matter which of the segmentation methods we decide to use.

The Compressed Mixture Model presented is using PAA representations as feature vectors and builds a probabilistic model of subsequences of a time series.

An important result of this study is that the similarity measure for SAX does not obey the triangular inequality. This fact is presented in section 2.3.2. Moreover, the simpler PAA representation is better than SAX at preserving resolution. This allows discrimination between relatively close subsequences in cases where SAX fails. For these reasons, we judge PAA to be preferable over SAX.

4.2 Applicability and Scalability

In this section, the two cluster approximation concepts Motifs and CMM are compared. They are both applied on the same data set with known features, $\mathbf{D}_1$, visualized in Figure 4.1.

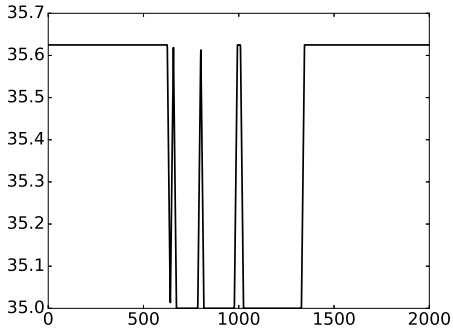
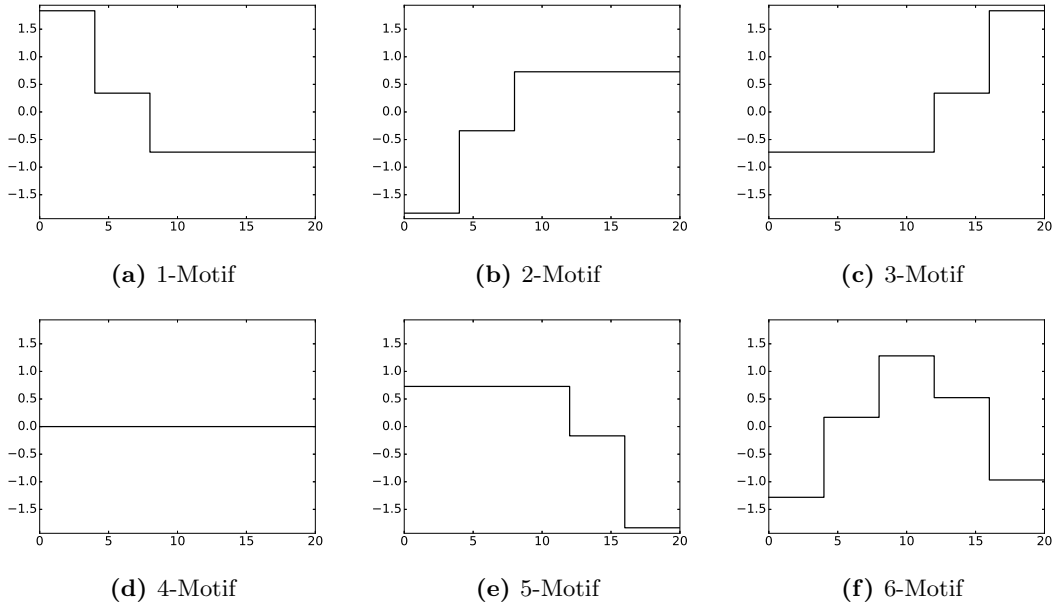


Figure 4.1: The data set of known features, $\mathbf{D}_1$, consisting of 2000 data points.



(a) 1-Motif **(b)** 2-Motif **(c)** 3-Motif

(d) 4-Motif **(e)** 5-Motif **(f)** 6-Motif

Figure 4.2: Motifs found in the data set $\mathbf{D}_1$ using the parameter values presented in Table 4.1 Motifs are SAX representations and is plotted in this figure using the 50-percentile of the symbol region.

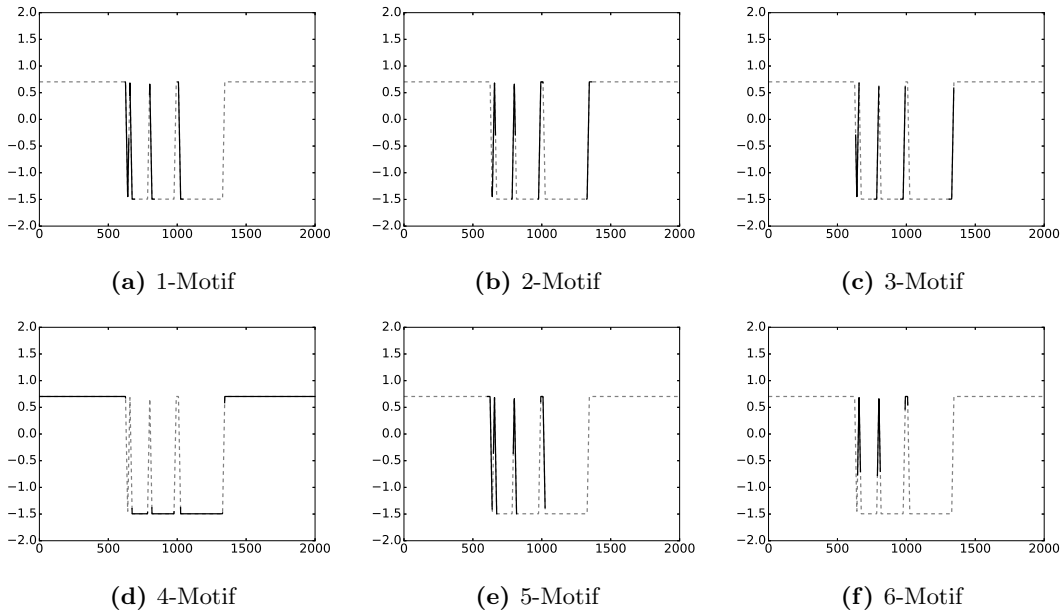
4.2.1 Motifs

The Motif finding algorithm (section 3.1.1) was applied on the data set $\mathbf{D}_1$ in order to find all the Motifs. The Motifs found are presented in Figure 4.2 using a parameter setting as displayed in Table 4.1.

Using the Motifs found as cluster centers, classification and anomaly detection (section 3.2.1) was performed on the same data set $\mathbf{D}_1$ using the same distance threshold $R = 0.6$ as during training. In Figure 4.3 the classification of subsequences in the data set $\mathbf{D}_1$ is visualized. There is one plot for each of the six Motifs found where each subsequence classified as that Motif is highlighted in black.

W (window size)	w (word length)	ϵ	α (alphabet size)	R (distance threshold)
20	5	10^{-4}	15	0.6

Table 4.1: Parameter settings used to find the Motifs present in the data set $\mathbf{D}_1$.



(a) 1-Motif **(b)** 2-Motif **(c)** 3-Motif

(d) 4-Motif **(e)** 5-Motif **(f)** 6-Motif

Figure 4.3: Data set $\mathbf{D}_1$ with highlighted areas according to the Motif labeling of subsequences.

One outlier was found using Motifs to model the data set $\mathbf{D}_1$. Each Motif holds information regarding its location and an outlier/anomaly is defined to be a subsequence further away than $2R$ from its closest Motif.

4.2.2 CMM

The Compressed Mixture Model (section 3.1.2) was trained on the data set $\mathbf{D}_1$ to find the components of the mixture. The distance metric used for training was the Hellinger distance (an

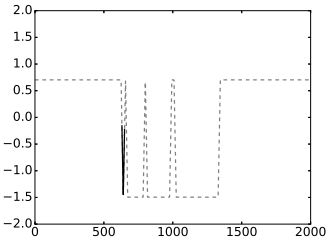
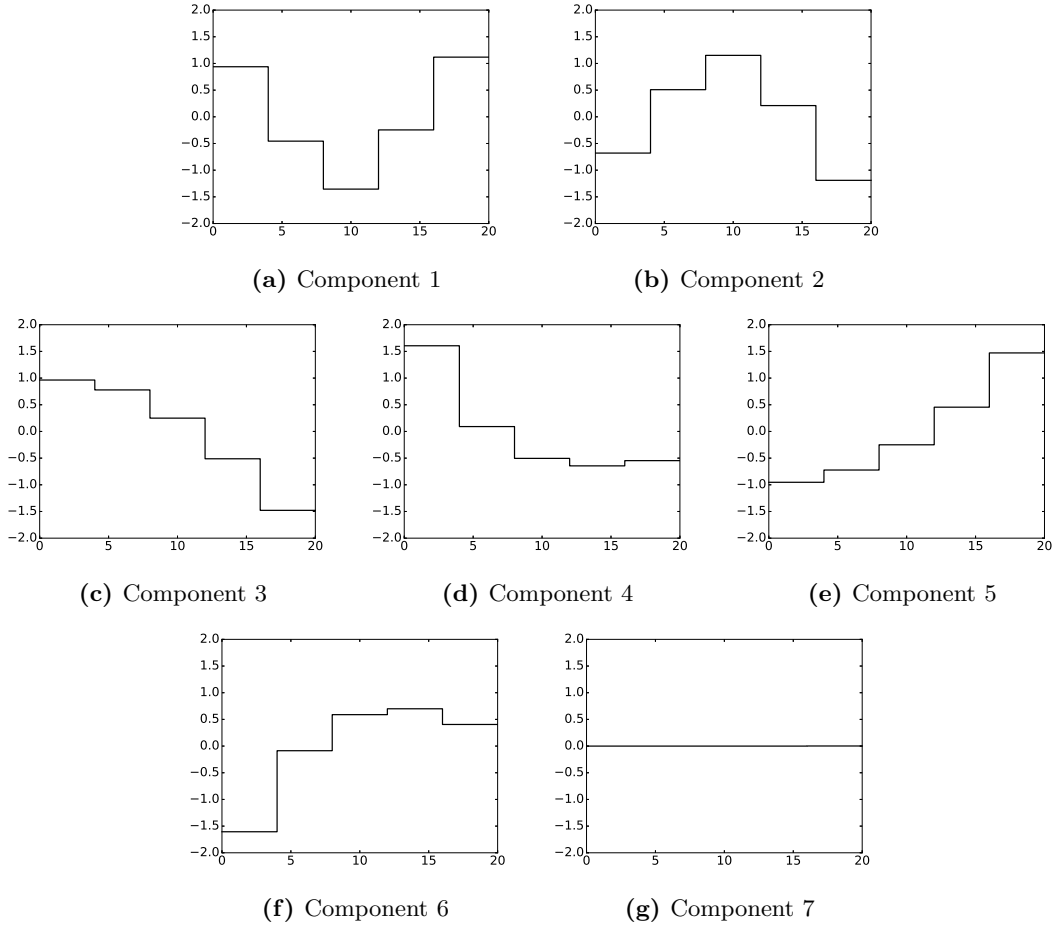


Figure 4.4: Outliers, i.e. subsequences not explained by the Motifs.



(a) Component 1 **(b)** Component 2

(c) Component 3 **(d)** Component 4 **(e)** Component 5

(f) Component 6 **(g)** Component 7

Figure 4.5: Component centers found in the data set $\mathbf{D}_1$ using the parameters presented in Table 4.2 and the Hellinger distance as the choice of distance method. The window size W, number of frames w as well as ϵ are given by Table 4.1.

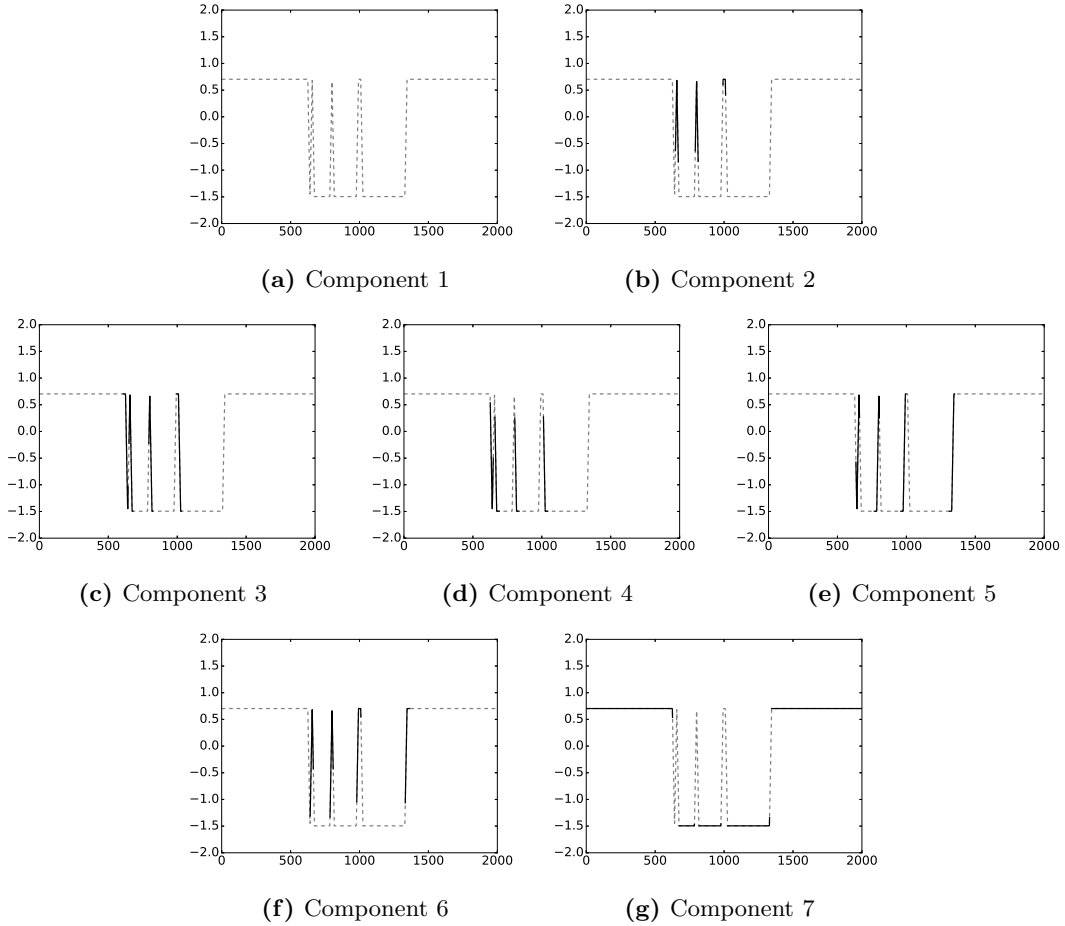
information theory distance). The training parameters are summarized in Table 4.2. The time series representation parameters, window size W, number of frames w as well as ϵ are the same as used when finding the Motifs given in Table 4.1. The component centers found with these settings are presented in Figure 4.5.

The order in which the components are presented in Figure 4.5 does not reflect their relative importance, i.e. the component weight, in contrast to the Motifs where the most representable subsequence is found as the 1-Motif, the second as the 2-Motif etc.

Kernel width	Merge loss threshold	Distance threshold
0.5	0.2	0.6

Table 4.2: Parameter settings used to train a CMM on the data set $\mathbf{D}_1$.

Using the components in the CMM as cluster densities, classification and anomaly detection



(a) Component 1 **(b)** Component 2

(c) Component 3 **(d)** Component 4 **(e)** Component 5

(f) Component 6 **(g)** Component 7

Figure 4.6: Data set $\mathbf{D}_1$ with highlighted areas according to the assigned component labeling of subsequences using a frequency threshold of 10.

(section 3.2.2) is performed on the same data set. The parameters as well as their values used are the same as during training with one exception. The frequency threshold for when a component is considered to be abnormal needs to be specified. Classification is performed using a frequency threshold of 5. In Figure 4.6 the subsequences classified as the respective component is highlighted in black.

No subsequences are labeled as component 1, which is due to that the frequency of that component is smaller than 5 and thus by definition is considered abnormal. The outlier or abnormal behavior found, Figure 4.7, correspond to subsequences similar to component 1.

4.2.3 Discussion regarding Applicability

A direct result from the applicability comparison between the two methods is that it is possible to produce similar results using either of the two methods by tuning the parameter values. When comparing Figure 4.2 with Figure 4.5 the two methods find similar cluster centers with the exception of Motifs finding one less. This is due to the definition that a subsequence can only

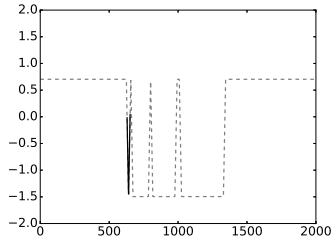


Figure 4.7: Outliers, i.e. subsequences not explained by the CMM.

be considered a Motif if it has at least one non-trivial match. Component 1 corresponds to a subsequence only occurring once, which is therefore not considered as a Motif but is considered as a component of the CMM. If one would pair the Motifs with the component centers they would pair up as: (1-Motif, Component 4), (2-Motif, Component 6), (3-Motif, Component 5), (4-Motif, Component 7), (5-Motif, Component 3) and (6-Motif, Component 2).

The two clustering methods differ greatly regarding their learning procedures. In the process of training a CMM one feature vector is added at a time in order to update the CMM, while the Motif finding procedure requires the whole time series to be available during learning. Thus, CMM possesses the ability of online clustering, classification and anomaly detection on streaming data.

Both of the clustering methods have the ability to learn the number of clusters and the cluster locations. The difference is that when a Motif is found its location is locked never to be altered in contrast to the locations of components in a CMM that are updated as new data becomes available.

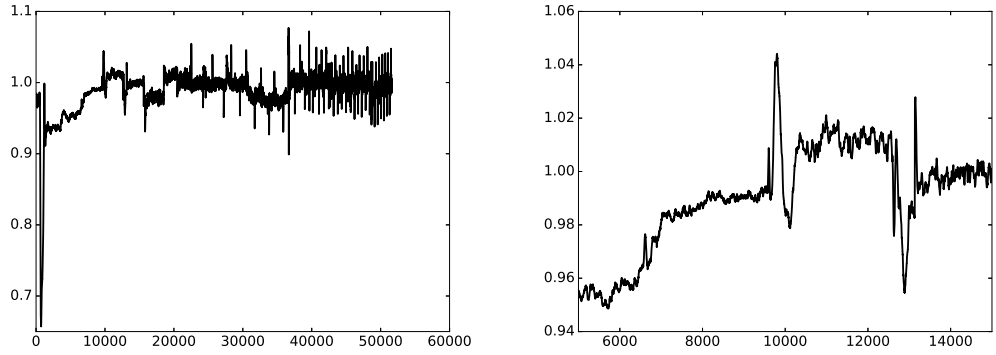
Also, approximations of clusters provided by the two methods differ. By definition a Motif is spherical in shape with a fixed radius defined by the user and provides a poor cluster approximation in many cases. In contrast, the components of a CMM are only initially set to a default shape as an atomic unit (default covariance matrix) and are free to adapt their shape (covariance matrix) during learning which results in better cluster approximations.

4.2.4 Discussion regarding Scalability

The proposed algorithm for finding time series Motifs has already been regarded as unsuitable for massive data sets as it is quadratic in the number of subsequences n i.e. $\mathcal{O}(n^2)$. No such implication has been made regarding the time complexity or scalability of the CMM update procedure. The Motif based algorithm does not scale well, since the pairwise distance between all subsequences has to be calculated in order to rank the subsequences with respect to their number of non-trivial matches.

We were not able to perform an in depth time complexity analysis of the CMM update procedure. The parameter settings for encoding the subsequence (i.e. PAA representation) as well as the merge loss threshold, distance threshold and kernel width all contributes to the number of components found during learning. Therefore, since the number of components vary throughout learning this introduces an additional level of difficulty analyzing the time complexity. Apart from these difficulties, it is clear from Algorithm 3.1 that the CMM update procedure is linear in the number of subsequences n, i.e. $\mathcal{O}(n)$.

Using the PAA representation as a substitute for the SAX representations when finding Motifs will not alter the time complexity. It will still be quadratic $\mathcal{O}(n^2)$ with respect to the number of



(a) Whole time series **(b)** Zoom in

Figure 4.8: The use case output signal, with a normalized y-axis. Here the x-axis is given by the sample number.

subsequences n.

The CMM update procedure is linear with respect to the number of subsequences. This is a great improvement in comparison to the Motif finding algorithm that is at least quadratic in the number of time series subsequences.

Improvements of the CMM update procedures running time could be performed by introducing a fast indexing method, e.g. R-tree indexing, to search for close components.

4.3 Use Case

In this section we will apply CMM on use case data to obtain features from subsequences in the time series.

The company for which this study was conducted has provided a set of questions that the CMM will help answer. For that, the CMM will first be trained in the use case output signal (see Figure 4.8).

The single most important question to answer is the number of unique patterns (clusters) building up a signal and also their shapes. By construction, the CMM is exploring the components/clusters as more data becomes available. The expected result using a CMM is that the number of components converges to the actual number of clusters in the dataset.

The upcoming results were obtained using the parameters: window size $W = 100$, number of frames $w = 5$, $\epsilon = 10^{-4}$ to find the PAA representations; and the parameters of kernel width of 0.3, component distance threshold of 0.5 and a merge loss threshold of 0.2 defining the CMM update procedure.

Training a CMM using the parameter values stated in the previous paragraph resulted in a learning of the number of components as presented in Figure 4.9. From the trace of number of components it is visible that the number of components seems to converge for the given data. Initially the number of clusters quickly increases until about 20000 subsequences when the number of components reaches its maximum value. After the maximum number of components is reached the components of the mixture continue to merge with each other to form larger clusters, while fewer new components are found. This is the explanation to why the number of component in the mixture decreases before converging to a more or less constant level.

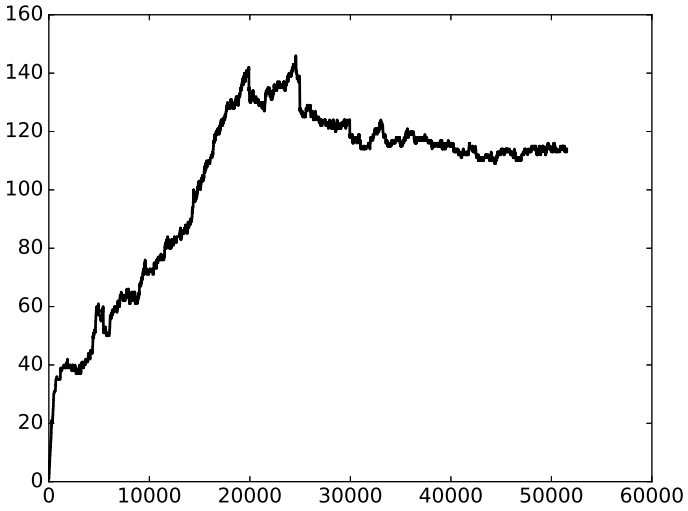


Figure 4.9: The trace of the number of components found in the use case output signal after training on 51533 subsequences. After completed training on one data set a total of 114 components were found.

Thus, the question regarding how many unique patterns builds up a signal is simply answered by the number of components (clusters) available in the CMM. Their shapes are given by the component center. This is of course assuming that all the possible subsequence scenarios have been fed to the CMM for learning. A CMM will deem all unseen subsequence scenarios, i.e. scenarios not explained by the model, as abnormal.

Another interesting question to answer is which subsequences are most- and less frequently occurring. By construction, the CMM stores how many subsequences that every component is build up from. Thus, this question is a very simple query to answer for the CMM. A histogram of the frequency of each component is presented in Figure 4.10.

In Figure 4.11 the most frequent subsequences of the use case output signal are presented and they correspond to the components producing the highest bars of the histogram in Figure 4.10. The less frequent subsequences are presented in Figure 4.12 and they correspond to the components producing the lowest bars of the histogram in Figure 4.10.

The four most frequently occurring subsequences correspond to an increase and decrease in the use case output signal. At a first glance of Figure 4.11, the two most frequently occurring components might seem too similar to be accounted for as different components. The same goes for the third and forth most frequent components. If a more generous merge loss and distance threshold had been chosen these components might have been merged into a composite, but for the current settings they are accounted for as two separate components. Interesting to note is that most of the frequently occurring subsequences also seem to possess a linear trend-like behavior. This could be explained by the fact that they possess a larger amount of trivial matches than patterns of higher variation. The frequency count of the most frequent components in decreasing order are: 2848, 1978, 1869, 1817, 1556.

In contrast to the very frequent subsequences for which a trend-like behavior was observed, the most unfrequent components in Figure 4.12 seem not to follow a linear trend. This statement

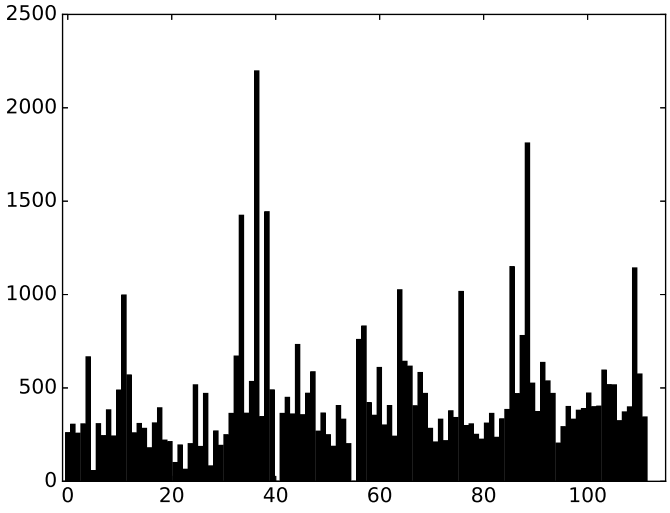
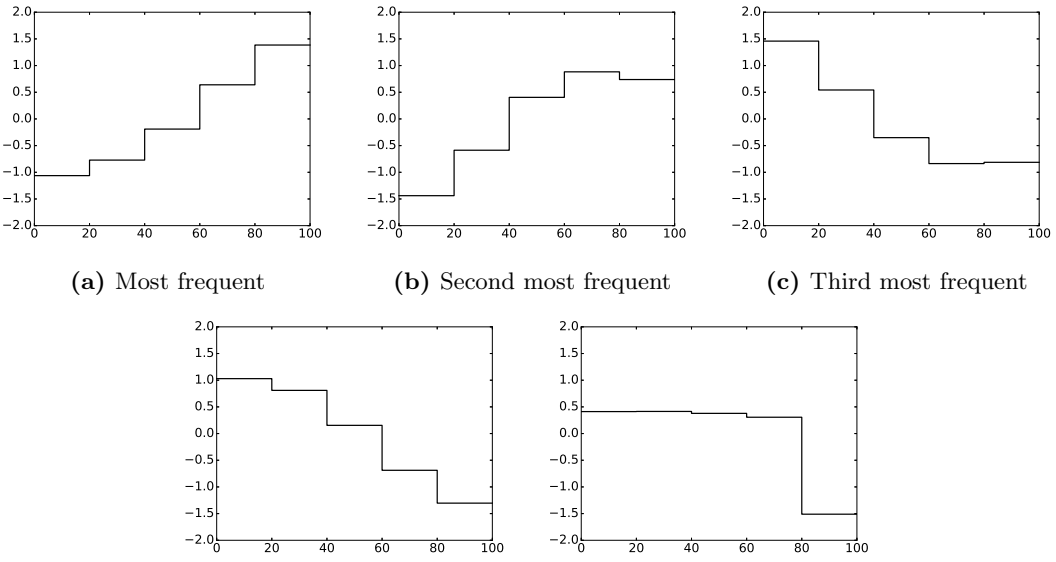


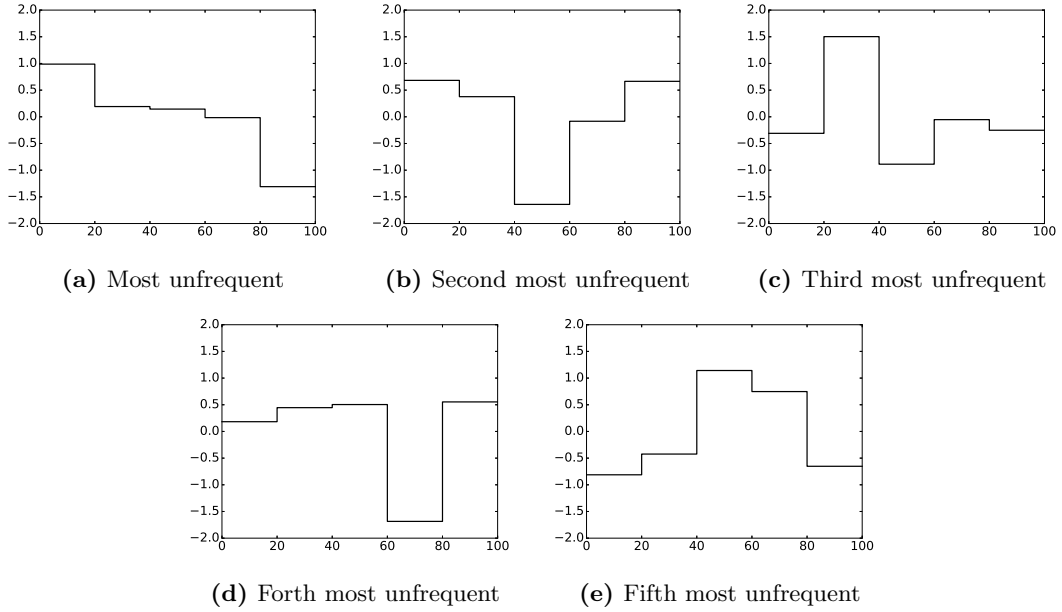
Figure 4.10: A histogram representing the component frequencies in the CMM trained on the use case output signal. Each bar corresponds to one component, and the height of a bar represents the frequency of that component.



(a) Most frequent **(b)** Second most frequent **(c)** Third most frequent

(d) Forth most frequent **(e)** Fifth most frequent

Figure 4.11: The five most frequent components in CMM, with their component centers plotted.



(a) Most unfrequent **(b)** Second most unfrequent **(c)** Third most unfrequent

(d) Forth most unfrequent **(e)** Fifth most unfrequent

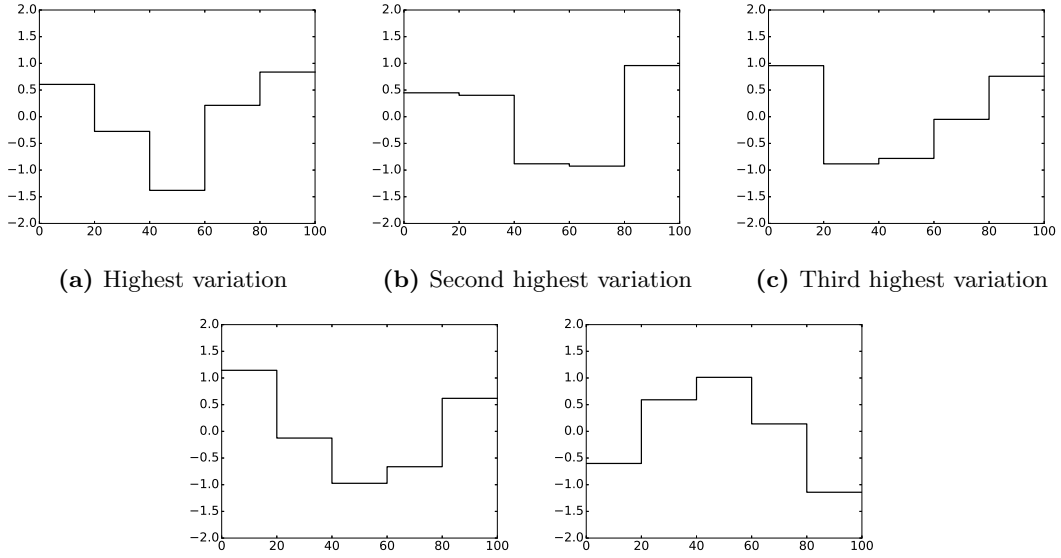
Figure 4.12: The five most unfrequent components in CMM, with their component centers plotted.

holds for four of the most uncommon components, but not for the most uncommon component which seems to have a more trend-like behavior than the others. The frequency counts of the most unfrequent components in increasing order are: 1, 5, 6, 6, 7. This means that the most unfrequent component was added as the CMM trained on the last provided feature vector, otherwise there would have been an additional number of trivial-matches merged into the most unfrequent component. All of these five low frequency components should by definition be considered abnormal since they only contain one subsequence and its trivial matches. As previously mentioned, the frequency threshold value should be set in the range of the largest amount of trivial matches.

The company is also interested in finding subsequences of large variation where the definition of variation is allowed various interpretations. In this study the subsequences are normalized before the PAA representations are obtained. Therefore any information regarding position in space and scaling is removed and no inference regarding absolute information of subsequences will be available. A suggestion of how to include absolute information for each component is suggested in the Future Work chapter. Finding components with the highest variance of cluster centers will therefore be non-informative (all component centers have standard deviations pending around 1). Instead a measure of variation of a component center (PAA) is measured based on its deviations from a linear regression fit. In Figure 4.13 the components of the highest variation, and considered normal, are presented.

Analyzing the use case output signal from the same kind of system, but three different objects, will provide interesting material to discuss. In Figure 4.14 the three objects were subjects to similar testing scenarios. During these tests the speed of the process was subject to change and every 5000 data point the process speed was increased.

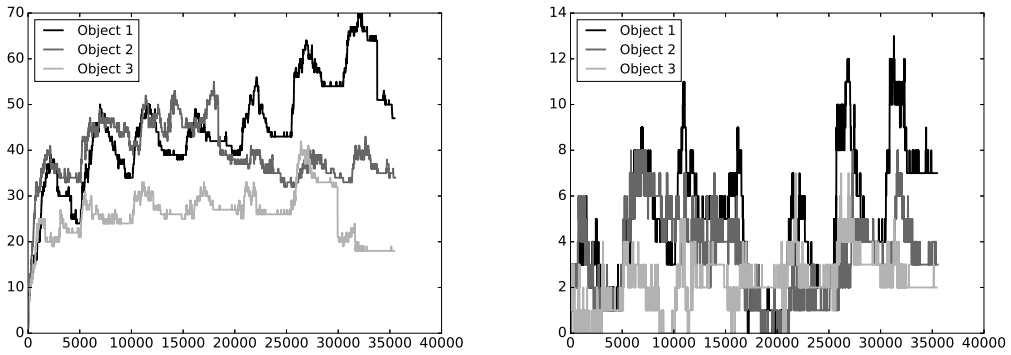
It is clear from the graphs in Figure 4.14 that every 5000 data point the number of components increases, which means that the CMM observes new subsequences not seen before. This happens every time the process speed increases for all three objects. After a number of subsequences



(a) Highest variation **(b)** Second highest variation **(c)** Third highest variation

(d) Forth highest variation **(e)** Fifth highest variation

Figure 4.13: The five components of highest variation in CMM classified as normal, with their component centers plotted.



(a) Number of components **(b)** Number of components classified as outliers

Figure 4.14: Evolution of CMM over time. Using $W = 100$, $w = 5$, $\epsilon = 1e - 4$ and merge loss threshold 0.2, distance threshold 0.5 kernel width 0.3. Every 5000 data points the test condition of the systems changes, in this case the process speed is increased.

have been observed by the CMM at the new process speed the number of components starts to decrease. This is due to the fact that components in the CMM merge together to form bigger components approximating larger clusters. It is not possible to assess if the number of components converge to a constant level from this test, since the number of data points in the time series at the different levels of process speeds are too small.

The behavior of increasing number of components when the process speed changes might be countered. Instead of using time to measure the process, another process speed adaptive measure should preferably be used.

5

Conclusions

Two univariate data mining tools, one Motif based and the other Mixture Model based, have been implemented and compared in this thesis. They both feature clustering, classification and anomaly detection of time series subsequences. Even though one of them shows a higher potential for further development, both techniques solve the crucial problem to cluster subsequences. This relates to solving the task of learning one model for each of the input time series in Figure 1.2 found in the Introduction chapter. The company for which this study was conducted will be able to use the data mining tool, Compressed Mixture Model, to achieve compression of time series data and to find all unique patterns.

Using the Compressed Mixture Model as a model of time series subsequences is beneficial in comparison to using Motif based model in many aspects, e.g.

- The time complexity for CMM's learning procedure is linear in the number of subsequences, while the time complexity for finding all Motifs is quadratic. The CMM learning procedures time complexity depends on the number of components. Thus it is beneficial having a proper indexing method to find the closest pair of components to minimize the actual running speed.
- CMM discovers and adapts the number of clusters as well as their location and shape in a feature space. The Motif based technique discovers the number of clusters and their location for a chosen radius of the multidimensional cluster, but not the shape which is spherical by definition (see Figure 3.1).
- The proposed technique, Compressed Mixture Model, is suitable for real time applications because of the possibility to incrementally update the model. This is not the case for the Motif based technique for which no good update procedure has been found.
- Classification and anomaly detection is simple to perform when a CMM has been trained, and it takes into account for the locations, shapes and frequencies of the components.

There is one aspect where the Motif based technique triumphs the CMM, it takes into account for trivial matches of subsequences. This is accounted for when finding anomalies in the CMM by setting a higher frequency threshold value. However, the number of trivial matches varies

between different patterns and therefore a more adequate method to deal with trivial matches would increase the value of CMM.

6

Future Work

This chapter contains possible improvements of the univariate time series data mining tool, CMM, proposed in this study. One improvement would be to extend the univariate CMM to also cover multivariate data mining. Another would be to include additional information regarding absolute values in the CMM. At last, including methods for automatic parameter selection would improve the useability of the CMM.

6.1 Multivariate Data Mining

The study conducted has solely focused on building a model for univariate time series data mining. Many of the interesting time series analysis applications feature a system of multiple input-/output signals, where it would be interesting to perform simultaneous data mining on all signals to capture cause-effect relations. The cause would be an event in one or multiple input signals, while the effect is an observed reaction to the event in one or multiple output signals. Capturing such relations would require that possible time delay between input- and output signals are known and accounted for. The company for which this study was conducted is interested in analyzing input-output relations in systems where the time delay is inconsiderable 8-16 ms, while patterns in the signals is in the order of 1 s. Therefore the cause-effect relations will be covered by the window size since it will be sufficiently large. Moreover, in order for such a multivariate cause-effect comparison to be meaningful, the window size W and the number of frames w should be chosen to be the same for all the signals. This would create one feature vector per dimension (signal) and it would be possible to proceed with the multivariate analysis in two separate directions.

6.1.1 Multivariate CMM

The first option is to create a multivariate kernel for all the signals and build a multivariate CMM.

Multivariate Gaussian Kernel

A multivariate kernel can be created by expanding the feature vector used in the univariate case to a vector of the form $(f_{11},\ldots,f_{1w},f_{21},\ldots,f_{2w},\ldots,f_{n1},\ldots,f_{nw})$ where f_{ij} is the feature value for signal i and feature j. The current implementation of the CMM could then be used to find multivariate clusters, their locations as well as shapes. No extra parameters will have to be added when comparing to the univariate case and the kernel width can be the same for all signals due to the normalization.

Using a multivariate Gaussian kernel for multivariate analysis (multivariate signals), the number of parameters to estimate would increase quadratically with respect to the number of signal dimensions. One problem introducing multiple signal dimensions is that the amount of data required to assess good cluster approximations quickly outgrows the amount of data available. Therefore, in order to obtain good cluster approximation using the data available at hand, it is important to make the aggregate method more robust especially when updating the covariance matrix.

An interesting suggestion of estimating the covariance matrix using the concept of *shrinkage* is defined by Ledoit and Wolf [11]. They propose a linear combination of an unstructured sample covariance matrix S (full covariance matrix) and a highly structured estimator of the sample covariance matrix F as the estimator for the true covariance matrix, $\delta F + (1-\delta)S$, where $\delta \in [0,1]$. The shrinkage covariance matrix estimator seems like a promising alternative to tackle the curse of dimensionality and to make the covariance matrix estimate more robust.

Markov Chain Model

As a direct result after expanding the CMM to handle multivariate signals, the multivariate feature vector could be built up by two consecutive feature vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ (consecutive in time). The two feature vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ both share the same data points but one. This means that it is possible to create a Markov chain CMM where the conditional distribution of the possible values of $\boldsymbol{x}_2$ is conditioned on the current value of $\boldsymbol{x}_1$.

6.1.2 Multiple Univariate CMMs

The second option is to build univariate CMMs for all the signals and produce multidimensional labels.

$$\begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix} = \begin{bmatrix} l_{11} & l_{12} & l_{13} & \ldots & l_{1N} \\ l_{21} & l_{22} & l_{23} & \ldots & l_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{M1} & l_{M2} & l_{M3} & \ldots & l_{MN} \end{bmatrix}$$

Here, element l_{ij} is the label of subsequence j in signal i. It is now possible to estimate how many multidimensional clusters there are by counting the number of unique multidimensional labels and how frequently occurring they are. It would also be possible to answer the question, conditioned on $l_{1j} = A$ how many unique multidimensional clusters exist and what are their frequencies. Moreover, it would be possible to easily remove a signal from the analysis by not considering its univariate label vector L_i.

6.2 Absolute Information

The CMM proposed in this study builds a model of the shape and trends of time series subsequences. No information regarding the absolute values of subsequences mean value and standard deviation is stored in the current implementation of CMM. One possible solution could be to include two additional 1-dimensional CMMs for each component and signal dimension. In one of them the mean value for a subsequence added to the component is inserted, while in the other the standard deviation is inserted. This would mean that every component can keep a probability model of absolute information and would introduce the possibility to draw conclusions if a subsequence is normal or not including information regarding the absolute values of mean and standard deviation.

6.3 Parameter Selection

The time series data mining tool, CMM, is sadly not parameter free. A number of parameter values needs tuning, and this task is non-trivial. As a recommendation for future improvements using CMM to model time series subsequences, this section highlight promising techniques to automate the parameter selection process.

6.3.1 Minimal Descriptive Length

Hu et al. [5] propose the use of Minimal Descriptive Length (MDL) to discover the natural intrinsic representation model, dimensionality and alphabet cardinality of a time series. They provide a general MDL algorithm and specific examples of how to apply it for model- and parameter selection, using a set of possible models and parameter values. The best model and parameter values are the ones that minimize the memory usage for decoding the time series, which is a trade-off between the model fit and the number of parameters.

To connect this to our work, it is possible to apply this method in order to determine the best set of parameters for the PAA representation given a time series. The parameters to consider for such a parameter selection technique are the window size and the number of frames defining the PAA representation.

6.3.2 Goodness-of-fit Measures

The problem of choosing the number of components of a Gaussian Mixture Model (GMM) using model selection techniques have previously been studied by e.g. McLachlan and Rathnayake [15] and Huang et al. [6]. Such model selection techniques assess the GMM's goodness-of-fit and make inference on the optimal number of components. In this study, the number of components is depending on the parameter choices of the kernel width, merge loss threshold and component distance threshold.

The idea is to make the parameter selection into a search problem for optimal fit, using the goodness-of-fit of the CMM (CMM is a GMM) for a set of parameters. It is necessary to define optimality of a CMM. Various possibilities exists e.g. optimal with respect to the data, the components and/or the merge loss.

Bibliography

[1] K.P. Burnham and D.R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer New York, 2003. ISBN 9780387953649. URL https://books.google.se/books?id=BQYR6js0CC8C.

[2] M. Butler and D. Kazakov. Sax discretization does not guarantee equiprobable symbols. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1162–1166, April 2015. ISSN 1041-4347. doi: 10.1109/TKDE.2014.2382882.

[3] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1): 12:1–12:34, December 2012. ISSN 0360-0300. doi: 10.1145/2379776.2379788. URL http://doi.acm.org/10.1145/2379776.2379788.

[4] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2):419–429, May 1994. ISSN 0163-5808. doi: 10.1145/191843.191925. URL http://doi.acm.org/10.1145/191843.191925.

[5] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh. Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *2011 IEEE 11th International Conference on Data Mining*, pages 1086–1091, Dec 2011. doi: 10.1109/ICDM.2011.54.

[6] T. Huang, H. Peng, and K. Zhang. Model Selection for Gaussian Mixture Models. *ArXiv e-prints*, January 2013.

[7] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371. ISSN 1573-756X. doi: 10.1023/A:1024988512476. URL http://dx.doi.org/10.1023/A:1024988512476.

[8] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowl. Inf. Syst.*, 8(2):154–177, August 2005. ISSN 0219-1377. doi: 10.1007/s10115-004-0172-7. URL http://dx.doi.org/10.1007/s10115-004-0172-7.

[9] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286. ISSN 0219-1377. doi: 10.1007/PL00011669. URL http://dx.doi.org/10.1007/PL00011669.

[10] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 44(10–11):2630 – 2642, 2011. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2011.03.019. URL `http://www.sciencedirect.com/science/article/pii/S0031320311001233`. Semi-Supervised Learning for Visual Content Analysis and Understanding.

[11] Olivier Ledoit and Michael Wolf. Honey, I shrunk the sample covariance matrix. Economics Working Papers 691, Department of Economics and Business, Universitat Pompeu Fabra, June 2003. URL `https://ideas.repec.org/p/upf/upfgen/691.html`.

[12] Qi Li and Jeff Racine. Nonparametric estimation of distributions with categorical and continuous data. *Journal of Multivariate Analysis*, 86(2):266 – 292, 2003. ISSN 0047-259X. doi: http://dx.doi.org/10.1016/S0047-259X(02)00025-8. URL `http://www.sciencedirect.com/science/article/pii/S0047259X02000258`.

[13] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM. doi: 10.1145/882082.882086. URL `http://doi.acm.org/10.1145/882082.882086`.

[14] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007. ISSN 1573-756X. doi: 10.1007/s10618-007-0064-z. URL `http://dx.doi.org/10.1007/s10618-007-0064-z`.

[15] Geoffrey J. McLachlan and Suren Rathnayake. On the number of components in a gaussian mixture model. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4 (5):341–355, 2014. ISSN 1942-4795. doi: 10.1002/widm.1135. URL `http://dx.doi.org/10.1002/widm.1135`.

[16] Emanuel Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09 1962. doi: 10.1214/aoms/1177704472. URL `http://dx.doi.org/10.1214/aoms/1177704472`.

[17] Cosma Rohilla Shalizi. *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, 2013.

A

Appendix

It is beneficial to understand the strengths and weaknesses of the time series representation techniques used, i.e. PAA and SAX. The following transformations presented in (A.1) are the interesting invariance scenarios.

$$\begin{aligned} y &= f(x) + \Delta\epsilon \\ y &= f(x) + \Delta y + \epsilon \\ y &= f(x) \cdot \Delta y + \epsilon \\ y &= f(x + \Delta x) + \epsilon \end{aligned} \tag{A.1}$$

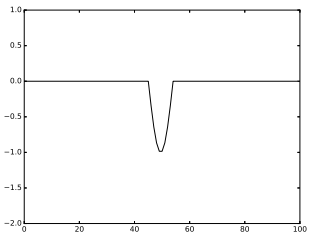
In (A.1), the first equation denotes a noise scaling transformation, the second offset transformation, the third signal scaling transformation and the forth time translation transformation. In this study the subsequences are normalized before represented by either PAA or SAX which removes the two invariance scenarios offset transformation and scaling transformation from the table. Both of the methods lack the time translation invariance property. It is of interest to assess how well the representations performs when noise is introduced.

How well the two representations perform is evaluated based on the distance between a reference signal s_{ref} and the same signal distorted with varying levels of noise. The reference signal together with examples of distorted reference signals is presented in in Figure A.1.

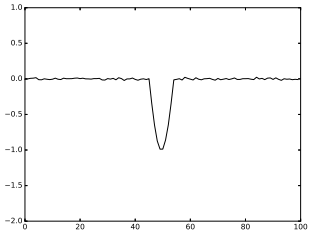
Noise scale invariance

The signal s_{ref} was defined as the underlying pattern with 100 data points and a half sinusoidal wave introduced of length 10 samples. Different levels of gaussian noise was added to the signal with mean 0 and varying standard deviation. The Signal-to-Noise Ratio (SNR) is defined as the ratio between the signal amplitude and the standard deviation of the noise.

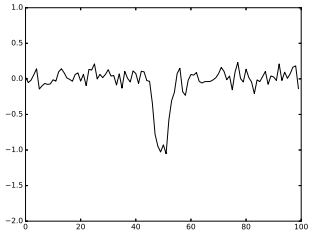
The PAA representation was obtained using varying number of frames (or word lengths) and an $\epsilon = 10^{-6}$. In Figure A.2 the distance between the reference signal and distorted signals of varying magnitude of noise is graphically presented. The distances for each set of parameters is calculated as the mean of 100 uniquely distorted signals.



(a) Reference signal s_{ref}



(b) Reference signal distorted $\sigma = 0.01$ (SNR = 100)

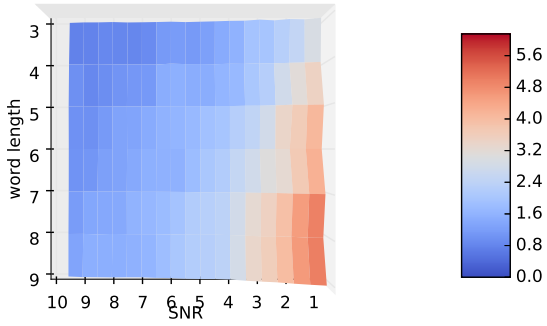


(c) Reference signal distorted $\sigma = 0.1$ (SNR = 10)

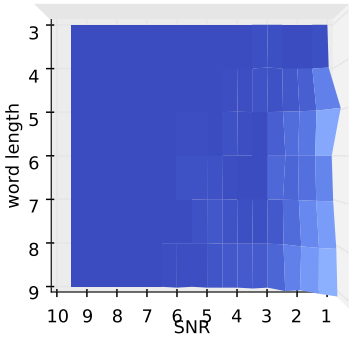
Figure A.1: The reference signal s_{ref} and two examples of distorted reference signal with various levels of Gaussian noise.

The plots in Figure A.2 points out that the finer the resolution in space, i.e. an increasing alphabet size α, used to obtain the SAX representation it more closely resembles the PAA distance. In the limit $\alpha \to \infty$ the PAA and SAX distances becomes equal.

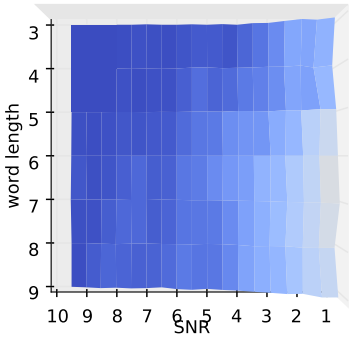
Regarding the noise invariance property. The SAX distance for smaller α's is close to zero which means that it is noise invariant. But it comes with a drawback of loosing too much information in the discretization procedure. It is easier to interpret the resulting PAA distances and for all the different choices of number of frames (or word length) as the SNR$\to \infty$ the distance tends to zero. Common SNRs for the signals of the application is usually larger than 20 and the word lengths are in the range of 5-10. For this parameter settings, the PAA representation can be considered to be noise invariant.



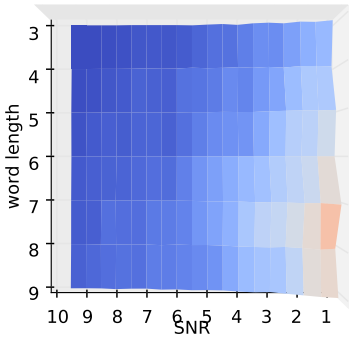
(a) PAA



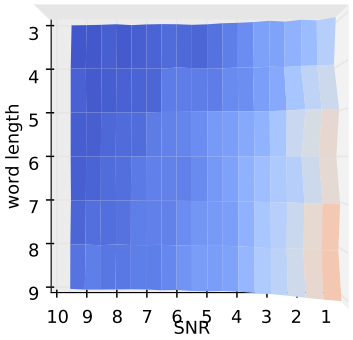
(b) SAX with $\alpha = 10$



(c) SAX with $\alpha = 15$



(d) SAX with $\alpha = 20$



(e) SAX with $\alpha = 25$

Figure A.2: The z-axis is defined as the distance (PAA distance and SAX distance respectively) between the signal s_{ref} and distorted signals of varying magnitude. The SAX representation was obtained using an alphabet size of 10 (upper left) 15 (upper right) 20 (lower left) 25 (lower right). They all share the same color bar.