Pattern Discovery in Time Series

Inaugural-Dissertation

zur

Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Gerhard Klassen

aus Kirowskij, Kasachstan

Düsseldorf, Mai 2021

aus dem Institut für Informatik der
Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: Prof. Dr. Stefan Conrad

Koreferent: Prof. Dr. Stefan Dietze

Tag der mündlichen Prüfung: 30.09.2021

Ich, Gerhard Klassen, versichere an Eides statt, dass die vorliegende Dissertation von mir selbstständig und ohne unzulässige fremde Hilfe unter Beachtung der *Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf* erstellt worden ist.

Die hier vorgelegte Dissertation habe ich eigenständig und ohne unerlaubte Hilfe angefertigt. Die Dissertation wurde in der vorgelegten oder in ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, Deutschland
11. Mai 2021

Gerhard Klassen

Dedicated to my uncle Gerhard Klassen

Acknowledgements

This dissertation is not only the result of my research, it also reflects the support I received from the people around me.

First of all, I would like to thank my supervisor *Prof. Dr. Stefan Conrad*. He gave me the freedom and support for my research that was necessary to complete this dissertation. His views on different topics have always been enriching and have had a lasting impact on my way of doing things. Additionally, I also thank *Prof. Dr. Stefan Dietze* who showed great interest in my research and agreed to review my work.

I also want to express my gratitude to *Martha Krakowski*, as it was the collaboration with her that made this dissertation possible. She always encouraged me and often gave me the motivation for my research. I cannot imagine a better colleague, so it makes me even sadder that our paths will soon part.

I thank *Jun-Prof. Dr. Marcus Bravidor*, who taught Martha and me the basics of business administration, always showed great interest in our work, and spent an incredible amount of time with us.

Dr. Michael Singhof not only supervised me during my Bachelor's and Master's studies, but has also become a good colleague and friend. His mathematical understanding seems unattainable to me to this day. The interrogation lamp, which not only brought us hours of fun but also revealed a few secrets, remains unforgotten. Thank you very much for this time.

It is unbelievable what we accomplished together with Martha Krakwoski and *Dr. Ludmila Himmelspach* in just one weekend. It will always remain in my memory. Thank you very much for that.

Julia Romberg entertained an entire bus with me in Stuttgart with a scientific discussion and although we often disagreed, I loved sharing a pizza funghi with her after the pub.

I would also like to thank *Filip Krakowski*, with whom I could discuss many things like with no one else.

My work would not be the same without my colleagues *Daniel Braun*, *Kirill Bogomasov*, and *Dr. Alexander Askinadze*. Thank you for the collaboration and fruitful scientific discussions.

Furthermore, I want to thank *Guido Königstein* and *Sabine Freese* for their excellent technical and administrative support.

I also want to express my deepest and warmest thanks to my friends and my family especially to my parents for their love, encouragement, and support. I am also grateful to my siblings *Christina* and *Waldemar* who often put their needs behind mine and thus gave me unforgettable moments.

Finally, I deeply want to thank Lilia, who always supported and believed in me. Without you it would not have been possible.

Abstract

The identification of groups in data sets, also called cluster analysis or clustering, is an important part of many analyses. Several algorithms from different research areas have already been developed for this purpose. These methods differ not only in their algorithmic procedure but also in the use of different comparison functions. In addition, many methods require the selection of one or more parameters, so that the results depend not only on the chosen method but also on the parameters selected. The question of the validity of the clusters found can only be answered, if at all, by experts in the relevant data domain. This problem affects all forms of data and can severely limit the usefulness of such an analysis.

Some types of data contain additional dependencies that can be used advantageously in such a cluster analysis. Time series, i.e. ordered sequences of observations, represent such a class of data. In many respects they determine our everyday life, whether on stock markets, in medicine or during the Corona pandemic in form of the course of infections. If the temporal component is properly taken into account, a cluster analysis can provide previously unknown information. However, the validity of the found clusters must first be ensured in order to prevent misinterpretations.

The explained problem is the motivation for CLOSE, a new method presented here, which is able to evaluate a clustering of time series. The developed evaluation is based on a novel stability measure for time series and clusters and provides a score, which makes different clusterings comparable. The circumstance that it is not only *crisp* clustering which is affected by the described problem, but also *fuzzy* clustering, led us to another method, called FCSETS, which is specialised on *fuzzy* clusterings.

We evaluate these methods using several data sets and clustering algorithms. We also present three applications and several variants which target the detection of outliers in time series. These applications are based on the findings in FCSETS and CLOSE. Additionally we present a clustering algorithm, which is based on a derived concept of CLOSE.

In an excursion chapter, we show the results of other machine learning techniques so that a comparison can be made with our applications. Our results are promising and enable users to choose a suitable clustering algorithm and the corresponding parameters without prior knowledge.

Zusammenfassung

Die Identifikation von Gruppen in Datensätzen, auch Clusteranalyse oder Clustering genannt, ist ein wichtiger Bestandteil vieler Analysen. Hierfür wurden bereits mehrere Algorithmen aus verschiedenen Forschungsbereichen entwickelt. Diese Methoden unterscheiden sich nicht nur in ihrem algorithmischen Vorgehen, sondern auch in der Verwendung unterschiedlicher Vergleichsfunktionen. Darüber hinaus erfordern viele Methoden die Wahl eines oder mehrerer Parameter, so dass die Ergebnisse nicht nur von der gewählten Methode, sondern auch von den gewählten Parametern abhängen. Die Frage nach der Gültigkeit der gefundenen Cluster kann, wenn überhaupt, nur von Experten in der jeweiligen Datendomäne beantwortet werden. Dieses Problem betrifft alle Formen von Daten und kann die Nützlichkeit einer solchen Analyse stark einschränken.

Einige Arten von Daten enthalten zusätzliche Abhängigkeiten, die in einer solchen Clusteranalyse vorteilhaft genutzt werden können. Zeitreihen, d.h. geordnete Folgen von Beobachtungen, stellen eine solche Klasse von Daten dar. Sie bestimmen in vielerlei Hinsicht unseren Alltag, ob an der Börse, in der Medizin oder während der Corona-Pandemie in Form von Infektionsverläufen. Wenn die zeitliche Komponente richtig berücksichtigt wird, kann eine Clusteranalyse viele bisher unbekannte Informationen liefern. Allerdings muss zunächst die Gültigkeit der gefundenen Cluster sichergestellt werden, um Fehlinterpretationen zu vermeiden.

Die erläuterte Problematik ist die Motivation für CLOSE, eine hier vorgestellte neue Methode, die in der Lage ist, ein Clustering von Zeitreihen zu bewerten. Die entwickelte Auswertung basiert auf einem neuartigen Stabilitätsmaß für Zeitreihen und Cluster und liefert einen Score, der verschiedene Clusterings vergleichbar macht. Der Umstand, dass nicht nur *hartes* Clustering von dem beschriebenen Problem betroffen ist, sondern auch *fuzzy* Clustering, führte uns zu FCSETS, einer weiteren Methode, die auf *fuzzy* Clustings spezialisiert ist.

Wir evaluieren diese Methoden anhand verschiedener Datensätze und Clustering-Algorithmen. Wir stellen außerdem drei Anwendungen und mehrere Varianten vor, die auf die Erkennung von Ausreißern in Zeitreihen abzielen. Diese Anwendungen basieren auf den Erkenntnissen in FCSETS und CLOSE. Zusätzlich stellen wir einen Clustering-Algorithmus vor, der auf einem abgeleiteten Konzept von CLOSE basiert.

In einem Exkurs zeigen wir die Ergebnisse anderer maschineller Lernverfahren auf, so dass ein Vergleich mit unseren Anwendungen möglich ist. Unsere Ergebnisse sind vielversprechend und ermöglichen es Anwendern ohne Vorkenntnisse einen geeigneten Clustering-Algorithmus und die entsprechenden Parameter auszuwählen.

Contents

1

Introduction

Data has never been worth as much as it is today. However, it is not the data itself that makes up its value, it is the potential for value creation that does. This is comparable to any other raw material, which only represents an actual value through its processing. Unlike other raw materials, however, data is not consumed in the process; once collected, it can serve many different purposes. How well an intended purpose is fulfilled depends, as with other raw materials, on their processing and their quality. In the context of data, the utilisation and thus the value creation depends on the analysis. Due to the many different types of data and different objectives, countless methods and approaches for their analysis have been developed over the past decades. Research in computer science and other areas play a key role here. Many people are fascinated by the results of this research and often experience it in everyday life as a form of artificial intelligence. Whether it is the autopilot of a Tesla[1] or the answering of questions by a voice assistant like Amazon's Alexa[2], in the background the result experienced is based on the analysis of the input.

1.1 Machine Learning

In computer science, data analyses are usually developed and implemented in research areas such as *machine learning*, *data science* or *data mining*. In the last decades, these research areas have introduced new approaches that are able to derive dynamic and stunningly precise rules from data. Although these approaches often have a statistical basis, they are referred to as *learning*. On the question of what distinguishes statistics from machine learning, Witten et al. write "Cynics, looking wryly at the explosion of commercial interest (and hype) in this area, equate data mining plus marketing" [53] and further "Some [data analysis techniques] derive from the skills taught in standard statistics courses, and others are more closely associated with the kind of machine

[1] https://www.tesla.com/autopilotAI
[2] https://developer.amazon.com/en-US/alexa

learning that has arisen out of computer science" [53]. Another interesting insight into this discussion is provided by Breiman in [10], where he divides the community into an algorithmic and a modeling culture. However, between those two cultures, there is a large consensus on the categorisation of machine learning methods. They are often divided into two, three or more categories. In the following, we will discuss a distinction into three categories and outline their differences. These categories are named after their learning approach and are called *reinforcement learning*, *supervised learning* and *unsupervised learning* [45]. It should not go unmentioned that there are also mixed methods that fall into two or more of these categories, those are equally important as the above mentioned categories, but are beyond the scope of this work.

The first category of learning methods is probably the most intuitive form of machine learning. It is based on the idea that humans learn by interacting with their environment [51]. Comparable to autodidactic learning, similar to a try-and-error process with reward and punishment functions. Systems based on *reinforcement learning* are promising and are also used in robotics. Here it fits perfectly, as it is important, that a robot is able to interact with its environment. However promising methods of this kind may be, in many cases the definition of gratification and punishment functions can be extremely complex.
The second of the above mentioned categories of learning approaches, *supervised learning*, is often used in classification tasks. These are tasks in which data objects are assigned to predetermined classes [7]. Probably one of the best-known classification task of our time is the filtering of spam in e-mails, in which a distinction is made between the classes *ham* and *spam* [44]. Here, the system decides independently whether an e-mail falls into the one or the other class. In the background, the system uses a mathematical model that makes this distinction possible. In *supervised learning*, the model is given the classes into which individual objects are to be classified [7]. With the known classes of a training set, the mathematical model can be adapted so that new, unknown data can also be correctly classified. Therefore, a distinction is made between a training phase, in which the system is trained with labelled data and an application phase in which the classes of the newly arriving objects are unknown. In order to evaluate the system, the application phase is simulated so that the actual known class memberships of objects remain unknown to the system [7]. A comparison of the classes recognised by the system and the actual classes then allows a statement about the quality of the system. This phase is generally called the *test phase* and is usually part in the development of the system. Methods of this type have enormous advantages in situations where the classes into which data objects are to be divided are already known in advance. An important aspect though is the balance of the data in the training phase. Imbalanced data sets can lead to better results in the *test phase* as the underlying model could adapt to this imbalance by returning the overrepresented class for every object. A simple example for such a situation is the prediction of rain in weather forecasting. Suppose it had been statistically shown that every week has exactly one rainy day. Predicting *no* rain for every single day would result in a correct statement in six of seven cases, which corresponds to a correct statement in about 86% of the cases.

The third category, *unsupervised learning*, does not require labelled data [44]. For this reason, a training phase like in *supervised learning* is omitted. The reference system in *unsupervised learning* is the data itself, as the goal is to find meaningful

patterns [44]. For this reason, *unsupervised learning* is also referred to as *descriptive learning* [45]. Methods in this category are suitable for a wide range of applications. Often, this form of learning is used for dimension reduction or for choosing meaningful features [11]. Furthermore, it is applied to the detection of groups in data, also called clustering. We go into this application in particular detail as it is a core element of this dissertation. The detection of outliers is another important application within the field of *unsupervised learning*. In this thesis it represents a higher level application and is discussed in Chapter 3.
Although it is often mentioned in the literature [14, 30], we think it is wrong to weigh the different types of machine learning with respect to a specific task. In our view, the different types have a complementary existence with different objectives and requirements. The choice of a method and thus the choice of a certain type therefore depends on the data at hand and the objective of the analysis.

1.2 Cluster Analysis

"Cluster analysis is the art of finding groups in data" [25], these groups are also referred to as clusters. Although the concept might be intuitive, a definition of a cluster is difficult. In his position paper, Vladimir Estivill-Castro argues, that there are so many different cluster algorithms because the term *cluster* cannot be defined precisely [21]. However, there is an agreement on the definition of clusters, in the sense that objects from one cluster are similar to each other and objects from different clusters are dissimilar to each other [22, 34]. The similarity or dissimilarity of objects can be interpreted in different ways. In many methods of cluster analysis, distance [6, 42] or density functions [5, 20] are used. From this, one can conclude that different algorithms lead to different results. In fact, it is even more complex: First of all, most algorithms require a selection of one or more parameters, some offer the choice of a similarity function and others also depend on a certain initialisation. All these factors may have a high impact on the results obtained. At the same time, the selection of these factors is crucial for a valid outcome of the analysis. Nevertheless, cluster analyses make important contributions in a wide range of domains. Among others, these include biology, psychology, medicine, marketing, computer vision and remote sensing [50].

However, all fields of application encounter the same problem, namely the validation of the clusters obtained. This is a major problem, which is particularly affecting naive users, since those often tend to misinterpret the results [50], but also domain experts may be misled in some situations. Jain and Dubes have recognized this risk and state:

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Jain and Dubes 1988, Algorithms for Clustering Data

The problem described affects all types of data and is probably more significant today than it was in 1988. Nonetheless, this circumstance may also represent a new opportunity: Solving the problem for one type of data could lead to solutions for other types of data and thus contribute to the overall solution. This dissertation is our

contribution to solving the problem formulated by Jain and Dubes and deals with the evaluation of clusterings of time-dependent data, also called time series.

1.3 Time Series

"There are no secrets that time does not reveal."
Jean Baptiste Racine

Time series occur in various areas and determine our everyday life. Without recognizing them as such, we experience them through the media - whether it is the development of the stock market, climate change or political polls. A single point in time has little significance. Only by considering several points in time a development can be derived, which usually reveals more detailed information about a certain situation. William W. S. Wei defines a time series as "an ordered sequence of observations" [52].

Probably the best known time series that has changed our lives in 2020 is that of the Corona infections. There is great interest in this time series, because its development determines measures that restrict our everyday lives. Another time series which dominated the media was the price development of the Bitcoin[3] [46]. While the current price of 50.000 US Dollars is not very meaningful, it becomes interesting, when it is mentioned that it cost around 4.000 US Dollars in the middle of 2020. In other domains, even this statement may not be sufficient. For this reason, news about the course of a certain stock often also mention the development of the according index (for example NASDAQ Composite). Here, the relationship of a stock to a certain group is important; if the share price develops in line with the reference group, it is probably not worth to mention.

In statistics, the idea to compare time series with each other exists for a long time. Therefore common correlation coefficients such as Pearson, Spearman, the cross-correlation [16, 29, 31], or the Granger causality [28] are applied to time series. Furthermore, Autoregressive-Moving Average (ARIMA) models are also used to compare time series to each other. [27]. Although these methods can be used to compare time series, they cannot be used to identify clusters of time series. Yet, such groups can serve as a reference and it is necessary to identify them if they are unknown. For this reason, time series are also clustered [38, 39, 54] which does not always lead to meaningful results [33]. In [33] Keogh et al. deal in particular with the clustering of subsequences. Therefore, a time series is divided into subsequences, which are then clustered. The objective of this procedure is to recognise patterns within a time series, which for example can represent reoccurring events. However, the procedure can trivially also be applied to several time series in order to find similar subsequences across time series. The main problem faced in these kind of approaches is the division of the time series into several subsequences. It can be assumed that this division depends on the time series in question. With several time series, the intervals per time series can differ, so that one would have to compare sequences of different lengths with each other. In the next chapter, we return to this circumstance and describe the problem addressed in this dissertation.

[3] https://www.bbc.com/news/business-56150425

1.4 Problem Description

The identification of groups in time series can be useful in many applications. In our works [R2, R3, R4, R6, R7] we address some of these. These include, for example, clustering countries over-time by economic features or grouping companies in terms of the history of their financial data. In latter, one could also use existing classification systems such as the Standard Industrial Classification (SIC)[4], but such systems are usually static and only map the industry of a company. If one wants to examine companies in terms of their development over time in relation to their reference group, one cannot avoid identifying its reference group first. The composition and number of groups can change at any time, which is why it makes sense to identify the groups anew at each time point.

This can be achieved by clustering at each individual timestamp. However, there are many different clustering algorithms, which depend on at least one, but often also several parameters. As already described in Chapter 1.2, this circumstance makes it extremely difficult to evaluate the clusters obtained. The problem can be summarised in a short question:

How can a good time series clustering be identified?

In this dissertation, our primary goal is not a clustering algorithm but the reduction of a clustering to a score that maps cluster stability. With this score we enable the user to choose the right algorithm and the right parameters in order to obtain valid clusters.

1.5 Contribution

Clustering of multidimensional data points is in many ways a very well researched field with established methods [50]. However, it is often difficult to measure the quality of the clusters obtained. For this reason, a variety of evaluation metrics such as the Rand Index [32] or the Silhoutte coefficient [48] have been proposed. Unlike the classical clustering of data points, the clustering of time series is much more complex. In most cases, the dimension of time cannot simply be considered as another feature dimension, as this could nullify its informative value. For this reason, new clustering methods have been developed specifically for time series [1, 15, 55], though methods evaluating time series clusters have not yet been developed. Without these, the validation of the obtained clusters is not possible and cluster analysis remains as a form of "black art" [50].

The main contribution of this thesis is the introduction of the *over-time stability*. This concept is used to evaluate the stability of time series in relation to each other. For this purpose, clusterings of time series are considered per time point and compared with each other. Due to the well known differences in *fuzzy* and *crisp* clustering, we present two variants of *over-time stability*, each tailored to the underlying logic. *Over-time stability* is the core concept of FCSETS (***F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries*) and CLOSE (***C**luster **O**ver-**T**ime **S**tability **E**valuation*), two methods for evaluating time series clustering. With the help of these methods, it is possible to

[4] https://www.osha.gov/data/sic-manual

assess the validity of time series clusters and clusterings. Furthermore, it is possible to compare the results of different clustering algorithms and different parameters and thus to choose the optimal configuration for an application. In particular, with the parameter selection by our methods, it is possible to obtain temporally linked clusterings without adapting the underlying clustering algorithms. The results are comparable to those of time series adapted clustering algorithms [R3]. These findings can be applied to synthetic as well as real-world data and open up new possibilities in the field of time series clustering.

To underline the usefulness of our methods, further applications based on the *over-time stability* are also presented in this dissertation. In particular, these applications represent an evaluation of the underlying concept of the *over-time stability*. Furthermore, we introduce a method for clustering time series based on *adaptability*, a concept closely related to *over-time stability*.

In the summary chapter, we discuss variations of *over-time stability* and evaluate them on further data sets. In addition, we show the interrelationships of the applications and the evaluation procedures and suggest future use cases.

In addition to some smaller contributions, we would like to highlight the core contributions:

- Idea and evaluation of the *over-time stability* (CLOSE).
- Idea, conception, implementation and evaluation of the fuzzy variant of the *over-time stability* (FCSETS).
- Idea, implementation and evaluation of the parameter selection for time series (crisp/fuzzy) clustering on real-world data.
- Discussion on the relationship between CLOSE and FCSETS.
- Idea, implementation and conception of a time series clustering algorithm.

1.6 Structure of the Thesis

We already have introduced the topic dealt with in this dissertation and defined the problem. So far, we have omitted mathematical definitions to give readers outside the field an insight into this research. However, in the respective chapters we discuss the mathematical formalisations in detail, to provide a profound understanding of our research. In the following chapter we introduce the evaluation of time series clusterings. Since our definition is based on a time-dependent stability assumption, the chapter is named *Time Series Over-Time Stability*. In the subchapters we discuss an approach for crisp clusterings and a method for fuzzy clusterings.

In Chapter 3 we introduce different variants of outlier detection methods, which are based on the ideas presented in the preceding chapter. Additionally we present a new clustering algorithm for time series which is based on a derived concept of the *Time Series Over-Time Stability*.

This is followed by an excursion in Chapter 4, in which we show that our applications provide competitive results to other well-known methods. Finally, we conclude the dissertation with an overall summary and an outlook for further research.

2

Time Series Over-Time Stability

In this chapter, we reference two fundamental papers that serve as the basis for the rest of this thesis. In particular, these works introduce the concept of *over-time stability* with respect to time series, clusters and whole clusterings.

The concept of *over-time stability* forms the basis for applications such as parameter search or outlier detection and can also be applied to others. Depending on the type of clustering at hand and the specific target of the analysis, we offer two methods in this chapter.

In Section 2.1. we first describe CLOSE, a method for evaluating the *over-time stability* of time series, clusters and whole clusterings. The method enables these elements to be considered in relation to one another and provides a basis for further analysis. For illustration purposes, we demonstrate the use case of parameter search for partitioning and density-based, time-independent clustering algorithms. Hence, we provide semantically meaningful clusterings of time series with algorithms designed for time-independent data.

In Section 2.2 we describe FCSETS, the fuzzy counterpart to CLOSE. It can be used with fuzzy clustering algorithms and is based on the *Hüllermeier-Rifqi index* [32], which was developed for comparing fuzzy partitions. According to the conditions that prevail in fuzzy environments, we have adapted the calculations of the over-time stability here. Nevertheless, the basic ideas from CLOSE remain the same.

Although one might get the impression that these works fall into the research area of *evolutionary clustering*, we use a different approach. While in the field of evolutionary clustering cluster algorithms are designed [15] or well-known cluster algorithms are adapted [1, 55], our method evaluates the over-time stability of time series, clusters and clusterings. This significant difference highlights the strengths of our approach because it is not limited to one application but forms a basis for many.

2.1 How is Your Team Spirit? Cluster Over-Time Stability Evaluation

---

Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. How is Your Team Spirit? Cluster Over-Time Stability Evaluation. In *Machine Learning and Data Mining in Pattern Recognition - 16th International Conference, MLDM 2016, New York, NY, USA, July 18-23, 2020, Proceedings*, Lecture Notes in Computer Science, pages 155–170. ibai-publishing, 2020.

Contributions: Gerhard Klassen contributed with the basic idea, the preprocessing, the acquisition of the data sets and the experiments with the Eikon data set (Section 5.1) and the GlobalEconomy data set (Section 5.2). The manuscript was jointly written by the two main authors Martha Krakowski (née Tatusch) and Gerhard Klassen under supervision of Jun.-Prof. Dr. Marcus Bravidor and Prof. Dr. Stefan Conrad.

Status: published

In the paper referenced in this section [R5] we present the ***C**luster **O**ver-**T**ime **S**tability **E**valuation* (CLOSE). A fundamental problem in clustering time series databases is the evaluation of the obtained clustering. As described in the introduction, this is a major problem, which also may lead to the misinterpretation of the results. In particular naive users are affected by this, but also domain experts may be facing difficulties [50]. Therefore, without a doubt, it is fundamental that each user be enabled to validate the clustering at hand.

In the paper [R5] we introduce an evaluation method, which allows the comparison of clusterings with the help of a simple score. It should support users to decide between clusterings, i.e. between cluster algorithms, parameters and possible initialisations, and thus contribute to the validation of clusterings. The ***C**luster **O**ver-**T**ime **S**tability **E**valuation*, presented in the paper [R5], offers a method to validate clusterings with regard to their *over-time stability*. *Over-time stability* is a novel concept and can be interpreted as an evaluation measure that examines how much the composition of clusters changes over time. As a by-product, it is also possible to evaluate not only clusterings but also individual clusters and even time series with regard to their *over-time stability*.

The presented procedure enables many further applications, some of which are discussed in later chapters. One application that becomes obvious through the evaluation of clusterings is the choice of parameters for cluster algorithms. We address this use case in the paper [R5] and illustrate the possibility of applying well-known time-independent clustering algorithms such as K-Means [42] or DBSCAN [20] in the context of time series.

The paper [R5] is part of the foundation of this dissertation, accordingly the concepts are fundamental for the further work.

How is Your Team Spirit? Cluster Over-Time Stability Evaluation

Martha Tatusch*[0000−0001−6302−6070], Gerhard Klassen*[0000−0002−1458−6546], Marcus Bravidor[0000−0003−1504−9889], and Stefan Conrad[0000−0003−2788−3854]

Heinrich Heine University, Universitätsstr. 1, 40225 D¨sseldorf, Germany
{tatusch,klassen,bravidor,stefan.conrad}@hhu.de

Abstract. Clustering of time series data is a major part of data mining. In this paper, we consider multiple multivariate time series and the clustering of their data points per timestamp. One of the major problems of this approach is that the temporal connection of clusterings at different times can neither be guaranteed nor tracked. For this reason we present CLOSE (**Cl**uster **O**ver-Time **S**tability **E**valuation): an internal evaluation measure for clusterings of temporal data. Our method evaluates not only the quality but also the over-time stability of the clusters. Time series with an equal cluster neighborhood over time are considered to be stable while those which change their neighbors often are considered as unstable. We applied our model to different data and present the results in this paper.

Keywords: Time Series Analysis · Clustering · Evaluation

1 Introduction

Information extraction from time series (TS) is well researched. There are many different approaches which all tackle specific problems. Often clustering the data has an important fraction in the concept of choice. While some of those methods divide the time series in parts, so called subsequences [2], others consider the whole time series at once [19], yet others extract feature sets [10, 26]. Although these approaches seem to solve a lot of problems and enable the discovery of knowledge, new problems like the choice of parameters arise. This parameter choice often ends up with many apparently good solutions and lacks an evaluation function which distinguishes the quality of clusterings properly. This problem grows with the amount of dimensions and requires an automatic rating of the available solutions.

In this paper we consider multiple multivariate time series with same length and equivalent time steps. We detect clusters for each point in time (called over-time clustering) with different parameters and identify the best overall clustering without knowing the ground truth. Therefore, we present an internal evaluation measure for temporal clusterings which can be used to rate and compare different clustering results of time series data. Our method, which is named CLOSE

* Both authors contributed equally to this research.

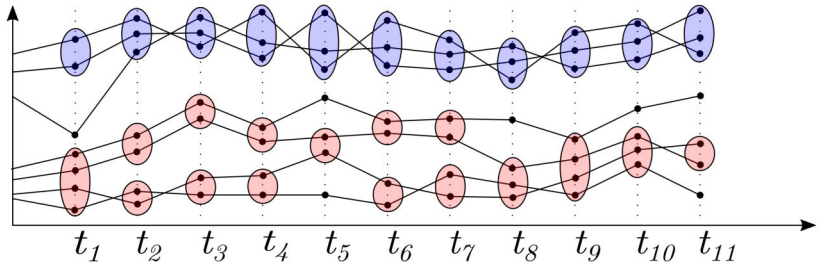


Fig. 1: Example of a time series over-time clustering [25]. The red clusters are less stable over time than the blue ones.

(**Cl**uster **O**ver-Time **S**tability **E**valuation), not only evaluates the quality of the individual clusterings per time point, but also the over-time stability. The temporal aspect is thus included in the evaluation. For the first time, this makes it possible to rate a clustering of time series data, in which the data points are clustered per timestamp, regarding the temporal linkage of clusters. Furthermore, the presented method is able to handle missing data points without adaptation. An example of an over-time clustering is illustrated in Figure 1. For a simple visualization, univariate time series are shown. Compared to whole time series clustering, this technique has a major advantage: similar partial sequences of undefined length can be found.

This approach is not only novel in the sense that it considers quality and over-time stability at the same time, but also because over-time stability differs from the stability usually represented in literature. It serves a different purpose and is based on transitions between clusters over time, which will be explained in more detail later in this work. The procedure for example may be useful when tracking topics in online forums. By clustering per point in time, the development of relationships between different terms can be investigated. When examining financial data, the procedure can lead to a gain in information as well. Assuming that the courses of different companies' financial data can be divided into groups – e.g. successful and less successful companies – clustering might be helpful to detect anomalies or even fraud. Since it cannot be guaranteed that all fraud cases are known – some may remain uncovered – this problem cannot be solved with fully supervised learning. The identification of meaningful groups would be a fundamental step. In General, the evaluation of temporal clusterings enables the identification of suitable hyper-parameters for different algorithms as a basis for further analysis such as outlier detection.

In the further course we will first discuss similar work (Section 2). We then define the considered problem (Section 3) and present our solution (Section 4). Finally, we give an overview of our experiments and their results (Section 5), discuss the method (Section 6) and draw a conclusion (Section 7).

2 Related Work

To the best of our knowledge, there does not exist any approach similar to ours, since clustering evaluation metrics usually do not contain a temporal component. For this reason, we refer on the one hand to related work with regard to time series clustering and on the other hand to time-independent evaluation of clusterings.

2.1 Time Series Clustering

In the field of time series analysis, there are different techniques for clustering time series data. When considering multiple time series, one approach is the clustering of the entire sequences [7, 19]. For our context, this procedure is not well suited as potential correlations between subsequences of different time series are not revealed. Additionally, the exact course of the time series is not relevant, but rather the trend they show. The transformation of entire sequences to feature vectors, which then are clustered [10], blurs the exact course and is a popular method. Still, the problem of not recognizing interrelated subsequences persists.

However, there is also the approach of clustering subsequences of a time series [2, 12]. Usually, this is done to find motifs in time series and therefore only a single time series is considered. In [14], Keogh et al. state that the clustering of subsequences of a single time series is meaningless, though. However, this statement is controversial, as Chen [4] argues that it is possible to obtain meaningful results if the correct distance measure is used. For this purpose, various distance measures have been introduced [23, 24].

There is also the approach of clustering partial sequences of multiple time series. Outliers may influence the results, though, and there is a need of finding a meaningful length of the subsequences, since the examination of subsequences of all lengths is usually very time-consuming. Our approach can provide more insights as subsequences of any length can selectively be investigated. However, under the assumption that the entire time course from the beginning is relevant, CLOSE only considers subsequences starting at the first point in time.

Methods for the clustering of streaming data [9, 18] are not comparable to our method, as they consider only one time series at a time and deal with other problems such as high memory requirements and time complexity.

2.2 Internal Evaluation Measures

There are many different evaluation measures for evaluating clusters and clusterings. Thereby, a distinction between *external* and *internal* measures ought to be made. In the case of the external evaluation, the ground truth is already known so that the results can be compared with expectations. In the internal evaluation, no information about the actual classes is known, so that the clusters are evaluated primarily on the basis of characteristics such as compactness or separation.

One metric that evaluates the compactness of clusters is the *Sum of Squared Errors*. It calculates the overall distance between the members and the *centroid* of a cluster. The centroid is usually the mean of all cluster members. The closer the objects of a cluster lie together, the smaller the error, the greater the compactness. However, this measure does not take into account the separation of different clusters.

The *Silhouette Coefficient* [22] evaluates the compactness as well as the separation of different clusters. This is achieved by using both the average distance of an object to members of its cluster and the average distance to members of the nearest cluster. These two properties are also addressed in the *Davies-Bouldin Index* [5] and the *Dunn Index* [6].

All these metrics cannot be directly compared to our method since they lack a temporal aspect. However, as we will show in the following, they can be applied in CLOSE.

2.3 Stability Evaluation

For the stability measurement of a clustering algorithm there are already several methods. The *Rand Index* [20], which is usually intended for the external evaluation of a clustering, can e.g. be used for this purpose. This evaluation measure rates the agreement of a clustering ζ_p with the expected result ζ_t (ground truth). Therefore it examines all object pairs that are located in the same cluster in ζ_p as well as ζ_t and all pairs that belong to different clusters in both clusterings. The number of corresponding object pairs is then set in relation to the number of all possible object pairs. Considering n objects, the number of all possible pairs is $\binom{n}{2}$.

Measuring the stability of a clustering algorithm is for instance made in order to find the optimal k for KMeans [17] or to determine the dependence of a clustering on its initialization. When considering m clusterings ζ_i ($1 \leq i \leq m$) with the same parameter k and random initialization, the Rand Index is calculated for every unordered pair of clusterings ζ_i, ζ_j with $i \neq j$ by assuming ζ_i is the ground truth without loss of generality. The stability is expressed by the average Rand Index across all pairs. Such stability measures, however, pursue a different objective and clearly do not take a temporal linkage into consideration [16].

An obvious idea would be to measure over-time stability by comparing clustering pairs of successive points in time. However, this approach is inflexible and would strongly weight variation between two points in time, although the clustering might deviate only at one point in time and otherwise remain quite stable. An ongoing change, on the other hand, would be punished only very slightly, since the variation between clusterings of two adjacent timestamps would be small, but in regard to the entire period the change would be very large. Furthermore a separation or merge of clusters would have a strong negative impact on the index. Even when comparing all possible clustering pairs of different time points these problems would persist. Our method handles such cases in a slightly different way.

In addition, the Rand Index exclusively evaluates the (over-time) stability of a clustering. But as stated in [3, 15], stability alone does not imply a *good* clustering. If this is not the case with constant data points, then certainly it is not the case with data points that change over time. CLOSE combines the evaluation of the over-time stability and the quality of a clustering to give an overall statement about an over-time clustering. However, changing values of the data objects is another problem that has to be faced when looking at time series data.

The identification of so called *Moving Clusters* [13] seems to be a closely related topic, but addresses a slightly different problem. In contrast to the evaluation of an over-time clustering, this field of research deals with the detection of clusters that remain mostly the same in regard to their members. In [13] an intuitive approach using the Jaccard Index is presented for the problem. If the Jaccard Index of two clusters of different timestamps is greater than θ, these clusters are identified as the *same cluster* for different timestamps. Apart from the fact that the clustering is not evaluated here, there is another difference to our approach: it is assumed that a cluster remains approximately the same size over time. In real data, however, this is not necessarily the case. This may apply to some tasks, such as herd tracking, which is examined in the paper, but in most cases this requirement is not satisfied.

3 Fundamentals

Since there are various approaches and definitions concerning TS analysis, we next clarify our understanding of some basic concepts regarding our approach.

Definition 1 (Time Series). *A time series $T = o_{t_1}, ..., o_{t_n}$ is an ordered set of n real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording, with t_1 and t_n indicating the first and last timestamp, respectively.*

Definition 2 (Data Set). *A data set $D = T_1, ..., T_m$ is a set of m time series of same length n and equivalent points in time.*

The vectors of all time series are denoted as the set $O = \{o_{t_1,1}, ..., o_{t_n,m}\}$. With the second index indicating the time series the data point originates from. We write O_{t_i} for all data points at a certain point in time.

Definition 3 (Cluster). *A cluster $C_{t_i,j} \subseteq O_{t_i}$ at time t_i, with $j \in \{1, ..., p\}$ being an unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm. This means that all clusters have distinct labels regardless of time.*

Definition 4 (Cluster Member). *A data point $o_{t_i,l}$ at time t_i, that is assigned to a cluster $C_{t_i,j}$ is called a member of cluster $C_{t_i,j}$.*

Definition 5 (Noise). *A data point $o_{t_i,l}$ at time t_i is considered as noise, if it is not assigned to any cluster. A data point that belongs to noise is also called an outlier.*

Definition 6 (Clustering). *A clustering is the overall result of a clustering algorithm for all timestamps. In concrete it is the set $\zeta = \{C_{t_1,1}, ..., C_{t_n,p}\} \cup Noise$.*

4 Method

A major disadvantage of creating clusters for every timestamp is an evident missing temporal link. In our approach we assume that different clusterings deliver different cluster connectedness and that this bond can be measured. In order to measure the temporal linking we make use of a stability function. Given a clustering ζ, we first analyze the behavior of every subsequence of a time series $T = o_{t_1}, ...o_{t_k}$, with $t_k \leq t_n$, starting at the first timestamp. This is done, because time series which separate from their clusters' members often, indicate a low temporal linkage. One could say we evaluate the *team spirit* of the individual time series. Further, we rate every cluster with a stability function, which depends on the subsequence analysis and the number of clusters merged into this cluster. Finally, we assign a score to the clustering, depending on the over-time stability of every cluster.

Let $C_{t_i,a}$ and $C_{t_j,b}$ be two clusters, with $t_i, t_j \in \{t_1, ...t_n\}$. In order to measure the stability of a time series we first introduce the *temporal cluster intersection*

$$\cap_t \{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \wedge o_{t_j,l} \in C_{t_j,b}\}, \tag{1}$$

with $l \in \{1, ..., m\}$. The temporal cluster intersection returns a set of time series, which contain data points grouped together in t_i as well as in t_j. Now the behavior of a subsequence from one cluster $C_{t_i,a}$ in t_i to another $C_{t_j,b}$ in t_j can be expressed by the proportion of members of $C_{t_i,a}$ remaining together in t_j

$$p(C_{t_i,a}, C_{t_j,b}) = \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a}|}, \tag{2}$$

with $t_i < t_j$. In the example in Figure 2 the proportion for $C_{t_i,l}$ and $C_{t_j,v}$ would be

$$p(C_{t_i,l}, C_{t_j,v}) = \frac{|\{a, b\}|}{|\{a, b\}|} = \frac{2}{2} = 1.0.$$

With the help of the proportion of clusters we now can rate all data points of a sequence with a *subsequence score*. It is defined as

$$subseq\_score(o_{t_k,l}) = \frac{1}{k_a} \cdot \sum_{i=1}^{k-1} p(cid(o_{t_i,l}), cid(o_{t_k,l})), \tag{3}$$

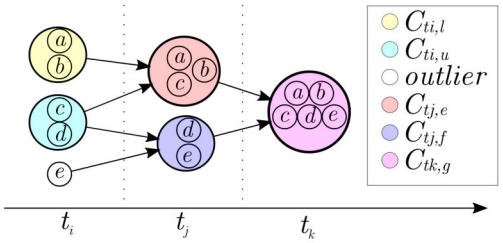


Fig. 2: Example for transitions of TS $a, .., e$ between clusters over time [25].

with $l \in \{1, ..., m\}$, $k_a \in [1, k-1]$ being the number of timestamps where the data point exists and is assigned to a cluster (therefore is **not** recognized as noise), and *cid*, the cluster-identity function

$$cid(o_{t_i,j}) = \begin{cases} \emptyset & \text{if the data point is not assigned to a cluster} \\ C_{t_i,l} & \text{else} \end{cases} \tag{4}$$

returning the cluster which the data point has been assigned to in t_i. Thus, in this equation, all time points in which an object is an outlier, are ignored. The *subsequence score* takes into account how many objects from the previous clusters have migrated together with the currently viewed object.

Regarding the example of Figure 2, the score of time series a in time point t_k would be:

$$subseq\_score(o_{t_k,a}) = \frac{1}{2} \cdot (1.0 + 1.0) = 1.0.$$

This value reflects the highest stability. The time series d, on the other hand, gets a lower value of $subseq\_score(o_{t_k,d}) = 0.75$ as it once changes the cluster without its cluster members.
The rating of clusters depends on two factors. The first factor is the number of merged clusters

$$m(C_{t_k,i}) = |\{C_{t_l,j} \mid t_l < t_k \wedge \exists a : o_{t_l,a} \in C_{t_l,j} \wedge o_{t_k,a} \in C_{t_k,i}\}|, \tag{5}$$

which describes the amount of different clusters of previous timestamps, that merged into the regarded cluster. The second factor is the sum of all *subsequence scores* of the data points within the regarded cluster. So the *over-time stability* of a cluster is defined as

$$ot\_stability(C_{t_k,i}) = \frac{\frac{1}{|C_{t_k,i}|} \cdot \sum_{o_{t_k,l} \in C_{t_k,i}} subseq\_score(o_{t_k,l})}{\frac{1}{k-1} \cdot m(C_{t_k,i})} \tag{6}$$

for $k > 1$. Note that the entire preceding time frame is considered. For the first timestamp we consider clusters to be stable and set $ot\_stability(C_{t_1,i}) = 1.0$. It is important to mention, that the number of merged clusters does not take outliers into account.

Regarding the example of Figure 2, the stability of the cluster $C_{t_k,g}$ would be:

$$ot\_stability(C_{t_k,g}) = \frac{\frac{1}{5} \cdot (1.0 + 1.0 + 0.75 + 0.75 + 1.0)}{\frac{1}{2} \cdot 4} = 0.45.$$

This low score can be explained by the fact that the cluster under consideration contains only a few data points, two of which already have an independent course of their clusters' members.
Finally we can rate the over-time stability of a clustering ζ:

$$CLOSE(\zeta) = \frac{1}{N_C} \cdot \Big(1 - \Big(\frac{k}{N_C}\Big)^2\Big) \cdot \Big(\sum_{C \in \zeta} ot\_stability(C) \cdot (1 - quality(C))\Big), \quad (7)$$

with N_C being the number of clusters of the whole clustering, k being the number of considered timestamps and *quality* being an arbitrary cluster rating function. We suggest the mean squared error (MSE) but density ratings like the local outlier factor (LOF) can also be used. Be aware using a function in the interval of $[0, 1]$ in order to get appropriate results. If greater values indicate a higher quality, $(1-quality(C))$ may e.g. be replaced by $(1-quality(C)^{-1})$ or $quality(C)$ depending on the quality measure.

When using normalized data with feature values in $[0, 1]$, and a measure function in $[0, 1]$, CLOSE as well returns a score between 0 and 1, with 1 indicating a good over-time clustering, as long as there is at least one cluster per timestamp.

The pre-factors result on the one hand from averaging by the number of clusters and on the other hand from the factor $1 - (\frac{k}{N_C})^2$. This is intended to counteract one large cluster, since such a clustering automatically receives a very high rate of over-time stability. The more clusters exist per time, the larger the factor. However, to prevent the creation of too many clusters, the influence of the fraction is diminished by squaring it.

Remark 1 (Time Point Comparison). In contrast to comparing pairs of consecutive points in time, CLOSE contains temporal information that is robust against outliers. By comparing clusterings of all preceding time points with the last timestamp of the considered subsequence, short-term changes to other clusters are weighted more lightly. In addition, long-term changes that develop slowly over time are punished more severely. Since the influence of the *over-time stability* is weighted with the *quality* of the cluster, the formula cannot be transformed to simply iterate over all cluster pairs.

Remark 2 (Handling Outliers). Our calculations are suitable for both cleaned and noisy data. Since outliers are neither considered in the subsequence score nor in the cluster stability, they have no influence at this point. However, they do have an indirect influence on the calculation of the clustering score. The prefactor favors a large number of clusters. Depending on the quality of the clusters, it may be more advantageous for the algorithm to assign data points to smaller clusters than to interpret them as noise and recognize only a few large clusters.

In view of the fact that over-time clustering might be used for outlier detection, this treatment of outliers is reasoned. In this case, the algorithm should not

be forced to assign every data object to a cluster. Nevertheless, the treatment of outliers may be extended in future work. One way to penalize noise would be, to replace k_a in the subsequence score with k. This would cause, that outliers would get the worst score of 0, as the timestamps would not be skipped.

Remark 3 (Merge & Split of Clusters). Considering the *subsequence score*, a merge of clusters has no negative impact on the score. On the contrary: if two clusters fuse entirely, the score is actually very good, since all objects move with all their cluster members. This circumstance is intended, as the focus is primarily on the cohesion of time series. As long as a group of time series remains together, it is not negative if more are joining.

If a split happens, however, the *subsequence score* decreases. This is also wanted, as a split indicates that time series that have formed a group at one point in time no longer hold together. This fact contradicts the desired cohesion and will be penalized in any case. If smaller clusters have previously been merged and then separated again in the same way as before, this has no great influence on the score over time, though.

Remark 4 (Additional Remark). A small sample size not only influences the stability when considering constant data points [3], but also leads to a high sensitivity to transitions between clusters when examining the over-time stability. The more data points are considered, the easier it is to give a meaningful statement about the (over-time) stability.

5 Experiments

To the best of our knowledge there are no comparable measures presented in literature. This is why we decided to make experiments to demonstrate the results of our measure. We show the transferability of our method to reality by performing two experiments on real data. Additionally, we present the results on two artificially generated data sets, that satisfy the necessary assumptions for the meaningful use of CLOSE, to show the impact of the over-time stability. For all experiments MSE was chosen as the cluster quality measure.

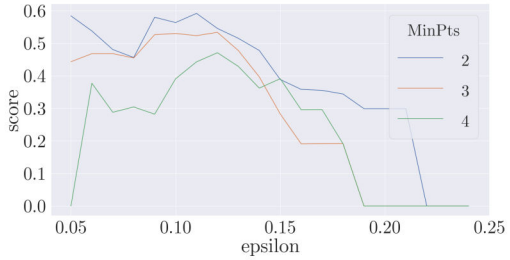


Fig. 3: Achieved CLOSE scores for $minPts \in [2, 4]$ depending on ϵ on the EIKON Financial data set.

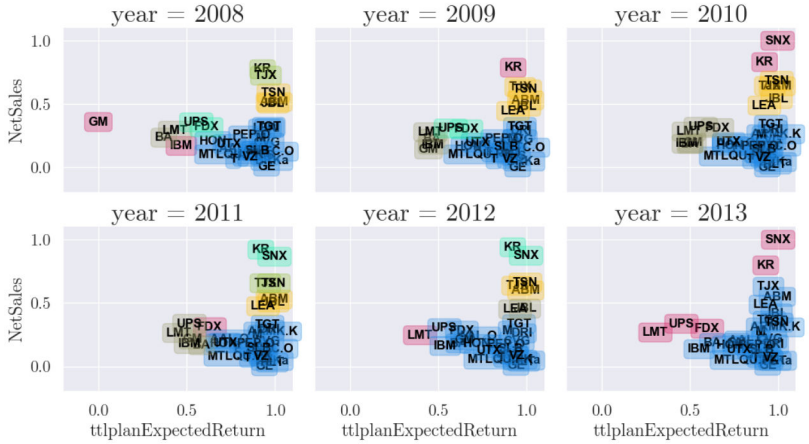


Fig. 4: Detected clusters by DBSCAN with $minPts = 2$ and $\epsilon = 0.11$ on the *EIKON Financial* data set. Red data points represent outliers.

5.1 EIKON Financial Data Set

The first data set is extracted from *EIKON* [21] which is a commercial set of software products released by Refinitiv (formerly Thomson Reuters Financial & Risk). It includes a database with financial information of thousands of companies. For the ease of visibility we chose two random features of fifty random companies. The features we chose are the net sales and the total plan expected return, which are figures taken from the balance sheet of the companies. Thomson Reuters named the according fields *TR-NetSales* and *TR-TtlPlanExpectedReturn*, respectively. The first feature represents the sales receipts for products and services without cash discounts, trade discounts, excise tax, sales returns and allowance. The second feature represents the total amount of expected return on all of a company's pension and post-retirement plans. We normalized the data through dividing the features by the total assets. This is a common approach in economics. The coefficient of correlation of these two features regarding our subset is 0.210. One time series represents the described features of one company over time.

In order to evaluate the CLOSE score on this data set we used the clustering algorithm DBSCAN [8] and applied a grid search with three different $minPts$ (2,3,4) in the epsilon interval $[0.05, 0.25]$. In Figure 3 it can be seen, that $minPts = 2$ reached the maximal CLOSE score of 0.59 at $\epsilon = 0.11$. Clusterings with $minPts = 3$ and $minPts = 4$ reached lower scores at higher epsilons. This is an expected behavior, since a higher $minPts$ would require a higher ϵ in order to create a cluster in this data set. A higher ϵ leads to a higher mse, which has a negative effect on the CLOSE score.

The resulting clustering is illustrated in Figure 4 and shows a very stable clustering. Especially notable is the subsequence from 2008 to 2012, which shows only minor variations.

5.2 GlobalEconomy Data Set

The second dataset is obtained from www.theglobaleconomy.com [1], which is a website that provides economic data for different countries. For this experiment we randomly selected two features, namely the "Unemployment Rate" and the "Public spending on education, percent of GDP". In order to make the chart clearer, we removed some countries and reduced it to the years from 2010 to 2013. Further we applied a min-max normalization.

In this experiment, we want to illustrate the differences of a clustering which received a good score and another clustering which received a worse one. Therefore we clustered the dataset with seeded KMeans and different k.

In Figure 5 it can be seen, that the clustering with $k = 8$ received a CLOSE score of 0.67, which represents the best score. The clustering itself can be seen in Figure 6. In order to compare this clustering to another with a lower score Figure 6 also holds the clustering result for $k = 3$.

In direct comparison the first differences that stand out are the cluster sizes. The clusters received with $k = 8$ are smaller than those of $k = 3$. This alone is no surprise but it leads to a smaller MSE and thus to a lower negative influence on the CLOSE score. In numbers, the average MSE for $k = 8$ is 0.0036. For $k = 3$ it is 0.0289. The second not so obvious observation is the average cluster stability. While the clustering with $k = 3$ has an average stability of 0.56, the agglomerations found with $k = 8$ got an average stability of 0.68. One example which leads to a higher stability is the behavior of the object BRB and its neighborhood. In the clustering with the highest CLOSE score, BRB has the same cluster neighbors in the first and the last year. In addition it is alone in a cluster in 2012, which means it moved with 50% of its neighbors from 2010. This is not the case in the clustering which was found with $k = 3$. In fact in the clustering with $k = 3$, BRB is never in a cluster with the same cluster members

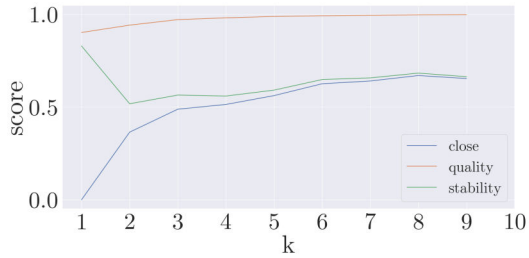


Fig. 5: Achieved scores for different k on the GlobalEconomy data set.

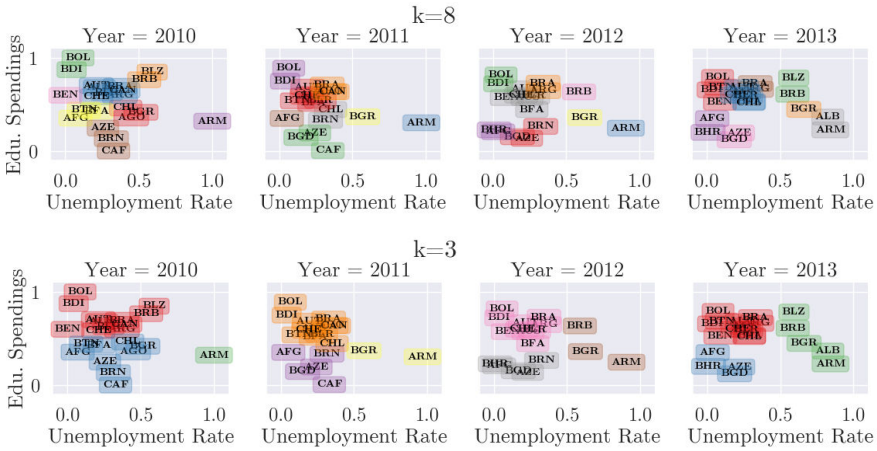
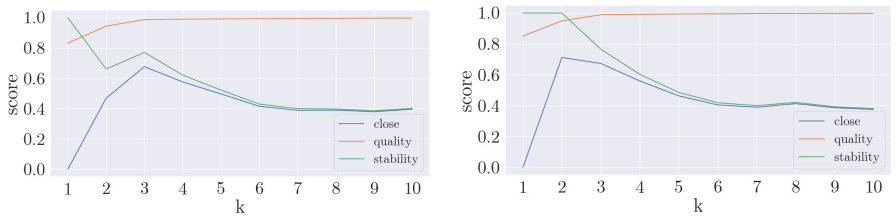


Fig. 6: KMeans Clusters with $k = 8$ and $k = 3$ on the GlobalEconomy data set. The datapoints contain ISO Countrycodes.

over two years. Another observation in the clustering with $k = 3$ is, that data points which change their cluster neighbors over time often move with a low number of other data points.

5.3 Artificially Generated Data Set

To show what a good clustering and the associated CLOSE score may look like, we generated two artificial data sets. In both cases, at first three random centroids with two features $\in [0, 1]$ were chosen. Then 20, 15 and 10 time series were placed next to these centroids, respectively. This means that the data points



(a) Data Set A (b) Data Set B

Fig. 7: Achieved CLOSE scores, average quality and average *ot_stability* for the two generated data sets depending on k. The quality line is given by 1 - MSE.

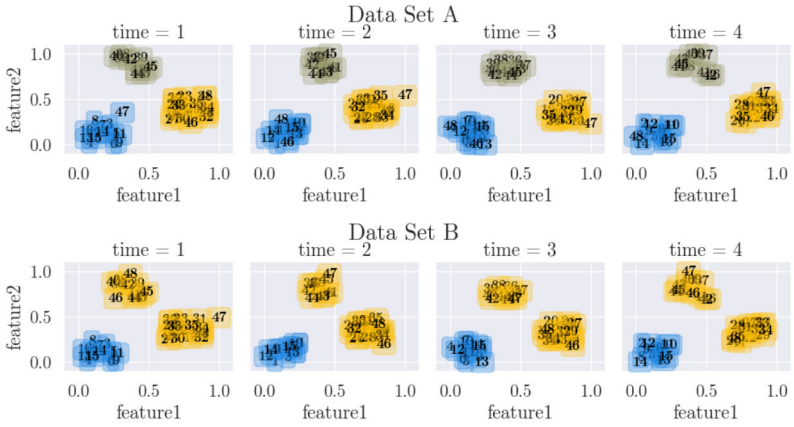


Fig. 8: Detected clusters by KMeans on the two artificially generated data sets.

of a time series for each time point were set with a maximal distance of 0.1 per dimension to the assigned centroid. Subsequently, data points for 3 time series (namely 46, 47 and 48) with random transitions between two of the three clusters were placed in the feature space. For overview purposes a total of 4 time points and 48 time series were examined. In Figure 8 the resulted data sets can be seen. Data set A contains transitions between the two lower clusters. In data set B there are transitions between the two upper clusters.

The clustering was performed with KMeans [17] for $1 \leq k \leq 10$. Figure 7 shows the achieved CLOSE scores, average quality and average *ot_stability* depending on k, whereby the quality line is given by 1 - MSE. While for data set A the best k is in accordance to the chosen centroids three, for data set B $k = 2$ is preferable. The corresponding clustering results are illustrated in Figure 8. The outcomes show that the best results regarding the CLOSE score may deviate from those of normal clustering if a fusion/split of clusters can increase the over-time stability without causing significant quality loss. As in data set B the clusters with bouncing time series are located close together, a merge of the two clusters is beneficial: the quality is only slightly affected, while the stability is significantly increased.

6 Discussion

Clustering time series is a challenging task. Besides the methodology, the user needs to choose parameters, which all lead to different results. Improving the results by adapting the parameters is often only possible with the help of a specialist. In this paper we provide a systematic approach for the determination of parameters in order to reach a given target. This enables users not only to

compare different clusterings, but also to choose a method and parameters suited for the data set without further knowledge.

Further more our work enables the user to use an arbitrary cluster algorithm and distance function, without further adaptation. If considering uniformly populated convex data groups, measures such as the mean squared error (MSE) or mean absolute error (MAE), and distance- or partition-based clustering algorithms such as KMeans are suitable. If the data set contains groups whose members are not approximately normally distributed, density-based measures such as the local outlier factor (LOF) and clustering algorithms such as DBSCAN might be more appropriate. Additionally, the formula of CLOSE (7) can be modified, so that quality measures for clusterings instead of clusters can be used. In that case, the average cluster stability avg_stab for every clustering ζ_{t_i} at time t_i can be considered:

$$CLOSE(\zeta) = \frac{1}{N_C} \cdot \Big(1 - \Big(\frac{k}{N_C}\Big)^2\Big) \cdot \Big(\sum_{\zeta_{t_i} \subset \zeta} avg\_stab(\zeta_{t_i}) \cdot (1 - quality(\zeta_{t_i}))\Big). \quad (8)$$

We are aware, that the presented method is computationally intensive but we are confident to enhance the approach in the future. Moreover, this is only a small drawback in view of the fact, that the complex manual search, which itself is very time-consuming anyway, gets simplified and guided.

7 Conclusion and Future Work

The presented method can be divided into two major parts: First the rating of time series and their subsequences, and second the evaluation of over-time clusterings. In this paper we focused on the latter. Therefore we presented a robust method which is able to rate over-time clusterings regarding a temporal linkage. This enables the comparison of different clusterings and their bond in time. We have performed several experiments and explained the influence of the major factors. The results show that our method is able to measure the over-time stability accurately for over-time clusterings of multiple multivariate time series. With the help of the presented measure, stable clusters are found. Due to the consideration of the quality, however, no unintuitive clusters are forced in favor of stability.

Based on CLOSE, much further research can be done. Apart from investigating different quality measures for clusterings, the treatment of outliers can be contextually adapted and analyzed. One way to penalize noise would be, to replace k_a in the subsequence score (3) with k. This would cause, that outliers would get the worst score of 0, as the timestamps would not be skipped. Besides, an intelligent initialization of the reference timestamp could be developed. Instead of examining the behavior with respect to the first point in time, e.g. the time with the highest clustering quality could be chosen. Furthermore, CLOSE can be used to detect anomalous subsequences using the *subsequence score* [25].

The presented measure could also be used in streaming environments. For example, it could indicate a significant change of data composition. Social media

could be an interesting field of application, too. The subsequence score of Instagram followers could e.g. be an indicator for their probability of remaining as a follower. In addition, the combination of CLOSE with contextual clustering [11] might lead to deeper insights about the resulting cluster compositions. Another interesting aspect would be the development of an over-time clustering algorithm using CLOSE as objective function. This would make the time-consuming search for optimal parameters per time point disappear.

Acknowledgement

We would like to thank the J¨rgen Manchot Foundation, which supported this work by financing the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University Duesseldorf.

References

1. Global economy, world economy, https://www.theglobaleconomy.com/, accessed: 13.01.2020
2. Banerjee, A., Ghosh, J.: Clickstream clustering using weighted longest common subsequences. In: Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining. pp. 33–40 (2001)
3. Ben-David, S., Von Luxburg, U.: Relating clustering stability to properties of cluster boundaries. In: 21st Annual Conference on Learning Theory (COLT 2008). pp. 379–390 (2008)
4. Chen, J.R.: Useful clustering outcomes from meaningful time series clustering. In: Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70. pp. 101–109 (2007)
5. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**(2), 224–227 (1979)
6. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics **3**(3), 32–57 (1973)
7. Ernst, J., Nau, G.J., Bar-Joseph, Z.: Clustering short time series gene expression data. Bioinformatics **21**(suppl_1), i159–i168 (2005)
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226–231 (1996)
9. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: Theory and practice. IEEE Trans. on Knowl. and Data Eng. **15**(3), 515–528 (2003)
10. Huang, X., Ye, Y., Xiong, L., Lau, R.Y., Jiang, N., Wang, S.: Time series k-means: A new k-means type smooth subspace clustering for time series data. Information Sciences **367-368**, 1 – 13 (2016)
11. J¨nichen, S., Perner, P.: Conceptual clustering and case generalization of two-dimensional forms. Computational Intelligence **22**(3-4), 177–193 (2006)
12. Jin, X., Lu, Y., Shi, C.: Distribution discovery: Local analysis of temporal rules. In: Advances in Knowledge Discovery and Data Mining. pp. 469–480 (2002)

13. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Advances in Spatial and Temporal Databases. pp. 364–381 (2005)
14. Keogh, E., Lin, J.: Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowledge and Information Systems **8**(2), 154–177 (2005)
15. Kuncheva, L.I., Vetrov, D.P.: Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1798–1808 (2006)
16. von Luxburg, U.: Clustering stability: An overview. Found. Trends Mach. Learn. **2**(3), 235–274 (2010)
17. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297 (1967)
18. O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: Streaming-data algorithms for high-quality clustering. In: Proceedings of IEEE International Conference on Data Engineering. p. 685 (2001)
19. Paparrizos, J., Gravano, L.: k-shape: Efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 1855–1870 (2015)
20. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association **66**(336), 846–850 (1971)
21. Reuters, T.: Eikon financial analysis and trading software
22. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53 – 65 (1987)
23. Schäfer, P.: Experiencing the shotgun distance for time series analysis. Trans. MLDM **7**, 3–25 (2014)
24. Schäfer, P.: Towards time series classification without human preprocessing. In: MLDM 2014: Machine Learning and Data Mining in Pattern Recognition. pp. 228 – 242 (2014)
25. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In: Data Mining. AusDM 2019. Communications in Computer and Information Science. vol. 1127, pp. 91–103 (2019)
26. Truong, C.D., Anh, D.T.: A novel clustering-based method for time series motif discovery under time warping measure. International Journal of Data Science and Analytics **4**(2), 113–126 (2017)

2.2 Fuzzy Clustering Stability Evaluation of Time Series

Gerhard Klassen, Martha Tatusch, Ludmila Himmelspach, and Stefan Conrad. Fuzzy Clustering Stability Evaluation of Time Series. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Communications in Computer and Information Science, pages 680–692, Cham, 2020. Springer International Publishing .

Contributions: Gerhard Klassen contributed the main idea of the paper, the implementation and the experiments. The formal concept was jointly developed by Gerhard Klassen and Ludmila Himmelspach. The manuscript was jointly prepared by Martha Krakowski (née Tatusch), Ludmila Himmelspach and Gerhard Klassen under the supervision of Prof. Dr. Stefan Conrad.

Status: published

Fuzzy logic is a mathematical concept that, unlike Boolean logic, works with degrees of state [9]. In the field of clustering, this can be a useful methodology, especially when working with noisy data. In such cases, fuzzy clustering algorithms often perform better than their crisp counterparts [37]. This is can encountered in image data where clusters tend to overlap [4], for example. In this context time series can be extracted and clustered from the images in videos. A corresponding application would have many advantages over crisp approaches, as noise can be filtered out easier. The identification of noise can potentially be improved by including a temporal context. Previous work has shown, that reference images were used to eliminate noise from videos [12]. Recalculating this reference image depending on the clustering at a certain timestamp could further improve this work.

Probably the best-known fuzzy clustering is the *Fuzzy C-Means* [8] algorithm, which is often regarded as a fuzzy variation of the K-Means [42] algorithm. However, this method and many other fuzzy clustering algorithms such as [24, 19] are usually not adapted to time-dependent data. If clustering time series, it is therefore necessary to evaluate the resulting agglomerations in order to be able to make a decision on a specific clustering. For this reason, we have developed the ***F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries* (FCSETS) and presented it in the referenced paper [R3].

The procedure is the fuzzy counterpart to the ***C**luster **O**ver-**T**ime **S**tability **E**valuation* and is based on already existing concepts such as the Hüllermeier-Rifqi Index [32]. It originally was developed for the comparison of fuzzy partitions and represents the basis for the calculation of *over-time stability*.

The method presented in [R3] is just as fundamental as CLOSE and serves as a basis for further applications. Similar to the previous section, here we focus on the illustration of the parameter determination in order to obtain a *stable over-time clustering*. In the further course of this dissertation, we present some applications such as outlier detection. Although the principles of the implementation are geared towards CLOSE, the considerations made can also be easily transferred to FCSETS.

Fuzzy Clustering Stability Evaluation of Time Series

Gerhard Klassen( ), Martha Tatusch, Ludmila Himmelspach, and Stefan Conrad

Heinrich Heine University, Universitätsstr. 1, 40225 Düsseldorf, Germany
{gerhard.klassen,martha.tatusch,ludmila.himmelspach,stefan.conrad}@hhu.de

Abstract. The discovery of knowledge by analyzing time series is an important field of research. In this paper we investigate multiple multivariate time series, because we assume a higher information value than regarding only one time series at a time. There are several approaches which make use of the granger causality or the cross correlation in order to analyze the influence of time series on each other. In this paper we extend the idea of mutual influence and present FCSETS (**F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries), a new approach which makes use of the membership degree produced by the fuzzy c-means (FCM) algorithm. We first cluster time series per timestamp and then compare the relative assignment agreement (introduced by Eyke Hüllermeier and Maria Rifqi) of all subsequences. This leads us to a stability score for every time series which itself can be used to evaluate single time series in the data set. It is then used to rate the stability of the entire clustering. The stability score of a time series is higher the more the time series sticks to its peers over time. This not only reveals a new idea of mutual time series impact but also enables the identification of an optimal amount of clusters per timestamp. We applied our model on different data, such as financial, country related economy and generated data, and present the results.

Keywords: Time series analysis · Fuzzy clustering · Evaluation

1 Introduction

The analysis of sequential data – so called time series (TS) – is an important field of data mining and already well researched. There are many different tasks, but the identification of similarities and outliers are probably among the most important ones. Clustering algorithms try to solve exactly these problems. There are various approaches for extracting information from time series data with the help of clustering. While some methods deal with parts of time series, so called subsequences [2], others consider the whole sequence at once [9,28], or transform them to feature sets first [17,34]. In some applications clusters may overlap, so that membership grades are needed, which enable data points to belong to more

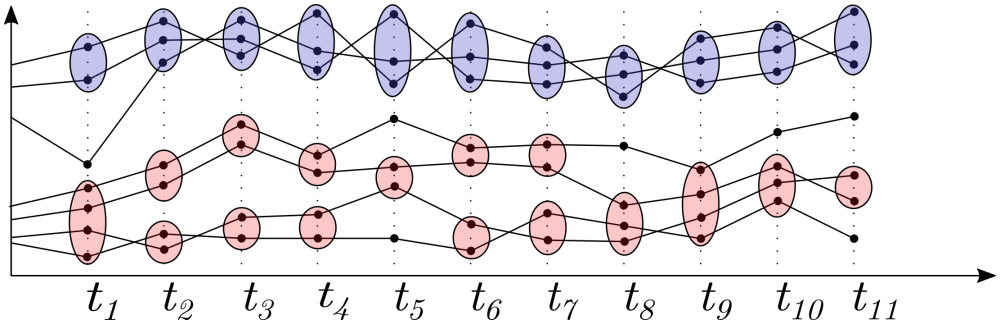


Fig. 1. Example for an over-time clustering of univariate time series [32]. The blue clusters are more stable over time than the red ones.

than one cluster to different degrees. These methods fall into the field of *fuzzy clustering* and they are used in time series analysis as well [24].

However, in some cases the exact course of time series is not relevant but rather the detection of groups of time series that follow the same trend. Additionally, time-dependent information can be meaningful for the identification of patterns or anomalies. For this purpose it is necessary to cluster the time series data per time point, as the comparison of whole (sub-)sequences at once leads to a loss of information. For example, in case of the euclidean distance the mean distance over all time points is considered. In case of Dynamic Time Warping (DTW) the smallest distance is relevant. The information at one timestamp has therefore barely an impact. The approach of clustering time series per time point enables an advanced analysis of their temporal correlation, since the behavior of sequences to their cluster peers can be examined. In the following this procedure will be called *over-time clustering*. An example is shown in Fig. 1. Note, that for simplicity reasons only univariate time series are illustrated. However, over-time clustering is especially valuable for multivariate time series analysis.

Unfortunately new problems like the right choice of parameters arise. Often the comparison of clusterings with different parameter settings is difficult since there is no evaluation function which distinguishes the quality of clusterings properly. In addition, some methods, such as outlier detection, require good clustering as a basis, whereby the quality can contextually be equated with the stability of the clusters.

In this paper, we focus on multiple multivariate time series with same length and equivalent time steps. We introduce an evaluation measure named FCSETS (**F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries) for the over-time stability of a fuzzy clustering per time point. For this purpose our approach rates the over-time stability of all sequences considering their cluster memberships. To the best of our knowledge this is the first approach that enables the stability evaluation of clusterings and sequences regarding the temporal linkage of clusters.

Over-time clustering can be helpful in many applications. For example, the development of relationships between different terms can be examined when tracking topics in online forums. Another application example is the analysis

of financial data. The over-time clustering of different companies' financial data can be helpful regarding the detection of anomalies or even fraud. If the courses of different companies' financial data can be divided into groups, e.g. regarding their success, the investigation of clusters and their members' transitions might be a fundamental step for further analysis. As probably not all fraud cases are known (some may remain uncovered) this problem cannot be solved with fully supervised learning.

The stability evaluation of temporal clusterings offers a great benefit as it not only enables the identification of suitable hyper-parameters for different algorithms but also ensures a reliable clustering as a basis for further analysis.

2 Related Work

In the field of time series analysis, different techniques for clustering time series data were proposed. However, to the best of our knowledge, there does not exist any approach similar to ours. The approaches described in [8,19,28] cluster entire sequences of multiple time series. This procedure is not well suited for our context because potential correlations between subsequences of different time series are not revealed. Additionally, the exact course of the time series is not relevant, but rather the trend they show. The problem of not recognizing interrelated subsequences also persists in a popular method where the entire sequences are first transformed to feature vectors and then clustered [17]. Methods for clustering streaming data like the ones proposed in [14] and [25] are not comparable to our method because they consider only one time series at a time and deal with other problems such as high memory requirements and time complexity. Another area related to our work is community detection in dynamic networks. While approaches presented in [12,13,26,36] aim to detect and track local communities in graphs over time, the goal of our method is finding a stable partitioning of time series over the entire period so that time series following the same trend are assigned to the same cluster.

In this section, first we briefly describe the fuzzy c-means clustering algorithm that we use for clustering time series objects at different time points. Then, we refer on the one hand to related work with regard to time-independent evaluation measures for clusterings. Finally, we describe a resampling approach for cluster validation and a fuzzy variant of the Rand index that we use in our method.

2.1 Fuzzy C-Means (FCM)

Fuzzy c-means (FCM) [4,7] is a partitioning clustering algorithm that is considered as a fuzzy generalization of the hard k-means algorithm [22,23]. FCM partitions an unlabeled data set $X = \{x_1, ..., x_n\}$ into c clusters represented by their prototypes $V = \{v_1, ..., v_c\}$. Unlike k-means that assigns each data point to exactly one cluster, FCM assigns data points to clusters with membership degrees $u_{ik} \in [0, 1]$, $1 \leq i \leq c$, $1 \leq k \leq n$. FCM is a probabilistic clustering

algorithm which means that its partition matrix $U = [u_{ik}]$ must satisfy two conditions given in (1).

$$\begin{aligned} \sum_{i=1}^{c} u_{ik} &= 1 \quad \forall k \in \{1, ..., n\}, \\ \sum_{k=1}^{n} u_{ik} &> 0 \quad \forall i \in \{1, ..., c\}. \end{aligned} \tag{1}$$

Since we focus on partition matrices produced by arbitrary fuzzy clustering algorithms, we skip further details of FCM and refer to the literature [4].

2.2 Internal Evaluation Measures

Many different *external* and *internal* evaluation measures for evaluating clusters and clusterings were proposed in the literature. In the case of the external evaluation, the clustering results are compared with a ground truth which is already known. In the internal evaluation, no information about the actual partitioning of the data set is known, so that the clusters are often evaluated primarily on the basis of characteristics such as compactness and separation.

One metric that evaluates the compactness of clusters is the *Sum of Squared Errors*. It calculates the overall distance between the data points and the cluster prototype. In the case of fuzzy clustering, these distances are additionally weighted by the membership degrees. The better the data objects are assigned to clusters, the smaller the error, the greater the compactness. However, this measure does not explicitly take the separation of different clusters into account.

There are dozens of fuzzy cluster validity indices that evaluate the compactness as well as the separation of different clusters in the partitioning. Some validity measures use only membership degrees [20,21], other include the distances between the data points and cluster prototypes [3,5,11,35]. All these measures cannot be directly compared to our method because they lack a temporal aspect. However, they can be applied in FCSETS for producing an initial partitioning of a data set for different time points.

2.3 Stability Evaluation

The idea of the resampling approach for cluster validation described in [30] is that the choice of parameters for a clustering algorithm is optimal when different partitionings produced for these parameter settings are most similar to each other. The *unsupervised cluster stability value* $s(c)$, $c_{min} \leq c \leq c_{max}$, that is used in this approach is calculated as average pairwise distance between m partitionings:

$$s(c) = \frac{\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} d(U_{ci}, U_{cj})}{m \cdot (m-1)/2}, \tag{2}$$

where U_{ci} and U_{cj}, $1 \leq i < j \leq m$, are two partitionings produced for c clusters and $d(U_{ci}, U_{cj})$ is an appropriate similarity index of partitionings. Our stability measure is similar to the unsupervised cluster stability value but it includes the temporal dependencies of clusterings.

Since we deal with fuzzy partitionings, in our approach we use a modified version of the *Hërmeier-Rifqi Index* [18]. There are other similarity indices for comparing fuzzy partitions like *Campello's Fuzzy Rand Index* [6] or *Frigui Fuzzy Rand Index* [10] but they are not reflexive.

The *Hërmeier-Rifqi Index (HRI)* is based on the *Rand Index* [29] that measures the similarity between two hard partitions. The Rand index between two hard partitions $U_{c\times n}$ and $\tilde{U}_{\tilde{c}\times n}$ of a data set X is calculated as the ratio of all concordant pairs of data points to all pairs of data points in X. A data pair (x_k, x_j), $1 \leq k, j \leq n$ is concordant if either the data points x_k and x_j are assigned to the same cluster in both partitions U and $\tilde{U}$, or they are in different clusters in U and $\tilde{U}$. Since fuzzy partitions allow a partial assignment of data points to clusters, in [18], the authors proposed an equivalence relation $E_U(x_k, x_j)$ on X for the calculation of the assignment agreement of two data points to clusters in a partition:

$$E_U(x_k, x_j) = 1 - \frac{1}{2} \sum_{i=1}^{c} |u_{ik} - u_{ij}|. \tag{3}$$

Using the equivalence relation $E_U(x_k, x_j)$ given in Formula (3), the Hërmeier-Rifqi index is defined as a normalized degree of concordance between two partitions U and $\tilde{U}$:

$$\mathrm{HRI}(U, \tilde{U}) = 1 - \frac{1}{n(n-1)} \sum_{k=1}^{n} \sum_{j=k+1}^{n} |E_U(x_k, x_j) - E_{\tilde{U}}(x_k, x_j)|. \tag{4}$$

In [31], Runkler has proposed the *Subset Similarity Index (SSI)* which is more efficient than the Hërmeier-Rifqi Index. The efficiency gain of the Subset Similarity Index is achieved by calculating the similarity between cluster pairs instead of the assignment agreement of data point pairs. We do not use it in our approach because we evaluate the stability of a clustering over time regarding the *team spirit* of time series. Therefore, in our opinion, the degree of the assignment agreement between time series pairs to clusters at different time stamps contributes more to the stability score of a clustering than the similarity between cluster pairs.

3 Fundamentals

In this chapter we clarify our understanding of some basic concepts regarding our approach. For this purpose we supplement the definitions from [32]. Our method considers multivariate time series, so instead of a definition with real values we use the following definition.

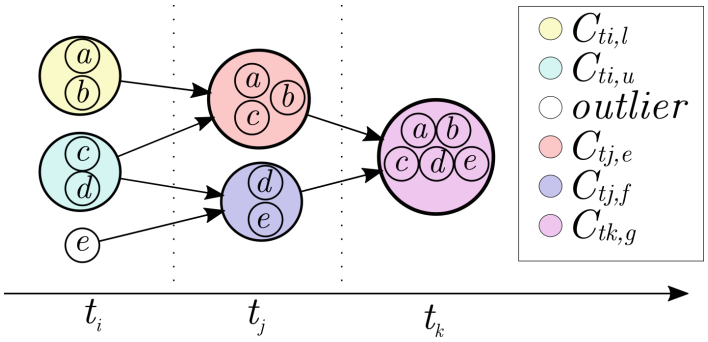


Fig. 2. Illustration of transitions of time series $T_a, .., T_e$ between clusters over time [32].

Definition 1 (Time Series). *A time series* $T = o_{t_1}, ..., o_{t_n}$ *is an ordered set of* n *real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording, with* t_1 *and* t_n *indicating the first and last timestamp, respectively.*

Definition 2 (Data Set). *A data set* $D = T_1, ..., T_m$ *is a set of* m *time series of same length* n *and equal points in time.*

The vectors of all time series are denoted as the set $O = \{o_{t_1,1}, ..., o_{t_n,m}\}$. With the second index indicating the time series the data point originates from. We write O_{t_i} for all data points at a certain point in time.

Definition 3 (Cluster). *A cluster* $C_{t_i,j} \subseteq O_{t_i}$ *at time* t_i*, with* $j \in \{1, ..., k_{t_i}\}$ *with* k_{t_i} *being the number of clusters at time* t_i*, is a set of similar data points, identified by a cluster algorithm.*

Definition 4 (Fuzzy Cluster Membership). *The membership degree* $u_{C_{t_i,j}}(o_{t_i,l}) \in [0, 1]$ *expresses the relative degree of belonging of the data object* $o_{t_i,l}$ *of time series* T_l *to cluster* $C_{t_i,j}$ *at time* t_i*.*

Definition 5 (Fuzzy Time Clustering). *A fuzzy time clustering is the result of a fuzzy clustering algorithm at one timestamp. In concrete it is the membership matrix* $U_{t_i} = [u_{C_{t_i,j}}(o_{t_i,l})]$*.*

Definition 6 (Fuzzy Clustering). *A fuzzy clustering of time series is the overall result of a fuzzy clustering algorithm for all timestamps. In concrete it is the ordered set* $\zeta = U_{t_1}, ..., U_{t_n}$ *of all membership matrices.*

4 Method

An obvious disadvantage of creating clusters for every timestamp is the missing temporal link. In our approach we assume that clusterings with different parameter settings show differences in the connectedness of clusters and that this connection can be measured. In order to do so, we make use of a stability function. Given a fuzzy clustering ζ, we first analyze the behavior of every subsequence of

a time series $T = o_{t_1}, ..., o_{t_i}$, with $t_i \leq t_n$, starting at the first timestamp. In this way we rate a temporal linkage of time series to each other. Time series that are clustered together at all time stamps, have a high temporal linkage, while time series which often separate from their clusters' peers, indicate a low temporal linkage. One could say we rate the *team spirit* of the individual time series and therefore their cohesion with other sequences over time. In the example shown in Fig. 2, the time series T_a and T_b show a good team spirit because they move together over the entire period of time. In contrast, the time series T_c and T_d show a lower temporal linkage. While they are clustered together at time points t_i and t_k, they are assigned to different clusters in between at time point t_j. After the evaluation of the individual sequences, we assign a score to the fuzzy clustering ζ, depending on the over-time stability of every time series.

Let U_{t_i} be a fuzzy partitioning of the data objects O_{t_i} of all times series in k_{t_i} clusters at time t_i. Similar to the equivalence relation in Hllermeier-Rifqi Index, we compute the relative assignment agreement of the data objects $o_{t_i,l}$ and $o_{t_i,s}$ of two time series T_l and T_s, $1 \leq l, s \leq m$ to all clusters in partitioning U_{t_i} at time t_i as follows

$$E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s}) = 1 - \frac{1}{2} \sum_{j=1}^{k_{t_i}} |u_{C_{t_i,j}}(o_{t_i,l}) - u_{C_{t_i,j}}(o_{t_i,s})|. \tag{5}$$

Having the relative assignment agreement of time series at timestamps t_i and t_r, $t_1 \leq t_i < t_r \leq t_n$, we calculate the difference between the relative assignment agreements of time series T_l and T_s by subtracting the relative assignment agreement values:

$$D_{t_i,t_r}(T_l, T_s) = |E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s}) - E_{U_{t_r}}(o_{t_r,l}, o_{t_r,s})|. \tag{6}$$

We calculate the stability of a time series T_l, $1 \leq l \leq m$, over all timestamps as an averaged weighted difference between the relative assignment agreements to all other time series as follows:

$$stability(T_l) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{r=i+1}^{n} \frac{\sum_{s=1}^{m} E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s})^m D_{t_i,t_r}(T_l, T_s)^2}{\sum_{s=1}^{m} E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s})^m}. \tag{7}$$

In Formula (7) we weight the difference between the assignment agreements $D_{t_i,t_r}(T_l, T_s)$ by the assignment agreement between pairs of time series at the earlier time point because we want to damp the large differences for stable time series caused by supervention of new peers. On the other hand we aim to penalize the time series that leave their cluster peers while changing cluster membership at a later time point.

Finally, we rate the over-time stability of a clustering ζ as the averaged stability of all time series in the data set:

$$FCSETS(\zeta) = \frac{1}{m} \sum_{l=1}^{m} stability(T_l). \tag{8}$$

As we already stated, the over-time stability of the entire clustering depends on the stability of all time series regarding staying together in a cluster with times series, that follow the same trend.

5 Experiments

In the following, we present the results on an artificially generated data set, that demonstrates a meaningful usage of our measure and shows the impact of the stability evaluation. Additionally, we discuss experiments on two real world data sets. One consists of financial figures from balance sheets and the other one contains country related economy data. In all cases fuzzy c-means was used with different parameter combinations for the number of clusters per time point.

5.1 Artificially Generated Data Set

In order to show the effects of a rating based on our stability measure, we generated an artificial data set with time series that move between two separated groups. Therefore, at first, three random centroids with two features $\in [0, 1]$ were placed for time point 1. These centroids were randomly shifted for the next timestamps whereby the maximal distance of a centroid at two consecutive time points could not exceed 0.05 per dimension. Afterwards 3, 4 and 5 time series were assigned to these centroids, respectively. This means that the data points of a time series for each time point were placed next to the assigned centroid with a maximal distance of 0.1 per feature. Subsequently, sequences with random transitions between two of the three clusters were inserted. Therefore 3 time series (namely 1, 2 and 3) were generated, that were randomly assigned to one of the two clusters at every time point. All together, a total of 4 time points and 15 time series were examined.

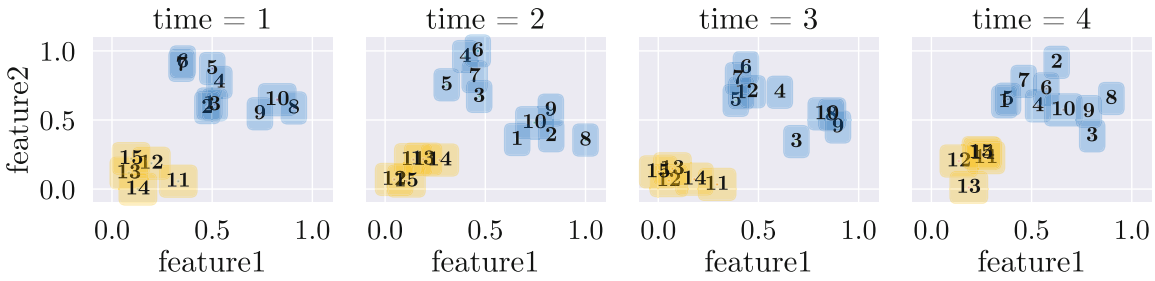


Fig. 3. Result of the most stable clustering on the artificially generated data set. (Color figure online)

To find the best stability score for the data set, FCM was used with various settings for the number of clusters per time point. All combinations with $k_{t_i} \in [2, 5]$ were investigated. Figure 3 shows the resulting fuzzy clustering with the highest FCSETS score of 0.995. For illustration reasons the clustering was

Table 1. Stability scores for the generated data set depending on k_{t_i}.

k_{t_1}	k_{t_2}	k_{t_3}	k_{t_4}	FCSETS score
2	2	2	2	**0.995**
2	3	2	2	0.951
2	3	3	2	0.876
2	3	3	3	0.829
3	3	2	2	0.967
3	3	3	3	0.9
2	3	4	5	0.71
5	3	4	2	0.908
3	10	3	10	0.577

defuzzyfied. Although it might seem intuitive to use a partitioning with three clusters at time points 1 and 2, regarding the over-time stability it is beneficial to choose only two clusters. This can be explained by the fact that there are time series that move between the two apparent groups of the upper (blue) cluster. The stability is therefore higher when these two groups are clustered together.

In Table 1 a part of the corresponding scores for the different parameter settings of k_{t_i} are listed. As shown in Fig. 3, the best score is achieved with k_{t_i} being set to 2 for all time points. The worst score results with the setting $k_{t_1} = 2$, $k_{t_2} = 3$, $k_{t_3} = 4$ and $k_{t_4} = 5$. The score is not only decreased because the upper (blue) cluster is divided in this case, but also because the number of clusters varies and therefore sequences get separated from their peers. It is obvious that the stability score is negatively affected, if the number of clusters significantly changes over time. This influence is also expressed by the score of 0.577 for the extreme example in the last row.

5.2 EIKON Financial Data Set

The first data set was released by Refinitiv (formerly Thomson Reuters Financial & Risk) and is called *EIKON*. The database contains structured financial data of thousands of companies for more than the past 20 years. For the ease of demonstration two features and 23 companies were chosen randomly for the experiment. The selected features are named as *TR-NetSales* and *TR-TtlPlanExpectedReturn* by Thomson Reuters and correspond to the net sales and the total plan expected return, which are figures taken from the balance sheet of the companies. Since it is a common procedure in economics, we divided the features by the company's total assets and normalized them afterwards with a min-max-normalization.

We generated the clusterings for all combinations of k_{t_i} from two to five clusters per timestamp. Selected results can be seen in Table 2. The actual maximum retrieved from the iterations (in the third row) is printed bold. The worst score can be found in the last row and represents an unstable clustering. It can be seen

Table 2. Stability scores for the EIKON financial data set depending on k_{t_i}.

k_{t_1}	k_{t_2}	k_{t_3}	k_{t_4}	k_{t_5}	k_{t_6}	k_{t_7}	k_{t_8}	FCSETS score
2	2	2	2	2	2	2	2	0.929
3	3	3	3	3	3	3	3	0.9
3	2	2	2	2	2	2	2	**0.945**
5	4	3	2	2	2	2	2	0.924
2	2	4	3	2	4	5	5	0.72

that the underlying data is well separated into three clusters in the first point in time and into two clusters at the following timestamps. This is actually a rare case but can be explained with the selection of features and companies. Actually *TR-TtlPlanExpectedReturn* is rarely provided by Thomson Reuters and the fact that we only chose companies which got complete data for all regarded points in time. This may have diminished the number of companies which might have lower membership degrees.

5.3 GlobalEconomy Data Set

The next data set originates from www.theglobaleconomy.com [1], which is a website that provides economic data of the past years for different countries. Again, two features were selected randomly for this experiment and were normalized with a min-max-normalization. Namely the features are the "Unemployment Rate" and the "Public spending on education, percent of GDP". For illustration reasons, we considered only a part of the countries (28) for the years from 2010 to 2017.

Table 3. Stability scores for the GlobalEconomy data set depending on k_{t_i}.

k_{t_1}	k_{t_2}	k_{t_3}	k_{t_4}	k_{t_5}	k_{t_6}	k_{t_7}	k_{t_8}	FCSETS score
2	2	2	2	2	2	2	2	**0.978**
3	3	3	3	3	3	3	3	0.963
3	2	2	2	2	2	2	2	0.945
5	3	4	2	2	2	2	2	0.955
2	3	2	2	4	5	5	5	0.837

The results are shown in Table 3. It can be seen that the best score is achieved with two clusters at every point in time. Evidently the chosen countries can be well separated into two groups at every point in time. More clusters or different numbers of clusters for different timestamps performed worse. In this experiment we also iterated over all combinations of k_{t_i} for the given points in time. The bold printed maximum, and the minimum, which can be found in the last row

of the table, represent the actual maximum and minimum within the range of the iterated combinations.

6 Conclusion and Future Work

In this paper we presented a new method for analyzing multiple multivariate time series with the help of fuzzy clustering per timestamp. Our approach defines a new target function for sequence-based clustering tasks, namely the stability of sequences. In our experiments we have shown that this enables the identification of optimal k_{t_i}s per timestamp and that our measure can not only rate time series and clusterings but also can be used to evaluate the stability of data sets. The latter is possible by examining the maximum achieved *FCSETS* score. Our approach can be applied whenever similar behavior for groups of time series can be assumed. As it is based on membership degrees, clusterings with overlapping clusters and soft transitions can be handled. With the help of our evaluation measure a stable over-time clustering can be achieved, which can be used for further analysis such as outlier detection.

Future work could include the development of a fuzzy clustering algorithm which is based on our formulated target function. The temporal linkage could therefore already be taken into account when determining groups of time series. Another interesting field of research could be the examination of other fuzzy clustering algorithms like the Possibilistic Fuzzy c-Means algorithm [27]. This algorithm can also handle outliers which can be handy for certain data sets. In the experiment with the GlobalEconomy data set we faced the problem, that one outlier would form a cluster on its own in every point in time. This led to very high FCSETS scores. The handling of outliers could overcome such misbehavior. Future work should also include the application of our approach to incomplete data, since appropriate fuzzy clustering approaches already exist [15,16,33]. We have faced this problem when applying our algorithm to the EIKON financial data set. Also, the identification of time series that show a good team spirit for a specific time period could be useful in some applications and might therefore be investigated. Finally, the examination and optimization of FCSETS' computational complexity would be of great interest as it currently seems to be fairly high.

Acknowledgement. We thank the J¨rgen Manchot Foundation, which supported this work by funding the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University D¨sseldorf.

References

1. Global economy, world economy. https://www.theglobaleconomy.com/
2. Banerjee, A., Ghosh, J.: Clickstream clustering using weighted longest common subsequences. In: Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining, pp. 33–40 (2001)

3. Beringer, J., Hüllermeier, E.: Adaptive optimization of the number of clusters in fuzzy clustering. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 1–6 (2007)
4. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
5. Bouguessa, M., Wang, S., Sun, H.: An objective approach to cluster validation. Pattern Recogn. Lett. **27**, 1419–1430 (2006)
6. Campello, R.: A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. Pattern Recogn. Lett. **28**(7), 833–841 (2007)
7. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybern. **3**(3), 32–57 (1973)
8. Ernst, J., Nau, G.J., Bar-Joseph, Z.: Clustering short time series gene expression data. Bioinformatics **21**(suppl-1), i159–i168 (2005)
9. Ferreira, L.N., Zhao, L.: Time series clustering via community detection in networks. Inf. Sci. **326**, 227–242 (2016)
10. Frigui, H., Hwang, C., Rhee, F.C.H.: Clustering and aggregation of relational data with applications to image database categorization. Pattern Recogn. **40**(11), 3053–3068 (2007)
11. Fukuyama, Y., Sugeno, M.: A new method of choosing the number of clusters for the fuzzy c-mean method. In: Proceedings of the 5th Fuzzy Systems Symposium, pp. 247–250 (1989)
12. Granell, C., Darst, R., Arenas, A., Fortunato, S., Gomez, S.: Benchmark model to assess community structure in evolving networks. Phys. Rev. E **92**, 012805 (2015)
13. Greene, D., Doyle, D., Cunningham, P.: Tracking the evolution of communities in dynamic social networks. In: Proceedings - 2010 International Conference on Advances in Social Network Analysis and Mining, ASONAM 2010, vol. 2010, pp. 176–183 (2010)
14. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: theory and practice. IEEE Trans. Knowl. Data Eng. **15**(3), 515–528 (2003)
15. Hathaway, R., Bezdek, J.: Fuzzy c-means clustering of incomplete data. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **31**, 735–44 (2001)
16. Himmelspach, L., Conrad, S.: Fuzzy *c*-means clustering of incomplete data using dimension-wise fuzzy variances of clusters. In: Carvalho, J.P., Lesot, M.-J., Kaymak, U., Vieira, S., Bouchon-Meunier, B., Yager, R.R. (eds.) IPMU 2016. CCIS, vol. 610, pp. 699–710. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40596-4 58
17. Huang, X., Ye, Y., Xiong, L., Lau, R.Y., Jiang, N., Wang, S.: Time series k-means: a new k-means type smooth subspace clustering for time series data. Inf. Sci. **367–368**, 1–13 (2016)
18. Hüllermeier, E., Rifqi, M.: A fuzzy variant of the rand index for comparing clustering structures. In: Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference, pp. 1294–1298 (2009)
19. Izakian, H., Pedrycz, W., Jamal, I.: Fuzzy clustering of time series data using dynamic time warping distance. Eng. Appl. Artif. Intell. **39**, 235–244 (2015)
20. Kim, Y.I., Kim, D.W., Lee, D., Lee, K.: A cluster validation index for GK cluster analysis based on relative degree of sharing. Inf. Sci. **168**, 225–242 (2004)

21. Le Capitaine, H., Frelicot, C.: A cluster-validity index combining an overlap measure and a separation measure based on fuzzy-aggregation operators. IEEE Trans. Fuzzy Syst. **19**, 580–588 (2011)
22. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
23. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
24. Möller-Levet, C.S., Klawonn, F., Cho, K.-H., Wolkenhauer, O.: Fuzzy clustering of short time-series and unevenly distributed sampling points. In: R. Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 330–340. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45231-7 31
25. O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: Streaming-data algorithms for high-quality clustering. In: Proceedings of IEEE International Conference on Data Engineering, p. 685 (2001)
26. Orlinski, M., Filer, N.: The rise and fall of spatio-temporal clusters in mobile ad hoc networks. Ad Hoc Netw. **11**(5), 1641–1654 (2013)
27. Pal, N., Pal, K., Keller, J., Bezdek, J.: A possibilistic fuzzy c-means clustering algorithm. IEEE Trans. Fuzzy Syst. **13**, 517–530 (2005)
28. Paparrizos, J., Gravano, L.: k-shape: efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD 2015, pp. 1855–1870. ACM, New York (2015)
29. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971)
30. Roth, V., Lange, T., Braun, M., Buhmann, J.: A resampling approach to cluster validation. In: Härdle, W., Rüz, B. (eds.) COMPSTAT, pp. 123–128. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-642-57489-4 13
31. Runkler, T.A.: Comparing partitions by subset similarities. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS (LNAI), vol. 6178, pp. 29–38. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14049-5 4
32. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In: Le, T.D., et al. (eds.) AusDM 2019. CCIS, vol. 1127, pp. 91–103. Springer, Singapore (2019). https://doi.org/10.1007/978-981-15-1699-3 8
33. Timm, H., Döring, C., Kruse, R.: Different approaches to fuzzy clustering of incomplete datasets. Int. J. Approx. Reason. **35**, 239–249 (2004)
34. Truong, C.D., Anh, D.T.: A novel clustering-based method for time series motif discovery under time warping measure. Int. J. Data Sci. Anal. **4**(2), 113–126 (2017). https://doi.org/10.1007/s41060-017-0060-3
35. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **13**(8), 841–847 (1991)
36. Zakrzewska, A., Bader, D.: A dynamic algorithm for local community detection in graphs. In: 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 559–564 (2015)

2.3 The Coexistence of FCSETS and CLOSE

In the previous two sections, we presented two methods for evaluating time series clusters. The aim of these methods is to evaluate existing clusterings and thus provide information about their validity. The comparison of clusterings can also be used to select a specific clustering algorithm or a parameter pair for it. Furthermore, these approaches build the basis for a variety of new methods, some of which we will discuss in the next chapter.

Similar to the coexistence of fuzzy and crisp clustering algorithms, we have also decided to present two methods. This decision is based on a variety of reasons, which are discussed in this chapter.

The exact course of a time series is (not) important.
The comparison of time series with the help of their cluster membership per time point, as postulated by us, does not consider the exact course of a time series. However, this statement does not apply without restrictions. In both, the crisp and the fuzzy case, the course of a time series and the position of the time series within the cluster has an effect on the evaluation result of both methods. However, this effect is weighted differently in the evaluation process of the two methods.

In the case of CLOSE, especially those time series have an influence on the overall result which are on the edge to other clusters over time. CLOSE would prefer a clustering that takes these time series into account and assigns clusters with the same composition at as many time points as possible. Although the influence is rather small, especially with larger data sets, it is nevertheless present.

Stable time series at the edge of clusters have no relevant influence on the result of the FCSETS evaluation. In such a case, the evaluation would not be subject to any negative influence because the development of the degrees of membership is constant. The situation is different with time series whose degrees of membership change significantly. In this case, it is even irrelevant if the maximum degree of membership at each time point is assigned to the cluster that would be best from the crisp point of view.

The choice of cluster centres thus has a greater influence in the fuzzy case than it does in the crisp case.

The quality of the clusters is (not) considered.
In the case of CLOSE, we have decided to introduce a quality function for the evaluation of the clusters per time point. This function is intended to ensure that the temporal aspect does not get out of hand in the calculation of CLOSE and produce clusters that have little or no significance in the time point itself. In FCSETS, we have dispensed with the evaluation. The reason for this is the already mentioned evaluation of the exact course of a time series, which corresponds to an indirect evaluation of a cluster centre or a cluster.

It should also be noted that due to the introduction of cluster quality in CLOSE, the calculation of the CLOSE score had to be solved via the *over-time stability* of the clusters. The indirect analysis of cluster quality in FCSETS, however, allows the calculation of the FCSETS score using the *over-time stability* of sequences only. For this reason, FCSETS does not include a stability score for clusters.

There are various ways to implement an *over-time stability* for clusters in FCSETS, but it is not of primary importance for the evaluation of clusterings and has therefore not been mentioned.

Is CLOSE (not) the crisp variant of FCSETS?
For the reasons mentioned above, a direct conversion of FCSETS into a crisp variant is difficult. It can certainly be said that CLOSE is one possible variant of FCSETS, but it is definitely not the only one. Both methods are adapted to the underlying logic and therefore have their raison d'être. Their complementary coexistence makes sense, similar to the case of fuzzy and crisp clustering algorithms, and offers the user a solution adapted to his application.

3

Applications

In the previous chapter, we introduced the *over-time stability* for time series, clusters and clusterings. In this section we describe two applications that make use of these definitions. First, we describe three variants of outlier detection in time series databases. The procedures are set up in such a way that partial sequences or entire time series can be recognised as outliers. The identified outliers represent a new type of outliers that has not been described before, as it is based on the clustering of time series.

In the papers referenced in this chapter [R1, R5, R8, R9] , we assume that the composition of the clusters to which a time series was assigned is decisive. If a time series changes its cluster peers more frequently, it is interpreted as a conspicuous feature. We refer to the migration of a time series into different clusters as the behaviour of time series.

The three methods in Section 3.1, Section 3.2 and Section 3.3 for detecting outlier sequences are all based on the idea of the behaviour of a time series, but they differ in several respects. The procedure in Section 3.1 uses the asymmetric comparison of clusters, while the procedure in Section 3.2 introduces a symmetric comparison using the Jaccard index. We call the result of this comparison the *proportion* of time series that occur together in two clusters from different points in time. In addition, we introduce different comparison metrics, which are oriented either to the most stable time series of a cluster or to the normal distribution. The procedure from section 3.3 shows a different approach based on the number of time series that have migrated together.

In section 3.4 we present a procedure for clustering time series. Unlike the outlier detection methods, the clustering in Section 3.4 is not based on the behaviour of time series, instead we introduce the concept of *time series adaptability*, which is intended to express how well a time series adapts to other time series.

3.1 Show Me Your Friends and I'll Tell You Who You Are. Finding Anomalous Time Series by Conspicuous Cluster Transitions

---

Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. Show Me Your Friends and I'll Tell You Who You Are. Finding Anomalous Time Series by Conspicuous Cluster Transitions. In *Data Mining*, Communications in Computer and Information Science, pages 91–103. Springer Singapore, Singapore, 2019.

Contributions: The presented method was developed jointly by Martha Krakowski (née Tatusch) and Gerhard Klassen. Martha Krakowski (née Tatusch) has formalised the idea mathematically and implemented the outlier detection algorithm. Gerhard Klassen was responsible for preprocessing and the experiments on the real world data. The manuscript was written in equal parts by the two main authors under the supervision of Jun.-Prof. Dr. Marcus Bravidor and Prof. Dr. Stefan Conrad.

Status: published

In this section we describe a first procedure for outlier detection. As noted earlier, this procedure is based on the stability measure introduced with the ***C**luster **O**ver-**T**ime **S**tability **E**valuation*.

As described in the previous chapter, a low *over-time stability* of a time series indicates a frequent change of cluster peers. The comparison of the *over-time stability* of time series provides a reference, which can be used in order to detect anomalous sequences.

In the work referred to in this section [R5], we assume that the expected *behaviour* of a time series is based on its previous *behaviour*. That means, if a time series is grouped with certain peers in the past, we expect it to be grouped with the same peers in the future, we refer to this as a *normal behaviour*. If a time series deviates from its normal *behaviour*, we consider this time series to be an outlier. In [R5] we consider the relation of a time series to the most stable time series of a given cluster. This time series can also be called the leader of the group and sets a kind of standard for the other time series in the respective cluster.

The title of the work refers to clusters whose time series are interpreted as *friends*. The evaluation of these *friends* reveals further information about the *over-time stability* of specific time series. The metaphor used includes the assumption that *good friends* stick together over time, while a *bad friend* (e.g. outlier) changes his peers.

In the next sections, we also describe further ways of interpreting the *over-time stability* of a time series in relation to other time series [R8, R9]. Our experiments in [R5] show the variety of possible applications and illustrate potential uses for real-world data. An implementation for fuzzy clustering is relatively simple to realise. It only requires the replacement of the *subsequence_score* with the analogous definition from the ***F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries*, as described in Section 2.2.

Show Me Your Friends and I'll Tell You Who You Are. Finding Anomalous Time Series by Conspicuous Cluster Transitions

Martha Tatusch(✉), Gerhard Klassen, Marcus Bravidor, and Stefan Conrad

Heinrich Heine University, Universitätsstr. 1, 40225 Düsseldorf, Germany
{tatusch,klassen,bravidor,conrad}@hhu.de

Abstract. The analysis of time series is an important field of research in data mining. This includes different sub areas like trend analysis, outlier detection, forecasting or simply the comparison of multiple time series. Clustering is also an equally important and vast field in time series analysis. Different clustering algorithms provide different analysis aspects like the detection of classes or outliers. There are various approaches how to apply cluster algorithms to time series. Previous work either extracted subsequences or feature sets as an input for cluster algorithms. A rarely used but important approach in clustering of time series is the grouping of data points per point in time. Based on this technique we present a method which analyses the transitions of time series between clusters over time. We evaluate our approach on multiple multivariate time series of different data sets. We discover conspicuous behaviors in relation to groups of sequences and provide a robust outlier detection algorithm.

Keywords: Outlier detection · Time series analysis · Clustering

1 Introduction

Time series data is collected in various domains. Not only the behavior of users on different platforms, but also the tracking of vehicles and objects or the recording of financial or weather data can be displayed as time series. For further analysis, the various data types can be converted into numerical (mostly discrete) values so that sequences of numerical vectors are derived. These can then be processed in a variety of ways. Information can be obtained through analyses such as clustering, prediction or comparison of time series and different outlier detection methods.

Depending on the context, different aspects can be relevant for the user. For example, not all clustering algorithms consider the same types of clusters, and outlier detection techniques do not always address the same types of outliers. In some cases, very special solutions have to be found for specific problems, whereby there are many algorithms that can be applied to a wide range of application areas.

T. D. Le et al. (Eds.): AusDM 2019, CCIS 1127, pp. 91–103, 2019.
https://doi.org/10.1007/978-981-15-1699-3_8

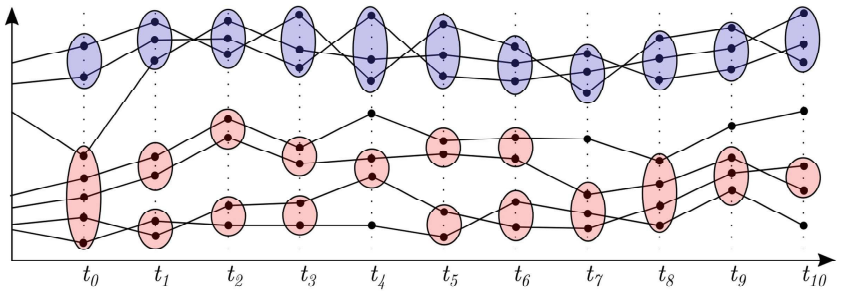


Fig. 1. Example for a time series over-time clustering. The blue color indicates stable clusters while red stands for instability. (Color figure online)

In this paper we focus on databases of multivariate time series with discrete values, same length and equivalent time steps. We detect anomalous subsequences with regard to groups of time series of the given database. Therefore we cluster the multivariate data of all time series per timestamp and analyze the stability of all subsequences over time. Thereby we call the resulting clustering *over-time clustering*. In Fig. 1 an example for such a clustering is displayed. For the sake of simplicity, only univariate time series are plotted. Since the data is clustered independently at each point in time, there is at first no time-related connection between the clusterings.

There are several proposals for clustering time series depending on the application. Some methods cluster the time series of a database as a whole [10,12,19], extract feature sets first [22], or consider subsequences of a single time series only [3]. However, these are not suitable when it comes to detecting irregularities or gathering information per time point.

Outlier detection in time series is in most cases not based on clustering. Because of various underlying data such as single or multiple time series with uni- or multivariate data points and different definitions of what an outlier is, there are several approaches to their identification. Some papers consider data points [1] or subsequences [15] that are anomalous with regard to a single time series [5,17], such as peaks. Others look for so called *change points* [6,16], that imply that the course of the considered time series significantly changes from that point on. Yet others analyse data from several time series that are very similar, such as sensor data, and detect irregularities in relation to the entire data set [1,11,13]. Finding these abnormalities usually presupposes that either the course of a single time series follows consistent patterns or that the courses of several time series are highly correlated.

In this paper we assume that the exact course of the individual time series is not important, but the trend which groups of sequences follow. By anomalies we denote subsequences that deviate from one trend and therefore cannot be assigned steadily to a group of sequences. In that case, we say that the sequence possesses a weak stability. We present an algorithm that identifies such unstable

sequences in a database of multivariate time series and is robust against missing data points.

2 Related Work

Anomaly detection in time series is a wide field of research. It can be distinguished in the detection of outliers within a single time series and the detection of outliers in multiple time series. Outliers in single time series are usually categorized in two classes:

Additive outliers, which represent surprisingly large or small values in a short period. In case additive outliers occur consecutively they are often summarized as additive outlier patches.

Innovational outliers are characterized by their impact on subsequent observations. Additionally the influence of innovational outliers can grow with time.

There are also several different categories of outliers, which can be described as a mix of both main classes. For example, additive outliers which cause a move of following observations to a new level are called *level shift outliers* and have a permanent impact on the ongoing time series. In case the influence of the level shift outlier is decreasing over time, it is called a *transient change outlier*. Additive outliers, which occur periodically are named *seasonal additive outliers*.

Additive and innovational outliers are often identified with extensions of autoregressive-moving-average (ARMA) models [2,18]. Other techniques include the use of decomposition methods such as STL, a seasonal-trend decomposition procedure based on LOESS [7]. Yet other methods evaluate derivatives of the dynamic time warping (DTW) [20] similarity in order to detect anomalies.

The detection of outliers in multiple time series is handled differently. Methods of this kind are often using the peers of a time series to determine whether it is anomalous or not. Beside other techniques, recent approaches use Probabilistic Suffix Trees (PST) [21] and Random Block Coordinate Descents (RBCD) [23] in order to detect outliers. However, while these approaches focus on the deviation of one time series to the others, we focus on the behaviour of a time series compared to its peers. More concretely, we assume that a time series which has a similar development to a group of other time series over a subsequence is expected to move on with the same group. Therefore we first cluster per point in time and then analyse the transition of time series regarding these clusters. This is realized by the analysis of cluster transitions of time series over time. Transitions of this kind are also analysed in cluster evolution methods. Landauer et al. [14] makes use of such a method in order to calculate an anomaly score for a single time series in a sliding window. In contrary to Landauer et al. we relate to multiple time series. The analysis of the time series behavior not only reveals new kinds of outliers but also detects different types of additive and innovational outliers.

This approach is very different from clustering whole time series or their subsequences, since the outlier detection would rely on the single fact whether a sequence is assigned to a cluster or not. Such an approach would not take

the cluster transitions of the time series into account, which can be an expressive feature on its own. Hence, our approach detects anomalous subsequences, although they would be assigned to a cluster in a subsequence clustering.

3 Fundamentals

In order to create a good basis of knowledge to avoid later misunderstandings, we will provide some definitions which our work is based on. As these terms are used in many different areas, it is useful to explain which interpretations are considered in this paper.

Definition 1 (Time Series). *A multivariate time series* $T = o_{t_1}, \ldots, o_{t_n}$ *is an ordered set of* n *real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording, with* t_1 *and* t_n *indicating the first and the last timestamp, respectively.*

Definition 2 (Data Set). *A data set* $D = T_1, \ldots, T_m$ *is a set of* m *time series of same length and equal points in time. The set of data points of all time series at a timestamp* t_i *is denoted as* O_{t_i}*.*

Definition 3 (Subsequence). *A subsequence* $T_{t_i,t_j,l} = o_{t_i,l}, \ldots, o_{t_j,l}$ *with* $j > i$ *is an ordered set of successive real valued data points beginning at time* t_i *and ending at* t_j *from time series* T_l*.*

Definition 4 (Cluster). *A cluster* $C_{t_i,j} \subseteq O_{t_i}$ *at time* t_i*, with* $j \in \{1, \ldots, q\}$ *being a unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm or human. This means that all clusters have distinct labels regardless of time.*

Definition 5 (Cluster Member). *A data point* $o_{t_i,l}$ *at time* t_i*, that is assigned to a cluster* $C_{t_i,j}$ *is called a member of cluster* $C_{t_i,j}$*.*

Definition 6 (Noise). *A data point* $o_{t_i,l}$ *at time* t_i *is considered as noise, if it is not assigned to any cluster.*

Definition 7 (Clustering). *A clustering is the overall result of a clustering algorithm or the set of all clusters annotated by a human for all timestamps. In concrete it is the set* $\zeta = \{C_{t_1,1}, \ldots, C_{t_n,q}\}$ *of all* q *clusters.*

In Fig. 2 an example for the above definitions can be seen. The data points of a data set containing five time series (T_a, T_b, T_c, T_d, T_e) are clustered for the timestamps t_i, t_j and t_k. For simplicity, all data points of a time series T_l are denoted by the identifier l.

In t_i the data points $o_{t_i,a}, o_{t_i,b}$ of time series T_a and T_b are cluster members of cluster $C_{t_i,l}$. The data point $o_{t_i,e}$ is marked as noise, as it is not assigned to any cluster in t_i. In total, the shown clustering consists of five clusters. It can be described by the set $\zeta = \{C_{t_i,l}, C_{t_i,u}, C_{t_j,v}, C_{t_j,f}, C_{t_k,g}\}$.

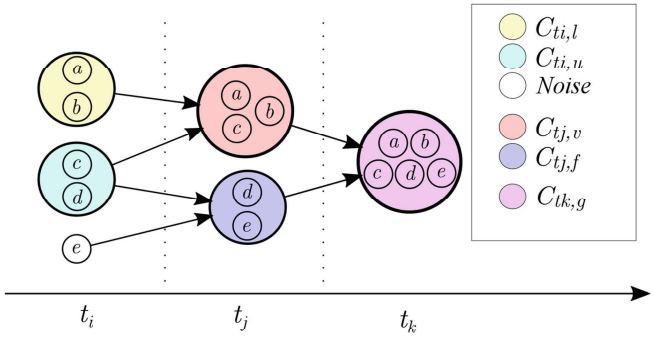


Fig. 2. Example for the transitions of time series $T_a, \ldots, T_e$ between clusters over time.

4 Method

After the clarification of important foundations, the basic idea of the algorithm is described. Therefore further terms have to be explained before.

Let $C_{t_i,a}$ and $C_{t_j,b}$ be two clusters, with $t_i, t_j \in \{t_1, \ldots t_n\}$. First, we introduce the term *temporal cluster intersection* for the purpose of measuring the stability of a time series:

$$\cap_t\{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \wedge o_{t_j,l} \in C_{t_j,b}\}$$

with $l \in \{1, \ldots, m\}$. The result is the set of time series that are assigned to both of the clusters under consideration. This means all sequences that were grouped together at time t_i and t_j. The transition of a time series from t_i to t_j can now be described by the proportion of cluster members from the corresponding cluster in t_i who migrated together into the cluster in t_j:

$$p(C_{t_i,a}, C_{t_j,b}) = \begin{cases} \emptyset & \text{if } C_{t_i,a} = \emptyset \\ \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a}|} & \text{else} \end{cases}$$

with $t_i < t_j$. In Fig. 2 an example for transitions of time series between clusters is sketched. There, the proportion for $C_{t_i,l}$ and $C_{t_j,v}$ would be

$$p(C_{t_i,l}, C_{t_j,v}) = \frac{|\{T_a, T_b\}|}{|\{o_{t_i,a}, o_{t_i,b}\}|} = \frac{2}{2} = 1.0.$$

This proportion can be used to measure the stability of a sequence with a *subsequence score*. It is defined as

$$subsequence\_score(T_{t_i,t_j,l}) = \frac{1}{k} \cdot \sum_{v=i}^{j-1} p(cid(o_{t_v,l}), cid(o_{t_j,l}))$$

with $l \in \{1, \ldots, m\}$, $k \in [1, j-i]$ being the number of timestamps between t_i and t_j where the data point exists and *cid*, the cluster-identity function

$$cid(o_{t_i,l}) = \begin{cases} \emptyset & \text{if the data point is not assigned to any cluster} \\ C_{t_i,a} & \text{else} \end{cases}$$

returning the cluster which the data point has been assigned to in t_i. The function returns an empty set, either if the object is classified as noise or if it does not exist at the considered time. Note, that the subsequence score is normalized to $[0, 1]$ by k, as the proportion p is a percentage between 0 and 1, as well.

In the example of Fig. 2, the score of time series T_a between time points t_i and t_k would be:

$$subsequence\_score(T_{t_i,t_k,a}) = \frac{1}{2} \cdot (1.0 + 1.0) = 1.0.$$

A notable characteristic is, that the score is always 0, if the last data point of the considered subsequence is marked as noise. However, this circumstance does not lead to any handicaps in most cases as all partial sequences of these subsequences are treated normally. Nevertheless, the handling of sequences with an endpoint that is labeled as noise will be analyzed in more detail later on.

For now describing the concrete procedure of detecting conspicuous sequences, we first provide a vague definition of them:

Definition 8 (Anomalous Subsequence). *A subsequence $T_{t_i,t_j,l}$ is called anomalous, if it is significantly more unstable than its cluster members at time t_j.*

With the help of the subsequence score which measures the stability of a subsequence, anomalous ones can now be distinguished by comparing the stability of grouped subsequences at a given time point. Every possible subsequence gets an outlier score indicating the probability of being anomalous, by calculating the deviation of its stability from the best subsequence score of its cluster. A formal description of the best subsequence score can be given by:

$$best\_score(t_i, C_{t_j,a}) = max(\{subsequence\_score(T_{t_i,t_j,l}) \mid cid(o_{t_j,l}) = C_{t_j,a}\})$$

The outlier score of a subsequence is then calculated as follows:

$$outlier\_score(T_{t_i,t_j,l}) = best\_score(t_i, cid(o_{t_j,l})) - subsequence\_score(T_{t_i,t_j,l})$$

As the best score lies between 0 and 1, an outlier score of 100% can only be achieved in completely stable clusters. The smaller the best score of the considered cluster is, the smaller is the greatest possible outlier score.

Regarding the example in Fig. 2, the time series T_d would get the following *outlier_score* between time points t_i and t_k:

$$outlier\_score(T_{t_i,t_k,d}) = 1.0 - (0.5 \cdot (0.5 + 1.0)) = 0.25$$

With the outlier score, now a more precise definition of an outlier can be given.

Definition 9 (Outlier). *Given a threshold $\tau \in [0,1]$, a subsequence $T_{t_i,t_j,l}$ is called an outlier, if its probability of being an outlier is greater than or equal τ. That means, if*

$$outlier\_score(T_{t_i,t_j,l}) \geq \tau.$$

Although τ is a constant, it can be interpreted as a dynamic threshold. That is, because the greatest possible deviation from the best subsequence score – and thus the greatest outlier score – depends on the best score of the considered cluster. Clusters with low stability have a lower probability of containing an outlier than stable ones, since all their cluster members show irregularities and that represents a pattern of instability. In this context, the small subsequence score is thus not conspicuous.

Intuitive outliers from the over-time clustering that were marked as noise get a special treatment. Subsequences that consist entirely of noise data points are automatically identified as outliers. Since subsequences whose last data point is labeled as noise are not assigned to a cluster from which the best score can be calculated, no outlier score can be determined for them. Therefore, they are not included in the regular outlier calculation. In the following we will differentiate between *anomalous subsequences*, *intuitive outliers* and *noise*.

Take another look at the case where the last element of an examined subsequence $T_{t_i,t_j,l}$ is marked as noise. Suppose the subsequence $T_{t_i,t_{j-1},l}$ gets a high outlier score and is detected as outlier. Then one would expect that the subsequence under consideration $T_{t_i,t_j,l}$ would be identified as an outlier as well. This will only be the case, if the previous data point was categorized as noise as well and the sequence was therefore recognized as an intuitive outlier. However, for the sequence $T_{t_i,t_k,l}$ with $k > j$, which at the last time point t_k is assigned to a cluster again for the first time this would also be the case. Thus in the end $T_{t_i,t_j,l}$ would be covered.

Yet a marginal case is when a data point is labeled as noise at the last time of the entire time series. In this scenario, a sequence with end time t_m would never be detected as an outlier if it is not marked as noise in t_{m-1}.

Remark 1 (Stability). The stability is not only influenced significantly by a small sample size when considering constant data points [4]. When examining the over-time stability, a small sample size leads to high sensitivity to cluster transitions, as well. As more data points are considered, the simpler it is to draw meaningful conclusions about the stability.

5 Experiments

In order to evaluate the presented method, we performed several experiments on different real world data. We also present two artificially generated data sets which are used to illustrate the handling of some marginal cases. In order to cluster the data per point in time, we used DBSCAN [9] with adapted parameters.

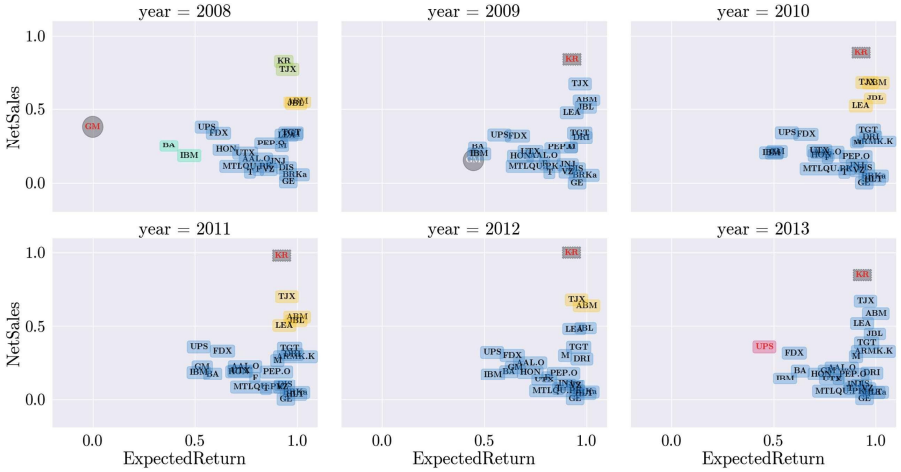


Fig. 3. Two dimensional experiment on the EIKON Financial Data Set with $\tau = 0.6$, $minPts = 2$ and $\epsilon = 0.15$. The colors indicate cluster belongings, whereby grey objects represent outliers. Circles are outliers by distance and boxes are intuitive outliers, as well. Red color or font indicates noise. (Color figure online)

5.1 EIKON Financial Data Set

Eikon is a set of software products released by Refinitiv (formerly Thomson Reuters Financial & Risk). It contains a database with financial data of thousands of companies for the past decades. For illustration reasons we randomly selected thirty companies and two features. The selected features are a figures which were taken from the balance sheet of the company. In economics it is common to normalize these figures by the companies' total assets in order to make it comparable to other companies. Beside this, we normalized the features with a min-max normalization. The clustering was done with DBSCAN and $\epsilon = 0.15$, $minPts = 2$ as parameters. The outlier detection parameter was chosen to be $\tau = 0.6$. In Fig. 3 one can see the illustrated results. The presented technique found two outlier subsequences. The first, which is labeled as GM is detected from the year 2008 until 2009. This is because GM is noise in the year 2008, which leads to a subsequence score of 0. In 2009 GM merges with a cluster, which has a high reference score. The second outlier detected is the subsequence $T_{t_{2009},t_{2013},KR}$. It is detected as an intuitive outlier.

5.2 Airline On-Time Performance Data Set

The Airline on-time performance data set [8] was originally collected by the U.S. Department of Transportation's Bureau of Transportation Statistics. It contains records of 3.5 million flights. Every flight has a set of 29 features, such as the departure delay, the delay reason, the arrival delay and the airline which processed the flight. In order to detect anomalies in this data set, we constructed

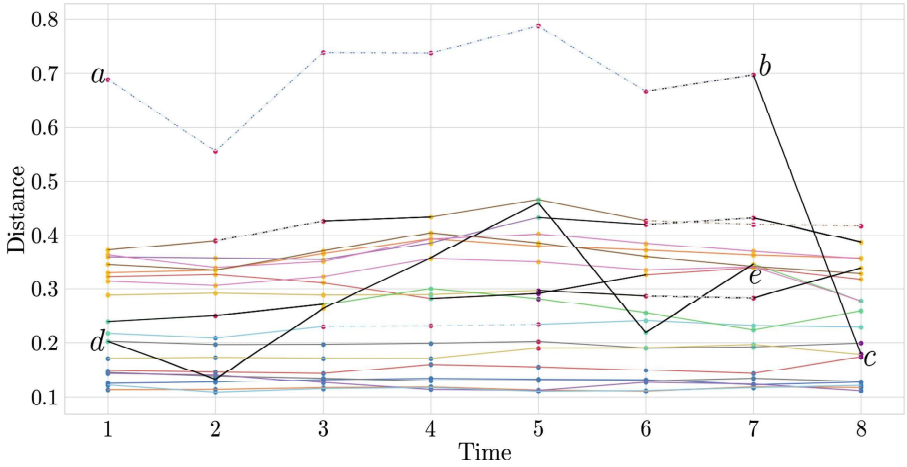


Fig. 4. One dimensional experiment on the Airline On-Time Performance Data Set with $\tau = 0.4$, $minPts = 3$ and $\epsilon = 0.03$. Black sequences represent anomalies, while white dashed ones stand for intuitive outliers. The color of the dots emphasize which cluster the data points are assigned to. Red dots represent noise. (Color figure online)

a time series for every airline by calculating the average of their features for every day. Before applying our technique, we normalized the data with the min-max normalization and clustered it with DBSCAN. Every observation represents a flight of an airline. In order to illustrate the results we executed our algorithm to one feature, namely the flight distance. We applied DBSCAN for eight time points with the following parameters: $minPts = 3$ and $\epsilon = 0.03$. Additionally we chose $\tau = 0.4$. The result can be seen in Fig. 4.

The figure shows two kinds of outliers: Intuitive outliers and outliers which were identified by their distance to a reference time series. Since the time series which is labeled with the points a, b and c has a large distance to other time series it is detected as an intuitive outlier from a to b. Due to this, the time series' accumulated subsequence score is zero and thus it is also detected as an outlier at the last time stamp c. From point a to b it is not detected as an outlier by it's distance to the reference subsequence score, since the neighborhood of the sequence at time point 8 have also a low stability score. Regarding the time points 1 to 8 and the objects in the neighborhood, there are at most two peers which remained together. The subsequence labeled with d and e is a good example for the presented method. It illustrates the detection of outliers by the change of cluster neighbors of the subsequence.

5.3 Simulated Data

In order to test our method in a targeted manner, two experiments were performed on simulated data. Both a univariate and a multivariate data set with two features are considered. In both cases, a time span of 8 time points is examined.

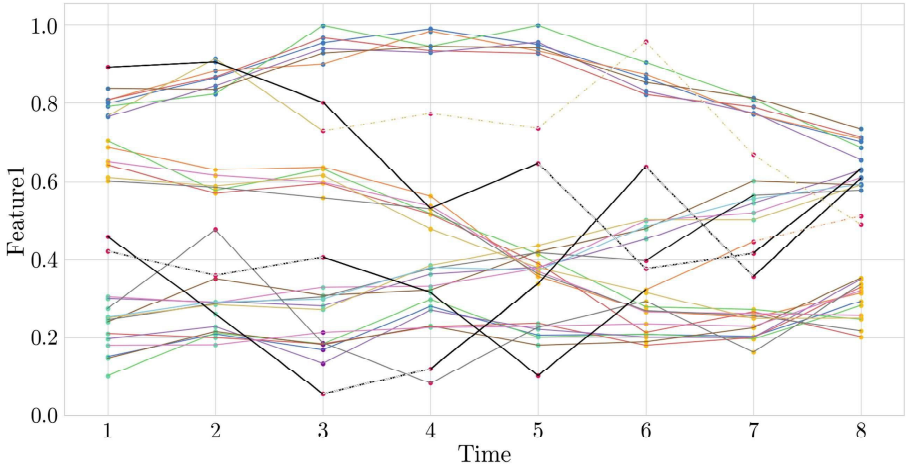


Fig. 5. Illustration of the detected outliers on the simulated one-dimensional data set with $\tau = 0.55$, $minPts = 3$ and $\epsilon = 0.05$. Black sequences represent anomalous subsequences, while white dashed ones stand for intuitive outliers. The color of the dots emphasize which cluster the data points are assigned to. Red dots represent noise. (Color figure online)

The one-dimensional data set was generated so that initially four starting points (for four groups) were selected. In addition, the maximum deviation from the centroid and the number of members were chosen for each group. The centroids were then calculated randomly for each time point, whereby the distance of the centroids of a cluster of two successive time points could not exceed 0.06. After generating the normal data points, 5 outlier sequences were randomly inserted. The starting points were chosen randomly and the distance between two consecutive points could not be greater than 0.3. For all points, care was taken to ensure that they were between 0 and 1.

As shown in Fig. 5, anomalous sequences from five time series have been found. Regarding the first time stamp the first and second black line show time series that are entirely recognized as conspicuous ones. Since their data points often switch between being noise (red dots) and different cluster members, this result is meaningful. Between time point 6 and 7 one additional black line in added. This can be explained by the stability of the sequence's cluster at time 7. All its cluster members migrate together from time point 6 to 7, so that an outlier is very conspicuous.

Looking at the completely randomly generated time series with the uppermost noise point at time 2, it is noticeable that it was not recognized by our algorithm. This is due to the fact that the purple cluster at time 3 and the turquoise cluster at time 5 do not have a high stability and the deviation of the sequence from the best possible score is therefore not very large. In the last time

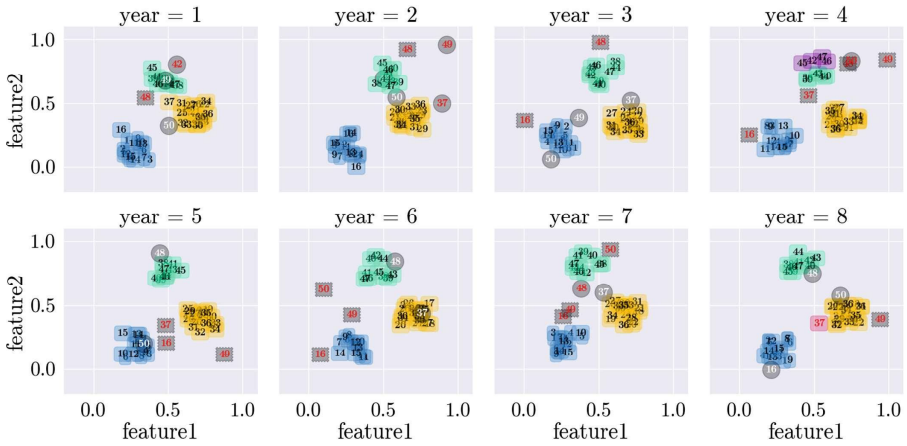


Fig. 6. Illustration of the detected outliers with $\tau = 0.5$, $minPts = 4$ and $\epsilon = 0.11$ on the artificially generated data two-dimensional set. The colors indicate cluster belongings, whereby grey objects represent outliers. Circles are outliers by distance and boxes are intuitive outliers, as well. Red color or font indicates noise. (Color figure online)

points, the time series migrates stably with the yellow cluster, so that it does not behave uncommonly.

If the data points of a time series change from one point in time to another from a cluster to noise, they are not initially interpreted as conspicuous. This is a problem if the time series remains as noise as the time at which it split from the cluster is not recognized as an intuitive outlier. This behavior can for example be seen in the striped line regarding the first time stamp. Between the times 6 and 7, the sequence was not detected as an outlier.

The second data set was created as follows: First, three starting points as centroids and the number of members of the three clusters were chosen. The maximum deviation of two consecutive centroids was set to 0.05 and that of the member data points to the centroid was set to 0.1. One time series was assigned to each group, which was allowed to deviate from the centroid by up to 0.25. Finally, two time series with completely random data points were added, so that a total of 5 outlier sequences should be noticeable. Here, too, we made sure that all data points are between 0 and 1 for each feature.

In Fig. 6 the results for an over-time clustering made by DBSCAN with $minPts = 4$ and $\epsilon = 0.11$ and an outlier threshold of $\tau = 0.5$ are shown. The time series 16, 37, 48 are generated with higher deviation and 49 and 50 completely random. It can be seen that all these time series were found by our algorithm as outliers (grey). Since the data points of these time series often are outliers as well as change their cluster members, this is a correct result. However, the first two time points are assumed to be normal for time series 16. This is desired too, as it moves stable with its cluster members at this time.

Although the data points of 42, 45, 46 and 47 split from their cluster members in time point 4, they are not identified as outliers. Since they migrate together and even merge back to their former cluster members in time point 5, their behavior is not conspicuous. The sequence 42 is identified as anomalous between time points 1 and 2 (turquoise cluster), since all its cluster members migrated completely stable from time point 1 to 2.

In total, the following outlier sequences can be read from Fig. 6: $T_{3,8,16}$, $T_{1,2,42}$, $T_{1,7,37}$, $T_{1,8,48}$, $T_{1,8,49}$, $T_{1,8,50}$. All are justified and correspond to the desired result. There is one striking observation, though: Although 37 is conspicuous over the entire period, it is only found as outlier between time 1 and 7. The reason for this is that the marginal case mentioned in Sect. 4 has occurred. Since the data point of the time series was classified as noise at the very last point in time, but not at the time before, the sequence is not found by our algorithm.

6 Conclusion and Future Work

In this work we presented a robust outlier detection algorithm for multiple multivariate time series. By analyzing the cluster transitions of time series over time, we are able to identify anomalous sequences. Instead of using sliding windows, our method performs an analysis of all possible subsequences. The shown results are sound and enable a new field of research. However, there are still some interesting aspects which may be examined in future work. The most important issue is the determination of the outlier detection parameter τ. We assume an interdependence of τ and hyperparameters that are used for the clustering algorithm. Further not all intuitive outlier sequences have to be conspicuous in regard to the time series database. Considering the deviation of time series can lead to an enhanced analysis of those. Finally, it could be useful to identify whole outlier clusters. Therefore a cluster score could be computed and evaluated.

References

1. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134–147 (2017)
2. Ahmar, A.S., et al.: Modeling data containing outliers using ARIMA additive outlier (ARIMA-AO). J. Phys: Conf. Ser. **954**, 012010 (2018)
3. Banerjee, A., Ghosh, J.: Clickstream clustering using weighted longest common subsequences. In: Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining, pp. 33–40 (2001)
4. Ben-David, S., Von Luxburg, U.: Relating clustering stability to properties of cluster boundaries. In: 21st Annual Conference on Learning Theory (COLT 2008), pp. 379–390 (2008)
5. Cheng, H., Tan, P.N., Potter, C., Klooster, S.: Detection and characterization of anomalies in multivariate time series. In: Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 413–424 (2009)

6. Cho, H., Fryzlewicz, P.: Multiple change-point detection for high-dimensional time series via sparsified binary segmentation (2014)
7. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: STL: a seasonal-trend decomposition procedure based on loess (with discussion). J. Official Stat. **6**, 3–73 (1990)
8. ASA Statistics Computing and Graphics: Airline on-time performance. http://stat-computing.org/dataexpo/2009/the-data.html. Accessed 15 July 2019
9. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
10. Ferreira, L.N., Zhao, L.: Time series clustering via community detection in networks. Inf. Sci. **326**, 227–242 (2016)
11. Hill, D.J., Minsker, B.S.: Anomaly detection in streaming environmental sensor data: a data-driven modeling approach. Environ. Model Softw. **25**(9), 1014–1022 (2010)
12. Huang, X., Ye, Y., Xiong, L., Lau, R.Y., Jiang, N., Wang, S.: Time series k-means: a new k-means type smooth subspace clustering for time series data. Inf. Sci. **367–368**, 1–13 (2016)
13. Keogh, E., Lonardi, S., Chiu, B.Y.C.: Finding surprising patterns in a time series database in linear time and space. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 550–556 (2002)
14. Landauer, M., Wurzenberger, M., Skopik, F., Settanni, G., Filzmoser, P.: Time series analysis: unsupervised anomaly detection beyond outlier detection. In: Su, C., Kikuchi, H. (eds.) ISPEC 2018. LNCS, vol. 11125, pp. 19–36. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99807-7_2
15. Lin, J., Keogh, E., Fu, A., Van Herle, H.: Approximations to magic: finding unusual medical time series. In: 18th IEEE Symposium on Computer-Based Medical Systems (CBMS 2005), pp. 329–334 (2005)
16. Liu, S., Yamada, M., Collier, N., Sugiyama, M.: Change-point detection in time-series data by relative density-ratio estimation. Neural Netw. **43**, 72–83 (2013)
17. Malhotra, P., Vig, L., Shroff, G.M., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: ESANN (2015)
18. Munir, M., Siddiqui, S.A., Chattha, M.A., Dengel, A., Ahmed, S.: FuseAD: unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. Sensors **19**(11), 2451 (2019)
19. Paparrizos, J., Gravano, L.: k-shape: efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1855–1870 (2015)
20. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (2007)
21. Sun, P., Chawla, S., Arunasalam, B.: Mining for outliers in sequential databases. In: ICDM, pp. 94–106 (2006)
22. Truong, C.D., Anh, D.T.: A novel clustering-based method for time series motif discovery under time warping measure. Int. J. Data Sci. Anal. **4**(2), 113–126 (2017)
23. Zhou, Y., Zou, H., Arghandeh, R., Gu, W., Spanos, C.J.: Non-parametric outliers detection in multiple time series a case study: power grid data analysis. In: AAAI (2018)

3.2 Behave or be detected! Identifying outlier sequences by their group cohesion

---

Martha Tatusch, Gerhard Klassen, and Stefan Conrad. Behave or Be Detected! Identifying Outlier Sequences by Their Group Cohesion. In *Big Data Analytics and Knowledge Discovery*, Lecture Notes in Computer Science, pages 333–347. Springer International Publishing, Cham, 2020.

Contributions: The idea and the conception of the presented paper were mainly developed and implemented by Martha Krakowski (née Tatusch) . Gerhard Klassen was responsible for obtaining and preprocessing the real world data sets and conducting the related experiments. The manuscript was prepared in equal parts by the two main authors under the supervision of Prof. Dr. Stefan Conrad.

Status: published

The work referenced in this section [R8] introduces some alternatives to the calculation of *over-time stability* and the calculation of the *outlier_score*. In particular, it should be noted that the *over-time stability* used so far is based on an asymmetric *proportion*. This *proportion* calculates the relative share of the same time series in two clusters. In the asymmetric case, this calculation is direction-dependent. The comparison of a cluster C_a at a time t with a cluster C_b from time $t+1$ is thus not identical with the comparison of C_b with C_a, if the clusters are not equal ($C_a \neq C_b$).

For some applications, however, symmetry can be advantageous. For this reason, we present the calculation of *over-time stability* using a *symmetric proportion* function. Moreover, we have recognised that users not always attach the same importance to the entire sequence. Instead, it is often perceived that the influence of recently passed points in time is significantly higher than that of long past ones. We used this insight to introduce a weighted *subsequence_score*, which allows the weighting of timestamps depending on their temporal distance to the considered time.

We evaluated and presented the combinations of adjustments with the help of some experiments. The results of the work [R8] are relevant to this dissertation, as the influence of symmetric and asymmetric proportions is demonstrated. The modifications to the outlier detection procedure can to a large extent also be adopted in the calculation of the ***C**luster **O**ver-**T**ime **S**tability **E**valuation*.

Behave or be detected! Identifying outlier sequences by their group cohesion

Martha Tatusch, Gerhard Klassen, and Stefan Conrad

Heinrich Heine University, Universitätsstr. 1, 40225 Düsseldorf, Germany
{tatusch,klassen,stefan.conrad}@hhu.de

Abstract. Since the amount of sequentially recorded data is constantly increasing, the analysis of time series (TS), and especially the identification of anomalous points and subsequences, is nowadays an important field of research. Many approaches consider only a single TS, but in some cases multiple sequences need to be investigated. In 2019 we presented a new method to detect behavior-based outliers in TS which analyses relations of sequences to their peers. Therefore we clustered data points of TS per timestamp and calculated distances between the resulting clusters of different points in time. We realized this by evaluating the number of peers a TS is moving with. We defined a stability measure for time series and subsequences, which is used to detect the outliers. Originally we considered cluster splits but did not take merges into account. In this work we present two major modifications to our previous work, namely the introduction of the jaccard index as a distance measure for clusters and a weighting function, which enables behavior-based outlier detection in larger TS. We evaluate our modifications separately and in conjunction on two real and one artificial data set. The adjustments lead to well reasoned and sound results, which are robust regarding larger TS.

Keywords: Outlier Detection · Time Series Analysis · Clustering.

1 Introduction

With increasing understanding about the value of data and the rising amount of connected sensors in the world of the IoT, more data is recorded every day than ever before. This enables a time aware analysis of the accumulated data by regarding it as time series. The time-driven data view not only allows the extraction of trends and seasons but also an interpretation of behavior. This is especially the case when several time series are considered at the same time. In our paper [20] we introduced an outlier detection algorithm based on the relative behavior of time series. As this was a novel approach we were aware of some drawbacks and application specific requirements. In concrete we noticed that earlier clusters had a high impact and that a cluster split would not be treated the same way as a cluster merge. While the latter is an application dependent circumstance the first causes a high dependence on early points in time, which is not wanted in most cases. In order to overcome these drawbacks we now introduce a weighting function and a new way of calculating the cluster

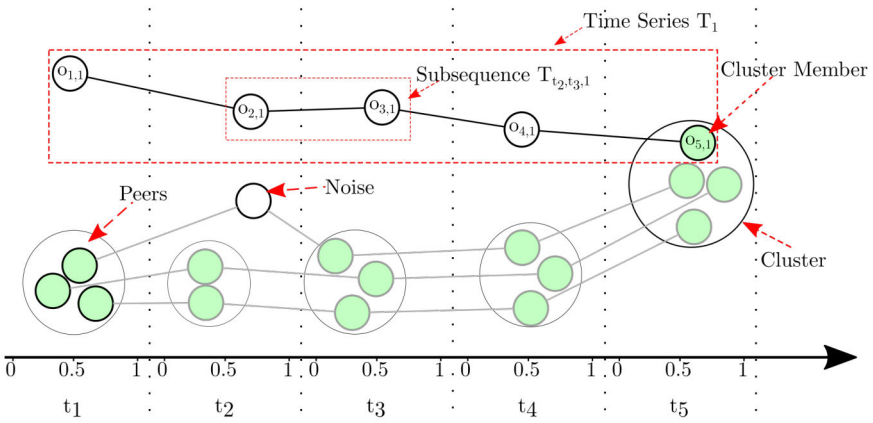


Fig. 1: Illustration of relevant terms regarding the time from t_1 to t_5.

proportions of two clusters. For this purpose we make use of the jaccard index which led to good results in [12] as well. Both extensions are tested separately and in conjunction. We highlight the differences and allow the user to choose carefully between those extensions - depending on his application.
Our original approach focuses on data sets with multivariate time series with discrete values, same length and equivalent time steps. Those time series are clustered per point in time and anomalous subsequences are detected by analysing the behavior of those. The behavior is defined as the change of peers over time. This leads us to the *subsequence_score* which represents the stability of a subsequence over time. An illustration of the relevant terms can be seen in Figure 1. With a calculated *outlier_score* for every subsequence and a threshold parameter τ we managed to detect anomalous time series. The outlier score depends on the subsequence score of a subsequence and the best subsequence score of a subsequence in the according cluster. In our work we differentiate three different types of outliers: *anomalous subsequences*, *intuitive outliers* and *noise*.
Our approach is different to other proposals which use cluster algorithms, as those either cluster time series as a whole [9] [11] [17], extract feature sets first [22], or consider subsequences of a single time series only [4]. None of the presented methods consider the cohesion of a time series regarding its peers. Our algorithm also differs from approaches which do not take time into account like [2] or which only regard subsequences of single time series [5] [15]. In contrary to those methods, we assume an information gain for one sequence from other sequences which have a semantic correlation.
In this paper we show once again, that we can identify an impact of other time series to one time series that is different to the granger causality [10] and that this influence can be used to detect anomalous subsequences. The adaptations of our original algorithm are well motivated and lead to different but sound results.

2 Related Work

Algorithms which detect outliers in time series are no novelty. There are actually various specialized approaches for different applications. Most methods deal with one time series only, while fewer ones regard multiple time series at the same time. There are different types of outliers, such as significantly deviating data points, uncommon subsequence patterns in periodic time series or changing points, which indicate that the further course of the sequence will change.
In many cases outliers of any type are identified with adapted autoregressive-moving-average (ARMA) models [3] [16]. Although these techniques are performing very well in most cases and factually are state-of-the-art, they lack the implementation of exterior information like other semantic correlated time series. There are also other methods which make use of decomposition techniques such as STL [6]. These methods work on time series which can be actually decomposed, but fail if this is not the case. Finally there are presented works which use dynamic time warping (DTW) [18] in order to detect anomalies.
There are also approaches which tackle the problem of finding outliers in multiple time series. Similar to our algorithm these methods are using peers of a time series to determine whether it is anomalous or not. The most recent works use Probabilistic Suffix Trees (PST) [19] or Random Block Coordinate Descents (RBCD) [23] in order to detect suspicious time series or subsequences. In contrary to our approach, in which the behavior of a time series is the central idea, the named methods analyse the deviation of one time series to the others. Our assumption that the change or the adherence of a time series to its peers is a crucial difference to all present methods. This behavior centered view is implemented by clustering time series per timestamp which is similar to identifying its peers per point in time. Then the movement of this time series relative to its peers is analysed. The result of this is described as a subsequence score, which also can be viewed as the stability over time of a time series regarding the adherence to its peers. The degree of change, also called transition, is an important factor to the subsequence score. It is also essential in cluster evolution methods such as [12], which try to match clusters of different time points. Works of this kind usually introduce a parameter which determines whether the dissimilarity of two clusters is too big to match. However, a match of clusters is a very subjective task and highly dependent on the used definitions. Further this is not necessary in order to detect outliers and thus not relevant for our work. The approach of Landauer et al. [14] uses an anomaly score, which is based on transitions of a single time series. This is different to our method, since we use the information of multiple time series.
The analysis of time series behavior like presented in this paper not only detects surprisingly deviating data points and subsequences with regard to a single time series, but also identifies new, behavior-based outliers. Our approach is also different from those which cluster whole time series, since such approaches do not consider the cluster transitions, which is an expressive feature on its own. The algorithm presented in this paper is able to detect anomalous subsequences, although they would have been assigned to one cluster in a subsequence clustering.

3 Fundamentals

Before introducing the method, some basic definitions regarding time series analysis used in the underlying paper [20] and this work are given, since they may vary in literature. An illustration of them can be seen in Figure 1.

Definition 1 (Time Series). *A multivariate time series $T = o_{t_1}, ..., o_{t_n}$ is an ordered set of n real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording, with t_1 and t_n indicating the first and the last timestamp, respectively.*

Definition 2 (Data Set). *A data set $D = T_1, ..., T_m$ is a set of m time series of same length and equivalent points in time. The set of data points of all time series at a timestamp t_i is denoted as O_{t_i}.*

Definition 3 (Subsequence). *A subsequence $T_{t_i,t_j,l} = o_{t_i,l}, ..., o_{t_j,l}$ with $j > i$ is an ordered set of successive real valued data points beginning at time t_i and ending at t_j from time series T_l.*

Definition 4 (Cluster). *A cluster $C_{t_i,j} \subseteq O_{t_i}$ at time t_i, with $j \in \{1, ..., q\}$ being a unique identifier (e.g. counter) and q being the number of clusters, is a set of similar data points, identified by a cluster algorithm or human. This means that all clusters have distinct labels regardless of time.*

Definition 5 (Cluster Member). *A data point $o_{t_i,l}$ from time series T_l at time t_i, that is assigned to a cluster $C_{t_i,j}$ is called a member of cluster $C_{t_i,j}$.*

Definition 6 (Noise). *A data point $o_{t_i,l}$ from time series T_l at time t_i is considered as noise, if it is not assigned to any cluster.*

Definition 7 (Clustering). *A clustering is the overall result of a clustering algorithm or the set of all clusters annotated by a human for all timestamps. In concrete it is the set $\zeta = \{C_{t_1,1}, ..., C_{t_n,q}\} \cup Noise$.*

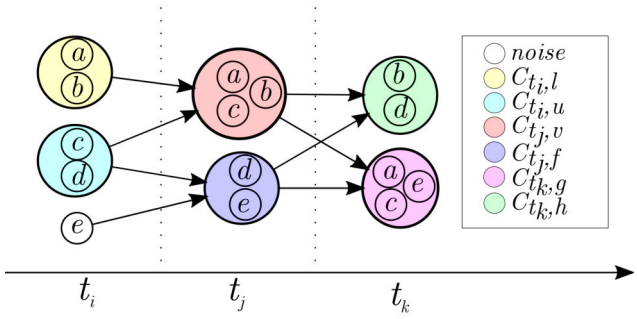


Fig. 2: Example for cluster transitions of time series $T_a, .., T_e$ over time.

An example for the above definitions can also be seen in Figure 2. Five time series of a data set $D = T_a$, T_b, T_c, T_d, T_e are clustered per timestamp for the time points t_i, t_j and t_k. The data points of a time series T_l are denoted by the identifier l for simplicity reasons. The shown clustering consists of six clusters. It can be described by the set $\zeta = \{C_{t_i,l}, C_{t_i,u}, C_{t_j,v}, C_{t_j,f}, C_{t_k,g}, C_{t_k,h}\} \cup \{o_{t_i,e}\}$. As $o_{t_i,e}$ is not assigned to any cluster in t_i, it is marked as noise for this timestamp. The data points $o_{t_i,a}, o_{t_i,b}$ of time series T_a and T_b in t_i are cluster members of the yellow cluster $C_{t_i,l}$.

4 Method

The cohesion of a sequence with its peers over time is described by the term *over-time stability*. Our approach is based on the assumption that unstable behavior over time indicates an irregularity. In order to rate the over-time stability of a sequence by means of a so called *subsequence_score*, the proportion of cluster members from earlier timestamps who migrated together into another cluster in later timestamps has to be calculated. For this reason, the *temporal cluster intersection* was introduced [20]:

$$\cap_t\{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \wedge o_{t_j,l} \in C_{t_j,b}\}$$

with $C_{t_i,a}$ and $C_{t_j,b}$ being two clusters, $t_i, t_j \in \{t_1, ...t_n\}$ and $l \in \{1, ..., m\}$. The proportion p of two Clusters $C_{t_i,a}$ and $C_{t_j,b}$ with $t_i < t_j$ is then calculated by:

$$p(C_{t_i,a}, C_{t_j,b}) = \begin{cases} 0 & \text{if } C_{t_i,a} = \emptyset \\ \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a}|} & \text{else} \end{cases}$$

As this proportion is asymmetric since it only describes the proportion of $C_{t_i,a}$ that is contained in $C_{t_j,b}$, a merge of clusters has no negative impact on the score. However, in some use cases it might be wanted to treat merges and splits equally, because a well-separated clustering is desired. With this calculation it is not possible to distinguish whether a time series has the best possible score because it always remains in its well-separated cluster or because its cluster only merged into other ones but never split off.
In order to punish merges and splits the same way, the jaccard index can be used to obtain the proportion. For this, we introduce the *temporal cluster union* of two clusters $C_{t_i,a}, C_{t_j,b}$:

$$\cup_t\{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \vee o_{t_j,l} \in C_{t_j,b}\}$$

with $l \in \{1, ..., m\}$. Now the proportion $\hat{p}$ can be calculated by the jaccard index of two clusters:

$$\hat{p}(C_{t_i,a}, C_{t_j,b}) = \begin{cases} 0 & \text{if } C_{t_i,a} = \emptyset \wedge C_{t_j,b} = \emptyset \\ \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a} \cup_t C_{t_j,b}|} & \text{else} \end{cases}$$

with $t_i < t_j$.
Regarding the example in Figure 2, the proportion p of cluster $C_{t_i,l}$ and $C_{t_j,v}$ would be

$$p(C_{t_i,l}, C_{t_j,v}) = \frac{|C_{t_i,l} \cap_t C_{t_j,v}|}{|C_{t_i,l}|} = \frac{2}{2} = 1$$

and therefore ideal. In contrast to that, the proportion $\hat{p}$ would be

$$\hat{p}(C_{t_i,l}, C_{t_j,v}) = \frac{|C_{t_i,l} \cap_t C_{t_j,v}|}{|C_{t_i,l} \cup_t C_{t_j,v}|} = \frac{2}{3} = 0.\overline{6}$$

as the merge of cluster $C_{t_i,l}$ and $C_{t_i,u}$ lowers the score.
Using the proportion, each subsequence $T_{t_i,t_j,l}$ of time series l beginning at timestamp t_i and ending at t_j is rated by the following *subsequence_score* in [20]:

$$subsequence\_score(T_{t_i,t_j,l}) = \frac{1}{k} \cdot \sum_{v=i}^{j-1} p(cid(o_{t_v,l}), cid(o_{t_j,l}))$$

with $l \in \{1, ..., m\}$, $k \in [1, j-i]$ being the number of timestamps between t_i and t_j where the data point exists and *cid*, the cluster-identity function

$$cid(o_{t_i,l}) = \begin{cases} \emptyset & \text{if the data point is not assigned to any cluster} \\ C_{t_i,a} & \text{else} \end{cases}$$

returning the cluster which the data point has been assigned to in t_i. In words, it is the average proportion of the sequence's clusters it migrated with from t_i to t_j. Here, the impact of all preceding time points to the score is weighted equally. For longer sequences, this can lead to a tendency towards a worse rating, since slow changes in cluster membership might influence the rating quite considerably. Assuming that the nearer past is more meaningful than the more distant past, we formulate a weighting that can be used in the subsequence score.
Regarding a time interval $[t_1, t_k]$, the proportion at time t_i with $t_1 \leq t_i \leq t_k$ gets the weighting $\frac{2 \cdot i}{k(k+1)}$ resulting by the division of i with the Gauss's Formula

$$\frac{i}{\sum_{a=1}^{k} a} = \frac{i}{\frac{k(k+1)}{2}} = \frac{2 \cdot i}{k(k+1)} \ .$$

The weighting function can easily be adjusted for time intervals starting at time $t_s > t_1$. The subsequence score is then calculated as follows:

$$weighted\_subseq\_score(T_{t_i,t_j,l}) = \sum_{v=i}^{j-1} \frac{2 \cdot (v-i+1)}{k(k+1)} p(cid(o_{t_v,l}), cid(o_{t_j,l}))$$

with $k \in [1, j-i]$ again being the number of timestamps between t_i and t_j where the data point exists. Since the sum of all weightings of a subsequence's timestamps is always 1, there is no need to normalize the score to an interval of

$[0, 1]$ by averaging it.
In the example of Figure 2, the score of time series T_a between time points t_i and t_k would be

$$subsequence\_score(T_{t_i,t_k,a}) = \frac{1}{2} \cdot (1.0 + 0.\overline{6}) = 0.8\overline{3}$$

whereby the rating with the weighted subsequence score would be

$$weighted\_subseq\_score(T_{t_i,t_k,a}) = (\frac{1}{3} \cdot 1.0 + \frac{2}{3} \cdot 0.\overline{6}) = 0.78$$

The second proportion which is smaller than 1 has thus more influence on the score now. The combination of the weighted subsequence score and the jaccard proportion $\hat{p}$ has the following result:

$$weighted\_jaccard\_score(T_{t_i,t_k,a}) = (\frac{1}{3} \cdot 0.\overline{6} + \frac{2}{3} \cdot 0.5) = 0.56$$

With the help of the subsequence's rating an outlier score can be calculated for each by determining the deviation of their stability from the best subsequence score of their cluster. Formally, the best score of a cluster $C_{t_j,a}$ for sequences starting at t_i and ending at t_j is given by

$$best\_score(t_i, C_{t_j,a}) = max(\{subsequence\_score(T_{t_i,t_j,l}) \mid cid(o_{t_j,l}) = C_{t_j,a}\}) \ .$$

A subsequence's outlier score is then described by

$$outlier\_score(T_{t_i,t_j,l}) = best\_score(t_i, cid(o_{t_j,l})) - subsequence\_score(T_{t_i,t_j,l}) \ .$$

The outlier score is therefore dependent on the over-time stability of the considered cluster's members. The smaller the best score is, the smaller is the highest possible outlier score. The detection of outlier sequences can be done by using a threshold τ [20]:

Definition 8 (Outlier). *Given a threshold $\tau \in [0, 1]$, a subsequence $T_{t_i,t_j,l}$ is called an outlier, if its probability of being an outlier is greater than or equal τ. That means, if*

$$outlier\_score(T_{t_i,t_j,l}) \geq \tau \ .$$

In addition to these outlier sequences, subsequences that consist entirely of noise data points from the clustering algorithm are identified as *intuitive outliers*. Sequences whose last data point is labeled as noise are not assigned to a cluster which the best score can be determined from, so they do not get an outlier score.

5 Experiments

In the following, several experiments on different (artificially generated and real world) data sets are performed in order to evaluate the effects of the

modifications of this paper regarding the original method. In all cases the density-based clustering algorithm *DBSCAN* [8] was used for clustering. We will differentiate between the *original method* from [20], the *jaccard method* (where the proportion is calculated by the jaccard index), the *weighted method* (where the weighting is included in the subsequence score), and the *weighted jaccard method* (where all modifications are integrated). In all experiments the same parameter settings for ϵ, $mitPts$ and τ were used for the investigated methods in order to make the results comparable. Please note, that dependent on the method in some cases another parameter choice could have been beneficial.

5.1 Artificially Generated Data Set

For a targeted evaluation of the properties, at first an artificially generated data set with 40 timestamps is considered. The data set was generated so that initially four starting points (for four groups of time series) were selected. In addition, the maximum distance of the centroids of two successive time points and the number of members were chosen for each group. The centroids as well as the members' data points were then calculated randomly for each time point, whereby the distance of the members to the centroids could not exceed 0.03. After generating the normal data points, one completely random outlier sequence and three targeted outlier sequences were inserted. For the completely random sequence all data points were chosen randomly and the distance between two consecutive points was set to not being greater than 0.1. The remaining outlier sequences were generated as follows: The data points were always set with a maximum distance of 0.06 to a centroid. The clusters were chosen randomly whereby the distance of the latest data point and the next centroid could not exceed 0.2. Additionally, the sequence always had to be allocated for at least 5

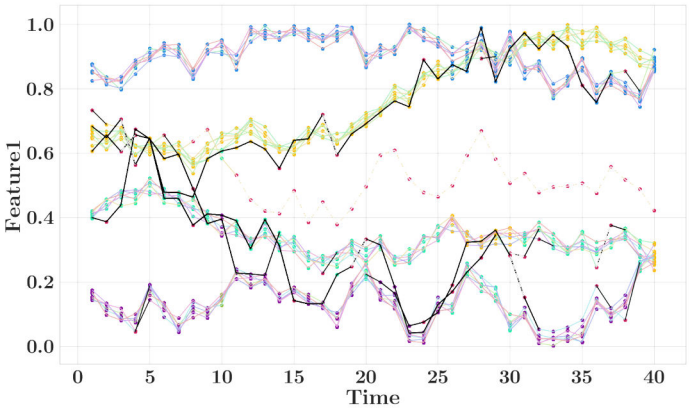


Fig. 3: Achieved results on the generated data set with $\epsilon = 0.025$, $minPts = 3$ and $\tau = 0.7$ by the original method.

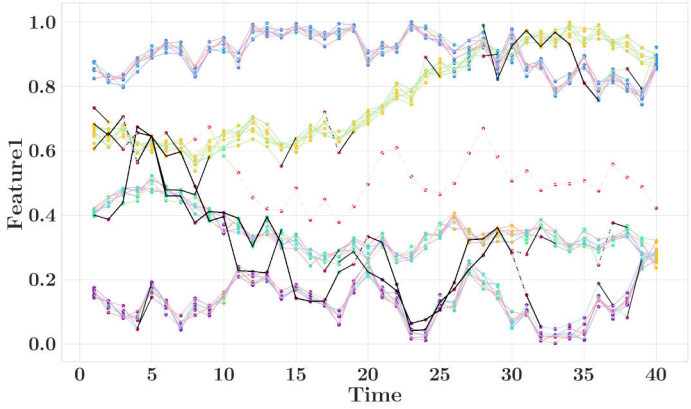


Fig. 4: Achieved results on the generated data set with $\epsilon = 0.025$, $minPts = 3$ and $\tau = 0.7$ by the weighted method.

time points to the same cluster before choosing the next one. For all points, care was taken to ensure that they were between 0 and 1.

The time series data was clustered per timestamp with the parameter setting $\epsilon = 0.025$ and $minPts = 3$. All four methods were performed on the clustering with the threshold $\tau = 0.7$. The results are illustrated in the Figures Fig. 3, Fig. 4, Fig. 5 and Fig. 6. Red dots represent noise data points while other colors indicate the cluster membership. Black lines stand for outliers that are found with the outlier score and dashed lines represent intuitive outliers.

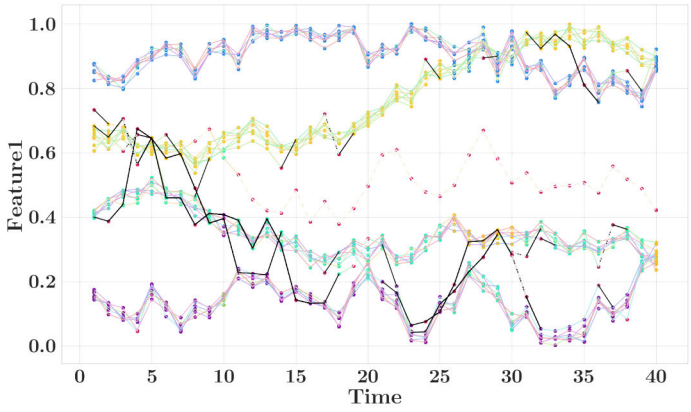


Fig. 5: Achieved results on the generated data set with $\epsilon = 0.025$, $minPts = 3$ and $\tau = 0.7$ by the jaccard method.

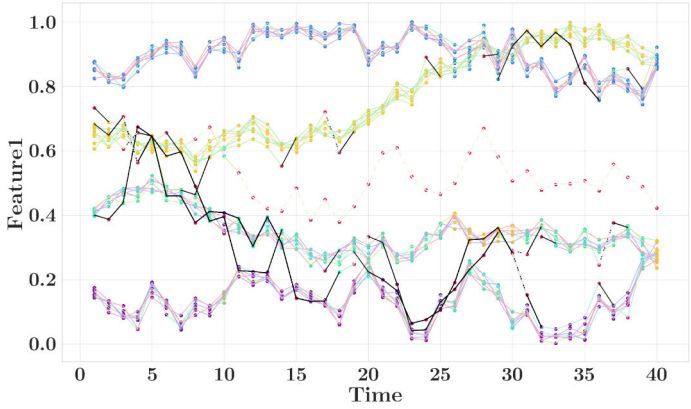
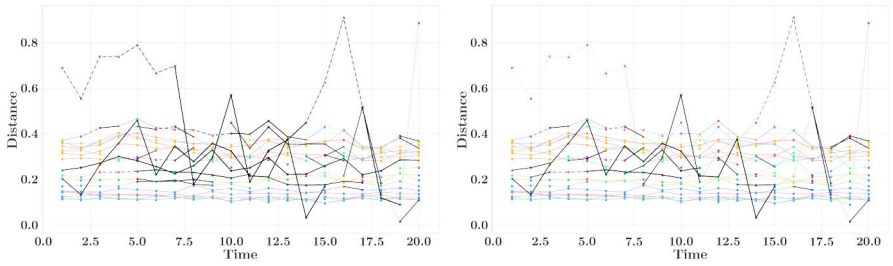


Fig. 6: Achieved results on the generated data set with $\epsilon = 0.025$, $minPts = 3$ and $\tau = 0.7$ by the weighted jaccard method.

The original method (Fig. 3) detects all four outlier sequences and marks almost the whole time series as such. However, some parts of the outlier sequence in the yellow clusters (second from the top) are quite stable and therefore should not be detected as outliers in regard to their over-time stability. When considering the results of the weighted method (Fig. 4) one can see, that some smaller parts of the time series are marked as outliers. The most obvious example is the outlier sequence of the yellow clusters. This effect shows, that the intention of the weighting, that the more distant past has a lower impact on the score than the nearer past, is therefore satisfied. The jaccard method (Fig. 5) leads to a more sparsely detection, as well. This can be explained by the fact that due to some merges (for example in the yellow clusters) the best subsequence score of the clusters is decreased and consequently the highest outlier score is decreased, too. The effect of the lower best score can also be seen between the timestamps 29 and 35. In contrast to the weighted method, the "M" shape is not marked completely. The combination of both modifications is illustrated in Figure 6. Since the nearer past is weighted more strongly here, the merge of the blue and yellow clusters at time point 26 has not as much influence on the best score. Therefore the "M" shape is detected as outlier. However, there are some differences in regard to the results of the weighted method. Overall fewer outlier sequences are found. An example can be seen in the first time stamps. This behavior is reasoned as the jaccard index lowers the best possible score in the clusters.

5.2 Airline On-Time Performance Data Set

This data set holds 29 features like the scheduled and actual departure time for flights reported by certified U.S. air carriers. In total it contains 3.5 million



(a) Original Method (b) Weighted Jaccard Method

Fig. 7: Achieved results on the Airline On-Time Performance Data Set with $\epsilon = 0.03$, $minPts = 3$ and $\tau = 0.5$.

records with each representing a flight. Originally this data set is provided by the U.S. Department of Transportation's Bureau of Transportation Statistics [7]. In order to make the data set suitable for our approach we interpreted the feature set of every airline as a sequence. Further we made these time series equidistant by calculating the average of their features for every day. Finally we normalized the data set with the min-max normalization and clustered it per timestamp.
In this experiment we compare the original method [20] with the modified approach presented in this paper. Both modifications are applied and the result is illustrated in Figure 7b. The first noticeable difference to the original [20] approach in Figure 7a is the lower amount of marked outliers. This can be explained with both adjustments: First of all, the introduced jaccard index leads to overall lower subsequence scores, thus the best score of a cluster is lower and therefore the outlier score is lower. Second, the weighting function allows time series to change their peers over time if it is done consequently. This means that time series are not considered to be suspicious if they made a stable change, which is to expect when regarding larger time series. Actually the original approach cannot handle the amount of points in time and tends to become more sensitive with rising amount of time stamps. In contrary, the adjusted version performs more robust and can handle more timestamps better.
On the second sight, one might notice that the adjusted method detects slightly different outliers than the original approach (e.g. the two upper outliers between timestamp 17.5 and 20.0). However, those differences in this example are too small to be reasoned with a specific modification.

5.3 GlobalEconomy Data Set

The GlobalEconomy data set is obtained from the website theglobaleconomy.com [1]. It holds over 300 indicators for different features for 200 countries over more than 60 years. For illustration reasons we chose 20 countries and two features, namely the education spendings and the unemployment rate. Please note, that

the amount of countries can vary per timestamp, because there are missing values in the data set.

The result of the original method and the modified approach presented in this paper, can be seen in Figure 8 and Figure 9. The colors represent the detected clusters, circles represent behavior-based outliers and red font is indicating noise which was detected by DBSCAN. In case a country is detected as a behavior-based outlier and as noise by DBSCAN it is represented as a circle with red font. The abbreviations are according to ISO 3166. At first glance it is noticeable that our original approach detected more outliers than the new method. Let us explain this by the example of Kyrgyzstan (KGZ) in the years 2010 and 2011: KGZ leaves the yellow cluster and at the same time joins the green cluster in 2011. In our original calculation KGZ is punished for this transition by applying the old cluster proportion function. At the same time the subsequence score of the Marshall Islands (ISL) is not influenced in 2011, because it was not assigned to a cluster in 2010. Thus the outlier score of Kyrgyzstan is negatively influenced. In the weighted jaccard method Kyrgyzstan is not detected as an outlier, because the Marshall Islands are punished for the merge with Kyrgyzstan in 2011. This leads to a lower *best_score* and at the same time to a lower *outlier_score* of Kyrgyzstan. In summary, Kyrgyzstan is not detected as an outlier, because the Marshall Islands are now punished for merging.

An example of finding new outliers is Honduras (HND) in the years from 2013 to 2015. The old technique did not identify Honduras as an outlier in the years 2014 and 2015, while the modified method does. Again this has to do with the low subsequence score of the Marshal Islands in 2014, but this time the cluster proportion of the original approach is punishing the Marshal Islands for splitting

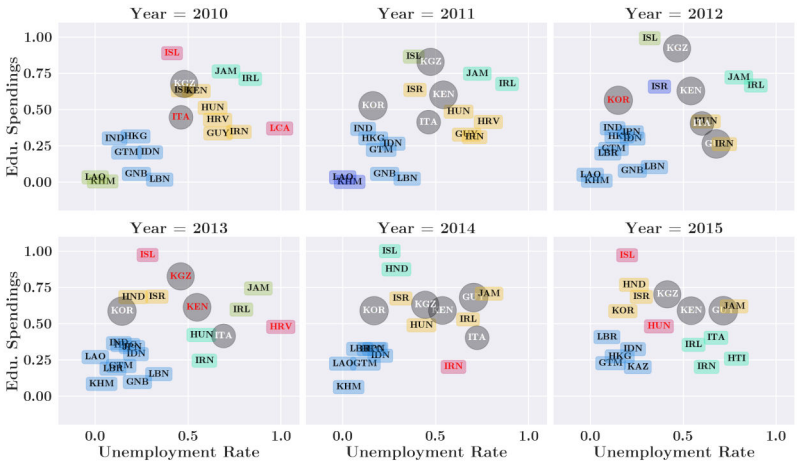


Fig. 8: Achieved results on the GlobalEconomy Data Set with $\epsilon = 0.18$, $minPts = 2$ and $\tau = 0.4$ by the original method.

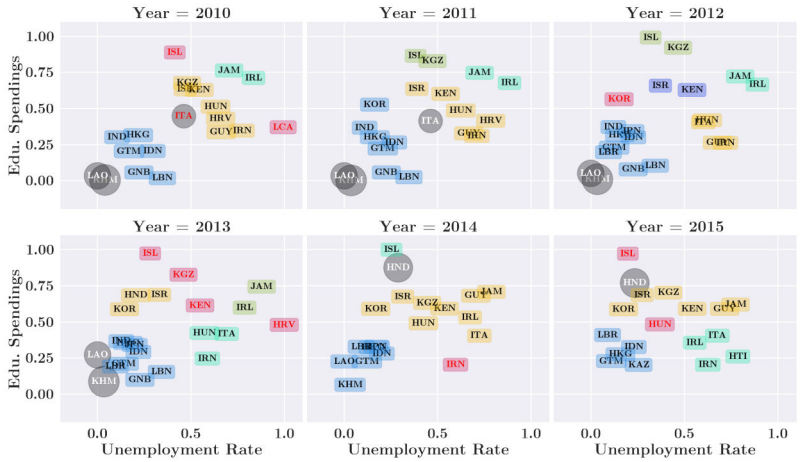


Fig. 9: Achieved results on the GlobalEconomy Data Set with $\epsilon = 0.18$, $minPts = 2$ and $\tau = 0.4$ by the weighted jaccard method.

from its peers in the previous years. However, this is not the only reason Honduras is not marked as an outlier. It actually benefits from rejoining the yellow cluster in 2015, although the yellow cluster contains more than twice the amount of countries now. In concrete that means, that the comparison of the years of 2013 and 2015 was influencing the subsequence score of Honduras in a positive way. The weighted jaccard approach takes the new constellation of the yellow cluster into account. In contrary to the original method, the comparison of the years 2013 and 2015 is not beneficial to its subsequence score. Further more the merge with the Marshall Islands in 2014 is punished by the jaccard index.
Another interesting observation is, that the jaccard index now enables the identification of outlier clusters. In Figure 9 one can observe, that Laos (LAO) and Cambodia (KHM) form a cluster in the years from 2010 to 2011. The merge to the big blue cluster in 2012 has a fairly high influence on their subsequence scores so that they are detected as outliers in 2013. Although the two countries are more stable in the subsequence from 2012 to 2013, they have not stabilized to the level of their peers in the blue cluster. This finally happens in the year of 2014.

6 Conclusion & Future Work

The analysis of time series data – especially the identification of conspicuous sequences – is an important field in data mining. So far, there are only a few approaches for the detection of outliers in multiple time series. In [20] we presented an outlier detection algorithm which analyses the behavior of groups of time series by clustering the data per timestamp using an arbitrary clustering

algorithm. As this was a novel approach, there were still some handicaps and application dependent properties. In this paper, we focused on two of these and proposed the following solutions: First, we presented another technique for the calculation of the proportion, which treats merges and splits of clusters equally. Second, we introduced a weighting function that causes a higher impact of a sequence's nearer past than the more distant one. Our results show, that the intended effects were achieved by our modifications. All results are meaningful and show individual qualities. Dependent on the application, one of the four investigated methods can be used for the detection of anomalous subsequences in regard to their over-time stability.
However, the aspects dealt with in this paper were only a part of the procedure's difficulties. There is still the problem of determining the best parameter τ and optimal hyperparameters for the clustering algorithms such as DBSCAN. Additionally, the treatment of noise data points could be improved. As proposed in [20], the inclusion of the time series' deviations might lead to an advanced analysis of those. Further, the detection of outlier clusters would be interesting. Partly they are already found by the modified method presented in this paper. Finally, the procedure could be adjusted to handle fuzzy clusterings. With the help of over-time stability measures for hard [21] and fuzzy clusterings [13] a good basis for the outlier detection can be provided.

7 Acknowledgement

We would like to thank the Jürgen Manchot Foundation, which supported this work by financing the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University Düsseldorf.

References

1. Global economy, world economy, https://www.theglobaleconomy.com/.
2. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134 – 147 (2017)
3. Ahmar, A.S., Guritno, S., Abdurakhman, Rahman, A., Awi, Alimuddin, Minggi, I., Tiro, M.A., Aidid, M.K., Annas, S., Sutiksno, D.U., Ahmar, D.S., Ahmar, K.H., Ahmar, A.A., Zaki, A., Abdullah, D., Rahim, R., Nurdiyanto, H., Hidayat, R., Napitupulu, D., Simarmata, J., Kurniasih, N., Abdillah, L.A., Pranolo, A., Haviluddin, Albra, W., Arifin, A.N.M.: Modeling data containing outliers using ARIMA additive outlier (ARIMA-AO). Journal of Physics: Conference Series **954** (2018)
4. Banerjee, A., Ghosh, J.: Clickstream clustering using weighted longest common subsequences. In: In Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining. pp. 33–40 (2001)
5. Cheng, H., Tan, P.N., Potter, C., Klooster, S.: Detection and characterization of anomalies in multivariate time series. In: Proceedings of the 2009 SIAM international conference on data mining. pp. 413–424 (2009)
6. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: Stl: A seasonal-trend decomposition procedure based on loess (with discussion). Journal of Official Statistics **6**, 3–73 (1990)

7. Computing, S., Graphics, S.: Airline on-time performance. http://statcomputing.org/dataexpo/2009/the-data.html, accessed: 2019-07-15
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226–231 (1996)
9. Ferreira, L.N., Zhao, L.: Time series clustering via community detection in networks. Information Sciences **326**, 227 – 242 (2016)
10. Granger, C.W.J.: Investigating causal relations by econometric models and cross-spectral methods. Econometrica **37**(3), 424 (Aug 1969)
11. Huang, X., Ye, Y., Xiong, L., Lau, R.Y., Jiang, N., Wang, S.: Time series k-means: A new k-means type smooth subspace clustering for time series data. Information Sciences **367-368**, 1 – 13 (2016)
12. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) Advances in Spatial and Temporal Databases. pp. 364–381 (2005)
13. Klassen, G., Tatusch, M., Himmelspach, L., Conrad, S.: Fuzzy clustering stability evaluation of time series. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems. pp. 680–692 (2020)
14. Landauer, M., Wurzenberger, M., Skopik, F., Settanni, G., Filzmoser, P.: Time series analysis: Unsupervised anomaly detection beyond outlier detection. In: ISPEC (2018)
15. Malhotra, P., Vig, L., Shroff, G.M., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: ESANN (2015)
16. Munir, M., Siddiqui, S.A., Chattha, M.A., Dengel, A., Ahmed, S.: Fusead: Unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. Sensors **19**(11) (2019)
17. Paparrizos, J., Gravano, L.: k-shape: Efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 1855–1870 (2015)
18. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (2007)
19. Sun, P., Chawla, S., Arunasalam, B.: Mining for outliers in sequential databases. In: in ICDM, 2006. pp. 94–106
20. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In: Data Mining. AusDM 2019. Communications in Computer and Information Science. vol. 1127, pp. 91–103 (2019)
21. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: How is your team spirit? cluster over-time stability evaluation (forthcoming). In: Machine Learning and Data Mining in Pattern Recognition, 16th International Conference on Machine Learning and Data Mining, MLDM (2020)
22. Truong, C.D., Anh, D.T.: A novel clustering-based method for time series motif discovery under time warping measure. International Journal of Data Science and Analytics **4**(2), 113–126 (2017)
23. Zhou, Y., Zou, H., Arghandeh, R., Gu, W., Spanos, C.J.: Non-parametric outliers detection in multiple time series a case study: Power grid data analysis. In: AAAI (2018)

3.3 Loners stand out. Identification of anomalous subsequences based on group performance

---

Martha Tatusch, Gerhard Klassen, and Stefan Conrad. Loners stand out. Identification of anomalous subsequences based on group performance. In *Advanced Data Mining and Applications - 16th International Conference, ADMA 2020, Foshan, China, November 12-14, 2020, Proceedings*, Lecture Notes in Computer Science. Springer, 2020.

Contributions: The concept and main idea of the paper was developed jointly by Martha Krakowski (née Tatusch) and Gerhard Klassen. Martha Krakowski (née Tatusch) implemented the proposed method and wrote the major part of the paper. Gerhard Klassen mainly had an advisory role in the implementation and writing of the paper. Prof. Dr. Stefan Conrad supervised the work.

Status: published

In this section we refer to a third approach [R9] for outlier detection in time series databases. The method is called ***D**etecting **A**nomalies based on **C**luster **T**ransitions* (DACT) and is also based on the principles introduced in CLOSE and FCSETS. However, in DACT we define the *over-time stability* of a time series differently. The focus here is on the number of different time series with which a time series has been in a cluster over time. We refer to this concept as the *shared time points* of time series. The *shared time points* are set in relation to the number of constant cluster peers over time (*peer count*), which results in a new definition of the *over-time stability*.

The procedure is an alternative to the symmetrical variant of the approach presented in the last section, i.e. in DACT not only the splitting of clusters but also the merging of clusters is penalised. As already discussed in the last section, this can be particularly useful in applications in which the clusters found should actually remain separate for semantic reasons. However, here the symmetry was not achieved with the help of the *jaccard-distance*, but by the simple ratio of the *shared time points* and the *peer count*.

The slightly different interpretation of *over-time stability* in the referenced work [R9] shows the variability of the concept. Many other variants are conceivable and should be analysed in future work.

Loners Stand Out. Identification of Anomalous Subsequences Based on Group Performance

Martha Tatusch[0000−0001−6302−6070], Gerhard Klassen[0000−0002−1458−6546], and Stefan Conrad[0000−0003−2788−3854]

Heinrich Heine University, Universitätsstr. 1, 40225 D¨sseldorf, Germany
{tatusch,klassen,stefan.conrad}@hhu.de

Abstract. Time series analysis is a part of data mining and nowadays an important field of research due to the increasing amount of data that is recorded sequentially by various systems. Especially the identification of anomalous subsequences arouses great interest, since a manual search for errors or malfunctions is not possible in most cases. Often outliers are defined as points or sequences that deviate significantly from the course of one or multiple time series, yet there are also applications where the trend rather than the exact course of time series is relevant. In that case, there is an approach of clustering the time series per time point and analyzing their cluster transitions over time. Sequences that change their cluster members suddenly or often, indicate an anomaly.
In 2019, a novel approach for the detection of these transition-based outliers was introduced [19]. Now, we present an algorithm called DACT (**D**etecting **A**nomalies based on **C**luster **T**ransitions) that is able to identify outlier sequences of the same type. It is a simple approach that stands out due to different results, although a similar type of anomalies is targeted. In the evaluation, we examine and discuss the differences. Our experiments show, that the results are competitive and reasonable.

Keywords: Outlier Detection · Time Series Analysis · Clustering.

1 Motivation

Due to the increasing popularity of digital systems such as social platforms, online shops or simple database applications in various industries, data analysis is of steadily growing importance. The analysis of sequential data forms an important part of this field of research and is known as *time series analysis*. There are several applications which consider either single or multiple time series whereby these can be univariate or multivariate. In this work, we focus on multiple multivariate time series and the behavior of subsequences with regard to their peers. There are many applications where these conditions apply. For example, when investigating a drug's tolerance on humans, one time series per patient can be extracted whereby various features per timestamp are recorded. In our approach, we examine the trend of groups of time series rather than the exact course, as it is not relevant in many applications. To do so, it is necessary to

previously cluster the data for each point in time. Regarding the drug tolerance behavior, the patients may be grouped by their state of health. Since every human body is unique, these clusters may change over time. Some of these changes are normal, but if a patient shows any irregularity, action must be taken. In order to detect such irregularities automatically, we introduce DACT (**D**etecting **A**nomalies based on **C**luster **T**ransitions), an anomaly detection algorithm for transition-based outliers. To the best of our knowledge, the first approach regarding this type of outliers was published in 2019 [19]. Hence, in the following

2 Foundation

In order to provide a good basis for the comparison of the two methods, the same definitions as given in [19] are used in this work.

Definition 1 (Time Series). *A multivariate time series* $T = o_{t_1}, ..., o_{t_n}$ *is an ordered set of n real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording.*

Definition 2 (Data Set). *A data set* $D = T_1, ..., T_m$ *is a set of m time series of same length and equivalent points in time. The set of data points of all time series at a timestamp* t_i *is denoted as* O_{t_i}*.*

Definition 3 (Subsequence). *A subsequence* $T_{t_i,t_j,l} = o_{t_i,l}, ..., o_{t_j,l}$ *with* $j > i$ *is an ordered set of successive real valued data points beginning at time* t_i *and ending at* t_j *from time series* T_l*.*

Definition 4 (Cluster). *A cluster* $C_{t_i,j} \subseteq O_{t_i}$ *at time* t_i*, with* $j \in \{1, ..., q\}$ *being a unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm or human.*

Definition 5 (Cluster Member). *A data point* $o_{t_i,l}$ *from time series* T_l *at time* t_i*, that is assigned to a cluster* $C_{t_i,j}$ *is called a member of cluster* $C_{t_i,j}$*.*

Definition 6 (Noise). *A data point* $o_{t_i,l}$ *from time series* T_l *at time* t_i *is considered as noise, if it is not assigned to any cluster.*

Definition 7 (Clustering). *A clustering is the overall result of a clustering algorithm or the set of all clusters annotated by a human for all timestamps. In concrete it is the set* $\zeta = \{C_{t_1,1}, ..., C_{t_n,q}\} \cup Noise$*.*

3 Related Work

There are various approaches for identifying irregularities in time series. In some applications, the detection of single anomalous data points is of interest. This problem is for example addressed by prediction-based algorithms like

auto-regressive-moving-average (ARMA) models [2, 6, 15]. In other cases, the identification of so called *changing points* [7, 13], which indicate a change of the previous course, are relevant. Although these techniques perform very well in most cases, they can not be used for our purpose. First, in contrast to DACT, they target single data points, not subsequences. Second, they lack the correlation of one time series to others. There are also other algorithms for the detection of outliers, which decompose the time series with techniques like STL [4] before analyzing them. However, these methods only work if the considered time series can be actually decomposed. In many applications, this is not the case. When regarding anomalous subsequences, there are various works using dynamic time warping (DTW) [17] for the comparison of time series or neural networks [3, 10, 16]. Another approach is the detection of the most unusual subsequences (discords) using a symbolic aggregation of a time series [8, 12, 9]. Even though these methods are aiming at subsequences, they only consider single time series and therefore can not be used in our case.

The most recent works for the detection of outlier subsequences in multiple time series use Probabilistic Suffix Trees (PST) [18] or Random Block Coordinate Descents (RBCD) [21] regarding the deviation of one time series to the others. In contrast to our approach, the behavior of a time series with regard to its peers is not analyzed here. We accomplish this analysis by clustering the time series data per timestamp and investigating a time series' transitions between clusters. Such an approach was already presented in 2019 [19]. However, the procedure has some particularities that might be unfavorable depending on the application. For example, the procedure in [19] only penalizes splits of a time series from a cluster, whereas merges of smaller clusters into larger ones do not have a negative influence on the outlier score of the sequences involved. In this paper we introduce a simple approach which resolves these difficulties.

4 Model Description

After the time series data has been clustered per timestamp using an arbitrary clustering algorithm like DBSCAN [5] or k-means [14], DACT can be applied. In short, the procedure is based on the analysis of the average number of points in time that a time series migrates with its peers, which indicates a subsequence's stability over time. The longer a sequence moves with its cluster members over time, the more stable it is.
For the following presentation of the components of DACT we first introduce the cluster identity function *cid* of a data point $o_{t_i,l}$, which returns the cluster of the time series l at the considered timestamp t_i:

$$cid(o_{t_i,l}) = \begin{cases} \emptyset & \text{if } o_{t_i,l} \text{ is not assigned to any cluster} \\ C_{t_i,a} & \text{else} \end{cases}$$

Now, we can calculate the number of time points in which two subsequences $T_{t_i,t_j,l}$ and $T_{t_i,t_j,x}$ share the same cluster. We call it the shared time points

count *stc*:

$$stc(T_{t_i,t_j,l}, T_{t_i,t_j,x}) = |\{t_k | cid(o_{t_k,x}) = cid(o_{t_k,l}) \ \wedge \ t_k \in [t_i, t_j]\}|$$

with $x \neq l$. In order to get the average number of time points a time series $T_{t_i,t_j,l}$ moves with its cluster members, we need to compute the number of peers of the time series during the considered time period. It describes the amount of distinct time series that are at least once assigned to the same cluster as T_l during the period. It can be calculated by the peer count *pc*:

$$pc(T_{t_i,t_j,l}) = |\{T_x \ | \exists t_k \in [t_i, t_j] : cid(o_{t_k,x}) = cid(o_{t_k,l})\}|$$

with $x \neq l$. We can now express the over-time stability OTS of a subsequence $T_{t_i,t_j,l}$ by

$$OTS(T_{t_i,t_j,l}) = \frac{\sum_{p=1}^{m} stc(T_{t_i,t_j,l}, T_{t_i,t_j,p})}{pc(T_{t_i,t_j,l}) \cdot k}$$

with k being the number of timestamps where T_l holds data. In order to detect anomalies in time series, this score needs to be included in an outlier score, which indicates whether a subsequence is conspicuous or not. In the following we propose two concepts for building the outlier score. Since we believe, that this score is dependent on the behavior of a subsequence's peers (an unstable sequence is not as conspicuous regarding an unstable cluster as it is in a stable one), both variants focus on the scores of the considered cluster. Before introducing these two concepts, we define the term *intuitive outlier*:

Definition 8 (Intuitive Outlier). *A sequence $T_{t_i,t_j,l}$ is called an intuitive outlier if its data points are marked as noise for every timestamp $t_k \in [t_i, t_j]$.*

This is necessary as the outlier score can only be calculated for subsequences whose data point at the last timestamp is assigned to a cluster. If it is not, it is not possible to determine a meaningful reference value.

4.1 Variant 1

The first approach focuses on the best stability score achieved in a cluster $C_{t_j,a}$ regarding a time period from t_i to t_j. Formally, it can be expressed by

$$best\_score(C_{t_j,a}, t_i) = max(\{OTS(T_{t_i,t_j,l}) \mid cid(o_{t_j,l}) = C_{t_j,a}\}).$$

It describes the highest score obtained by subsequences from t_i to t_j ending in cluster $C_{t_j,a}$. The outlier score DACT of a subsequence is then given by the deviation of its stability score from the best score:

$$DACT(T_{t_i,t_j,l}) = best\_score(cid(o_{t_j,l}), t_i) - OTS(T_{t_i,t_j,l}).$$

Obviously, the *best_score* represents the upper bound for the outlier score within a cluster for a given time period. This causes, that clusters containing stable subsequences are more sensitive to deviations than the ones containing less stable sequences. Finally, an outlier can be formally described using the outlier score.

Definition 9 (Outlier – Variant 1). *Given a threshold τ, a sequence $T_{t_i,t_j,l}$ is called an outlier if*

$$DACT(T_{t_i,t_j,l}) > \tau \ .$$

Since the best subsequence score of a cluster influences the highest possible outlier score, the threshold τ often has to be chosen rather central in the interval $[0, 1]$. Additionally, the best threshold differs for data sets with different distributions of the data points. The more scattered the data, the lower the threshold.

4.2 Variant 2

The second approach follows the statistical assumption that anomalies can be found with the help of their deviation from the standard deviation. For this, the mean of a cluster's stability scores regarding the start time t_i has to be determined first. Regarding a cluster $C_{t_j,a}$ for the time period from t_i to t_j, it is given by

$$\mu(C_{t_j,a}, t_i) = \frac{1}{|C_{t_j,a}|} \cdot \sum_{o_{t_j,l} \in C_{t_j,a}} OTS(T_{t_i,t_j,l}).$$

The standard deviation of a cluster's stability scores regarding the start time t_i can then be calculated by

$$\sigma(C_{t_j,a}, t_i) = \sqrt{\frac{1}{|C_{t_j,a}|} \cdot \sum_{o_{t_j,l} \in C_{t_j,a}} (\mu(C_{t_j,a}, t_i) - OTS(T_{t_i,t_j,l}))^2}.$$

In order to compare it later with the standard deviation, we formulate the outlier score sDACT of a subsequence $T_{t_i,t_j,l}$ as the absolute difference of its stability score and the mean of its last cluster:

$$sDACT(T_{t_i,t_j,l}) = |\mu(cid(o_{t_j,l}), t_i) - OTS(T_{t_i,t_j,l})|.$$

We call it sDACT in order to express, that the *statistical* variant is used. In the following, this score can be used to detect outliers by inspecting the deviation of it from the standard deviation. With the help of a factor ρ it can be formally described.

Definition 10 (Outlier – Variant 2). *Given a threshold ρ, a sequence $T_{t_i,t_j,l}$ is called an outlier if*

$$sDACT(T_{t_i,t_j,l}) > \rho \cdot \sigma(cid(o_{t_j,l}), t_i) \ .$$

Again, the outlier score is highly dependent on the performance of the considered cluster's members. Since the standard deviation is considered, the outlier score is even less sensitive to deviations, especially in the case of a rather unstable cluster. Therefore in most cases the default value of $\rho = 3$ will probably be to high in order to detect inconsistencies. In our method, frequently a value of around $\rho \approx 2$ is recommended. This factor naturally is also dependent on the distribution of the data.

5 Experiments

Following, experiments on a synthetic and a real world data set are discussed to evaluate the performance of the presented methods. In order to simplify referencing the approaches we will name them as follows:

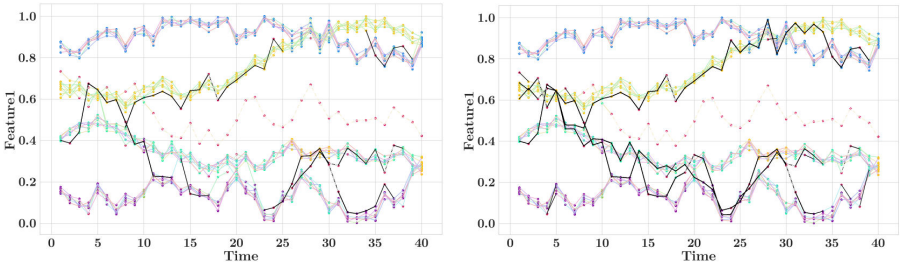
- *referred method* – describes the approach from [19].
- *DACT* – stands for the presented method using variant 1 for the detection of outliers.
- *sDACT* – represents the approach using variant 2.

5.1 Artificially Generated Data Set

The first considered data set was artificially generated and contains 28 univariate time series (TS) with 40 timestamps. Initially four groups of TS were randomly generated. Afterwards, three targeted and one completely random outlier sequence were inserted. All data points of the completely random outlier TS were chosen randomly, whereby the distance between two consecutive points was set to not being greater than 0.1. The remaining outlier sequences were generated so that their data points were always located near to a cluster's centroid. An outlier sequence could change its cluster at the earliest if it was located for at least 5 time points in a cluster.

The experiment was performed with DACT and the referred method. In order to get comparable results, the same parameter settings for both approaches were chosen. For the clustering DBSCAN [5] was used with $\epsilon = 0.025$ and $minPts = 3$. The threshold τ was set to 0.55. Figure 1 shows the detected anomalies by DACT and the referred method. The colored dots represent cluster belongings whereby red dots indicate noise. The detected outlier sequences are illustrated as and intuitive outliers as dashed lines.

Both methods managed to detect the completely random as well as parts of the three targeted outliers. The referred method, however, marked a lot more



(a) DACT (b) referred method

Fig. 1: Detected outliers on the generated data set with $\boldsymbol{\tau = 0.55}$, $\boldsymbol{minPts = 3}$ and $\boldsymbol{\epsilon = 0.025}$.

parts as outliers than DACT. Regarding the uppermost outlier sequence from time point 10 to 39, there is a difference between both methods between time 25 and 34. DACT did not mark this part of the TS as an outlier although the referred method did. This can be explained by the fact, that the TS moves stably with most of its cluster members in this period. The merge of the two upper clusters causes lower stability scores, but since the size of both clusters is approximately the same, all cluster members are affected equally. The same applies to the split.

Considering the second lowest outlier sequence between timestamp 30 and 38, it is the other way around. While DACT marks the sequence as an outlier for the whole period, the referred method interprets the course between timestamp 34 and 36 as normal. On the one hand, this is caused by the decrease of the stability scores in the second lowest cluster. As there were merges and splits in the history of the cluster, all scores were negatively affected. On the other hand, there are only few members in the considered cluster and another sequence is marked as noise at time point 32, too. Between timestamp 34 and 36 the considered time series behaves stable, so that it does not stand out in contrast to its cluster members, regarding this short period. In contrast to that, DACT is more sensitive concerning short term changes, if only few time series are considered.

5.2 GlobalEconomy Data Set

The second data set is provided by the website theglobaleconomy.com [1]. It consists of over 300 indicators for different features of 200 countries for more than 60 years. For the experiments, we considered 20 different countries and two features (namely the education spendings and the unemployment rate) within the

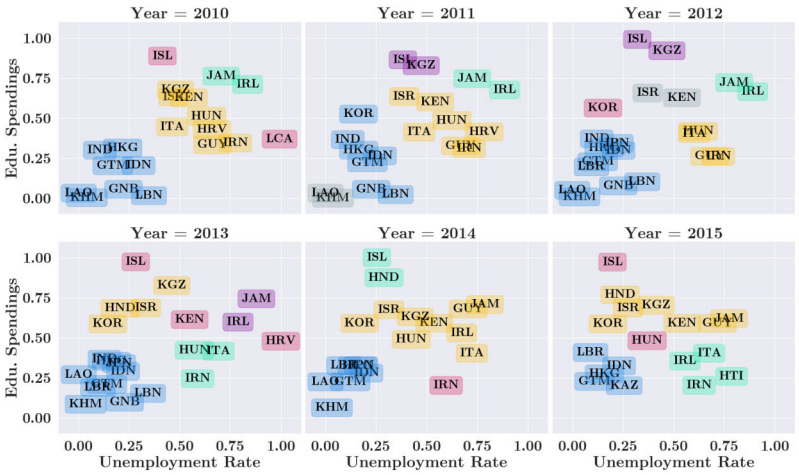


Fig. 2: Resulted clustering by DBSCAN with $\boldsymbol{minPts = 2}$ and $\boldsymbol{\epsilon = 0.19}$ on the GlobalEconomy data set.

period from 2010 to 2015 to enable a manageable illustration. Since the database is not complete for all country-year combinations, the amount of countries per

The experiment was run with all three methods using DBSCAN with $\epsilon = 0.19$ and $minPts = 2$. Since the underlying clustering for all three approaches is the same, it is illustrated separately in Figure 2. Different colors represent different cluster belongings and noise data points are marked red. The resulting outlier sequences are listed in Table 1. The list was shortened so that in case of overlaps only the longest detected subsequence of a country is included per method. This time, the threshold parameters τ and ρ were chosen for all methods separately, as the first experiment showed that the same parameter setting led to considerably more outlier sequences with the referred method than with DACT. An individual parameter choice might therefore be appropriate.

Country	Start	End	DACT	sDACT	referred
GUY	2012	2015	–	–	x
HND	2013	2015	x	–	–
HND	2014	2015	x	x	–
IRL	2010	2014	x	–	–
JAM	2010	2014	x	–	–
KEN	2010	2015	–	–	x
KEN	2013	2014	–	x	x
KGZ	2010	2014	–	–	x
KOR	2011	2014	x	–	x

Table 1: Resulting outlier sequences by DACT ($\boldsymbol{\tau = 0.3}$), sDACT ($\boldsymbol{\rho = 2}$) and the referred method ($\boldsymbol{\tau = 0.35}$) on the GlobalEconomy data set.

It can be seen, that sDACT produces significantly less outlier sequences than DACT and the referred method. While those approaches detect both five anomalous subsequences, sDACT only finds two. This can be explained by the fact, that there are many clusters with only few cluster members. In addition, there are only a few TS, that are very stable over time. This causes, that the mean stability score per cluster is rather low. In order to stand out, a sequence needs therefore a very bad stability score. This only happens in two cases. First, Honduras (HND) does badly from 2014 to 2015, as it moves away from its only cluster member Iceland (ISL) and merges into a large cluster. The second case is Kenya (KEN) from 2013 to 2014, where it turns from noise to a large cluster's member. While the first anomaly sounds reasonable, the second one appears rather groundless, depending on the context. In contrast to DACT, which only found the first and not the second discussed outlier sequence, the referred method had exactly the opposite result. In fact, the only anomaly DACT and the referred method share, is the subsequence of Korea (KOR) from 2011 to 2014. This result is desired, since KOR changes its cluster members at every timestamp in this period.

The outlier sequences IRL and JAM show DACT's sensitivity regarding small clusters merging into large ones. Although those two countries stay stably together from 2010 to 2014, even when merging into the larger cluster, both are detected as outlier sequences. The referred method does not detect those sequences, because it does not penalize merges of clusters. However, although KEN stays with many cluster members over time, it is marked as outlier from 2010 to 2015. This is caused by the split from its cluster in 2012 and 2013. Another outlier detected by the referred method is Guyana (GUY) from 2012 to 2015. In 2013, the data is missing and this is the crucial point. In 2012 GUY is

grouped with Hungary (HUN), Italy (ITA) and Iran (IRN). The merge into a larger cluster in 2014 is not penalized, but the following split from HUN, ITA and IRN in 2015 has a very negative effect on the stability, though.

6 Conclusion

In this paper, we introduced two approaches of finding transition-based outliers in time series databases. We examined the differences of the results and evaluated our methods against their competitor from [19], which targets the same problem definition. The results showed that both approaches find reasonable outliers, thus they differ in some characteristics. While the referred method does not penalize merges of clusters but only splits, DACT and sDACT treat both cases the same way. Furthermore, DACT is more sensitive regarding short term changes in small data sets. These differences lead to slightly different results, whereby the methods agree in clear cases. Depending on the application, both approaches provide a benefit.

We are aware of some shortcomings in DACT, that provide incentives for future work. For example, the handling of noise data points from the clustering could be improved. Currently, all subsequences consisting exclusively of noise data points are marked as intuitive outliers. In some cases, this behavior may not be legitimate. Furthermore, DACT is reliant on the assumption, that the underlying clustering is reasonable. Apart from inventing an evaluation measure for over-time clusterings [11, 20] in order to support the user in finding the right parameter settings, a new clustering algorithm tailored to the intention of an over-time clustering with temporal linkage would be useful.

References

1. Global economy, world economy, https://www.theglobaleconomy.com/.
2. Ahmar, A.S., Guritno, S., Abdurakhman, Rahman, A., Awi, Alimuddin, Minggi, I., Tiro, M.A., Aidid, M.K., Annas, S., Sutiksno, D.U., Ahmar, D.S., Ahmar, K.H., Ahmar, A.A., Zaki, A., Abdullah, D., Rahim, R., Nurdiyanto, H., Hidayat, R., Napitupulu, D., Simarmata, J., Kurniasih, N., Abdillah, L.A., Pranolo, A., Haviluddin, Albra, W., Arifin, A.N.M.: Modeling data containing outliers using ARIMA additive outlier (ARIMA-AO). Journal of Physics: Conference Series **954** (2018)
3. Chambon, S., Thorey, V., Arnal, P.J., Mignot, E., Gramfort, A.: A deep learning architecture to detect events in eeg signals during sleep. In: 28th Int. Workshop on Machine Learning for Signal Processing. pp. 1–6 (2018)
4. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: Stl: A seasonal-trend decomposition procedure based on loess (with discussion). Journal of Official Statistics **6**, 3–73 (1990)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226–231 (1996)

6. Hill, D.J., Minsker, B.S.: Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. Environ. Model. Softw. **25**(9), 1014–1022 (2010)
7. Kawahara, Y., Sugiyama, M.: Change-point detection in time-series data by direct density-ratio estimation. In: Proceedings of the 2009 SIAM International Conference on Data Mining. pp. 389–400. SIAM (2009)
8. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: Fifth IEEE International Conference on Data Mining (ICDM'05). p. 226233 (2005)
9. Keogh, E., Lonardi, S., Chiu, B.Y.c.: Finding surprising patterns in a time series database in linear time and space. In: Proceedings of the 8th Int. Conference on Knowledge Discovery and Data Mining. pp. 550–556 (2002)
10. Kieu, T., Yang, B., Jensen, C.S.: Outlier detection for multidimensional time series using deep neural networks. In: 2018 19th IEEE Int. Conference on Mobile Data Management (MDM). pp. 125–134 (2018)
11. Klassen, G., Tatusch, M., Himmelspach, L., Conrad, S.: Fuzzy clustering stability evaluation of time series. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU (2020)
12. Lin, J., Keogh, E., Ada Fu, Van Herle, H.: Approximations to magic: finding unusual medical time series. In: 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05). pp. 329–334 (2005)
13. Liu, S., Yamada, M., Collier, N., Sugiyama, M.: Change-point detection in time-series data by relative density-ratio estimation. Neural Netw. **43**, 72–83 (2013)
14. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297 (1967)
15. Munir, M., Siddiqui, S.A., Chattha, M.A., Dengel, A., Ahmed, S.: Fusead: Unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. Sensors **19**(11) (2019)
16. Munir, M., Siddiqui, S.A., Dengel, A., Ahmed, S.: Deepant: A deep learning approach for unsupervised anomaly detection in time series. IEEE Access **7**, 1991–2005 (2018)
17. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (2007)
18. Sun, P., Chawla, S., Arunasalam, B.: Mining for outliers in sequential databases. In: ICDM, 2006. pp. 94–106
19. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In: Data Mining. AusDM 2019. Communications in Computer and Information Science. vol. 1127, pp. 91–103 (2019)
20. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: How is your team spirit? cluster over-time stability evaluation. In: Machine Learning and Data Mining in Pattern Recognition, MLDM (2020)
21. Zhou, Y., Zou, H., Arghandeh, R., Gu, W., Spanos, C.J.: Non-parametric outliers detection in multiple time series a case study: Power grid data analysis. In: AAAI (2018)

3.4 Clustering of Time Series Regarding their Over-Time Stability

---

Gerhard Klassen, Martha Tatusch, and Stefan Conrad. Clustering of Time Series Regarding Their Over-Time Stability. In *Big Data Analytics and Knowledge Discovery*, Lecture Notes in Computer Science, pages 333–347. Springer International Publishing, Cham, 2020.

Contributions: The concept and main idea of the paper were mainly designed by Gerhard Klassen with the support of Martha Krakowski (née Tatusch). The methods and experiments were mainly implemented by Gerhard Klassen. Except for the *Method* and *Evaluation* sections, the paper was written entirely by Gerhard Klassen. The paper was supervised by Prof. Dr. Stefan Conrad.

Status: published

There are many different approaches to clustering time series. Many methods refer to one time series and try to identify recurring patterns in it [23, 36, 40]. However, the results achieved are not always meaningful, because the data basis must comply with certain properties [33].

The algorithm presented in the referenced paper [R1], however, does not refer to a single time series but to time series databases. The objective is to recognise patterns across time series. To do this, we always consider the entire data set. In our previous work (CLOSE and FCSETS), we looked at the temporal aspect after cluster analysis. This approach may have some disadvantages, for example, the cluster algorithms considered may not be able to construct a more stable clustering. For this reason, it can be advantageous to consider the temporal aspect at the time of clustering.

In this section we reference a method [R1] for clustering time series. The basis of this procedure is the idea of so-called *adaptability* of time series. This property is very closely related to the *over-time stability*, which cannot be used here because it requires clustered data. *Adaptability* is a measure that expresses how similar an object is to all other objects. In the introduced *connection factor* it represents a weighting for the similarity of two time series at one time point. The *temporal connection factor* is then used to include the temporal aspect at the time of clustering. The clusters found correspond to our ideas of stable clustering, but may lead to many outliers. Overall, the referenced work [R1] presents a possible interpretation of *over-time stability* included in a clustering algorithm.

Clustering of Time Series Regarding Their Over-Time Stability

1st Gerhard Klassen
Department of Computer Science
Heinrich Heine University
Düsseldorf, Germany
klassen@hhu.de

2nd Martha Tatusch
Department of Computer Science
Heinrich Heine University
Düsseldorf, Germany
tatusch@hhu.de

3rd Stefan Conrad
Department of Computer Science
Heinrich Heine University
Düsseldorf, Germany
stefan.conrad@hhu.de

***Abstract*—The clustering of time series data is still a challenging task. There are different approaches which consider either multiple time series or a single one. While some interpret the whole sequence as one feature vector, others examine subsequences or extract relevant features first. Because of these various perspectives, very different statements result. In this paper, we present the clustering algorithm C(OTS)^2 for multivariate time series data sets, that delivers a clustering per time point. It not only optimizes the quality of the clusters regarding intuitive demands, such as the spatial closeness of objects to their neighborhood within a cluster, but also the stability over time. Additionally, it can easily handle missing data points. The algorithm is of benefit whenever a cohesion of groups of time series can be assumed. One advantage is, that it requires only one parameter. Our experiments on different synthetic and real world data sets show, that our method works reasonable and fulfills the intention of finding temporal stable clusters without presupposing that the exact courses of the time series resemble.**

***Index Terms*—Time Series Analysis, Clustering Methods, Unsupervised Learning.**

I. Introduction

The analysis of sequentially registered data, so called time series, has strongly grown in interest over the past years, as there are more and more data sources for temporal data, such as online shops, IoT devices or medical sensors. The research field, in which databases of multiple multivariate time series are considered, often focuses on the task of classifying these or parts of them to investigate different properties. In many cases, the desired classes are not known beforehand, so that clustering algorithms need to be used. Various approaches consider the whole time series as a vector or extract feature vectors first. Some make use of a decomposition into seasonal, trend and other components.

In this paper, we present C(OTS)^2, a clustering algorithm for multivariate time series data, that clusters the data points per timestamp without missing the temporal context. The presented idea is based on two connection factors: one which expresses the connection factor of one object to another at a certain timestamps, and the other including the temporal context. With the help of these factors and a sliding window, we are able to construct a graph which describes the distance- and time-based cohesions. Its connected components represent the resulting clusters. The basic intention of C(OTS)^2 is the detection of clusters that are stable over time. The intuitive definition of compact clusters, where cluster members have a low distance to each other, is thereby mostly maintained. The maximization of the so called *over-time stability* does not force unintuitive clusters that are spread over the feature space. Still, it has a significant impact on the resulting clusters. Since the temporal components can easily be removed from the cluster calculation, the algorithm can also be used for clustering non-temporal data. Figure 1 shows two examples of clusterings provided by C(OTS)^2.

In contrast to approaches like the detection of Moving Clusters [1], our assumption is, that time series might change their cluster members over time and that this transition would consist of important information. Therefore we build a foundation for further analysis, whereby the detection of outliers is already partly included as our clustering algorithm is able to handle distance- and time-based noise.

Besides the use cases of tracking topics in online forums or the analysis of customer's purchasing behavior, our algorithm is useful whenever it can be assumed that there are groups of time series that behave in a similar way over time. In finance for example, the identification of misstatements regarding the annual financial statements of companies is of great interest. One approach is to interpret these statements as anomalous points with regard to a *common* behavior. This might be described by the behavior of other companies' financial statements that showed a similar behavior over time. With C(OTS)^2 these groups and possibly even the outliers may be identified and a solid foundation for further analysis could be provided. Another example is the analysis of the effectiveness and tolerance of medication regarding different patients. Every human body responds different to different medications. Thus, the formation of groups of patients whose bodies react similar to the drugs, can be assumed. However, it is possible that patients change their groups over time due to different circumstances, for example simply because their body is unique and responds different to the medication than its former cluster members. The group transitions of patients can be an indicator for an anomaly or the necessity of a

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

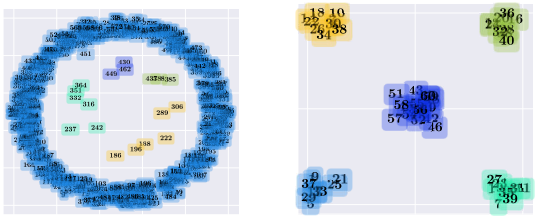


Fig. 1: Examples for resulting clusterings by C(OTS)^2, when only one timestamp is considered. Different colors indicate different cluster belongings.

change in medication, and might help in the prediction of future disease progression. With C(OTS)^2 this behavior can be discovered.

II. Fundamentals

Before we explain our method in detail in the next section, we clarify some important terms. Generally we stick to the definitions of Tatusch et al.'s paper [2], which deals with the detection of outliers with the help of a time series clustering per timestamp. We only make small adaptations and introduce one new definition. In addition to the mathematical statements, the most important definitions are illustrated in Figure 2.

Definition 1. *Time Series*
A time series $T_l = (o_{l,1}, ..., o_{l,n})$ is a tuple of n data points with $o_{l,i} \in \mathbb{R}^d$, $d \geq 1$. The data points are chronologically ordered by their time of recording.

Definition 2. *Time Series Data Set*
A time series data set $D = \{T_1, ..., T_m\}$ is a set of m time series with equivalent points in time. The set of data points of all time series at a timestamp t_i is denoted as O_{t_i}.
Different lengths and missing values are acceptable, as long as the time points can be mapped to a uniform scheme.

Definition 3. *Subsequence*
A subsequence $T_{l,i,k} = (o_{l,i}, ..., o_{l,k})$ with $i > k$ is a tuple of successive data points from time series T_l beginning at time t_i and ending at t_k.

Definition 4. *Sliding Window*
A sliding window of size s is a set of timestamps and is denoted as $w_{i,s} = \{t_{i-\lceil \frac{s-1}{2} \rceil}, ..., t_i, ..., t_{i+\lfloor \frac{s-1}{2} \rfloor}\}$. In case a timestamp is not represented by any time series of the data set, it does not occur in the set.

Definition 5. *Cluster*
A cluster $C_{t_i,j} \subseteq O_{t_i}$ at time t_i, with $|C_{t_i,j}| \geq 2$ and $j \in \{1, ..., q\}$ being a unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm.

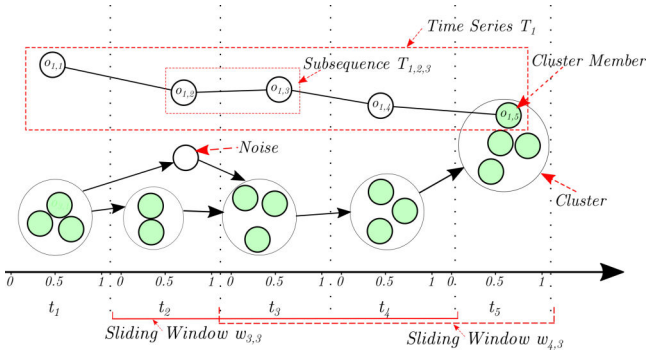


Fig. 2: Illustration of the most important definitions. Note, that a black arrow represents the development of a cluster, while a black line between objects of a time series represents the development of the sequence.

Definition 6. *Cluster Member*
A data point $o_{l,i}$ from time series T_l at time t_i, that is assigned to a cluster $C_{t_i,j}$ is called a member of cluster $C_{t_i,j}$.

Definition 7. *Noise*
A data point $o_{l,i}$ from time series T_l at time t_i is considered as noise, if it is not assigned to any cluster.

Definition 8. *Clustering*
A clustering is the overall result of a clustering algorithm for all timestamps. In concrete it is the set $\zeta = \{C_{1,1}, ..., C_{n,q}\} \cup Noise$.

III. Related Work

With the growing amount of time-dependent data in many applications, researches have presented different approaches to classify time series. The Time Series Classification Repository (TSCR) [3] established by the University of California, Riverside (UCR) and the University of East Anglia (UEA) led to a growth in the number of algorithms for time series classification problems. Besides the hosting of suitable data, the repository also offers a performance comparison on algorithms to the data. The methods presented in the TSCR target the identification of groups, so called classes, and the assignment of objects to those. Therefore, in contrast to our approach, these techniques regard the time series as a whole, so that the classification task refers to the entirety of the sequences. Referring the classification problems presented in the UCR this is reasonable, especially since Eamonn Keogh and Jessica Lin remarkably argued that clustering of time series subsequences is "meaningless" [4]. The problem we are tackling in this paper is different. Instead of identifying the class of a time series, we are interested in the behavior of time series in relation to other sequences. Especially changes in their behavior can contain significant information. This problem is related to cluster evolution over time [1], [5] with the difference, that instead of recognizing earlier clusters at later timestamps, our approach targets a clustering as a basis for the identification

of anomalous subsequences. Therefore we introduce the term *time series adaptability* which reflects a time series' ability to adapt to other sequences, that means its average similarity in relation to the data set. It may also be understood as the degree of team spirit of a sequence. Besides the tracking of topics in online forums, detecting outliers in financial data or fMRI scans, there are various other applications which could profit by our clustering algorithm.

The idea of using graphs in clustering algorithms is not new. In 2003, Stuetzle [6] proposes a graph-based clustering algorithm based on runt test for multimodality [7]. The clusters are identified by breaking edges in the minimal spanning tree of the regarded data set. Other graph-based clustering approaches make use of Delaunay Triangulations [8], which represent the dual graph of the Voronoi diagram for a discrete point set. There are techniques which make use of a user input as a threshold for the construction of the graph [9] and methods like AUTOCLUST [10] which do not require any user input. However, in contrary to our algorithm, these approaches do not take the temporal aspect of time series into account.

Finally classic clustering algorithms like k-Means [11] or DBSCAN [12] could be adapted to time series. Since the initial design does not handle time-based data, the modification is not simple. Regarding subsequences as vectors does not reflect the impact of time accordingly. Developing a distance function that includes the temporal aspect might be more promising. However, this is again a complex problem as a time series' neighborhood has to be considered as well. Of course, the naive approach of clustering the data at all time points independently of each other should also be taken into account. Obviously this approach lacks the temporal linkage, but in addition the clustering algorithm's hyperparameters have to be determined for every timestamp. In all cases, an analysis of cohesion post clustering has to be made. This would further influence the time complexity in a negative way. The design of our algorithm is targeted to time series, hence the cohesion analysis is done on the fly during the determination of clusters.

IV. Method

Our algorithm is designed to detect stable over-time clusters. That means, that the actual position of an object at one timestamp is not as important as its surrounding. We accept a certain deviation of an object to a cluster, if it moved with the same cluster members over a certain time period. The sliding window is optional. If not given we regard the whole time series. Our method is based on an arbitrary distance function which is normalized by the maximum distance $d_{\max}$ and minimum distance $d_{\min}$ over all timestamps. This is necessary to convert the distance measure to a similarity measure. For two sequences T_a, T_b we define the distance d at time point t_j as follows.

$$d(T_a, T_b, t_j) = dist(o_{a,j}, o_{b,j}) \tag{1}$$

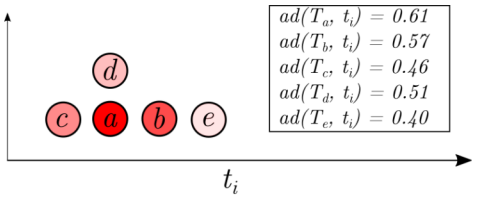


Fig. 3: A high opacity represents a high adaptability. The central location and the low distances to its neighbors lead to a high adaptability of a. In contrary, the high distance and peripheral location of e causes a low adaptability for it.

Here, $dist$ is an arbitrary distance function. Now the similarity at timestamp t_j of two time series can be calculated by

$$sim(T_a, T_b, t_j) = \left(1 - \left(\frac{d(T_a, T_b, t_j) - d_{\min}}{d_{\max} - d_{\min}}\right)\right)^2. \tag{2}$$

We square the term to get overall smaller similarities with a greater difference to each other. Later this becomes handy, when determining the only parameter in our method. The creation of clusters not only depends on the similarity of two objects, but also on the factor how adaptive an object is. We denote an object as being adaptive if it has a high similarity to many objects. It is the average similarity of the regarded object to every other object. Mathematically expressed, the adaptability ad of a time series at the timestamp t_j is defined as

$$ad(T_a, t_j) = \frac{1}{m-1} \cdot \sum_{x \in [1,m], x \neq a} sim(T_x, T_a, t_j)$$

with m describing the total number of objects at timestamp t_j.

In Figure 3 an illustration for the adaptability can be seen. It is important to understand this concept in order to rate the later influence of it to the resulting clustering.

The combination of the adaptability of an object and its similarity to another object represents the connection factor cf at time point t_j:

$$cf(T_a, T_b, t_j) = sim(T_a, T_b, t_j) \cdot ad(T_a, t_j). \tag{3}$$

Because of the adaptability of time series T_a, the connection factor between two time series at a certain timestamp is not symmetric. More precisely, in most cases it is $cf(T_a, T_b, t_j) \neq cf(T_b, T_a, t_j)$.

Finally, we introduce the temporal linkage by the average connection factor avg_cf, which then is incorporated in the temporal connection factor $temp\_cf$. The calculation of avg_cf is introduced with a sliding window, but can be adapted to the whole time series easily. This is particularly useful when short time series are considered. First have a look at the definition of avg_cf with a sliding window $w_{j,s}$:

$$avg\_cf(T_a, T_b, w_{j,s}) = \frac{1}{|w_{j,s}| - 1} \cdot \sum_{i \in w, i \neq j} cf(T_a, T_b, t_i).$$

In case the application on the whole time series is wanted, the size of the sliding window can be set to the number of time points of the largest time series. As noted in the previous section, points in time which are not present in the data set, do not occur within the set of the sliding window. The temporal connection factor tmp_cf is then defined as the average of cf at time point t_j and avg_cf. This leads to a higher influence of cf at the regarded point in time.

$$tmp\_cf(T_a, T_b, t_j) = \frac{cf(T_a, T_b, t_j) + avg\_cf(T_a, T_b, w_{j,s})}{2}.$$

With the help of the temporal connection factor tmp_cf and a parameter min_cf indicating the minimum connection factor to build an edge between two data points, an undirected graph $G_{t_j} = (V, E)$ can be created for every timestamp. $V = O_{t_j}$ denotes the set of nodes in the graph which are given by the data points of all time series at time t_j. The set $E \subseteq \{\{o_{a,j}, o_{b,j}\} | \forall o_{a,j}, o_{b,j} \in O_{t_j}\}$ contains all undirected edges between pairs of nodes of the graph. Using the minimum connection factor min_cf, an edge between $o_{a,j}$ and $o_{b,j}$ is added to the graph whenever the temporal connection of $o_{a,j}$ to $o_{b,j}$ or the temporal connection of $o_{b,j}$ to $o_{a,j}$ is greater or equal min_cf. So E is defined as

$$E = \{\{o_{a,j}, o_{b,j}\} | tmp\_cf(T_a, T_b, t_j) \geq min\_cf \vee tmp\_cf(T_b, T_a, t_j) \geq min\_cf\}. \quad (4)$$

After the graph is built, the clusters can be extracted by calculating the connected components[1] of it. Each component represents one cluster, whereby single-element components are marked as noise. Due to the usage of the introduced connection factors non-convex cluster shapes can be detected.

Since the connection factor cf and the adaptability ad are based on the similarity sim of the time series at a timestamp, the threshold min_cf highly depends on the closeness of objects belonging to the same cluster. The more compact the groups of data points are the higher min_cf must be set. Because of avg_cf the over-time stability of course has impact on the parameter choice as well. The more stable the time series are, the clearer the gradation of their connection factors, since cf and avg_cf are converging.

The summarized algorithm can be seen in Algorithm 1. The time complexity of the calculation of tmp_cf for all object pairs is in $O(n^2)$ as it can be done by matrix multiplication and all its components, like calculating the distance, are in $O(n^2)$. Since tmp_cf must be calculated for all m timestamps, the time complexity gets $O(m \cdot n^2)$. The graph again can be created in quadratic runtime and the connected components can even be extracted in linear time. So the overall time complexity of C(OTS)^2 is $O(m \cdot n^2)$ with m being the number of timestamps and n being the number of time series. Compared to the use of k-Means, which is in $O(n^2)$ and would have to be applied for every timestamp, which results in $O(m \cdot n^2)$, too, this time complexity is competitive.

[1] "A connected component of an undirected graph is a maximal set of nodes such that each pair of nodes is connected by a path." – https://www.sci.unich.it/~francesc/teaching/network/components.html

Algorithm 1 C(OTS)^2

procedure COTS(D, min_cf) ▷ $D = \{T_1, .., T_m\}$
  $clusters \leftarrow list\ of\ empty\ dictionaries$
  $V \leftarrow \{\}, E \leftarrow \{\}$
  for $t_i \in \{t_1, .., t_n\}$ **do**
    for all $(o_{a,i}, o_{b,i}) \in O^2_{t_i}$ **do**
      $calculate\ tmp\_cf(T_a, T_b, i)$ ▷ use the aforementioned formula tmp_cf
      if $tmp\_cf(T_a, T_b, i) \geq min\_cf \wedge (o_{a,i}, o_{b,i})\ not\ in\ E$ **then**
        $E \leftarrow E \cup \{(o_{a,i}, o_{b,i})\}$
      end if
    end for
    $G \leftarrow (V, E)$
    $components \leftarrow extract\_connected\_components(G)$
    $components \leftarrow mark\_noise(components)$ ▷ one-element sets denote noise
    $clusters \leftarrow clusters \cup components$
  end for
  return $clusters$
end procedure

As the temporal connection factor is based on the average connection factor, which is zero when considering only one timestamp, the approach can also be used for clusterings of non-temporal data using only the connection factor. Examples of resulting clusterings with C(OTS)^2 on non-temporal data are illustrated in Figure 1.

V. Experiments

Since our approach is a novelty in the field of time series analysis, unfortunately there is no appropriate quality measure which consists of the over-time stability as well as a shape-based measure for clusters. Therefore, we evaluate the accuracy of C(OTS)^2 by visual inspection. For illustration reasons, we generated three different data sets G_1, G_2, G_3 with time series containing two dimensional features, and between four and eight timestamps. Additionally, we consider two real world data sets comprising financial figures from the annual financial statements of publicly listed companies and a data set based on macroeconomic features of countries.

All experiments are explained along with figures. For reasons of illustrations the time series shown are never lasting for more than eight years and do not hold a high number of objects per timestamp. This does not indicate, that our method is not capable of handling greater data sets with more points in time. Quite in the contrary, especially the use of the sliding window, allows us the application to longer sequences. The amount of data points per timestamp changes the results, as it is expectable of a clustering algorithm but the results are still reasonable as can be seen in the experiments with the generated data sets. The shown explanatory figures always follow the same color code. Red indicates outliers, while other colors indicate a cluster.

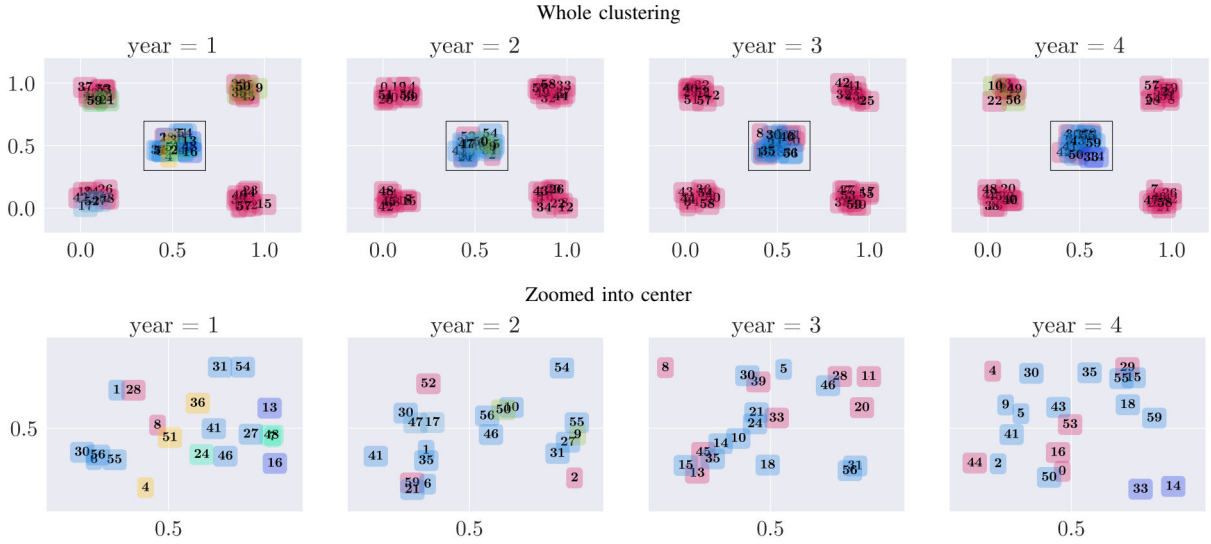


Fig. 4: Resulting clustering by C(OTS)^2 with $min\_cf = 0.35$ and $s = 3$ on the generated data set G_1 with an overall low over-time stability.

A. Synthetic Data

The first data set G_1 is supposed to represent a very *un*stable data set over time. It includes 60 time series and four timestamps. In order to point out the impact of the temporal behavior on the resulting clustering, five clusters were positioned fix in the feature space for all timestamps. For each point in time every time series was placed randomly into one of the five clusters. The data set and C(OTS)^2 clustering result can be seen in Figure 4. The first row shows the resulting clustering. The second row illustrates the excerpt from the upper clustering marked by a rectangle. Red data points indicate noise while other colors represent cluster belongings. Classic clustering algorithms which do not include a temporal aspect would have found five clusters per timestamp as there obviously are always five dense groups of data points. C(OTS)^2, however, marks most objects as noise and finds only small clusters. This can be explained by the fact, that only a few time series move with their cluster members over time. Most of them behave individually and therefore do not show a good team spirit. When considering the zoomed illustration in the second row, it is noticeable, that there are points in the center of the group, which are marked as noise or a separate cluster. This as well is caused by the over-time stability, which is aimed to be optimized in C(OTS)^2.

Note, that this experiment was executed with different parameter settings, thus never a good clustering result could be achieved, except of the case, when only one big cluster results. This is a desired behavior, since regarding the over-time stability, this data set can not be reasonably clustered. An example of the same cluster formation but perfectly stable time series can be seen in Figure 1. The result is the same for one or multiple timestamps if the time series behave stable over time.

The second data set G_2 consists of 15 stable time series and 4 timestamps, and intends to show an over-time clustering that slightly differs from a clustering per time point without temporal context. This effect can be caused by inserting time series which move between two clusters or a merge of clusters over time. In our case there is both, transitions as well as a merge. The result is shown in Figure 5. Since the data points

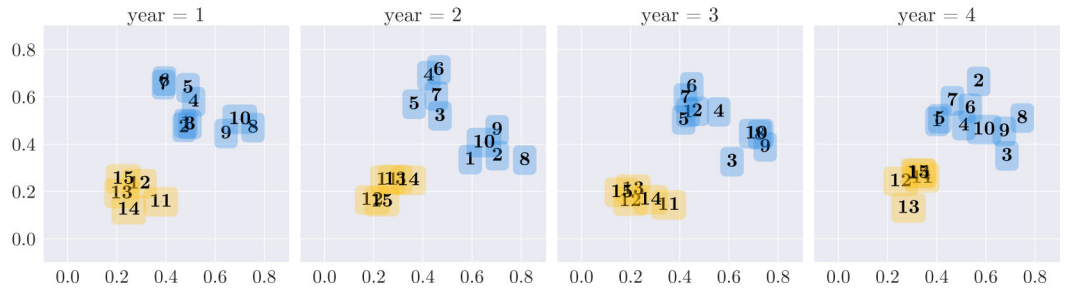


Fig. 5: Resulting clustering by C(OTS)^2 with $min\_cf = 0.15$ and $s = 3$ on the artificially generated data set G_2.

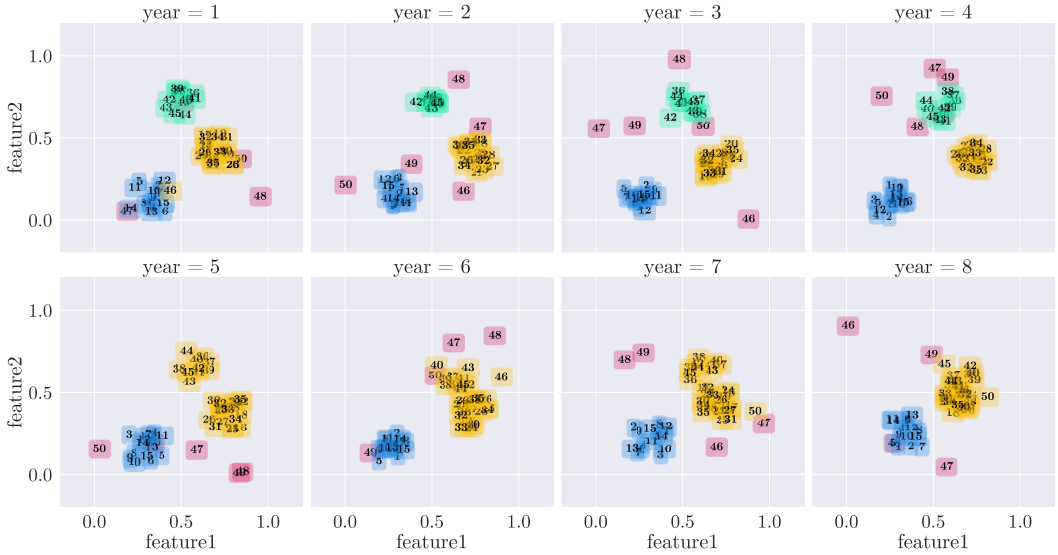


Fig. 6: Resulting clustering by C(OTS)^2 with $min\_cf = 0.32$ and $s = 3$ on the artificially generated data set G_3.

of the upper cluster are not very close to each other, min_cf has to be chosen comparatively small.

In every time point there can be seen two up to three groups of data points. C(OTS)^2 always identifies two clusters, although at least in timestamp two, a classic clustering algorithm without temporal component probably would have found three clusters. This can be explained on the one hand by the transitions of time series 1, 2 and 3 between the two upper clusters, and on the other hand by the fact that the aforementioned clusters merge at a later point in time. The result of a subsequence clustering would probably look different, too, as the exact course of the individual time series differs a lot. The time series 1, 2 and 3 might for example be recognized as noise, because their curves stick out with regard to the other time series.

In the third data set G_3, 3 clusters for 8 timestamps and a total of 50 time series were generated. Five sequences, namely 46 – 50, were inserted as outliers, by placing them randomly in the feature space for every point in time. Figure 6 illustrates the clustering result of C(OTS)^2 with $min\_cf = 0.32$ and a sliding window with size $s = 3$. Since the cluster members of each cluster lie close to each other and the time series are rather stable over time, all connection factors get higher values, so that min_cf is chosen higher than for example in Figure 5.

At first sight it is visible that C(OTS)^2 manages to detect all outlier sequences as such. As in the last two timestamps time series 50 is positioned near to the right cluster, the algorithm assigns it to it. This behavior is reasonable, because both, the similarity and the stability are given. In time point 3 on the other hand, time series 50 is not assigned to the upper cluster although it is located very near to the cluster's members. That is the effect of the considered over-time stability.

In the first four points in time C(OTS)^2 detects three clusters, which probably would also be recognized by common clustering algorithms. From time point five there are only two clusters, which is apparently a good choice especially for the timestamps six to eight. Although in timestamp five there are three obvious groups of time series, C(OTS)^2 merges the upper ones in terms of the further course. Because of the sliding window with width 3, the time points 4, 5 and 6 are considered in order to make a clustering for time point 5. Since the connection factors in timestamp 6 are generally higher than in timestamp 4, as more objects lie in small distance to each other, this timestamp has a higher impact on the clustering in timestamp 5. Therefore, the merge already happens in time point 5.

B. Real World Data

In order to test our method on real world data, we present two data sets. After presenting a financial data set, we present a macroeconomic data set and demonstrate how one could discover knowledge with the help of our algorithm.

First, we chose a financial data set which we obtained from EIKON [13], a product provided by Revinitiv (former provided by Thomson Reuters). We selected 30 arbitrary companies and two random features, namely *SoftAssets* and *Pension* over a timespan of eight years (2007 to 2014). The latter represents reserves for retirement plans of workers, while the first represent assets which have no physical nature such as patents, copyrights and trademarks. Unfortunately not every feature is available for every company at every point in time, therefore new companies may appear and other companies may disappear over time. In Figure 7 one can see the results of C(OTS)^2 applied with a sliding window of size five. Every box represents a company, the label corresponds to the stock symbol of the company. In 2009 one can observe a good example for the time aspect of this clustering algorithm.

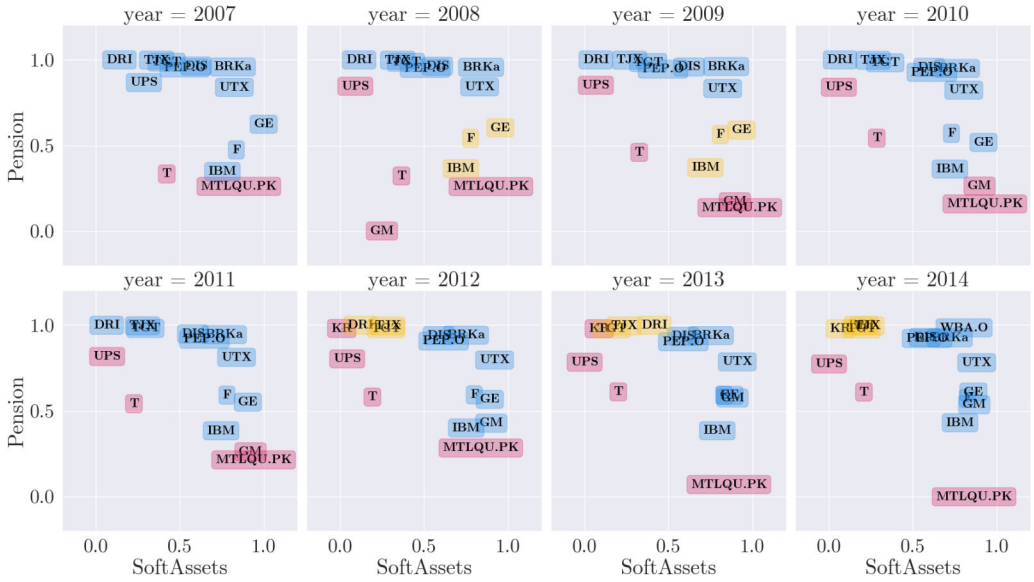


Fig. 7: Resulting clustering by C(OTS)^2 with $min\_cf = 0.26$ and $s = 5$ on the financial data set.

Although the companies *GM* and *MTLQU.PK* are very close to each other our method marks both as noise instead of creating a new cluster for them. This has two main reasons, first the *adaptability* of *GM* and *MTLQU.PK* is low in all three years from 2008 to 2010, second the *connection factor* of *MTLQU.PK* and *GM* in 2008 is very low and therefore also lowers it in the *temporal connection factor*. Comparing this to a clustering per timestamp, most cluster algorithms would assign *GM* and *MTLQU.PK* to a new cluster in 2009. It is also noticeable, that *MTLQU.PK* is detected as an outlier in the year 2012, where it is actually very close to a small agglomeration. In contrary the behavior of *GM* adapts to those of *F*, *GE* and *IBM* from 2012 on. The small splitting of *IBM* in 2013 is not punished and the stability of this cluster is preserved. On the other hand, the small splitting of *UPS* from 2007 to 2008 causes *UPS* being recognized as an outlier for the rest of the time. Since the distance of *UPS* to its peers is rising over time, this is a correct behavior of our algorithm. Another interesting aspect is the handling of overall outliers. In this excerpt AT&T, which is represented by the symbol *T* is always far away from the other companies. Therefore it is also always marked as noise. The second data set is obtained from theglobaleconomy.com [14]. It contains different features to countries over several years. We have chosen two figures to illustrate our algorithm on this data set. Additionally, we have chosen 2007 to be the beginning and 2012 to be the end of the regarded time. Because of this time span and the data basis, only 19 countries remained. The first figure we chose, is the household consumption as percent of the GDP and the second feature is the unemployment rate. The results of our method can be seen in Figure 8. We chose the given timespan, because of the financial crisis in 2008. The first observation, we made is, that the number of outliers increases from the year 2010. In conclusion that means, that the unemployment rate and the household consumption as percent of the GDP did not develop everywhere in the same way after the crisis. It can also be said, that the effect of the crisis is long-term, especially when inspecting numbers later than 2012. While the unemployment rate in some countries remained almost the same as before the crisis, some countries had a bad development. For example Estonia (*EST*) and Spain (*ESP*) had an increase of unemployment from 2009 on. While the change of Estonia is still close to the majority in 2009, Spain had a worse development and therefore is marked as an outlier. In 2010, Estonia almost has the same unemployment rate as Spain and both countries are far away from the majority in the blue cluster. Finally, Estonia somehow reacted on the crisis and could significantly lower its unemployment rate, so that it came very close to those of the majority. Spain on the contrary, had a rising unemployment rate until the last regarded year. In econometrics, this could be a helpful and quick analysis, which puts economic figures into the relation of groups of other countries.

VI. Conclusion & Future Work

The clustering of time series data is a broad field of research. Depending on the application there exist various approaches. When considering multiple multivariate time series, often whole sequences or parts of them are clustered using different preprocessing. In this paper, we presented a novel approach for clustering multivariate time series data. Our over-time clustering algorithm is named C(OTS)^2 and produces clusterings for every timestamp. One particularity of our approach is, that the exact course of (parts of) time series not necessarily has to resemble but the spatial location with regard to other time series over time. Another advantage is, that C(OTS)^2 requires

Fig. 8: Resulting clustering by C(OTS)^2 with $min\_cf = 0.34$ and $s = 3$ on the globaleconomy data set.

only one parameter. The results on various data sets showed, that the resulting clusters are stable over time, while satisfying intuitive demands on clusters, like spatial closeness of objects belonging to the same cluster. Since the calculation is based on two components of which one is time-independent, our algorithm can be used on non-temporal data for connection-based clusterings, as well. Additionally, it can easily handle missing data points. Because of a sliding window, the user is furthermore able to control the temporal impact on the clustering. We are keen to see a development in this field of research. It is important to benchmark the results against other algorithms with the same objective. However, regarding our algorithm, improvements still can be done. Although, we had no real difficulties to find good values for min_cf, a determination method would be very helpful. We are also aware of the optional second parameter s, the size of the sliding window. However, we think, that this is depending on the targeted analysis and should be determined by the domain specialist. In addition, we believe that runtime optimization could make the algorithm even faster, than it already is. Finally, it would be interesting to develop a fuzzy derivative of C(OTS)^2, where data points can belong to more than one cluster, as there are many applications where a hard clustering is not possible or wanted.

VII. Acknowledgement

This work was partly supported by the Jürgen Manchot Foundation, which funds the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University Duesseldorf.

References

[1] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In Claudia Bauzer Medeiros, Max J. Egenhofer, and Elisa Bertino, editors, *Advances in Spatial and Temporal Databases*, pages 364–381, 2005.

[2] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In *Data Mining. AusDM 2019. Communications in Computer and Information Science*, volume 1127, pages 91–103, 2019.

[3] Anthony J. Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn J. Keogh. The UEA multivariate time series classification archive, 2018. *CoRR*, 2018.

[4] 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.

[5] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 554–560, New York, NY, USA, 2006. Association for Computing Machinery.

[6] Werner Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *J. Classification*, 20:025–047, 05 2003.

[7] J. Hartigan and Surya Mohanty. The runt test for multimodality. *Journal of Classification*, 9(1):63–70, 1992.

[8] Xiankun Yang and Weihong Cui. A novel spatial clustering algorithm based on delaunay triangulation. *JSEA*, 3:141–149, 01 2010.

[9] G. Papari and N. Petkov. Algorithm that mimics human perceptual grouping of dot patterns. In *Proceedings of the First International Conference on Brain, Vision, and Artificial Intelligence*, BVAI'05, page 497–506, Berlin, Heidelberg, 2005. Springer-Verlag.

[10] Vladimir Estivill-Castro and Ickjai Lee. Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets. In *In Proceedings of the 5th International Conference on Geocomputation*, pages 23–25, 2000.

[11] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[13] Thomson Reuters. Eikon financial analysis and trading software.

[14] Global economy, world economy.

4

Excursion: Outlier Detection in Financial Data

In this excursion chapter, we introduce two works [R4, R7] that both present methods to predict so-called financial restatements of published balance sheets. Restatements always occur when intentional or unintentional errors were discovered in the original balance sheet. Although intentional errors are usually associated with far greater economic damage, as they usually constitute fraud [49], non-intentional errors can also lead to such damage [26]. It can therefore be advantageous for all parties involved to anticipate such restatements when the first balance sheet of a year is published.

While in the first work [R4] in section 4.1 we consider the data of a company as a time series and cluster vectors from one time point to another, in the work [R7] in section 4.2 we apply common machine learning algorithms without time reference. The results obtained are comparable, although it can be noted that the isolation forest [41] performs best in all respects.

The introduced work [R4, R7] is important for this dissertation because, on the one hand, it shows problems in clustering time series and, on the other hand, it provides results that can be compared with the outlier detection methods from Chapter 3. Such a comparison may provide further information about the potential of the outlier detection methods presented [R5, R8, R9].

4.1 Predicting Erroneous Financial Statements Using a Density-Based Clustering Approach

---

Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. Predicting Erroneous Financial Statements Using a Density-Based Clustering Approach. In *Proceedings of the 4th International Conference on Business and Information Management*, ICBIM '20, New York, NY, USA, 2020. Association for Computing Machinery.

Contributions: The main idea for this work was developed by Martha Krakowski (née Tatusch) and Gerhard Klassen . Both main authors implemented the described approach and the associated experiments in equal parts. Jun.-Prof. Dr. Marcus Bravidor motivated the work and provided the business context. Prof. Dr. Stefan Conrad supervised the work.

Status: published

In the first excursion paper [R4] we present in this chapter, we address the problem of detecting restatements using DBSCAN [20], a density-based clustering method. We consider about 9000 companies and look at the annual data of the companies as a time series. Because we hope to gain information from the temporal context, we do not cluster the raw data, but rather the vectors between two time points in each case. It has been shown that this relatively simple approach leads to comparatively good results. Although the state of the art method of Dechow et al. [17] is based on a different data set, we consider the results comparable. Especially with regard to precision, i.e. the ratio of correctly recognised (true positives) to the set of all recognised (true positives + false positives), our method performs significantly better. From our point of view, this method demonstrates the advantages of including the temporal aspect. It served as an introduction to the matter and the development of the ***C**luster **O**ver-**T**ime **S**tability **E**valuation* and ***F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries* .

Predicting Erroneous Financial Statements Using a Density-Based Clustering Approach

Martha Tatusch
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
tatusch@hhu.de

Gerhard Klassen
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
klassen@hhu.de

Marcus Bravidor
Department of Business Administration
HHU Düsseldorf
Düsseldorf, Germany
bravidor@hhu.de

Stefan Conrad
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
stefan.conrad@hhu.de

ABSTRACT

In this paper, we present a novel machine-learning approach to detect and predict restated financial statements. Our approach is based on DBSCAN, a cluster analysis algorithm. In contrast to prior methods, we assume that firms which perform different than their peers (outliers) are more likely to restate. By modifying DBSCAN to also incorporate temporal variation of these differences, we optimize the algorithm to fit financial data. We test our model for US data and benchmark against prior findings in accounting research. Our results show that the modified version of DBSCAN is more efficient than prior approaches. Best results are obtained if we cluster based on only two or three features. We outperform prior approaches regarding the precision to identify restatements. As with prior results, detection error increases for material restatements.

CCS Concepts

•Computing methodologies→Machine learning→Learning paradigms→Unsupervised learning→Cluster analysis

•Computing methodologies→Machine learning→Learning paradigms→Unsupervised learning→Anomaly detection

•Applied computing→Operations research→Forecasting

•Applied computing→Law, social and behavioral sciences→Economics

Keywords

Clustering, Prediction, Anomaly Detection, Time Series Analysis, Financial Restatements

1. INTRODUCTION

Restatements weaken the reliability of financial information. They are an important signal for investors and have (negative) long-term financial consequences [1]. Even though the number of restatements declined from more than 800 in 2009 to around 550 in 2017 [2], the number is still troublesome. Therefore, it is important for financial statement users, particularly investors, to be able to identify potential restatements. In this paper, we aim to detect financial restatements with a modified, dynamic version of the DBSCAN [5] clustering algorithm.

To evaluate our model, we use data from Thomson-Reuters (TR) EIKON [8] for the 20-year period between 1998 to 2017. For benchmarking, we use four different sets of restatements. First, restatements are defined as changes in any notable financial statement position (see Appendix.A1) recorded in EIKON. Second, relevant restatements that change either sales, operating cash flow, net income or shareholders' equity. Third, relevant and material restatements which are similar to the second definition but the change for any position must be at least 5%. Note that the first definition is the least strict, with two and three increasing in strictness, respectively. The fourth definition has to be considered separately as it is given by restatements reported by Audit Analytics [1].

To detect financial restatements, we use our modified DBSCAN algorithm. The idea behind this approach is that similar firms should have similar attributes and changes in a similar fashion. Hence, once a firm behaves abnormally in a sense that it shows different development in attributes than its peers, we assume it to be an "outlier". The advantage of using an unsupervised machine-learning algorithm like DBSCAN is that we need no *ex ante* expectations on (a) why firms behave differently, and (b) the threshold in differences that makes a firm an "outlier". Firms are clustered within industry groups (defined as four-digit Thomson-Reuters Business Classification codes) and there are one up to six attributes (features) provided.

Our results show that our approach correctly classifies more than 50% of all firm-years as (non-)restatement years for all four restatement definitions. On first sight, this result falls short of prior approaches (e.g., [3] report an accuracy of more than 65%). However, DBSCAN excels in precision of the results. We report values of 65.6%, 52.7%, 33.5% and 16.4% for the four restatement

definitions, respectively. [3] score 0.7%. Put differently, our approach is many times more likely to correctly classify restatements (as opposed to non-restatements).

We contribute to the literature in several ways. First, prior models used to identify financial restatements usually relied on either extensive multiple regression models or supervised machine-learning approaches. Whereas the first require a lot of firm-level data, the latter are often time opaque and the results difficult to understand. We address both issues and implement an efficient clustering algorithm which can detect restatements based on two or three firm-level items. Second, DBSCAN was initially build to work with 'static' data. We introduce a modified, dynamic version of DBSCAN which can detect cluster outliers by changes over different periods. Put differently, our approach is suitable to track changes in yearly firm-level data.

The paper is structured as follows. In section 2, we briefly discuss some background information on financial restatements as well as prior studies in accounting and computational science research. In section 3, we introduce our modified, dynamic version of DBSCAN. Evaluation results and benchmarks are presented in section 4. The paper ends with some concluding remarks.

2. BACKGROUND

An important distinction in the first place is the difference between erroneous and fraudulent financial statements. Fraudulent statements are the product of (management's) intention to mislead the user. Erroneous statements are simply (partly) false. The reasons can be manifold: fraud / intention, clerical or technical errors, etc. Once such an (material) error is found, the company must file a restatement. In our case, we are not interested in the reason behind the error. Therefore, we look at restatements as an indicator for any kind of erroneous financial statement.

In their extensive survey of accounting research, [10] differentiate between the causes and consequences of financial statements. Especially smaller firms, growth firms, and firms with a low earnings and/or reporting quality are more likely to restate. Most of these studies use logistic regressions to identify the causes (e.g., [3]). In their methodological review on data mining techniques used to identify financial restatements, [4] show that most studies in this realm build upon artificial neural networks, Bayesian Belief Networks and other forms of supervised learning. We follow their call to explore other complementary techniques. In this case, cluster analysis as another form of unsupervised machine-learning.

3. MODEL DESCRIPTION

In the following, we explain our method, the related parameter and feature selection as well as necessary basics of the original DBSCAN algorithm.

3.1 DBSCAN

DBSCAN is an algorithm for discovering clusters in large databases with noise [5]. In contrary to other clustering algorithms, DBSCAN determines clusters with the spatial density property of the regarded data points. In addition, the problem of identifying the right number of clusters is omitted, since it is automatically established. The algorithm differentiates three types of data points:

- **Core points:** A point p is a core point if there are at least *minPts* points in the ε-neighborhood of it (including p). The ε-neighborhood of p is defined as the spatial region with center p and radius $\varepsilon>0$.
- **Density-reachable points:** A density-reachable point is a point that is located within the ε-neighborhood of a core point.
- **Noise points:** A noise point is a point that is neither a core point nor a density-reachable point.

A cluster consists of at least one core point and *minPts-1* density-reachable points. Core points of different clusters are not density-reachable regarding each other. Points that are not assigned to any cluster are interpreted as noise.

3.2 Our Approach

As outlined above, current approaches for the detection of financial restatements have two major shortcomings. First, they rely on an "estimate-predict" idea. Take the case of a logistic regression. In order to predict whether a firm-year is likely to be restated, one has to first estimate the model parameters, then reverse and fill in the "blanks" with firm-year specific data. This approach requires a lot of data (out of sample predictions) and judgment (e.g., thresholds). Second, to draw meaningful inferences, the variables in the prediction model have to be selected on *ex ante* expectations.

For our approach, we require no *ex ante* expectations. We assume that similar firms behave in a similar and comparable manner. Similar deviations or periodic changes represent shared economic characteristics. To cover a broad set of economic factors we use the set of variables from [3]. Furthermore, we consider real activities manipulation proxies (RAM, [9]), and accrual-based earnings management (AEM) based on the modified Jones-model ([7]; [6]). We expect these features to hold more information about restatements than usual balance sheet figures.

In order to avoid the comparison of highly distinctive companies and industry-specific circumstances such as seasonal changes, our algorithm is applied to every industry sector separately. We define industry sectors based on four-digit TR Business Classification codes. We analyzed about 30 features (see the full list in Appendix.A1) and targeted a solution with less input features than the state-of-the-art approaches. Therefore we looked into different feature sets. A feature set f for a year t is described as $f_t = a_{t1},\dots,a_{tn}$ with $n \in \{2,3,4,5\}$. Then the development vector is calculated as $d_{t+1} = f_{t+1} - f_t$.

In order to identify misstatements the development vectors of companies in the same section are clustered with DBSCAN. Finally, development vectors which are not assigned to a cluster are regarded as misstatements.

3.3 Feature & Parameter Selection

In order to determine the best feature set for our approach and the most suitable setting of DBSCAN, we set a few constraints and iterated through all possible combinations. Since the most values are between 0 and 1, the possibilities for the radius ε of DBSCAN were set to [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0]. *minPts* was set to integers between 3 and 6.

To avoid the curse of dimensionality only subsets containing one to maximal five features were considered. In Appendix.A1 a complete list of all features is given. The original fields from EIKON (prefix “TR-”) have been normalized with the assets of the year before. All fields with the suffix “Error” contain the difference between the actual value and the expected regression value. For example, in *FF-CFOError* the difference between the actual cashflow from operating activities and the expected value of Roychowdhury's regression is stored. All regressions were calculated for every firm-year in every sector.

Note that the processed values and the original fields were considered individually. That means, that the subsets of all fields with the prefix "FF-" and all subsets of the fields with the prefix "TR-" have been tested. This was done to check whether, in our method, the popular proxies from [3], [6], [7] and [9] are more expressive than the original data.

4. EVALUATION

In the following, we will present our experiments, starting with the experimental setup, and discuss the results.

4.1 Experimental Setup

We analyzed 9300 companies with available data from 1998 to 2017. The data has been retrieved from TR EIKON which also holds restated figures.

In this paper, we observe four definitions of restatements:

- $Definition_{all}$: A statement is considered a restatement, if one or multiple financial values have been changed afterwards.
- $Definition_{relevant}$: A statement is considered a restatement, if at least one relevant financial value has been changed afterwards. Relevant values are: net income, shareholder's equity, operating cashflow, and sales.
- $Definition_{relevant5\%}$: A statement is considered a restatement, if one or multiple relevant financial values (see $Definition_{relevant}$) have been changed by at least 5% afterwards.
- $Definition_{audit}$: A statement is considered a restatement, if it is reported as restatement by Audit Analytics.

In case necessary data is missing, the firm-year is not included. This leads to different observations for different feature sets. All feature sets and parameters were evaluated on the HPC-Cluster of the Heinrich-Heine-University Düsseldorf. We analyzed about 500000 combinations of features and parameters. Every feature set was submitted as a job to the cluster.

4.2 Results

Apart from the type I and type II error, we consider three other measures:

- $accuracy = \frac{correct\ classified\ firm\ years}{all\ firm\ years}$
- $precision = \frac{correct\ classified\ as\ restatement}{all\ classified\ as\ restatements}$
- $recall = \frac{correct\ classified\ as\ restatement}{all\ real\ restatements}$

The detection of restatements is a challenging task. Since the data set is often very unbalanced, as there are only few misstatements, it is important to find a model that is not only strong in recognizing objects of one class.

The accuracy is not a suitable measure to verify this property. Suppose a dataset consists of 90% non-restatements and 10% restatements. If a model classifies every firm-year as non-restatement, an accuracy of 90% is achieved. This behavior is not desirable as the model does not make decisions based on features but on the fact, that most of the firm-years are non-restatements.

Precision and recall are more informative measures in this task. The precision indicates the relative value of how many elements of those classified as restatements have been correctly detected. The recall specifies the percentage of all restatements that have been recognized. If the recall is small, it is likely that the model rarely classifies objects as restatements. If the precision is small, this means that the probability, that a classification as a restatement is correct, is very low.

Table 1. Results for $Definition_{all}$.

Original Data	**obs. \ pred.**	**Rest.**	**No Rest.**	∑
	Rest.	4066	3719	7785
	No Rest.	2140	2171	4311
	∑	6206	5890	12096
	Rest.	**52.2%**	47.8%	64.4%
	No Rest.	49.6%	**50.4%**	35.6%
	Precision:	65.6%		
	Recall:	52.2%		
	Accuracy:	51.6%		
Processed Data	**obs. \ pred.**	**Rest.**	**No Rest.**	∑
	Rest.	7816	7704	15520
	No Rest.	5976	6215	12191
	∑	13792	13919	27711
	Rest.	**50.4%**	49.6%	56.0%
	No Rest.	49.0%	**51.0%**	44.0%
	Precision:	56.7%		
	Recall:	50.3%		
	Accuracy:	50.6%		

Top: TR-AccountsPayable, TR-NetIncome, TR-TtlPlanExpectedReturn, $\varepsilon = 0.01$, minPts = 5. Bottom: FF-CH_CS, FF-CH_EMP, FF-CFF, FF-TAX, $\varepsilon = 0.1$, minPts = 5.

In Table 1, 2, 3 and 4 the best results after iterating all possibilities are shown for the four restatement definitions from 4.1. In this paper, results are considered as *good* if the hit rates are higher than the error values, and the precision and recall are near to their maximum. Unfortunately, many (original and processed) values are missing, so that only combinations with at least 10000 observations are discussed in the following.

Table 1 shows the results for $Definition_{all}$. The hit rates are highlighted using bold font. With this definition both, the original values from EIKON as well as the processed data lead to good results. It is notable, that with $Definition_{all}$ combined with the best settings, the restatements form the majority with 64.4% and 56.0%. The classification into restatement and non-restatement is balanced. This means that the model does not only focus on one class. The precision indicates that the model detects restatements with a certainty of 65.6% with the original data and 56.7% with the processed features. [3] use 7 different features (variables, model 1) and report a precision of $\frac{339}{48621} = 0.7\%$. Of course, the values are not directly comparable. One reason are the different data sets and the definition of the outcome variables (restatements vs. accounting and enforcement actions). On the other hand, [3] achieve a better recall with 68.6%. This means that they are more likely to detect restatements. However, this may be due to the fact that the number of restatements was smaller. Although there is a gap between the achieved precision with the original data and the processed proxies, both outcomes are competitive regarding the results in [3], as the recall is only slightly lower, but the precision is significantly higher.

Nevertheless, these results are not fully comparable due to different definitions.

Table 2. Results for *Definition*$_{relevant}$.

Original Data	**obs. \ pred.**	**Rest.**	**No Rest.**	**Σ**
	Rest.	2938	2487	5425
	No Rest.	2639	2644	5283
	Σ	5577	5131	10708
	Rest.	**54.2%**	45.8%	50.7%
	No Rest.	33.4%	**66.6%**	49.3%
	Precision:	52.7%		
	Recall:	54.2%		
	Accuracy:	52.1%		
Processed Data	**obs. \ pred.**	**Rest.**	**No Rest.**	**Σ**
	Rest.	5911	5725	11636
	No Rest.	7881	8194	16075
	Σ	13792	13919	27711
	Rest.	**50.8%**	49.2%	42.0%
	No Rest.	49.0%	**51.0%**	58.0%
	Precision:	42.9%		
	Recall:	50.8%		
	Accuracy:	50.9%		

Top: TR-NetSales, TR-TtlPlanExpectedReturn. Bottom: FF-CH_CS, FF-CH_EMP, FF-TAX. In both cases ε = 0.01 and minPts = 5.

The results for *Definition*$_{relevant}$ can be seen in Table 2. The original values as well as the processed proxies achieved good results with subsets of two and three features. The best performance, however, has been reached with the original data using two values: *TR-NetSales* and *TR-TtlPlanExpectedReturn*.

The data set and classification are nearly balanced. Nevertheless, it can be observed that non-restatements are better identified than restatements. Although only two original values are used, the hit rates can compete with those of [3]. Furthermore the precision is again significantly better. Using three proxies from [3] the hit rates are around 50%. However, we reach better precision values.

In Table 3 the results for *Definition*$_{relevant5\%}$ are shown. The calculations show that original values as well as the processed ones achieve good results with subsets of two up to five elements. The pure financial ratios again delivered better scores than the processed proxies. This time restatements have a higher hit rate than non-restatements in both cases. The hit rates of the processed data are similar to the ones for *Definition*$_{all}$ in Table 1.

Last but not least the results for *Definition*$_{audit}$ are shown in Table 4. The best settings are very similar to the ones regarding *Definition*$_{relevant5\%}$. One reason for this could be that both definitions are quite granular. Only 13.8% of the data (15.3% respectively)

is considered as restatement. It is striking that for the first time better results can be achieved when using the processed data with a precision of 16.4%. However, the difference between the results of the different data is not very considerable.

One notable finding is that precision decreases for more granular or strict definitions of restatements. Hereby, the recall slightly increases. In our approach, the popular proxies for the processed

Table 3. Results for *Definition*$_{relevant5\%}$.

Original Data	**obs. \ pred.**	**Rest.**	**No Rest.**	∑
	Rest.	2077	1741	3818
	No Rest.	4129	4149	8278
	∑	6206	5890	12096
	Rest.	**54.4%**	45.2%	31.6%
	No Rest.	49.9%	**50.1%**	68.4%
	Precision:	33.5%		
	Recall:	54.4%		
	Accuracy:	51.5%		
Processed Data	**obs. \ pred.**	**Rest.**	**No Rest.**	∑
	Rest.	3754	3308	7062
	No Rest.	10038	10611	20649
	∑	13792	13919	27711
	Rest.	**53.2%**	46.8%	25.5%
	No Rest.	48.6%	**51.4%**	74.5%
	Precision:	27.2%		
	Recall:	53.2%		
	Accuracy:	51.8%		

Top: TR-AccountsPayable, TR-NetIncome, TR-TtlPlanExpectedReturn, ε = 0.01, minPts = 5. Bottom: FF-CFF, FF-CH_CS, FF-CH_EMP, FF-TAX, ε = 0.1, minPts = 5.

data generally perform worse than the original data for the restatements extracted from EIKON. Only when considering the Audit Analytics information, the processed data scores slightly better. Altogether, our model performs best on a broader definition with the original data (financial ratios). It could be shown, that our approach works better with the original data than with economic proxies, so that in total only up to four features are necessary to achieve the shown results.

5. CONCLUDING REMARKS

In this paper, we introduced a modified, dynamic version of the DBSCAN clustering algorithm. We use this algorithm to detect financial restatements. Overall, our approach is highly efficient. We reach more than 50% accuracy with just two or three features. Remarkably, the modified version of DBSCAN performs particularly in detecting restatement years as compared to non-restatement years.

Our results should be of interest to practitioners and standard-setters. We demonstrate that it is not the amount of data alone but the data processing method that can make a difference. Furthermore, we would like to point to the difficulty of assessing the superiority of one approach to another based on different evaluation criteria. Whereas our approach scores low values of

accuracy compared to [3], it is much better suited to identify restatements (precision).

One major shortcoming of our paper is the limited knowledge about the generalizability of results. More testing is required to analyze

the dependence of the results on the characteristics of the underlying sample as well as deriving evidence on the predictive power of the results.

Table 4. Results for *Definition*$_{audit}$.

Original Data	obs. \ pred.	Rest.	No Rest.	Σ
	Rest.	882	789	1671
	No Rest.	4739	5686	10425
	Σ	5621	6475	12096
	Rest.	**52.8%**	47.2%	13.8%
	No Rest.	45.5%	**54.5%**	86.2%
	Precision:	15.7%		
	Recall:	52.8%		
	Accuracy:	54.3%		
Processed Data	**obs. \ pred.**	**Rest.**	**No Rest.**	**Σ**
	Rest.	2264	2000	4264
	No Rest.	11563	12125	23688
	Σ	13827	14125	27952
	Rest.	**53.1%**	46.9%	15.3%
	No Rest.	48.8%	**51.2%**	84.7%
	Precision:	16.4%		
	Recall:	53.1%		
	Accuracy:	51.5%		

Top: *TR-AccountsPayable, TR-NetIncome, TR-TtlPlanExpectedReturn*, $\varepsilon = 0.01$, *minPts* = 4. Bottom: *FF-CH_CS, FF-CH_EMP, FF-TAX*, $\varepsilon = 0.05$, *minPts* = 3.

6. ACKNOWLEDGMENTS

This work was partly supported by the Jürgen Manchot Foundation which funds the research Group *Decision-making with the help of Artificial Intelligence* at HHU Düsseldorf.

7. REFERENCES

[1] AuditAnalytics: https://www.auditanalytics.com

[2] AuditAnalytics: 2017 financial restatements review, https://www.auditanalytics.com/blog/2017-financial-restatements-review/

[3] Dechow, P., Ge, W., Larson, C., Sloan, R.: Predicting material accounting misstatements. *Contemporary Accounting Research* 28, 1 (2010), 17–82.

[4] Dutta, I., Dutta, S., Raahemi, B.: Detecting financial restatements using data mining techniques. *Expert Systems with Applications* 90 (2017), 374–393.

[5] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd Int. Conference on Knowledge Discovery and Data Mining* (1996). 226–231.

[6] Jones, J.J.: Earnings management during import relief investigations. *Journal of Accounting Research* 29, 2 (1991), 193–228.

[7] Kothari, S.P., Leone, A.J., Wasley, C.E.: Performance matched discretionary accrual measures. *Journal of Accounting and Economics* 39 (2005), 163–197.

[8] Reuters, T.: Eikon financial analysis and trading software

[9] Roychowdhury, S.: Earnings management through real activities manipulation. *Journal of Accounting and Economics 42* (2006), 335–370

[10] Sievers, S., Sofilkanitsch, C.: Financial Restatements: Trends, Reasons for Occurrence, and Consequences. A Survey of the Literature.

APPENDIX

Table A1. List of features included

Fields		Source
FF-CFF FF-CH_CM FF-CH_EMP FF-CH_INV FF-CH_REC FF-EXFIN FF-LEASEDUM FF-OPLEASE FF-RSST_ACC FF-TAX	FF-CH_BACKLOG FF-CH_CS FF-CH_FCF FF-CH_PENSION FF-CH_ROA FF-ISSUE FF-LEVERAGE FF-PENSION FF-SOFT_ASSETS FF-WC_ACC	Dechow et al. [3]
FF-CFOError FF-DISEXPError FF-PRODError	FF-COGSError FF-INVError	Roychow-dhury [9]
FF-ACC_JONESError		Jones [6]
FF-ACC_KOTHARIError		Kothari et al. [7]
TR-Employees TR-LTDebt TR-LTDebtIssued TR-LTDebtNet TR-LTInvestments TR-NetIncome TR-TaxDefTot TR-Revenue TR-NetSales TR-TotalEquity TR-TotalInventory TR-ValueBacklog	TR-NetIncomeAfterTaxes TR-NetIncomeBeforeTaxes TR-PreferredStockNet TR-SgaExpenseTotal TR-ShortTermInvestments TR-TotalCurrLiabilities TR-TotalLiabilities TR-TotalOperatingExpense TR-TotalReceivablesNet TR-TtlPlanExpectedReturn TR-TtlPreferredSharesOut TR-CostOfRevenue	EIKON [8]
TR-ResearchAndDevelopment TR-SaleIssuanceOfCommonPreferred TR-TotalEquityAndMinorityInterest TR-TotalOperatingLeasesSuppl TR-AdvertisingExpense TR-CapitalExpenditures TR-CapitalLeaseObligation TR-CashandEquivalents TR-CashAndSTInvestments TR-CashFromFinancingAct TR-CashFromOperatingAct TR-CommonStockNet TR-CostOfRevenueTotal TR-DepreciationAmort TR-IncomeTaxesPayable TR-LTDebtMaturingYear1 TR-PptyPlantEqpmtTtlGross TR-SaleIssuanceOfCommon		EIKON [8]

4.2 Evaluating Machine Learning Algorithms in Predicting Financial Restatements

---

Gerhard Klassen, Martha Tatusch, Weisong Huo, and Stefan Conrad. Evaluating Machine Learning Algorithms in Predicting Financial Restatements. In *Proceedings of the 4th International Conference on Business and Information Management*, ICBIM '20, New York, NY, USA, 2020. Association for Computing Machinery .

Contributions: The idea for this paper came from Martha Krakowski (née Tatusch) and Gerhard Klassen . Weisong Huo implemented the algorithms under the guidance of Gerhard Klassen. The paper was written entirely by Gerhard Klassen. Martha Krakowski (née Tatusch) was mostly involved in an advisory capacity. Prof. Dr. Stefan Conrad supervised the work.

Status: published

In developing our outlier detection procedures, the question arises as to how well our procedures actually detect certain outliers. However, this evaluation alone is not very meaningful if you do not compare it with other methods already available. For this reason, we have applied common machine learning methods to the same problem in the paper [R7]. This paper serves as a reference and allows the evaluation of the results from our outlier detection methods.

In the referenced paper [R7], we apply six well-known machine learning methods to corporate financial data obtained from [47]. We present the results and additionally show the influence of XGBoost [13], a featureboost algorithm, to the results.

Evaluating Machine Learning Algorithms in Predicting Financial Restatements

Gerhard Klassen
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
klassen@hhu.de

Martha Tatusch
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
tatusch@hhu.de

Weisong Huo
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
weisong.huo@hhu.de

Stefan Conrad
Department of Computer Science
HHU Düsseldorf
Düsseldorf, Germany
stefan.conrad@hhu.de

ABSTRACT
The identification of financial statements which were willfully or accidentally misstated is important for all involved parties: Investors can expect improved returns, analysts preserve their reputation and auditors avoid costly litigation. In this paper, we chose six state-of-the-art machine learning methods which we analyze in their ability to detect misstatements. In addition to that we investigated the influence of a FeatureBoost algorithm, namely XG-Boost to all of the six machine learning methods. The underlying data is retrieved from Eikon [6], a financial database provided by Refinitiv (former provided by Thomson Reuters). In order to take out our experiments we chose about 9000 US-companies and 757 features per year over ten years. We offer six definitions of ground truth of which three can be calculated with the data extracted from the Eikon database. The other three definitions are created with the help of an external data source provided by Audit Analytics Europe [8]. Our well structured results give an overview on the performance of current machine learning methods in order to identify misstatements.

CCS Concepts
•Computing methodologies→Machine learning→Machine learning approaches→Classification and regression trees

•Computing methodologies→Machine learning→Machine learning algorithms→Ensemble methods→Boosting

•Computing methodologies→Machine learning→Machine learning algorithms→Feature selection

•Applied computing→Law, social and behavioral sciences→Economics

Keywords
Classification, Ensemble Methods, Financial Restatements

1. INTRODUCTION
"One tiny drop changes everything" was the advertising slogan of Theranos, founded in 2003, promising a technology which would detect cancer with only a drop of blood. In 2018 it revealed to be one of the most scandalous fraud cases of the last century. Although the whole scope was not known to publicity immediately, it was assumed that Theranos got a long history of misstating their finances. The SEC confirmed this later with a press release [15]. At this point the harm was already done: Reputations were forever damaged, billions of Dollars were burned and the hope in the advertised technology destroyed. The story of Theranos is not unique. Similar stories could be told about the accounting scandals of Enron, WorldCom, Tyco and many other fraudulent companies. While reading about these cases one question comes to the fore: "Couldn't this have been predicted?". There are many different perspectives and approaches to answer this question. Since various analysts and investors were deceived by the fraudsters for several years, one approach may be the use of artificial intelligence. However, not every AI method is suitable for this task, since the reason for a classification of a misstatement is at least as important as the classification itself. Finally, false detection of misstatements could also cause damage in many ways. Hence, in this paper we do not investigate the performance of neural networks, since these got a black-box character, which is a subject of current research.

Although fraud is one motivation for misstatements, it only represents a small fraction of companies. In fact, most false statements happen due to human made mistakes [16]. While some mistakes are detected and corrected quickly, others cause huge damage similar to fraudulent statements [16].

It is undoubted, that the detection of misstatements is an important field of research for all involved parties. The information that a company misstated a financial statement can make a huge difference in investment decisions. It also got a high impact on the market, especially if a misstatement was done willingly. However, the detection of false statements remains to be a difficult task, especially when trying to detect those automatically. The problem begins with a definition of a misstatement. While detected false statements are forced to be restated, unveiled ones remain hidden. This is a difficulty when applying supervised learning algorithms, which require a labeled training set. In this paper we present six machine learning algorithms of which five are supervised and one is unsupervised. All algorithms are taken out with and without

XGBoost [3], which is a representative for feature-boost algorithms. In order to train the supervised models we introduce six different definitions of misstatements, all based on restatements. We analyze about 9000 US-firms with 757 features per year from 1998 to 2017. We retrieved our data from Eikon [6], which is provided by Refinitiv (former provided by Thomson Reuters) and Audit Analytics Europe [8].

2. RELATED WORK

Although there are several other works which use machine-learning techniques in order to identify misstatements, none of them analyses the impact of feature-boost algorithms. Actually most of them like [4,5] use feature-sets selected by domain specialists. Being aware of the fact, that the knowledge of domain specialists can enhance the results, we added 28 features from [11]. These features had a great impact in the presented work and we assume that they could also have a positive influence in this work. In contrary to [4] and [5] we use way more features and show the impact of a feature-boost algorithm to the results. Other works try to uncover hidden misstatements [1]but do not apply their model to actual restatements. There are also works which present models for fraud detection [10]. In contrary to [10], we do not use neural networks, because of their black-box character. We assume a higher gain from results which potentially can be explained, since this could also explain false-positives. Finally there are also approaches which regard the problem from the perspective of someone who would manipulate a financial statement. One popular work in this field of research is [14]. Roychowdhury makes use of regression equations in his work. Since we want to evaluate the strengths and weaknesses of machine-learning methods in detecting misstatements, the approach of [14] is not really comparable to our approach. All in all there to the best of our knowledge there is no other work which provides an extensive machine-learning and feature-boost evaluation to the presented dataset.

3. DATASETS AND DEFINITIONS

In this study we make use of two different data sources. The first data source is Eikon [6] provided by Refinitiv (former provided by Thomson Reuters). From Eikon we retrieved 732 financial figures of 9263 companies from 1998 to 2017. Additionally we added 28 features, which were meaningful in [11]. We regard the financial figures as features and use them as input for the machine learning algorithms.

As stated in the introduction, we use five supervised algorithms which require a training phase. In order to realize the training, we require labeled data. For revealed misstatements, namely those which were restated, we offer six definitions. Three of those are calculated with the help of the Eikon data. The other three definitions are based on data retrieved from Audit Analytics Europe [8].

Since we can only evaluate with unveiled and corrected misstatements we make use of financial restatements. Eikon provides two versions for every financial figure: The actual figure stated by the company and a restated figure. In case a firm corrected a number, the restated figure differs from the actual figure. It must be noted though, the reason for the correction is not given by the database. In order to obtain the values the Python Eikon API offers the parameter *ReportingState*, which can be either set to *Orig* (original) or *Rstd* (restated). Audit Analytics Europe differs two different types of restatements. Those which got a positive effect and those which got a negative effect. In the following section we provide all six definitions of misstatements.

4. MODEL DEFINITIONS

In this section we give the six misstatement definitions. For those we solely use restatements, since these are the only misstatements which got revealed and accessible to the public. We define the restatements as follows:

1. **Eikon based definitions**
    a. **all**: If any figure has been restated in a certain year, we label the company to have misstated in this year.
    b. **relevant**: If at least one of the relevant figures has been restated by a company in a certain year, we mark this year as a misstatement for this firm. We consider the following five figures as being relevant: Net income, shareholder's equity, operating cashflow and sales.
    c. **relevant5%**: If at least one of the relevant figures has a restated value which is 5% higher or lower than the actual stated figure, we mark the statement of the to be a misstatement.
2. **Audit Analytics based definitions**
    a. **positive**: The restatement had a positive effect on the originally stated figures.
    b. **negative**: The restatement had a negative effect on the originally stated figures.
    c. **positive or negative**: The original financial statement was restated according to Audit Analytics Europe.

In order to detect the defined misstatements, we make use of six machine learning methods. Additionally, we analyze the impact of XGBoost [3], a feature-boost algorithm to the results. Three of the applied machine learning methods are classic algorithms: The K-Nearest-Neighbor classification algorithm (KNN), the Support Vector Machine (SVM) [17] and the Decision Tree [12]. In the following we denote these models as *simple*. The other three machine learning algorithms are so called *ensemble methods*. In concrete that means, that these are algorithms which combine the results of several classifiers. Therefor we applied the Random Forest [2], the Isolation Forest [18] and AdaBoost [9].

5. EVALUATION

In order to evaluate the performance of the machine learning methods we make use of a three-fold cross-validation and use the common measures, namely precision, recall and the f1-score. First we will have a look on the performance of the machine learning algorithms without applying XGBoost.[3]. Then we present the results with XGBoost applied before using the classifiers. In some cases some results would not give further insight, this is why we left those out. This applies especially to the first two restatement definitions of every data source. Note, that the label in the tables represent the two classes *restatement* (=1) and *no-restatement* (=0). Another important remark is that we did not tune the parameters of the machine-learning algorithms. Instead we used the proposed standard parameters from scikit-learn, a python machine-learning library [13].

5.1 Evaluation without Feature Boost

In this section we present the results without XGBoost being applied priorly. In Table 1 one can see the results for the first three classic models applied on all 760 features. It can be clearly seen, that the K-Nearest-Neighbor algorithm outperformed the other algorithms, although the amount of false negatives (Type II error)

is extremely high. The high precision of the SVM in this classification task can be explained with the imbalanced dataset. The SVM is actually predicting almost every data point as being a non-restatement.

Table 1. Results of simple models regarding the restatement definition *all*.

Algorithm	Label	Precision	Recall	F1-score
KNN	0	0.89	0.91	0.90
	1	0.66	0.61	0.64
Decision Tree	0	0.79	0.98	0.88
	1	0.60	0.13	0.21
SVM	0	0.78	1.00	0.87
	1	0.97	0.00	0.00

In Table 2 one can observe the results of the *simple* methods for the restatement definition *relevant*. Although the definition is stricter than the *all* definition, the results can be compared. The K-Nearest-Neighbor algorithm is again outperforming the other methods. It is also the one which actually detects the most misstatements.

Table 2. Results of simple models regarding the restatement definition *relevant*.

Algorithm	Label	Precision	Recall	F1-score
KNN	0	0.91	0.95	0.93
	1	0.63	0.49	0.55
Decision Tree	0	0.86	0.98	0.91
	1	0.57	0.16	0.25
SVM	0	0.84	1.00	0.91
	1	0.90	0.00	0.00

The last Eikon based definition is also the strictest. The results of the three *simple* algorithms can be seen in Table 3. All algorithms perform worse with this restatement definition, especially the Decision Tree tends to classify all data points as being no restatements. This leads to an extremely high precision and an even higher recall regarding the firm years which were labeled as no restatement.

Table 3. Results of simple models regarding the restatement definition *relevant5%*.

Algorithm	Label	Precision	Recall	F1-score
KNN	0	0.93	0.98	0.95
	1	0.57	0.31	0.40
Decision Tree	0	0.91	1.00	0.95
	1	0.00	0.00	0.00
SVM	0	0.91	1.00	0.95
	1	0.80	0.00	0.00

Restatements retrieved from Audit Analytics Europe [8] have an even worse detection ratio than the Eikon based definitions. Neither the restatements with a *positive*, nor the restatements with a *negative* effect can be detected well by any of the three *simple* methods. Actually all algorithms tend to classify almost every firm year to be stated correctly. This is why we did not show the results here. The only acceptable result is achieved by the K-Nearest-Neighbor algorithm and the *positive or negative* (Table 4) definition of restatements, although 1916 misstatements were not detected as such.

Table 4. Results of simple models regarding the restatement definition *positive or negative*.

Algorithm	Label	Precision	Recall	F1-score
KNN	0	0.94	0.99	0.96
	1	0.44	0.12	0.18
Decision Tree	0	0.93	1.00	0.96
	1	0.00	0.00	0.00
SVM	0	0.93	1.00	0.96
	1	0.00	0.00	0.00

Overall the ensemble methods perform better, in particular the Isolation Forest is outperforming every other algorithm. In Table 5 it can be seen, that with the strictest Eikon based definition *relevant5%* the Isolation Forest also outperforms the K-Nearest-Neighbor algorithm. This is also the case for all other Eikon based definitions. The other two ensemble methods show similar performance as the *simple* algorithms.

Table 5. Results of ensemble models regarding the restatement definition *relevant5%*.

Algorithm	Label	Precision	Recall	F1-score
Random Forest	0	0.91	1.00	0.95
	1	0.00	0.00	0.00
Isolation Forest	0	0.89	0.98	0.93
	1	0.89	0.54	0.67
AdaBoost	0	0.92	0.99	0.95
	1	0.54	0.13	0.21

The Isolation Forest also performs better with the *positive or negative* restatement definition, retrieved from Audit Analytics Europe. Comparing Table 4 and Table 6, one can see the Isolation Forest again outperforms the K-Nearest-Neighbor algorithm. Regarding the *positive* and *negative* definitions of restatements, the Isolation Forest has a similar performance to the *positive or negative* definition.

4.2 Evaluation with Feature Boost

In this subsection we present the feature-boosted results of the six machine-learning methods. XGBoost [3] selected only 93 of the 757 features. However, unlike one would expect this does not influence the results significantly. As you can see in Table 7, KNN profits the most by XGBoost, regarding the restatement definition *relevant5%*. Although it is losing one percent of the recall at thenon-restatement firm-years, it is gaining three percent in the classification of misstatements. Regarding the other Eikon based restatement definitions, the results are pretty similar to the one in Table 8 The Audit Analytics Europe definition of a restatement (*positive or negative*) has still a poor detection rate with the *simple*

machine learning methods. In Table 8 one can see that KNN has the maximum gain, which is three percent at detecting misstatements.

Table 6. Results of ensemble models regarding the restatement definition *positive or negative*.

Algorithm	Label	Precision	Recall	F1-score
Random Forest	0	0.93	1.00	0.96
	1	0.33	0.00	0.00
Isolation Forest	0	0.87	0.98	0.92
	1	0.75	0.26	0.38
AdaBoost	0	0.93	1.00	0.96
	1	0.00	0.00	0.00

.

Table 7. Results of simple models with XGBoost applied, regarding the restatement definition *relevant5%*.

Algorithm	Label	Precision	Recall	F1-score
KNN	0	0.93	0.97	0.95
	1	0.56	0.34	0.42
Decision Tree	0	0.91	1.00	0.95
	1	0.00	0.00	0.00
SVM	0	0.91	1.00	0.95
	1	0.81	0.00	0.00

Table 8. Results of simple models with XGBoost applied, regarding the restatement definition *positive or negative*.

Algorithm	Label	Precision	Recall	F1-score
KNN	0	0.94	0.99	0.96
	1	0.46	0.15	0.22
Decision Tree	0	0.93	1.00	0.96
	1	0.00	0.00	0.00
SVM	0	0.93	1.00	0.96
	1	0.00	0.00	0.00

There is also a rather small influence on the ensemble methods. In Table 9, one can see that the Isolation Forest slightly profits by the prior application of XGBoost, while AdaBoost has worse results than without priorly applied the feature-boost algorithm. However, the change is not significant and accounts maximum to only +0.04 for the Isolation Forest and the recall of misstatements and -0.03 for precision of the misstatements for AdaBoost.

Feature-boosting with XGBoost [3] has its highest impact on ensemble methods in combination with the *negative or positive* restatement definition. Comparing Table 6 and Table 10 one can see, that the impact on the precision of the Isolation Forest in detecting restatements is 0.07 higher with XGBoost than without it. Although the amount of detected misstatements is still very low, the precision of detecting them is also higher for the Random Forest, if applying XGBoost first.

Table 9. Results of ensemble models with XGBoost applied, regarding the restatement definition *relevant5%*.

Algorithm	Label	Precision	Recall	F1-score
Random Forest	0	0.91	1.00	0.95
	1	0.00	0.00	0.00
Isolation Forest	0	0.90	0.98	0.94
	1	0.91	0.58	0.70
AdaBoost	0	0.92	0.99	0.95
	1	0.51	0.12	0.19

Table 10. Results of ensemble models with XGBoost applied, regarding the restatement definition *positive or negative*.

Algorithm	Label	Precision	Recall	F1-score
Random Forest	0	0.93	1.00	0.96
	1	0.45	0.00	0.00
Isolation Forest	0	0.87	0.99	0.93
	1	0.82	0.26	0.39
AdaBoost	0	0.93	1.00	0.96
	1	0.00	0.00	0.00

5. CONCLUDING REMARKS

Our extensive evaluation has shown that the detection of misstatements of any definition presented in this paper is a difficult task. The strictness of the restatement definition has a high impact on the performance of the machine learning algorithms. Especially the KNN algorithm produced worse results, the stricter the restatement definition was. Beside the Isolation Forest, all ensemble methods were also struggling with this classification task. Our assumption is, that the reason for the results is the highly unbalanced dataset. The stricter the restatement definition becomes, the less firm-years are labeled as actual misstatements. This makes some algorithm classify all firm-years as good stated, as this is the majority class.

According to the results, the impact of feature-boosting with XGBoost [3] was rather small. However, if the same results can be achieved with 93 of 757 features this has a high impact on the runtime of the machine-learning algorithms. In addition to that the last experiment with the ensemble methods and the restatement definition *positive or negative* has shown that XGBoost actually can boost the results by a two digit number.

5. FUTURE WORK

Detecting restatements is an important task for all involved parties. As this survey has shown, the results have plenty of air at the top. In our opinion, the usage of neural networks is no alternative, since it is hardly possible to get insight to the decision process. In the future we would like to see other machine learning methods to be applied on the presented combination of data. These could be other clustering algorithms, like DBScan [7] or classification algorithms like Naïve Bayes.

6. ACKNOWLEDGMENTS

This work was partly supported by the Jürgen Manchot Foundation by funding the research Group *Decision-making with the help of Artificial Intelligence* at HHU Düsseldorf.

7. REFERENCES

[1] J Bertomeu, E Cheynel, E Floyd, and W Pan. 2018. Ghost in the Machine : Using Machine Learning to Uncover Hidden Misstatements. (2018), 1–32.

[2] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794.

[4] Patricia M. Dechow, Weili Ge, Chad R. Larson, and Richard G. Sloan. 2011. Predicting Material Accounting Misstatements*. *Contemporary Accounting Research* 28, 1 (2011), 17–82.

[5] Ila Dutta, Shantanu Dutta, and Bijan Raahemi. 2017. Detecting financial restatements using data mining techniques. *Expert Systems with Applications* 90 (2017), 374–393.

[6] Thomson Reuters Eikon. 2018. Retrieved February 1, 2018 from https://eikon.thomsonreuters.com/index.html. (2018).

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd Int. Conference on Knowledge Discovery and Data Mining* (1996), 226–231.

[8] Audit Analytics Europe. 2020. Retrieved December 12, 2019 from https://eikon.thomsonreuters.com/index.html. (2020).

[9] Yoav Freund and Robert E Schapire. 1999. A Short Introduction to Boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann,* 1401–1406.

[10] Chyan Long Jan. 2018. An effective financial statements fraud detection model for the sustainable development of financial markets: Evidence from Taiwan. *Sustainability (Switzerland)* 10, 2 (2018).

[11] B Brian Lee and William Vetter. 2015. Critical Evaluation of Accrual Models in Earnings Management Studies. *Journal of accounting and Finance* 15, 1 (2015), 62–72.

[12] Dan H. Moore. 1987. Classification and regression trees, by Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone.. *Cytometry* 8, 5 (1987), 534–535.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

[14] Sugata Roychowdhury. 2006. Earnings management through real activities manipulation. *Journal of Accounting and Economics* 42, 3 (2006), 335–370. https://doi.org/10.1016/j.jacceco.2006.01.002

[15] U.S. Securities and Exchange Commission. 2020. Theranos, CEO Holmes, and Former President Balwani Charged With Massive Fraud, Retrieved March 12, 2020 from https://www.sec.gov/news/press-release/2018-41. (2020).

[16] Soenke Sievers and Christian Sofilkanitsch. 2018. Financial Restatements: Trends, Reasons for Occurrence, and Consequences - A Survey of the Literature. *SSRN Electronic Journal* (2018).

[17] Alex J. Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. , 199–222 pages.

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest In *2008 Eighth IEEE International Conference on Data Mining* (2008).

5

Conclusion and Future Work

5.1 Overall Summary

Gerhard Klassen, Martha Tatusch, and Stefan Conrad. Cluster-Based Stability Evaluation in Time Series Data Sets. 2021.

Contributions: The presented methods were developed in earlier papers. The new experiments concerning the time series cluster evaluation were carried out by Gerhard Klassen. The real world data sets were aquired and preprocessed by Gerhard Klassen. The manuscript was written in equal parts by the two main authors under the supervision of Prof. Dr. Stefan Conrad. The paper has been submitted to ACM Transactions on Knowledge Discovery from Data (TKDD) in May 2021.

Status: submitted

In the paper referenced in this section [R2], we once again establish the connection between the evaluation procedures and the applications based on them. In principle, the most stable clustering is first sought for an application based on CLOSE or FCSETS. In the following paper, different clustering methods are evaluated and their results are presented. In addition, we present a new data set, the COVID-19 data from Hopkins University [18], which, when paired with our methods, provides interesting insights into the global evolution of the pandemic.

Cluster-Based Stability Evaluation in Time Series Data Sets

GERHARD KLASSEN*, MARTHA TATUSCH*, and STEFAN CONRAD, Heinrich Heine University, Germany

In modern data analysis, time is often considered just another feature. Yet time has a special role that is regularly overlooked. Procedures are usually only designed for time-independent data and are therefore often unsuitable for the temporal aspect of the data. This is especially the case for clustering algorithms. Although there are a few evolutionary approaches for time-dependent data, the evaluation of these and therefore the selection is difficult for the user. In this paper, we present a general evaluation measure that examines clusterings with respect to their temporal stability and thus provides information about the achieved quality. For this purpose, we examine the temporal stability of time series with respect to their cluster neighbors, the temporal stability of clusters with respect to their composition, and finally conclude on the temporal stability of the entire clustering. We summarise these components in a parameter-free toolkit that we call **Cl**uster **O**ver-Time **S**tability **E**valuation (CLOSE). In addition to that we present a fuzzy variant which we call FCSETS (**F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries). These toolkits enable a number of advanced applications. One of these is parameter selection for any type of clustering algorithm. We demonstrate parameter selection as an example and evaluate results of classical clustering algorithms against a well-known evolutionary clustering algorithm. We then introduce a method for outlier detection in time series data based on CLOSE. We demonstrate the practicality of our approaches on two real world data sets and one generated data set.

CCS Concepts: • **Computing methodologies** → **Anomaly detection**; • **Information systems** → *Clustering*; • **Mathematics of computing** → **Time series analysis**; **Cluster analysis**.

Additional Key Words and Phrases: Time Series Clustering, Over-Time Stability Evaluation, Anomalous Subsequences

ACM Reference Format:
Gerhard Klassen, Martha Tatusch, and Stefan Conrad. 2021. Cluster-Based Stability Evaluation in Time Series Data Sets. *ACM Trans. Knowl. Discov. Data.* 0, 0, Article 0 (August 2021), 27 pages. https://doi.org/DOI...

1 INTRODUCTION

With the increase of time series (TS) data, their analysis is becoming more and more important. There are many different approaches which are all suitable for different setups. However, most of the methods target the analysis of individual time series, while only a few aim to analyse whole TS databases. Without any doubt, the information gained from a time series database can have a significant influence on the results, especially compared to an analysis applied to only one time series of the database.

*Both authors contributed equally to this research.

Authors' address: Gerhard Klassen, gerhard.klassen@hhu.de; Martha Tatusch, martha.tatusch@hhu.de; Stefan Conrad, stefan.conrad@hhu.de, Heinrich Heine University, Universitätsstraße 1, Düsseldorf, Germany, 40225.

A setting which illustrates this circumstance is the stock market: During an economic crisis most of the shares lose value. Regarding only one share at a time could lead to a false interpretation (e.g. an outlier sequence within the time series), while regarding all time series simultaneously the assessment would result differently.

Although the mentioned setting describes extreme circumstances, it is obvious that similar problems in analysis and interpretation also occur under normal conditions. The examination of this kind of setups can prove to be very difficult, since it can be useful not to look at the whole database at once, but to look at specific groups instead. This requires the identification of groups which is often accomplished by applying suitable clustering algorithms. Although this is a well researched topic for time independent data, approaches for time series are often insufficient, sometimes to an extent that the produced results are meaningless [21]. As this has been identified as a major problem in time series clustering, the research field *evolutionary clustering* developed. According to [7] evolutionary clustering is producing a clustering per timestamp, hence a series of clusterings. Each clustering should be similar to the clustering of its predecessor, while accurately reflecting the properties of its own data. As this definition is not regarding a certain clustering algorithm, this leads to a variety of approaches adapted to different clustering algorithms (read more about this in Section 2). There are also approaches which try to define the necessary adjustments to a standard clustering algorithm to receive an evolutionary clustering algorithm [7, 9]. However, the amount of different approaches and different clustering algorithms makes it difficult to select a suitable method for a certain task.

The detection of groups in time series can provide important insights into the data at hand. The application areas of our toolkits and the methods based on them, such as outlier detection, are diverse. One conceivable application of our methods is on medical data, where patients could be identified who were initially grouped with healthy patients and whose medical values then slowly move away from this group. Another area of application is the financial market, where, for example, companies can be grouped that behave similarly over time, so that classic company classifications such as the Standard Industrial Classification (SIC) or the North American Industry Classification System (NAICS) can be usefully supplemented. Companies that change their group more frequently in relation to other companies may have anomalies that our outlier detection method would identify. Analyses of the current Corona epidemic are also conceivable. Using data from the Coronavirus Government Response Tracker at the University of Oxford [1], the effectiveness of government measures to contain the epidemic could be analysed. It would also be possible to identify how good the respective chosen timing of a measure was. There are countless applications where our methods can be used. In addition, all our methods are transparent and provide explainable results.

In this paper we describe two fundamental methods to evaluate time series clustering according to its over-time stability. The first algorithm CLOSE (**Cl**uster **O**ver-Time **S**tability **E**valuation) [46] is designed for multivariate time series in crisp cluster environments. Hereby we use an extended definition of evolutionary clustering: Instead of targeting the similarity of two successive clusterings we demand the similarity of a clustering to all previous clusterings. We call this the *over-time stability* and introduced it, because small changes between two timestamps could develop to huge changes over several time steps. Those changes would be overseen by considering only two consecutive timestamps. A simple example for this problem is the Covid-19 infection rate in different countries: If one country changes its cluster peers from one point in time to the other, this may be reasonable. However, if the country is changing its cluster peers in every time point, regarding solely the previous timestamp is not sufficient, since it does not hold the historical changes. Therefore this country could not be directly compared to other countries, since among those there

[1] https://www.bsg.ox.ac.uk/research/research-projects/coronavirus-government-response-tracker

might be countries which have changed its peers as well. Hence, the changes before the previous timestamp contribute to the overall stability.

In the course of this paper we will show that this adaptation of the definition is especially handy for certain applications like outlier detection or parameter selection for time series clustering. Our methodology is also very different from other approaches in this field of research. In contrary to a framework or an adapted clustering algorithm, CLOSE is a ready-to-use toolkit. It does not require any customization of the user-chosen clustering algorithm, instead it analyses the produced clusterings per timestamp and returns a stability score. This can be used to find the best parameter setting for the underlying clustering algorithm.

The second algorithm FCSETS (**F**uzzy **C**lustering **S**tability **E**valuation of **T**ime **S**eries) [26] is a toolkit developed for fuzzy clustering environments. It makes use of the relative assignment agreement similar to the equivalence relation in the Hüllermeier-Rifqi Index [17] and achieves a stability score by regarding the average weighted difference between the relative assignment agreements of one time series to the others. The methodology of FCSETS is very similar to the one of CLOSE and therefore further adjustments of the chosen underlying fuzzy clustering algorithm are not required. Further we are presenting an outlier detection algorithm [45] which is an application of CLOSE. We give two variants [47] of the procedure which focus on cluster transitions and therefore are capable to detect a new sort of outliers, which are based on the *behavior* of time series in relation to its cluster peers. The implementation of the approaches as well as the generated data sets are available on Github[2].

In order to present the results of the introduced algorithms we use two real world data sets and one generated data set. We apply CLOSE in combination with DBSCAN [13] and K-Means [33] and FCSETS in combination with Fuzzy C-Means [5] to the selected data sets to get the best parameter settings. We qualitatively analyse the resulting clusterings and in the case of K-Means we subsequently use the possibility to compare the CLOSE score with that of the evolutionary K-Means from [7]. Further, we apply the outlier detection algorithms to the data sets and explain the results in detail.

2 RELATED WORK

Since this work addresses many different problems and approaches, such as the (over-time) stability evaluation of (fuzzy) clusters and the detection of anomalous subsequences, this chapter deals with related works from various domains, as well.

2.1 Time Series Clustering

There are various techniques for clustering TS data in the field of time series analysis. In [51] the approaches are divided into three categories: *raw-data-based*, *feature-based* and *model-based* clustering algorithms. The first type describes approaches, which consider the TS data without any preprocessing. The second one works with feature vectors extracted from the time series. In the third case, models are approximated for the representation of the TS data.

When considering approaches that work with the unprocessed TS data that is given, a common approach is clustering subsequences of a time series [2, 18]. As this is usually done in order to find motifs in time series, only a single TS is considered at once. This approach is controversial, since Keogh et al. state in [21] that the clustering of subsequences of a single time series is meaningless. Chen, however, argues that it is possible to obtain meaningful results if the correct distance measure is used [8]. In our context, the clustering has to be applied to multiple time series, though. Clustering

[2]https://github.com/tatusch/ots-eval

subsequences has some disadvantages. First, outlier data points may have a negative impact on the results. Second, the determination of a meaningful length for the considered subsequences is difficult but needed, since the examination of subsequences of all lengths is usually very time-consuming. In our approaches, subsequences of any length can selectively be investigated and therefore provide more insights. Nevertheless, it has to be noted, that only subsequences starting at the first existing timestamp are considered. This is reasoned by the assumption, that the entire time course from the beginning is relevant for the analysis.

Another raw-data-based approach is the clustering of entire sequences [12, 27, 36]. Since potential correlations between subsequences of different TS are not recognized, this procedure is not suitable for our applications.

In our context, the exact course of time series is not relevant, but rather the trend they follow. This can be achieved by algorithms of the second type, where the sequences are transformed to feature vectors first [16]. By extracting relevant features, the exact course gets blurred. However, the problem of not recognizing correlating subsequences still persists.

When considering the third type of TS clustering, a major approach is the usage of auto-regressive moving-average models (ARIMA) [37, 53]. Therefore, an ARIMA model/mixture for every time series is fitted. Those sequences, whose models are similar to each other, are grouped to the same cluster. Also, the sequences can be modeled by the Haar Wavelet decomposition [49], their approximated seasonality [28] or with the help of Markov Chains [38]. However, all approaches share the idea of clustering whole time series. In our application, correlating subsequences and the movement of sequences with regard to their neighbors are of interest. Therefore, those methods are not applicable.

Approaches, which deal with the clustering of streaming data [15, 35] are also not comparable to our method, as they deal with other problems such as high memory requirements and time complexity, and in addition to that usually consider only one sequence at once.

2.2 Evolutionary Clustering

Evolutionary clustering describes the task of clustering temporal data per timestamp under the consideration of two criteria: on the one hand, the clustering should be reasonable for the current data, and on the other the clustering should not deviate significantly from one timestamp to another [7]. Different frameworks have been developed, which meet both criteria regarding streaming data [9], TS data [54] and dynamic networks [23]. The framework, which is presented in [7], for instance, is developed for streaming data and therefore an incremental approach, which for each timestamp t tries to find a clustering C_t that optimizes the following formula:

$$sq(C_t, M_t) - cp \cdot hc(C_{t-1}, C_t)\,, \tag{1}$$

where $sq(C_t, M_t)$ is the *snapshot quality* regarding an object relationship matrix M_t, cp is a change parameter and $hc(C_{t-1}, C_t)$ is the *history cost*. The *snapshot quality* measures the quality of a clustering at a certain time point with respect to the calculated $n \times n$ matrix M_t which represents the relationship of all n objects to each other. The history cost is calculated by the comparison of the clusterings of two consecutive time points, whereby the comparison may be applied on different data levels. For example, simply the partitions of both clusterings may be compared, or the best matching between two sets of centroids regarding KMeans [33]. The change parameter $cp > 0$ is a hyperparameter which trades off between sq and hc. With this flexible framework a stable over-time clustering may be achieved, which can be used as the underlying clustering for our outlier detection algorithm. Yet, due to the comparison of only consecutive time points, short-term changes may have a strongly negative impact on the result and large long-term changes may occur, which is not desirable.

The problem of identifying so called *Moving Clusters* [19] seems to be a closely related topic, but addresses a slightly different task. In contrast to clustering time series, this field of research deals with the detection of already given clusters that remain mostly the same with regard to their members. In addition, it is assumed that a cluster remains approximately the same size over time. This may apply to some tasks, such as herd tracking, which is examined in [19], but in most cases this requirement can not be met.

2.3 Internal Cluster Evaluation Measures

For the evaluation of clusters and clusterings, various evaluation measures have been developed over the years. There are two types of cluster evaluation measures: *external* and *internal* measures. The difference between the two is, that while the expected result – also known as *ground truth* – is known for the external measures, it is missing for the internal ones. Therefore, external evaluation measures make a qualitative comparison between the expected and the real result. Internal measures, however, focus on other describing characteristics, such as the compactness or separation of clusters in order to evaluate the quality of the result.

One common metric is the *Sum of Squared Errors* (SSE) that evaluates the compactness of clusters. In case of fuzzy clusterings this measure can be used by weighting the membership degrees. The SSE is based on the calculation of the overall distance between the members and the *centroid* of a cluster. The centroid usually describes the mean of all cluster members. Since this measure only considers the compactness of clusters, further validity measures have been developed, which evaluate the compactness as well as the separation. Common examples are the *Silhouette Coefficient* [41], *Davies-Bouldin Index* [10] or *Dunn Index* [11]. When considering fuzzy clusterings, there are for example validity measures which use only membership degrees [24, 31] or include the distances between data points and cluster prototypes [4, 6, 14, 52].

However, all these metrics cannot directly be compared to our method since they lack a temporal aspect, but they can be applied in our stability evaluation methods.

2.4 Stability Evaluation of Clusterings

There are also several approaches addressing the stability measurement of a clustering algorithm. One example is the *Rand Index* [39], which is usually intended for the external evaluation of a clustering. Given the clustering ζ_p and the expected result ζ_t, it examines on the one hand all object pairs that are located in the same cluster in ζ_p as well as ζ_t, and on the other hand all pairs that belong to different clusters in both clusterings. The measure is defined by the number of corresponding object pairs in relation to the number of all possible object pairs.

The measurement of the stability of a clustering algorithm is for instance executed when searching for the optimal parameter setting. In 2002, Roth et. al [40] introduced the resampling approach for cluster validation. Roth et. al put forward the hypothesis, that if multiple partitionings of a clustering algorithm for the same parameter setting are similar to each other, the parameter setting is good. The higher the similarity, the better is the parameter choice.

The *unsupervised cluster stability value* $s(c)$ that is used in Roth et. al's approach [40] is calculated as the average pairwise distance between m partitionings:

$$s(c) = \frac{\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} d(U_{ci}, U_{cj})}{m \cdot (m-1)/2}, \tag{2}$$

where U_{ci} and U_{cj}, $1 \leq i < j \leq m$, are two partitionings produced for c clusters and $d(U_{ci}, U_{cj})$ is an arbitrary similarity index of partitionings. The Rand Index can be used for stability evaluation by including it in this formula. Such stability measures pursue a different objective and obviously do not take a temporal linkage into consideration [50]. Our stability measure is similar to the unsupervised cluster stability value but it includes the temporal dependencies of clusterings. An intuitive idea for achieving a temporal linkage would be to simply compare clustering pairs of successive points in time. This approach would strongly weight variation between two points in time and neglect long-term changes. An ongoing change would for instance be punished only slightly, since consecutive clusterings would be very similar, while short-term deviations would stand out, although the overall behavior might be stable. Also, the index would be strongly negatively affected by separations or merges of clusters of successive time points. Even when comparing clustering pairs of all different time points these problems would persist.

In addition, the referred methods exclusively evaluate the (over-time) stability of clusterings. As stated in [3, 29], however, stability alone is not sufficient for a proper evaluation of a clustering. CLOSE takes both into account, the over-time stability as well as the quality of a clustering, to give an overall rating for an over-time clustering.

2.5 Anomaly Detection in Time Series

When regarding works dealing with outlier detection in time series, various definitions of the term *outlier* can be found. Many approaches consider only single conspicuous data points such as additive outliers or change points [20, 32] and focus on a single time series [1, 34, 43]. However, in our context the detection of anomalous subsequences is considered, so that only algorithms, which either handle outlier subsequences or analyse the group behavior of multiple time series over time, are relevant.

For the latter, approaches such as Probabilistic Suffix Trees (PST) [44], Random Block Coordinate Descents (RBCD) [55] and various neural networks [22] have been developed and been shown to achieve convincing results. However, while these methods examine the deviation of one time series to all others in the data set, we focus on the behavior of a time series compared to its steady neighbors, since the consideration of the whole data set is only meaningful, if all TS have a similar course. This is for example the case in sensor data. In order to analyse the group behavior over time, we first have to identify continuous peers by clustering the TS data per time point. Then, the transitions of sequences between different clusters over time can be analysed. This type of transitions is also evaluated in cluster evolution methods. Landauer et al. [30] make use of such a method in order to calculate a prediction-based anomaly score for a single data point. Similar to our approach, the TS data is clustered per timestamp. The cluster transitions of a considered time series are then analysed by cluster evolution methods in order to approximate a model which predicts the next data point. Although groups of time series are identified, the detection of outliers is therefore based on the prediction of a single sequence. In contrast to Landauer et al. we refer to several time series.

Our approach is very different from clustering whole time series or their subsequences, since in that case the outlier detection relies on the single fact whether a sequence is assigned to a cluster or not. Such an approach does not take the cluster transitions of a sequence into account, which may be an expressive feature on its own. Hence, our approach might recognize anomalous subsequences which in a subsequence clustering would have been assigned to a cluster and therefore not been marked as outlier.

Apart from clustering subsequences, there are also other approaches for the detection of conspicuous subsequences or so called *discords* [21]. Those often consider only a single time series at once. Therefore, only anomalous behavior with regard to the course of one sequence is recognized. Though, in the context of the whole data set, this behavior might for example be normal. Such methods are thus not applicable in our context.

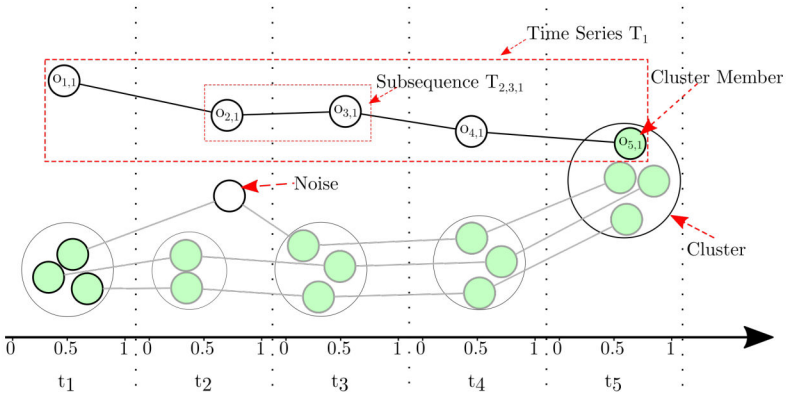


Fig. 1. Illustration of the most important definitions. Lines between objects of a time series represent the development of the sequence [25].

3 METHODOLOGY

The agglomeration of similar time series is a problem which arises in many applications. There are various approaches and a lot of research happened in this field. Since the definitions differ in related works, we first present our notations of relevant concepts for our work. Subsequently, we will describe the principles of our approaches CLOSE [46] and FCSETS [26].

3.1 Notations

The following definitions are based on our previous works [26, 45, 46].

Definition 3.1 (Data Set). A data set $D = \{T_1, ..., T_m\}$ is a set of m time series of same length n and equivalent points in time. Equivalent means, that they are either identical or they can be mapped to a reference timestamp.

Definition 3.2 (Time Series). A time series $T = o_{t_1}, ..., o_{t_n}$ is an ordered set of n real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording, with t_1 and t_n indicating the first and last timestamp, respectively.

The vectors of all time series are denoted as the set $O = \{o_{t_1,1}, ..., o_{t_n,m}\}$, with the second index indicating the time series where this data point originates from. For the ease of reference we write O_{t_i} for all data points at a certain point in time.

Definition 3.3 (Subsequence). A subsequence $T_{t_i,t_j,l} = o_{t_i,l}, ..., o_{t_j,l}$ with $j > i$ is an ordered set of successive real valued data points beginning at time t_i and ending at t_j from time series T_l.

Definition 3.4 (Cluster). A cluster $C_{t_i,j} \subseteq O_{t_i}$ at time t_i, with $j \in \{1, ..., N_C\}$ being a unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm, where N_C is the number of clusters. This means that all clusters have distinct labels regardless of time.

Definition 3.5 (Cluster Member). A data point $o_{t_i,l}$ at time t_i, that is assigned to a cluster $C_{t_i,j}$ is called a member of cluster $C_{t_i,j}$.

Definition 3.6 (Noise). A data point $o_{t_i,l}$ at time t_i is considered as noise, if it is not assigned to any cluster. A data point that belongs to noise is also called an *outlier*. *Noise* describes the set of noise data points of all timestamps, i.e. $Noise = \bigcup_k Noise_{t_k}$.

Definition 3.7 (Clustering). A clustering is the overall result of a clustering algorithm for all timestamps. It is defined by the set $\zeta = \{C_{t_1,1}, ..., C_{t_n,N_C}\} \cup Noise$.

Definition 3.8 (Time Clustering). A time clustering is the result of a clustering algorithm at one timestamp. It is defined by the set $\zeta_{t_k} = \{C_{t_k,a}, ..., C_{t_k,b}\} \cup Noise_{t_k}$ of all clusters at time t_k.

Definition 3.9 (Fuzzy Cluster Membership). The membership degree $u_{C_{t_i,j}}(o_{t_i,l}) \in [0, 1]$ expresses the relative degree of belonging of the data object $o_{t_i,l}$ of time series T_l to cluster $C_{t_i,j}$ at time t_i.

Definition 3.10 (Fuzzy Time Clustering). A fuzzy time clustering is the result of a fuzzy clustering algorithm at one timestamp. It is defined by the membership matrix $U_{t_i} = [u_{C_{t_i,j}}(o_{t_i,l})]$.

Definition 3.11 (Fuzzy Clustering). A fuzzy clustering of time series is the overall result of a fuzzy clustering algorithm for all timestamps. It is defined by the ordered set $U = U_{t_1}, ..., U_{t_n}$ of all membership matrices.

An example for the above definitions can also be seen in Figure 1 and 2. In Figure 2, five time series of a data set $D = T_a, T_b, T_c, T_d, T_e$ are clustered per timestamp for the time points t_i, t_j and t_k. The data points of a time series T_l are denoted by the identifier l for simplicity reasons. The shown clustering consists of six clusters. It can be described by the set $\zeta = \{C_{t_i,l}, C_{t_i,u}, C_{t_j,v}, C_{t_j,f}, C_{t_k,g}, C_{t_k,h}\} \cup \{o_{t_i,e}\}$. As $o_{t_i,e}$ is not assigned to any cluster in t_i, it is marked as noise for this timestamp. The data points $o_{t_i,a}, o_{t_i,b}$ of time series T_a and T_b in t_i are cluster members of the yellow cluster $C_{t_i,l}$. The subsequences $T_{t_i,t_j,a}$ and $T_{t_i,t_j,b}$ from time series T_a and T_b move both from the yellow ($C_{t_i,l}$) to the red ($C_{t_j,v}$) cluster. The green ($C_{t_k,h}$) and pink ($C_{t_k,g}$) cluster can be summarized by the time clustering ζ_{t_k} at time t_k.

3.2 Over-Time Stability Evaluation

Since we want to measure the stability of an over-time clustering, whereby the partitioning may be produced by an arbitrary (evolutionary) clustering algorithm, we assume that different clusterings constitute different cluster connectedness based on the underlying TS members. Time series which separate from their clusters' members often, indicate a low over-time stability. For this reason, we first analyse the behavior of every subsequence of a time series $T = o_{t_1}, ...o_{t_k}$, with $t_k \leq t_n$, starting at the first timestamp. In case of a hard clustering, subsequently, every cluster is rated by a stability function, based on the previous subsequence analysis of its members and the number of clusters that merged into the considered cluster. The final over-time stability score for the whole clustering can then be calculated with the rating of each cluster. When regarding fuzzy clusterings, the over-time clustering is directly rated based on the subsequence scores.

3.2.1 CLOSE. Given a TS data set $D = \{T_l | 1 \leq l \leq m\}$ with n timestamps and an over-time clustering ζ, let $C_{t_i,a}$ and $C_{t_j,b}$ be two clusters of ζ, with $t_i, t_j \in \{t_1, ...t_n\}$. The *temporal cluster intersection*, which is used for the stability evaluation of a subsequence, is defined as follows

$$\cap_t \{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \wedge o_{t_j,l} \in C_{t_j,b}\}, \tag{3}$$

with $l \in \{1, ..., m\}$. The resulting set consists of time series, which contain data points that are grouped to the same cluster in t_i and t_j. The transition of a subsequence from one cluster $C_{t_i,a}$ in t_i to another $C_{t_j,b}$ in t_j along with its

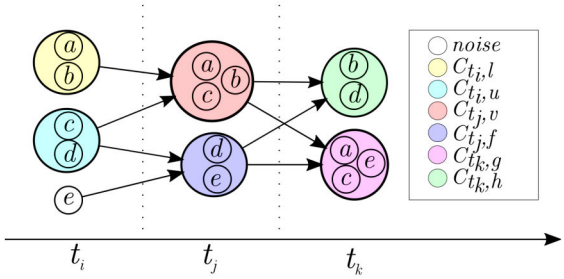


Fig. 2. Example for cluster transitions of time series $T_a, .., T_e$ over time [47].

group behavior, which may be interpreted as *team spirit*, can now be expressed by the proportion of members of $C_{t_i,a}$ remaining together in t_j

$$p(C_{t_i,a}, C_{t_j,b}) = \begin{cases} 0 & \text{if } C_{t_i,a} = \emptyset \\ \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a}|} & \text{else} \end{cases} \tag{4}$$

with $t_i < t_j$. Regarding the example in Figure 2 the proportion for $C_{t_i,l}$ and $C_{t_j,v}$ is defined by

$$p(C_{t_i,l}, C_{t_j,v}) = \frac{|\{a,b\}|}{|\{a,b\}|} = \frac{2}{2} = 1.0 \, .$$

This proportion can be used to evaluate the over-time stability of a subsequence by rating its history with a *subsequence score*. In order to address the clusters a data point is assigned to, we first need to introduce an auxiliary function, which we call *cluster-identity function*:

$$cid(o_{t_i,j}) = \begin{cases} \emptyset & \text{if the data point is not assigned to a cluster} \\ C_{t_i,l} & \text{else} \end{cases} \tag{5}$$

For a data point $o_{t_i,j}$ at time t_i the function returns the cluster it is assigned to. The subsequence score is then defined by

$$subseq\_score(o_{t_k,l}) = \frac{1}{k} \cdot \sum_{i=1}^{k-1} p(cid(o_{t_i,l}), cid(o_{t_k,l})) \, , \tag{6}$$

with $l \in \{1, ..., m\}$ and k being the number of timestamps where the data point exists. That means, that all time points in which an object is an outlier, get the worst possible score of 0. The *subsequence score* takes into account how many cluster members of the object from the previous timestamps have migrated together over time.

In the example of Figure 2, the score of time series T_a at time point t_k would be:

$$subseq\_score(o_{t_k,a}) = \frac{1}{2} \cdot (\frac{2}{2} + \frac{2}{3}) = 0.83 \, .$$

This value reflects a quite high stability, which can be explained by the fact that T_a moves with most of its cluster members over the time period. The time series d, gets a significantly lower value of $subseq\_score(o_{t_k,d}) = 0.5$ as it never moves with any of its cluster members. Note, that the impact of transitions of single TS becomes significantly lower when considering larger data sets.

The stability of a cluster can now be evaluated focussing on two factors. The first one is the number of different clusters of previous timestamps, that merged into the regarded cluster. This can be expressed by

$$m(C_{t_k,i}) = |\{C_{t_l,j} \mid t_l < t_k \wedge \exists a : o_{t_l,a} \in C_{t_l,j} \wedge o_{t_k,a} \in C_{t_k,i}\}| \,, \tag{7}$$

Furthermore, a cluster's stability score depends on the subsequence rating of all its cluster members. The second factor is therefore the sum of all *subsequence scores* of the data points within the considered cluster. Hence, the *over-time stability* of a cluster is defined as

$$ot\_stability(C_{t_k,i}) = \frac{\frac{1}{|C_{t_k,i}|} \cdot \sum_{o_{t_k,l} \in C_{t_k,i}} subseq\_score(o_{t_k,l})}{\frac{1}{k-1} \cdot m(C_{t_k,i})} \tag{8}$$

for $k > 1$. For a cluster at time point t_k the entire preceding time frame $[t_1, t_{k-1}]$ is considered. We define clusters at the first timestamp to be stable and set $ot\_stability(C_{t_1,i}) = 1.0$. In order to make clusters comparable, the sum of *subseq_score* is averaged by the number of data points in the viewed cluster, while the number of merged clusters is averaged by the number of timestamps before the regarded cluster. There are clustering algorithms which do not assign a cluster to every data point. Those data points are usually denoted as outliers. It is important to mention, that the number of merged clusters does not take these outliers into account.

Regarding the example of Figure 2, the stability of the cluster $C_{t_k,g}$ is given by:

$$ot\_stability(C_{t_k,g}) = \frac{\frac{1}{3} \cdot (0.83 + 0.58 + 0.25)}{\frac{1}{2} \cdot 4} = 0.28 \,.$$

This low score can be explained by the fact that the cluster under consideration contains only three data points. One of those (T_e) has a completely independent course of its clusters' members and the remaining two are not perfectly stable either.

Finally, the over-time stability of a clustering ζ can be calculated by

$$CLOSE(\zeta) = \frac{1}{N_C} \cdot \left(1 - \left(\frac{n}{N_C}\right)^2\right) \cdot \left(\sum_{C \in \zeta} ot\_stability(C) \cdot (1 - quality(C))\right) , \tag{9}$$

with N_C being the number of clusters of the over-time clustering ζ, n being the number of timestamps and *quality* being an arbitrary cluster evaluation measure. When working with normalised data $\in [0,1]^d$, we suggest the mean squared error (MSE), but any other rating function can also be used. Please make sure of using a function, whose results lie in the interval of $[0,1]$ in order to get appropriate results. When using a function for evaluating the quality instead of the deficiency of a clustering – that means, higher values indicate a higher quality – the term $(1 - quality(C))$ may e.g. be replaced by $(1 - quality(C)^{-1})$ or $quality(C)$ depending on the quality measure.

As long as the output of the quality function is between 0 and 1 and there exists at least one cluster per timestamp, CLOSE as well returns a score between 0 and 1, with 1 indicating a good over-time clustering.

The first pre-factor results from averaging by the number of clusters. The second factor $1 - (\frac{n}{N_C})^2$ is intended to counteract one large cluster to get a high score. Since such a clustering automatically exhibits a very high over-time stability, the CLOSE score rises. Note, that the clusters of the first point in time are also included in the evaluation measure. Since they are assumed to have a stability of 1.0, the score is in general slightly increased and for the first timestamp only influenced by the quality of the clusters.

Remark 3.1 (Time Point Comparison). In contrast to the evaluation function integrated in evolutionary clustering [7, 23, 54], where only consecutive points in time are compared, CLOSE compares clusterings of all preceding time points with the last timestamp of the considered subsequence. This has multiple effects. First, the stability score is robust against outliers. Second, short-term transitions between clusters are weighted more lightly. Simultaneously, long-term changes that develop slowly over time are punished more severely, which forms the third effect. Note: The formula cannot be transformed to simply iterate over all cluster pairs. Since the *over-time stability* is weighted with the *quality* of the cluster, the results would differ.

Remark 3.2 (Handling Outliers). Our calculations are suitable for both cleaned data and data with noise. Currently, outliers have only a minor impact on the score. That is, because they are solely considered in the subsequence score and not in the cluster stability. However, apart from decreasing the subsequence score, they have an additional indirect influence on the clustering score. Since the pre-factor in Formula 9 favors a large number of clusters, it may be more advantageous for the clustering algorithm to assign data points to smaller clusters than to interpret them as noise and recognize only a few large clusters.

This weak treatment of outliers is reasoned considering the idea, that the over-time clustering might be used for outlier detection. In this case, the algorithm should not be pushed into assigning every data object to a cluster. Nevertheless, different strategies for treating outliers might be investigated in future work.

One way to penalize noise more strongly would be, to insert an *exploitation term* which represents the number of data points that are assigned to a cluster N_{co} in relation to the number of all existing data points N_o. In order to achieve high CLOSE scores, this term should be maximized then. The formula including the exploitation term is given by

$$CLOSE(\zeta) = \frac{1}{N_C} \cdot \left(1 - \binom{n}{N_C}^2\right) \cdot \left(\sum_{C \in \zeta} ot\_stability(C) \cdot (1 - quality(C))\right) \cdot \binom{N_{co}}{N_o}, \tag{10}$$

Remark 3.3 (Merge & Split of Clusters). Considering the *subsequence score* (Formula 6), a merge of clusters do not have a negative impact on the score. On the contrary: if two clusters fuse entirely, the score is actually increased, as all objects move together with all their cluster members and therefore show a good team spirit. This is intended, since the focus lies primarily on the cohesion of time series. A good team spirit is rewarded in every case.

When considering cluster splits, though, the *subsequence score* is lowered. Since a split indicates that time series which have been members of the same cluster at some point in time separate from each other, this behavior is also wanted. Note, that in the case, where smaller clusters have previously merged together and then separated again in the same way as before, the influence on the score is not high and vanishes over time.

However, in some applications the punishment of cluster merges might be desired. As we will show in Section 4 regarding our proposed outlier detection algorithm, the Jaccard Index can be used in the proportion calculation, in order to penalize merges and splits in the same way.

Remark 3.4 (Additional Remarks). As Ben et. al stated, the sample size has a high impact on the stability evaluation of a clustering [3]. This is not only the case, when considering constant data points. When examining the over-time stability of a clustering, a small sample size also leads to a high sensitivity to transitions between clusters. The greater the considered data set, the easier a statement about the (over-time) stability can be made. In order to extend the method for a broader field of quality measures, the formula of CLOSE can be modified, so that quality measures for clusterings instead of clusters can be used. Therefore, the average cluster stability *avg_stab* per time clustering ζ_{t_i} must

be considered. The score is then normalised using the number of timestamps n:

$$CLOSE(\zeta) = \frac{1}{n} \cdot \left(1 - \left(\frac{n}{N_C}\right)^2\right) \cdot \left(\sum_{\zeta_{t_i} \subset \zeta} avg\_stab(\zeta_{t_i}) \cdot (1 - quality(\zeta_{t_i}))\right). \tag{11}$$

3.2.2 FCSETS. Given a TS data set $D = \{T_i | 1 \leq i \leq m\}$ with n timestamps and a fuzzy over-time clustering U. Let $U_{t_i} \subset U$ be a fuzzy partitioning of the data objects O_{t_i} of all times series at time t_i in k_{t_i} clusters. The relative assignment agreement of two data objects $o_{t_i,l}$ and $o_{t_i,s}$ from time series T_l and T_s to all clusters in the partitioning U_{t_i} at time t_i can be calculated using the equivalence relation from Hüllermeier-Rifqi Index (HRI) [17]:

$$E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s}) = 1 - \frac{1}{2} \sum_{j=1}^{k_{t_i}} \left| u_{C_{t_i,j}}(o_{t_i,l}) - u_{C_{t_i,j}}(o_{t_i,s}) \right|, \tag{12}$$

with $u_{C_{t_i,j}}(o_{t_i,l})$ being the membership degree of the data point $o_{t_i,l}$ regarding the cluster $C_{t_i,j}$ (see Definition 3.9). In order to measure the relation of two time series T_l and T_s, we calculate the difference between their relative assignment agreements by subtracting the relative assignment agreement values:

$$D_{t_i,t_r}(T_l, T_s) = \left| E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s}) - E_{U_{t_r}}(o_{t_r,l}, o_{t_r,s}) \right|. \tag{13}$$

Leaning on the Hüllermeier-Rifqi Index [17] – which deals with a slightly different task by calculating the normalised degree of concordance between two partitions – we define the over-time stability of a time series T_l as the average weighted difference between the relative assignment agreements to all other time series:

$$stability(T_l) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{r=i+1}^{n} \frac{\sum_{s=1}^{m} E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s})^m D_{t_i,t_r}(T_l, T_s)^2}{\sum_{s=1}^{m} E_{U_{t_i}}(o_{t_i,l}, o_{t_i,s})^m}. \tag{14}$$

The difference between the assignment agreements $D_{t_i,t_r}(T_l, T_s)$ is weighted by the assignment agreement between pairs of TS at a previous time point in order to damp large differences for stable time series caused by supervention of new peers. On the other hand, time series that leave their cluster peers when changing their cluster membership are penalized.

The over-time stability of a fuzzy clustering U can now be expressed by the average over-time stability of all time series in the data set:

$$FCSETS(U) = \frac{1}{m} \sum_{l=1}^{m} stability(T_l). \tag{15}$$

A more efficient approach as a substitute for the HRI proposed by Runkler [42] is the *Subset Similarity Index (SSI)*. The efficiency gain is reasoned by the similarity calculation, which in SSI considers cluster pairs while HRI concentrates on the assignment agreement of data point pairs. In our context, where the clustering should be used for further analysis such as outlier detection, we aim to describe the over-time stability of clustering by the *team spirit* of the considered time series. Therefore, we believe, that the degree of the assignment agreement between TS pairs to clusters at different timestamps provide a greater information gain than the similarity between cluster pairs. For this reason the SSI is not suitable for our over-stability evaluation.

4 APPLICATIONS

Our evaluation measures can not only be used for the over-time stability evaluation of clusterings, but also for further analyses such as parameter selection or outlier detection [45, 47, 48]. Therefore, for example the part of CLOSE, where subsequences are evaluated, can be used.

In [45], we present an approach called DOOTS (**D**etecting **O**utliers regarding their **O**ver-**T**ime **S**tability) for finding conspicuous subsequences of all lengths with an underlying over-time clustering regarding the following definition:

Definition 4.1 (Anomalous Subsequence). A subsequence $T_{t_i,t_j,l}$ is called anomalous, if it is significantly more unstable than its cluster members at time t_j.

For this, the subsequence score from Formula 6 has to be reformulated in order to handle subsequences with arbitrary starting points. The *subsequence score* of a subsequence $T_{t_i,t_j,l}$ of time series T_l starting at t_i and ending at t_j is defined as

$$subsequence\_score(T_{t_i,t_j,l}) = \frac{1}{k} \cdot \sum_{v=i}^{j-1} p(cid(o_{t_v,l}), cid(o_{t_j,l})) \tag{16}$$

with $l \in \{1, ..., m\}$, $k \in [1, j-i]$ being the number of timestamps between t_i and t_j where the time series exists [45].

One noteworthy aspect is that the score is always 0, if the last data point of the considered subsequence is marked as noise. In most cases, this does not lead to any handicaps regarding the analysis, since all partial sequences of these subsequences are treated normally, though. Nevertheless, a more detailed discussion of such situations will be provided in the further course of this work.

As already mentioned, the used *proportion* from Formula 4 is asymmetric and punishes splits while ignoring merges. In order to counteract this circumstance, the jaccard index can be used, as proposed in [47]. Therefore, the *temporal cluster union* of two clusters $C_{t_i,a}, C_{t_j,b}$ has to be introduced first:

$$\cup_t \{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \vee o_{t_j,l} \in C_{t_j,b}\} \tag{17}$$

with $l \in \{1, ..., m\}$. The proportion $\hat{p}$ can then be expressed by the jaccard index of two clusters:

$$\hat{p}(C_{t_i,a}, C_{t_j,b}) = \begin{cases} 0 & \text{if } C_{t_i,a} = \emptyset \wedge C_{t_j,b} = \emptyset \\ \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a} \cup_t C_{t_j,b}|} & \text{else} \end{cases} \tag{18}$$

with $t_i < t_j$. In contrast to the proportion from Formula 4 regarding the example in Figure 2 the jaccard proportion is

$$\hat{p}(C_{t_i,l}, C_{t_j,v}) = \frac{|\{a,b\}|}{|\{a,b,c\}|} = \frac{2}{3} = 0.67$$

since the merge of (parts of) the yellow ($C_{t_i,l}$) and turquoise ($C_{t_i,u}$) cluster gets punished.

Another characteristic of the subsequence score from CLOSE (Formula 6) is the equal impact of all considered timestamps regarding the over-time stability of a subsequence. When considering longer sequences, however, this may lead to a tendency towards a worse rating, since slow changes in cluster memberships might influence the score considerably. Assuming that the nearer past is more significant than the more distant past, a weighting function can be integrated in the subsequence score.

Using the Gauss' Formula, the weighting of the proportion at time t_i regarding the time interval $[t_1, t_k]$ can be calculated by

$$\frac{i}{\sum_{a=1}^{k} a} = \frac{i}{\frac{k(k+1)}{2}} = \frac{2 \cdot i}{k(k+1)} . \tag{19}$$

Adjusting this weighting function to a time interval with arbitrary starting point $t_s \geq t_1$, the subsequence score is then defined by

$$weighted\_subseq\_score(T_{t_i,t_j,l}) = \sum_{v=i}^{j-1} \frac{2 \cdot (v-i+1)}{k(k+1)} \, p(cid(o_{t_v,l}), cid(o_{t_j,l})) \; . \tag{20}$$

with $k \in [1, j-i]$ again being the number of timestamps between t_i and t_j where the considered time series exists [47]. There is no need to normalize the score to an interval of $[0, 1]$ by averaging it, as the sum of all weightings of a subsequence's timestamps is always 1 due the division by the Gauss's Formula.

In contrast to the subsequence score, regarding the example in Figure 2 the weighted subsequence score is given by

$$subseq\_score(o_{t_k,a}) = \frac{1}{3} \cdot \frac{1}{2} + \frac{2}{3} \cdot \frac{2}{3} = 0.61$$

which is a bit higher, since the immediately preceding (higher) score gets a greater weighting than the more distant one.

In summary, four options can be used: (i) the ordinary subsequence score (DOOTS), (ii) the weighted subsequence score (wDOOTS), (iii) the ordinary subsequence score using the jaccard proportion (jDOOTS) and (iv) the weighted subsequence score using the jaccard proportion (jwDOOTS).

With this score, a subsequence can now be compared with its cluster members, in order to determine, if its over-time stability stands out. In this respect we consider the following assumptions:

Assumption 4.1. If the score of a subsequence is significantly lower than those of its cluster members, its over-time behavior is conspicuous.

Assumption 4.2. If the score of a subsequence is low, but so are those of its cluster members, its over-time behavior is not conspicuous, since this low over-time stability shows a pattern of regularity.

In order to find outlier sequences of all lengths, every possible subsequence receives an outlier score indicating the probability of being anomalous. The outlier score describes the deviation of a subsequence's stability from the best subsequence score of its cluster. Figuratively, one can imagine that the time series with the highest subsequence score represents a kind of leader and that a large deviation from this leader is to be considered conspicuous. The best subsequence score of a cluster $C_{t_j,a}$ regarding subsequences starting at time t_i is expressed by the following formula:

$$best\_score(t_i, C_{t_j,a}) = max(\{subsequence\_score(T_{t_i,t_j,l}) \mid cid(o_{t_j,l}) = C_{t_j,a}\}) \tag{21}$$

The outlier score can then be calculated by

$$outlier\_score(T_{t_i,t_j,l}) = best\_score(t_i, cid(o_{t_j,l})) - subsequence\_score(T_{t_i,t_j,l}) \; . \tag{22}$$

With respect to Assumption 4.1 and 4.2, the outlier score depends on the best score of a cluster's members. Therefore, an outlier score of 100% can only be achieved in clusters consisting exclusively of completely stable subsequences. On the other hand, a cluster with small stabilities only, can lead to a situation where no subsequence score is considered conspicuous, no matter how low it is. As mentioned in Assumption 4.2, this behavior is desired.

Using the outlier score and a threshold parameter τ, a more precise definition of an outlier can now be given.

Definition 4.2 (Outlier). Given a threshold $\tau \in [0, 1]$, a subsequence $T_{t_i,t_j,l}$ is called an outlier, if its probability of being an outlier is greater than or equal τ. That means, if

$$outlier\_score(T_{t_i,t_j,l}) \geq \tau \; .$$

Even though the parameter τ is constant, it can be considered as a dynamic threshold, since the greatest possible deviation from the best subsequence score – and simultaneously the greatest outlier score – is dependent on the best score of the considered cluster. Leaning on Assumption 4.2, clusters which show a low stability have a lower probability of containing an outlier than stable ones, because all their cluster members exhibit irregularities, which represents a pattern of instability. Thus, in this case, a small subsequence score is not conspicuous.

Subsequences that consist entirely of noise data points are automatically identified as outliers and are called *intuitive outliers*. This special treatment is needed, since subsequences whose last data point is labeled as noise do not have any cluster members which the best score can be calculated from. Therefore, no outlier score can be determined for them. Hence, in our outlier detection we consider three types of outliers: *anomalous subsequences* regarding Definition 4.2, *intuitive outliers* and data points marked as *noise* by a clustering algorithm.

Imagine examining a subsequence $T_{t_i,t_j,l}$ whose last data point at time t_j is marked as noise. In addition suppose its subsequence $T_{t_i,t_{j-1},l}$ getting a high outlier score and therefore being detected as an outlier. Intuitively, one would expect the subsequence under consideration $T_{t_i,t_j,l}$ being identified as an outlier as well. In our approach, this would only be the case, if the sequence was recognized as an intuitive outlier i.e. the previous data point was categorized as noise, too. Anyway, the subsequence $T_{t_i,t_k,l}$ with $k > j$, which for the first time is assigned to a cluster again at its last time point t_k, would be detected as an outlier. Thus, in the end $T_{t_i,t_j,l}$ would be covered.

Still, in the marginal case where a data point is labeled as noise at the last time of the entire time series, a subsequence with end time t_m would never be detected as an outlier, if it is not marked as noise in t_{m-1}. This drawback should be investigated in future works.

Remark 4.1 (Modifications). As DOOTS is leaned on the presented evaluation measure, the modification of the proportion calculation using the Jaccard index as well as the weighting function for the *subsequence_score* may naturally also be applied to CLOSE, if desired.

5 EVALUATION

In this section we present several experiments. First we describe the different data sets, which we use in order to illustrate our results. Then we present clusterings calculated with K-Means [33] and DBSCAN [13]. In order to create those clusterings we use common methods to identify good parameters per timestamp. Afterwards we compare the results with clusterings whose parameters were identified with the help of CLOSE. These results are then compared to those of the evolutionary clustering presented in [7]. We also evaluate clusterings retrieved by Fuzzy C-Means [5] and focus on the achieved FCSETS scores. Finally, the comparison of clusterings is followed by applications to the outlier detection algorithm. We finish the section with qualitative analyses of the results.

5.1 Data Sets

In the following we present the three data sets our analyses are based on.

5.1.1 COVID-19 Data Set. The COVID-19 pandemic is currently affecting the whole world. In this context the hashtag #FlattenTheCurve is intended to encourage people all over the world to behave in a way that prevents the distribution of infections over time and thus counteracts overloading of the health care systems. Although the hashtag is used in an inflationary way, few people realise that the curve is actually a time series. Because of the current relevance of the data set, it is an excellent candidate for applying our methods. We obtained the data from the official GitHub repository of

Johns Hopkins University[1]. Specifically, we used the daily reports on worldwide COVID-19 infections for our analyses. Depending on the country, the data set contains data on the individual regions (such as federal states) of the country concerned. We have aggregated these data so that for each available country only one entry per point in time has been created. Over time, other features such as incidence were added. In order to provide the incidence for all points in time, we calculated the incidence using population data for the countries. For this purpose we have obtained the population data for the respective countries from theglobaleconomy.com[2]. We then calculated the seven-day incidence for the countries. The incidence reflects the number of infections in the last seven days per 100,000 inhabitants. Due to the low infection figures at the beginning of the pandemic, the incidence value is particularly low at some times. For this reason, we give the number of infections per 10,000,000 inhabitants. In addition, we do not consider directly consecutive days, because the fluctuation in these is relatively small. Instead, we look at every seventh day, reflecting the development within a week.

5.1.2 TheGlobalEconomy.com Data Set. We extracted this data set from theglobaleconomy.com[2]. The website offers over 400 indicators on 200 countries for over 80 years. The indicators include data such as GDP, inflation, population data, employment rates and many more. All available data have been obtained from reliable official sources. From the large number of available indicators we selected two for illustration purposes, namely the unemployment rate and the education spending. The two features are on the one hand the educational expenditure and on the other hand the unemployment rate. In addition, we have only considered twenty countries for the purpose of the overview.

5.1.3 Generated Data Set. In order to show specific characteristics of CLOSE and our outlier detection algorithm, we generated two artificial data sets. The first contains 40 time series with 6 time points and two dimensional feature vectors in $[0, 1]^2$. For every timestamp four cluster centroids have been set, which 10 time series were assigned to with a maximal distance of 0.1 each. The cluster members remain the same for the whole time period, but the clusters merge and split over time. More precisely, at any time point only three clusters are visible, since at the moment where one cluster splits (t_4), two others merge into one.

For the evaluation of our outlier detection algorithm, three transition-based outliers have been inserted in the data set. For each timestamp, the outlier sequences have been randomly assigned to a cluster centroid with a maximal distance of 0.1.

5.2 Density-based Clustering

Since to the best of our knowledge there are no other evaluation measures for the over-time stability of clusterings-per-timestamp, a quantitative evaluation against other measures is not possible. The comparison to other common stability measures is not meaningful either, as the targeted stability definition differs. Nevertheless, the evaluation of clusterings retrieved with parameter settings determined by CLOSE against those of evolutionary clustering algorithms, may surrogate such an analysis as the objective function which is optimized in evolutionary clustering includes a similar definition of over-time stability. Apart from the comparison with evolutionary clusterings, our evaluation section deals with different experiments on real world and artificially generated data sets in order to discuss different characteristics of CLOSE and its applications.

In the first experiment we investigate the behavior of the CLOSE score depending on the parameter setting of DBSCAN regarding the GlobalEconomy data set. In Figure 4 this behavior is illustrated. For each *minPts* a colored

[1]https://github.com/CSSEGISandData/COVID-19

[2]https://www.theglobaleconomy.com

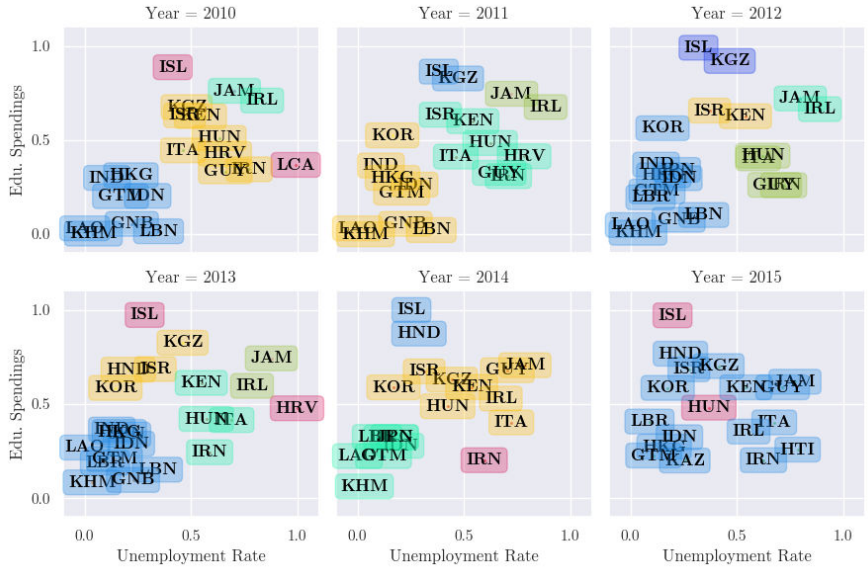


Fig. 3. Best resulting clustering with DBSCAN ($\epsilon = 0.2$, $minPts = 2$) achieving a CLOSE score of 0.514 on the GlobalEconomy data set.

line is drawn, which shows the CLOSE score depending on ϵ. We tested all $minPts \in [2, 6]$ and $\epsilon \in [0.1, 0.4]$ with a step size of 0.01. The best result was achieved with $minPts = 2$ and $\epsilon = 0.2$ and is shown in Figure 3.

As can be seen, the resulting clustering is quiet stable although the data set is rather dispersed and some of its data objects have irregular movements. For example, Jamaica (JAM) and Ireland (IRL) are completely stable over time as they are always together in one cluster. Such a stability can only be achieved with $minPts = 2$ since bigger clusters would lead to more cluster transitions. This characteristic can also be read off the diagram in Figure 4, where the curve of $minPts = 2$ reaches higher CLOSE scores than the others in most cases. Obviously, regarding this data set, it is difficult to determine *one* optimal ϵ since the groups of objects move towards each other. The choice of one fix parameter setting leads for example to the creation of a single cluster in the last considered timestamp. Although it is not desired to have only one cluster, since it does not lead to a high information gain, it is an intuitive result in this case, though. When choosing a smaller ϵ in order to counteract this circumstance, the over-time stability would be significantly decreased.

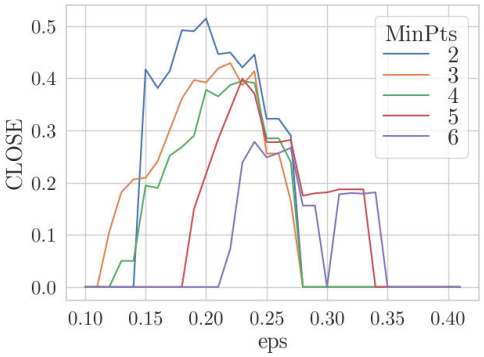


Fig. 4. Resulting CLOSE score for different $minPts$ depending on ϵ.

When considering the line of $minPts = 6$ in Figure 4, the results might seem unintuitive since the CLOSE score is 0 for most of the time and it gets higher with $\epsilon > 0.3$ although it already reached a score of 0 before. The first characteristic can be explained by the high $minPts$ value since ϵ has to be chosen relatively high in order to reach enough data points to put together in one cluster. The second characteristic is caused by the pre-factor of CLOSE which sets the score to 0, if there are not at least k clusters, where k is the number of timestamps. For $\epsilon = 0.3$ only one cluster per timestamp is found which causes a high amount of outliers. By increasing ϵ new clusters are created, whose members have been marked as noise for lower ϵ. This applies in particular to the years 2012 and 2013.

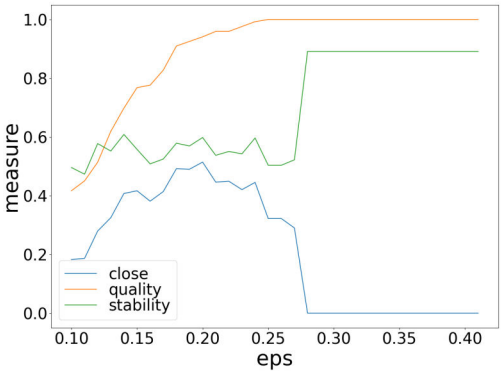


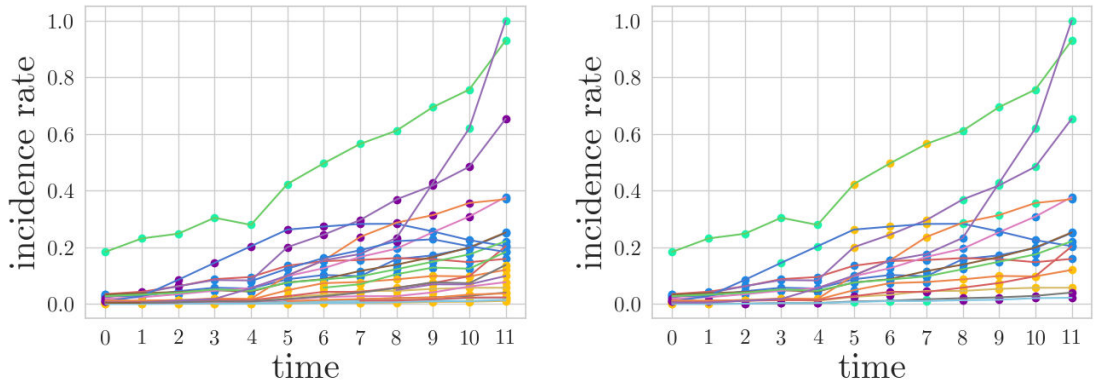
Fig. 5. Resulting CLOSE score, stability and quality for $minPts = 2$ depending on ϵ.

In Figure 5 the behavior of the *ot_stability*, *quality* and the CLOSE score (see Formula 9) depending on ϵ can be compared. $minPts$ was set to 2, as it proved to be the best choice on the GlobalEconomy data set. The quality was measured by the amount of objects that are assigned to a cluster in relation to all objects at the considered time point. The usage of such a simple measure can be justified by the fact that the density of the resulting clusters is already indirectly evaluated by the clustering algorithm DBSCAN. Also, evaluation measures addressing the separation and compactness of clusters are not suitable for density-based clustering algorithms. Therefore, the aim is to minimize the amount of outliers as they are not caught in the formula of CLOSE. The diagram shows that, as long as the quality is lower than the stability ($\epsilon \leq 0.13$), it has a high impact on the CLOSE score. Afterwards, the curve of CLOSE is very similar to the stability. For $\epsilon > 0.26$ the CLOSE score gets worse, although the quality as well as the stability increases. The CLOSE score decreases rapidly to 0, which is caused by the fact, that the number of clusters falls below the number of timestamps. In other cases the score would highly depend on the number of clusters as long as they exceed the number of timestamps, if the quality and stability remain almost the same.

5.3 K-Means

In this paragraph we compare the achievable CLOSE score of K-Means with those of the evolutionary K-Means of [7]. For this, we first used the one-dimensional COVID-19 data set. The evolutionary clustering approach from [7] softens the definition of partitioning clustering: At a point in time, the space is classically partitioned into k regions, but the assignment of individual elements to a cluster is also based on the partitionings of the previous points in time. The assignment function is therefore based on two components, the so-called history costs and the distance to the cluster centre. The user must specify a weighting for these two components in advance. In addition, this approach requires an unknown function f that maps clusters from two points in time to each other. Although this function seems intuitive at first glance, it constitutes a separate field of research. Despite the problems mentioned above, evolutionary clustering has a decisive advantage that becomes more relevant when calculating stability. The assignment function of evolutionary clustering from [7] can assign objects to a cluster even if they lie in a different cluster from the point of view of a classical partitioning method. This can positively influence the stability of time series, clusters and thus also clusterings.

An adaptation of the classical K-Means to previous points in time can be realised with the help of varying ks. A search for the most stable clustering with varying ks is also possible with CLOSE, but we consider this scenario impractical



(a) Standard K-Means Clustering with $k = 4$. Achieved *CLOSE-Score* is 0.55.

(b) Evolutionary Clustering from [7] with $k = 4$ and $cp = 0.5$. Achieved *CLOSE-Score* is 0.51.

Fig. 6. K-Means and evolutionary K-Means from [7] applied to the COVID-19 Data Set.

because the number of configurations to be tested would increase considerably: For 10 time points and a $k \in [2, 5]$, this would already be $4^{10} = 1048576$ combinations. A corresponding evaluation of the stability for time-dependent k would therefore be difficult to realise. For this reason, we search for one k that fits best for all time points. The clustering that achieves the highest CLOSE score is then compared with evolutionary clustering. In the following evaluations, the asymmetric proportion and the mean squared error as quality measure were used.

5.3.1 K-Means and Evolutionary K-Means Applied to the COVID-19 Data Set . The results of the two clustering algorithms applied to the COVID-19 data set are very different. First, the best k was identified for both approaches using CLOSE. Here, all ks in the interval of $[2, 10]$ were examined. For both algorithms, $k = 4$ was identified as the k that leads to the most stable clustering.

For the evolutionary approach, the *change parameter* was set to 0.5. The results can be viewed in Figure 6. The differences are particularly striking at times five to seven. These can be explained by the previously extended assignment function of the evolutionary approach. In this specific case, however, the evolutionary approach does not lead to a higher CLOSE score than the classical approach. Specifically, the standard approach produces a clustering that is 0.04 more stable than the evolutionary approach. This may not be a big difference, but it shows that the adjustments from [7] made for the evolutionary approach do not necessarily lead to better CLOSE score.

5.3.2 K-Means and Evolutionary K-Means Applied to the Generated Data Set. In contrast to the results with the COVID-19 data set, the clusterings of the classical K-Means and the evolutionary K-Means [7] are identical. The result can be seen in Figure 7. This is mainly due to the nature of the generated data set. As mentioned earlier, the generated data set actually contains four clusters at each time point, two of which split off from each other and merge in t_4 respectively. Although intuitively one would identify three clusters at each time point, both algorithms identified only two clusters each. This result shows that both methods recognise that categorisation into three clusters would lead to more changes within the clusters and thus to less cluster stability. The only clustering that could compete with this clustering in terms of stability would be one in which all four original clusters were identified. However,

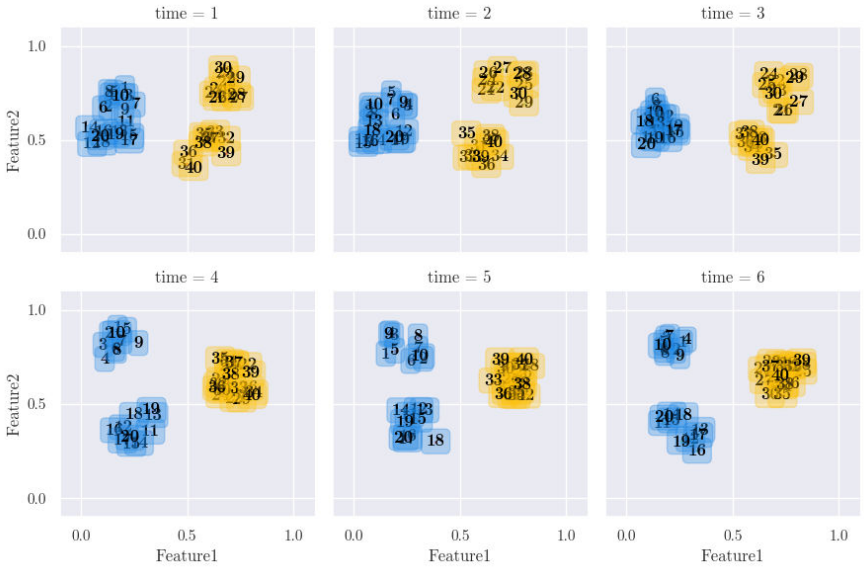


Fig. 7. K-Means and evolutionary K-Means from [7] applied to the Generated Data Set.

this result is not achievable due to the partitioning property of K-Means. The relatively large distance between the clusters does not prevent the evolutionary algorithm from recognising only two clusters. This can be explained by the high influence of history costs. In this case, we have set the weighting of the change parameter to 0.5 again; it can be assumed that the result will be different with a lower weight. In fact, a much lower weight leads to the detection of three clusters. We identified the k that leads to a clustering with the highest stability for both approaches with CLOSE ($k = 2$). Here we examined all k in the interval $[2, 6]$. In Figure 8 one can see the development of the CLOSE score as a function of the chosen k for the classical K-Means. In this data set, the highest CLOSE score is reached at $k = 2$. Higher ks lead to lower CLOSE scores. Figure 7 gives the impression that three clusters would be more intuitive at any point in time, but the problem is that such a setup would lead to more data points changing their cluster peers over time. This circumstance then leads to less stability of the individual time series, clusters and thus the entire clustering. More clusters lead to distributions in which objects have even more changing cluster peers. It should be noted that in a scenario with more clusters, quality increases but stability decreases. Together with the stability, the pre-factor then has a higher influence than the quality.

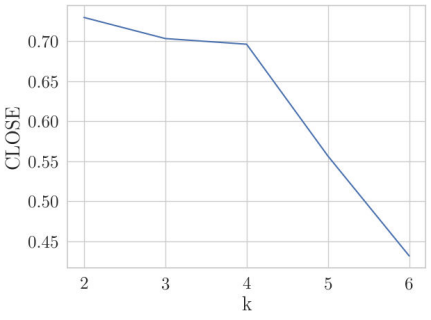
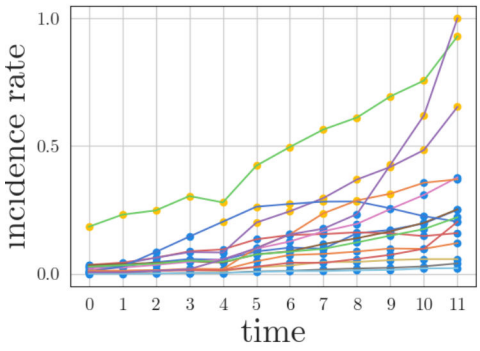
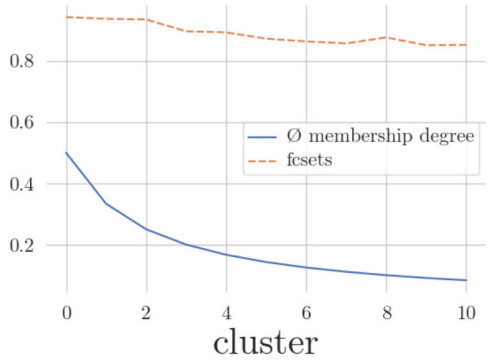


Fig. 8. Resulting CLOSE score for standard K-Means with different ks.



(a) Fuzzy C-Means Clustering with $c = 2$. Achieved FCSETS-Score is 0.941.



(b) Ø membership degree/ FCSETS Score in regard of the amount of clusters per timestamp.

Fig. 9. Fuzzy C-Means applied to the COVID-19 data set.

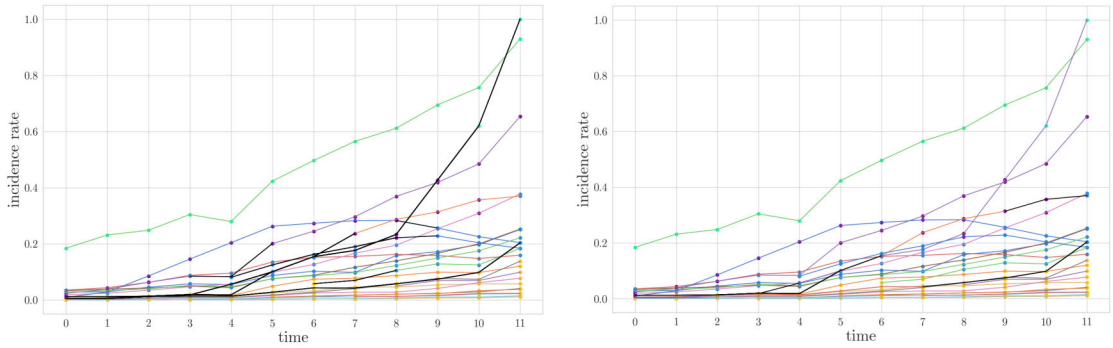
5.4 Fuzzy C-Means

In this section we discuss the results of FCSETS on the COVID-19 data set. The clusterings evaluated here were created using fuzzy C-Means, a fuzzy variant of K-Means. Figure 9b) shows the development of the FCSETS score as a function of the number of clusters. In addition, the average membership degree can be observed. The average membership degree decreases as the number of clusters increases; this development is expected because with more clusters a data object also has more assignment options. The assignment is then smaller per cluster with more clusters. The development of the FCSETS score is largely independent of the membership degree. This behavior was also expected since the FCSETS score compares the membership degrees of different time points with each other and this is independent of the actual membership degrees.

It is noticeable that the FCSETS scores achieved are significantly higher than the CLOSE scores. This is mainly due to the fact that there is no function for evaluating the cluster quality. While the highest CLOSE score was achieved with four clusters, the highest FCSETS score was reached with two clusters (0.941). The fact that both methods evaluate different numbers of clusters with the best score is expected due to the different approaches of the underlying clustering algorithms. This also means that other clustering algorithms could achieve better or worse results in the crisp but also fuzzy case. The decisive factor for the evaluation of an over-time clustering in the fuzzy case is the change in the degrees of membership over time. Fuzzy C-Means achieves the smallest change in these with two clusters per timestamp, which also reflects the most stable result in this case.

The main reason for this is the rate of change of membership degrees from one time point to another. In the case of the COVID-19 data set, a higher number of clusters provides a higher rate of change, so that the cluster membership is less stable over time. This is especially the case when the movement of objects within clusters is high. However, usually the movement has only little influence on the highest degree of membership of an object to a cluster, but the other degrees of membership change strongly. In the case of the COVID-19 data set, this change is strongest with ten clusters.

In Figure 9a) we have visualised the clustering with the highest FCSETS score. We have assigned the objects to the cluster to which they have the highest membership degree.



(a) DOOTS with asymmetric proportion

(b) DOOTS with symmetric proportion.

Fig. 10. Detected outliers on the COVID-19 data set with $\tau = 0.6$. black lines represent outliers. Clustering identified with CLOSE (K-Means, k = 4).

5.5 Outlier Detection

In this part of the paper, we present a qualitative analysis of the presented outlier detection and its variants. In particular, we address the effects of the different proportions and weightings chosen and illustrate this using the COVID-19 data set and the generated data set. In all the analyses presented, to identify the most stable clustering, we applied CLOSE to determine the parameters.

5.5.1 COVID-19 Data Set. In this section we compare the effect of asymmetric proportion and symmetric (jaccard) proportion on outlier detection. For this purpose we use the one-dimensional COVID-19 data set because it is particularly suitable for illustration. We clustered the data with K-Means, identifying the most stable clustering ($k = 4$) with CLOSE. In Figure 10 we can see the results obtained. The black graphs correspond to the outliers found. At first glance, it is immediately apparent that the outlier detection method with the symmetric jaccard proportion detects significantly fewer outliers than its asymmetric counterpart. This is due to the different evaluation of merged clusters. While merges of clusters have no influence with the asymmetrical proportion, the symmetrical jaccard proportion evaluates them negatively. This has a direct impact on the *subsequence_score*s, in the sense that they all become smaller in our example. This is reflected accordingly in the *best_score*, which corresponds to the maximum *subsequence_score* of a cluster. Overall smaller *subsequence_scores* also lead to smaller *outlier_scores*, because the difference between the *best_score* and the individual *subsequence_score*s also becomes smaller. With constant τ, as in this example, this leads to a smaller outlier detection rate. So in the case of the COVID-19 data set, we would prefer the outlier detection method with asymmetric proportion. The one-dimensional example also illustrates the type of outliers detected. In particular, we notice a time series that was detected as a whole by the system and has the highest incidence rate at the end. This time series is the incidence value of Luxembourg. There, on 31 May 2020, the highest incidence value of the European countries we looked at was reported. The high number of changes in the cluster environment is particularly striking. The first change occurs from week four to week five, followed by the change in week seven to week eight and finally the change from week nine to week ten. The constant change of cluster members leads to a relatively small *subsequence_score*, which then shows a high difference to the *best_score*s of the individual clusters.

Another rather inconspicuous time series detected by outlier detection has an incidence rate just above 0.2 at the last time point. This time series reflects the development of the pandemic in Romania. It is detected mainly because it

completely changes its cluster members twice. Firstly, the incidence rate in Romania at time one does not develop like that of its cluster members at time zero: In contrast to Romania's cluster members at time zero, the incidence rate in Romania does not continue to rise but remains at about the same level. The other change occurs from time ten to time eleven: Here, Romania's incidence rate jumps within one week, so that it is now in a cluster with countries of a higher infection level.

In this example, the difference between the two applied proportions is not only that the asymmetric proportion detects more outliers. The jaccard proportion also detects other outliers. Exemplary for this is the sequence of the top orange-colored time series. This is detected by the outlier detection with jaccard proportion, since a merge of clusters takes place in the last time point and this is penalised by the symmetrical proportion. This is not the case with the asymmetric proportion, the merge has no effect on the *subsequence_score* of the time series.

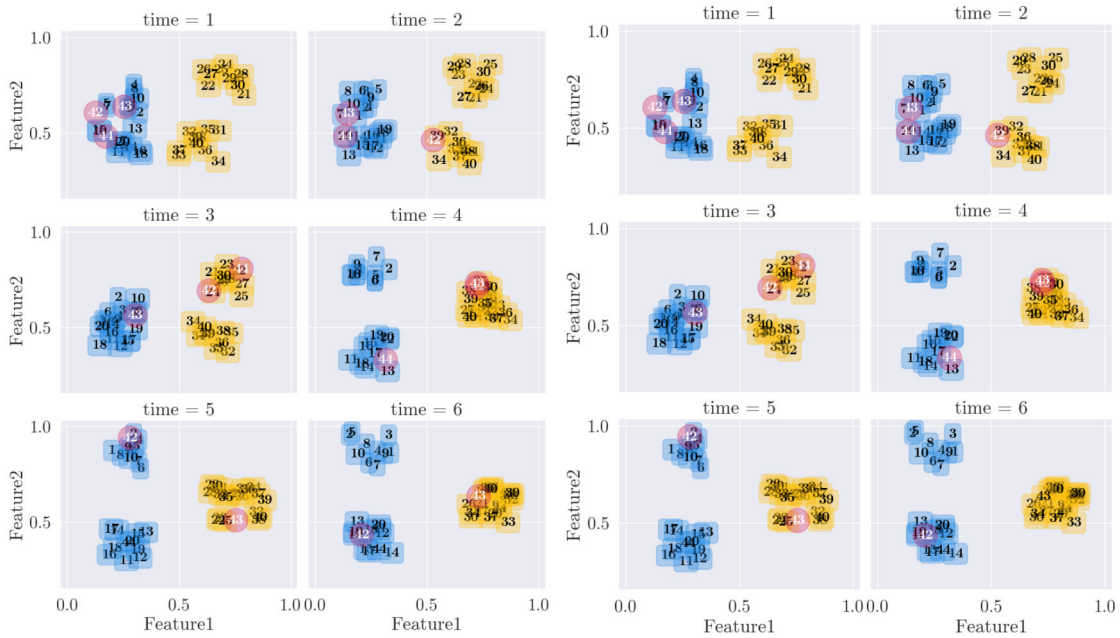
Overall, relatively many outliers are found in this example. This is mainly due to the choice of the parameter τ and the relatively over-time stable composition of the time series. The clustering has many time series that remain in a cluster over time with comparatively many time series. This leads to high *subsequence_score*s and thus to high *best_score*s. Time series that change their cluster members only once have a comparatively low *subsequence_score*, which also leads directly to classification as outliers due to the selected τ. This example also shows how to deal with missing data: A time series only begins in the sixth week, its sequence from week six to week eight is recognised as an outlier. On the one hand, this can be explained by the change in the cluster composition from week seven to week eight and, on the other hand, by the shortness of the time series.

5.5.2 Generated Data Set. For the evaluation of DOOTS on the generated data set, the clustering setting achieving the best CLOSE score was chosen as underlying clustering. Therefore, K-Means with $k = 4$ was used. Figure 11 shows the detected outlier sequences on the bivariate data set. All four proposed derivatives of our algorithm have been tested: the *original* method (DOOTS), the one using the *jaccard* index in the proportion calculation (jDOOTS), the one using a *weighting* in the subsequence score (wDOOTS) and the method combining the *weighting* and the *jaccard* index (jwDOOTS).

As can be seen, both approaches using the weighting function got the same results (11b). The same applies to the remaining two (11a). Both results are very similar to each other, as they differ only at one timestamp and that is the last one. Each method detects all three outlier sequences (42, 43, 44) in the first four timestamps. At time 5, all approaches are in agreement that there are only two outliers: 42 and 43. But at the last timestamp the weighted methods mark only one sequence (42) as an outlier, while the other ones additionally detect the time series 43.

In the first three timestamps, the detection of 42 and 44 are intuitive as they have transitions between the blue (left) and the yellow (right) cluster. In order to understand, why the sequence 43 has been marked as outlier, however, the fourth timestamp has to be inspected. Here, the sequence moves from the blue to the yellow cluster. Since both clusters have many members, which move stably over time, one transition can suffice for a high outlier score. Since all pairs of timestamps (t_i, t_j) with $i < j$ are considered in the calculation of the subsequence score of a sequence ending at t_j, the stable behavior of 43 from time 4 to 6 decreases the subsequence score even more after the transition. The subsequences $T_{t_1,t_3,43}$ and $T_{t_4,t_6,43}$ get high scores, since those sequences have a perfectly stable behavior. In context of the whole sequence, however, the score is very low, as half of the time there are completely different cluster members near the sequence than the rest of the time span.

In contrast to that, the sequence 44 is not marked as an outlier in the last two timestamps although it has more transitions than 43. This can be explained by the fact, that for 5 of 6 timestamps it is assigned to the blue cluster.



(a) DOOTS and jDOOTS.

(b) wDOOTS and jwDOOTS.

Fig. 11. Detected outliers on the generated data set with $\tau = 0.5$. Red data points represent outliers.

Therefore, only the transition to the yellow cluster at timestamp 3 is suspicious. As already explained before, this transition has a high impact on the outlier score caused by the high stability of the other cluster members.

The impact of the weighting function gets clear considering the sequence 43 in the last timestamp. While it is marked as an outlier in 11a), it does not get a high outlier score in the weighted approaches (b). Since the impact of the timestamps of the nearer past is weighted higher than this of the more distant one, the stability of 43 after its transition at timestamp 4 is rewarded. Due to the stable behavior in the later timestamps the negative influence of the transition is compensated.

6 CONCLUSION & FUTURE WORK

In this paper, we gave a short overview for a tool set specialised on time series analysis for databases containing multiple multivariate time series. The presented over-time stability evaluation measures CLOSE and FCSETS are useful tools for the evaluation of fuzzy and hard clusterings retrieved by evolutionary or time-independent clustering algorithms. With the help of CLOSE/FCSETS fitting hyperparameters for a stable over-time clustering using common clustering algorithms like K-Means [33] or DBSCAN [13] and evolutionary clustering algorithms such as evolutionary K-Means [7] may be determined. The considered definition of over-time stability varies slightly from the one usually used e.g. in evolutionary clustering. Instead of rating the actual movement of a sequence or cluster in the feature space, the behavior of a sequence is analysed in comparison to its peers. The stability of a cluster is thereby driven by its members. Also, not only the immediately preceding timestamp is considered, but the whole history of a sequence. Based on CLOSE various further TS analyses may be derived. In this paper, we e.g. propounded an outlier detection algorithm, called

DOOTS, for the detection of transition-based outliers, which were firstly introduced in [45]. Two application-based modifications regarding the calculation of the *proportion* and the *subsequence_score* are shown, which may be applied to DOOTS as well as CLOSE. Because of that, the presented methods are quite flexible which makes them applicable to a broad field of applications.

The discussed experiments showed, that all depicted methods fulfill the desired intention. With the help of CLOSE common clustering algorithms are able to compete against evolutionary clusterings regarding a stable over-time clustering. In addition, CLOSE can be helpful when using evolutionary clustering algorithms in order to find the optimal parameter setting. Due to the variable components in CLOSE, such as the quality measure, it can be adapted for different types of clusterings, e.g. partition-based and density-based clusterings, in order to ensure a high quality apart from the over-time stability. This has been shown by experiments on different artificial and real-world data sets, and various clustering algorithms. Also, the influence of different parameter settings on the CLOSE score may be discovered by plotting a diagram similar to our experiment, which allows a further analysis of the underlying data.

With an underlying over-time stable clustering, the outlier detection algorithm can be applied. Our experiments showed that the desired outlier type has been detected. On lucid data sets with one or two features, those outliers may be easy to recognize with the human eye, but considering multivariate time series with higher dimension, the problem gets quite complex. Therefore, an outlier detection algorithm addressing this type of outliers might be helpful.

Apart from the presented ones, further methods based on CLOSE may be developed, e.g. an over-time clustering algorithm [25] or the prediction of the further course of sequences or clusters. Similar subsequences and patterns may already be identified by investigating the resulting clusters. Of course, an automation might easily be implemented. Since CLOSE only considers the past history of a sequence, it also may be adapted for streaming data. This could e.g. be realised by using a sliding window, which also could be included in order to speed up the run time. Generally, future work might focus on run time optimization leading to the usage of CLOSE becoming more attractive.

ACKNOWLEDGMENTS

We thank the Jürgen Manchot Foundation, which funds the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University Duesseldorf that partly supported this work. In addition, we want to thank Ludmila Himmelspach for helpful and inspiring discussions, and Pascal Braband for helping implementing the evolutionary clustering algorithms for the evaluation section.

REFERENCES

[1] Ansari Saleh Ahmar, Suryo Guritno, Abdurakhman, Abdul Rahman, Awi, Alimuddin, Ilham Minggi, M Arif Tiro, M Kasim Aidid, Suwardi Annas, Dian Utami Sutiksno, Dewi S Ahmar, Kurniawan H Ahmar, A Abqary Ahmar, Ahmad Zaki, Dahlan Abdullah, Robbi Rahim, Heri Nurdiyanto, Rahmat Hidayat, Darmawan Napitupulu, Janner Simarmata, Nuning Kurniasih, Leon Andretti Abdillah, Andri Pranolo, Haviluddin, Wahyudin Albra, and A Nurani M Arifin. 2018. Modeling Data Containing Outliers using ARIMA Additive Outlier (ARIMA-AO). *Journal of Physics: Conference Series* 954 (2018).

[2] Arindam Banerjee and Joydeep Ghosh. 2001. Clickstream Clustering Using Weighted Longest Common Subsequences. In *Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining*. 33–40.

[3] Shai Ben-David and Ulrike Von Luxburg. 2008. Relating Clustering Stability to Properties of Cluster Boundaries. In *21st Annual Conference on Learning Theory (COLT 2008)*. 379–390.

[4] Jürgen Beringer and Eyke Hüllermeier. 2007. Adaptive Optimization of the Number of Clusters in Fuzzy Clustering. In *Proceedings of the IEEE International Conference on Fuzzy Systems*. 1–6.

[5] James C Bezdek. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer Science & Business Media.

[6] Mohamed Bouguessa, Shengrui Wang, and Haojun Sun. 2006. An Objective Approach to Cluster Validation. *Pattern Recognition Letters* 27 (2006), 1419–1430.

[7] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. 2006. Evolutionary Clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. 554–560.

[8] Jason R Chen. 2007. Useful Clustering Outcomes from Meaningful Time Series Clustering. In *Proceedings of the Sixth Australasian Conference on Data mining and Analytics*, Vol. 70. 101–109.

[9] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L Tseng. 2009. On Evolutionary Spectral Clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3, 4 (2009), 1–30.

[10] David L. Davies and Donald W. Bouldin. 1979. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1, 2 (1979), 224–227.

[11] John C. Dunn. 1973. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3, 3 (1973), 32–57.

[12] Jason Ernst, Gerard J. Nau, and Ziv Bar-Joseph. 2005. Clustering Short Time Series Gene Expression Data. *Bioinformatics* 21, suppl_1 (2005), i159–i168.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 226–231.

[14] Yoshiki Fukuyama. 1989. A New Method of Choosing the Number of Clusters for the Fuzzy C-Mean Method. In *Proceedings of the 5th Fuzzy Systems Symposium*. 247–250.

[15] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. 2003. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering* 15, 3 (2003), 515–528.

[16] Xiaohui Huang, Yunming Ye, Liyan Xiong, Raymond Y.K. Lau, Nan Jiang, and Shaokai Wang. 2016. Time Series K-Means: A New K-Means Type Smooth Subspace Clustering for Time Series Data. *Information Sciences* 367-368 (2016), 1–13.

[17] Eyke Hüllermeier and Maria Rifqi. 2009. A Fuzzy Variant of the Rand Index for Comparing Clustering Structures. In *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference*. 1294–1298.

[18] Xiaoming Jin, Yuchang Lu, and Chunyi Shi. 2002. Distribution Discovery: Local Analysis of Temporal Rules. In *Advances in Knowledge Discovery and Data Mining*, Ming-Syan Chen, Philip S. Yu, and Bing Liu (Eds.). 469–480.

[19] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. 2005. On Discovering Moving Clusters in Spatio-temporal Data. In *Advances in Spatial and Temporal Databases*. 364–381.

[20] Yoshinobu Kawahara and Masashi Sugiyama. 2009. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, 389–400.

[21] Eamonn Keogh and Jessica Lin. 2005. Clustering of Time-Series Subsequences is Meaningless: Implications for Previous and Future Research. *Knowledge and Information Systems* 8, 2 (2005), 154–177.

[22] Tung Kieu, Bin Yang, and Christian S. Jensen. 2018. Outlier Detection for Multidimensional Time Series Using Deep Neural Networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. 125–134.

[23] Min-Soo Kim and Jiawei Han. 2009. A Particle-and-Density Based Evolutionary Clustering Method for Dynamic Networks. *Proceedings of the VLDB Endowment* 2, 1 (2009), 622–633.

[24] Young-Il Kim, Dae-Won Kim, Doheon Lee, and Kwang Lee. 2004. A Cluster Validation Index for GK Cluster Analysis Based on Relative Degree of Sharing. *Information Sciences* 168 (2004), 225–242.

[25] Gerhard Klassen, Martha Tatusch, and Stefan Conrad. 2020. Clustering of Time Series Regarding Their Over-Time Stability. In *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI)*.

[26] Gerhard Klassen, Martha Tatusch, Ludmila Himmelspach, and Stefan Conrad. 2020. Fuzzy Clustering Stability Evaluation of Time Series. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems, 18th International Conference, IPMU 2020*. 680–692.

[27] Katarina Košmelj and Vladimir Batagelj. 1990. Cross-Sectional Approach for Clustering Time Varying Data. *Journal of Classification* 7, 1 (1990), 99–109.

[28] Mahesh Kumar, Nitin R. Patel, and Jonathan Woo. 2002. Clustering Seasonality Patterns in the Presence of Errors. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. 557–563.

[29] Ludmila I. Kuncheva and Dmitry P. Vetrov. 2006. Evaluation of Stability of k-Means Cluster Ensembles with Respect to Random Initialization. *IEEE Transactions of Pattern Analysis and Machine Intelligence* 28, 11 (2006), 1798–1808.

[30] Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Peter Filzmoser. 2018. Time Series Analysis: Unsupervised Anomaly Detection Beyond Outlier Detection. In *ISPEC*. 19–36.

[31] Hoel Le Capitaine and C. Frelicot. 2011. A Cluster-Validity Index Combining an Overlap Measure and a Separation Measure Based on Fuzzy-Aggregation Operators. *IEEE Transactions on Fuzzy Systems* 19 (2011), 580–588.

[32] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. 2013. Change-point Detection in Time-series Data by Relative Density-ratio Estimation. *Neural Networks* 43 (July 2013), 72–83.

[33] James MacQueen. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. 281–297.

[34] Mohsin Munir, Shoaib Ahmed Siddiqui, Muhammad Ali Chattha, Andreas Dengel, and Sheraz Ahmed. 2019. FuseAD: Unsupervised Anomaly Detection in Streaming Sensors Data by Fusing Statistical and Deep Learning Models. *Sensors* 19, 11 (2019), 2451–2465.

[35] Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. 2001. Streaming-Data Algorithms for High-Quality Clustering. In *Proceedings of IEEE International Conference on Data Engineering*. 685–694.

[36] John Paparrizos and Luis Gravano. 2015. k-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. 1855–1870.

[37] Domenico Piccolo. 2008. A Distance Measure for Classifying ARIMA Models. *Journal of Time Series Analysis* 11 (2008), 153 – 164.

[38] Marco Ramoni, Paola Sebastiani, and Paul Cohen. 2000. Multivariate Clustering by Dynamics. In *AAAI/IAAI*. 633–638.

[39] William M Rand. 1971. Objective Criteria for the Evaluation of Clustering Methods. *J. Amer. Statist. Assoc.* 66, 336 (1971), 846–850.

[40] Volker Roth, Tilman Lange, Mikio Braun, and Joachim Buhmann. 2002. A Resampling Approach to Cluster Validation. *COMPSTAT* (2002), 123–128.

[41] Peter J. Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65.

[42] Thomas A. Runkler. 2010. Comparing Partitions by Subset Similarities. In *Proceedings of the 13th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU*. 29–38.

[43] Stan Salvador and Philip Chan. 2007. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[44] Pei Sun, Sanjay Chawla, and Bavani Arunasalam. 2006. Mining for Outliers in Sequential Databases. In *Proceedings of the 2006 SIAM International Conference on Data Mining*. 94–106.

[45] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. 2019. Show Me Your Friends and I'll Tell You Who You Are. Finding Anomalous Time Series by Conspicuous Cluster Transitions. In *Data Mining. AusDM 2019. Communications in Computer and Information Science*, Vol. 1127. 91–103.

[46] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. 2020. How is Your Team Spirit? Cluster Over-Time Stability Evaluation. In *Machine Learning and Data Mining in Pattern Recognition, 16th International Conference on Machine Learning and Data Mining, MLDM 2020*. 155–170.

[47] Martha Tatusch, Gerhard Klassen, and Stefan Conrad. 2020. Behave or Be Detected! Identifying Outlier Sequences by Their Group Cohesion. In *Big Data Analytics and Knowledge Discovery, 22nd Int. Conference, DaWaK 2020*. 333–347.

[48] Martha Tatusch, Gerhard Klassen, and Stefan Conrad. 2020. Loners Stand Out. Identification of Anomalous Subsequences Based on Group Performance. In *Advanced Data Mining and Applications, ADMA 2020*. 360–369.

[49] Michail Vlachos, Jessica Lin, Eamonn Keogh, and Dimitrios Gunopulos. 2003. A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time Series. In *In Proceedings of the Workshop on Clustering High Dimensionality Data and its Applications*.

[50] Ulrike von Luxburg. 2010. Clustering Stability: An Overview. *Foundations and Trends in Machine Learning* 2, 3 (2010), 235–274.

[51] T. Warren Liao. 2005. Clustering of Time Series Data — A Survey. *Pattern Recognition* 38, 11 (2005), 1857 – 1874.

[52] Xuanli Lisa Xie and Gerardo Beni. 1991. A Validity Measure for Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13, 8 (1991), 841–847.

[53] Yimin Xiong and Dit-Yan Yeung. 2002. Mixtures of ARMA Models for Model-Based Time Series Clustering. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 717 – 720.

[54] Kevin S Xu, Mark Kliger, and Alfred O Hero Iii. 2014. Adaptive Evolutionary Clustering. *Data Mining and Knowledge Discovery* 28, 2 (2014), 304–336.

[55] Yuxun Zhou, Han Zou, Reza Arghandeh, Weixi Gu, and Costas J. Spanos. 2018. Non-Parametric Outliers Detection in Multiple Time Series A Case Study: Power Grid Data Analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. 4605–4612.

5.2 Further Research

Clustering time series is a major challenge for all disciplines involved. The question arises not only about the appropriate method but also about the choice of parameters for a selected method. We have presented first approaches for an evaluation of clusterings and have shown that these approaches are also suitable for further applications. However, this has also raised new research questions that should be addressed in future work. One of these questions relates to the fact that many data sets are incomplete. While it can be said by definition that missing values are tolerated, the question arises as to how much they influence the evaluation result. In order to make a corresponding statement, further research is necessary so that users can rely on the achieved results in scenarios in which data is missing.

Another interesting question relates to applications based on FCSETS. Due to the similarity to CLOSE, it can be assumed that such applications work similarly well, but this requires proof, which could be provided in future work.

The range of applications should also be expanded to confirm the performance of the evaluation procedures in other ways. For example, methods for predicting cluster compositions or data points or methods for identifying meaningful features would be conceivable. As mentioned in the introduction, the foundation we have laid can also contribute to solving the problem for other data types. To what extent our methods can be transferred to other data types still needs to be researched.

6

Publications

6.1 Related Publications

[R1] Gerhard Klassen, Martha Tatusch, and Stefan Conrad. Clustering of Time Series Regarding Their Over-Time Stability. In *Big Data Analytics and Knowledge Discovery*, Lecture Notes in Computer Science, pages 333–347. Springer International Publishing, Cham, 2020.

Contributions: The concept and main idea of the paper were mainly designed by Gerhard Klassen with the support of Martha Krakowski (née Tatusch). The methods and experiments were mainly implemented by Gerhard Klassen. Except for the *Method* and *Evaluation* sections, the paper was written entirely by Gerhard Klassen. The paper was supervised by Prof. Dr. Stefan Conrad.

Status: published

[R2] Gerhard Klassen, Martha Tatusch, and Stefan Conrad. Cluster-Based Stability Evaluation in Time Series Data Sets. 2021.

Contributions: The presented methods were developed in earlier papers. The new experiments concerning the time series cluster evaluation were carried out by Gerhard Klassen. The real world data sets were aquired and preprocessed by Gerhard Klassen. The manuscript was written in equal parts by the two main authors under the supervision of Prof. Dr. Stefan Conrad. The paper has been submitted to ACM Transactions on Knowledge Discovery from Data (TKDD) in May 2021.

Status: submitted

[R3] Gerhard Klassen, Martha Tatusch, Ludmila Himmelspach, and Stefan Conrad. Fuzzy Clustering Stability Evaluation of Time Series. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Communications

in Computer and Information Science, pages 680–692, Cham, 2020. Springer International Publishing .

Contributions: Gerhard Klassen contributed the main idea of the paper, the implementation and the experiments. The formal concept was jointly developed by Gerhard Klassen and Ludmila Himmelspach. The manuscript was jointly prepared by Martha Krakowski (née Tatusch), Ludmila Himmelspach and Gerhard Klassen under the supervision of Prof. Dr. Stefan Conrad.

Status: published

[R4] Gerhard Klassen, Martha Tatusch, Weisong Huo, and Stefan Conrad. Evaluating Machine Learning Algorithms in Predicting Financial Restatements. In *Proceedings of the 4th International Conference on Business and Information Management*, ICBIM '20, New York, NY, USA, 2020. Association for Computing Machinery .

Contributions: The idea for this paper came from Martha Krakowski (née Tatusch) and Gerhard Klassen . Weisong Huo implemented the algorithms under the guidance of Gerhard Klassen. The paper was written entirely by Gerhard Klassen. Martha Krakowski (née Tatusch) was mostly involved in an advisory capacity. Prof. Dr. Stefan Conrad supervised the work.

Status: published

[R5] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. Show Me Your Friends and I'll Tell You Who You Are. Finding Anomalous Time Series by Conspicuous Cluster Transitions. In *Data Mining*, Communications in Computer and Information Science, pages 91–103. Springer Singapore, Singapore, 2019.

Contributions: The presented method was developed jointly by Martha Krakowski (née Tatusch) and Gerhard Klassen. Martha Krakowski (née Tatusch) has formalised the idea mathematically and implemented the outlier detection algorithm. Gerhard Klassen was responsible for preprocessing and the experiments on the real world data. The manuscript was written in equal parts by the two main authors under the supervision of Jun.-Prof. Dr. Marcus Bravidor and Prof. Dr. Stefan Conrad.

Status: published

[R6] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. How is Your Team Spirit? Cluster Over-Time Stability Evaluation. In *Machine Learning and Data Mining in Pattern Recognition - 16th International Conference, MLDM 2016, New York, NY, USA, July 18-23, 2020, Proceedings*, Lecture Notes in Computer Science, pages 155–170. ibai-publishing, 2020.

Contributions: Gerhard Klassen contributed with the basic idea, the preprocessing, the acquisition of the data sets and the experiments with the Eikon data set (Section 5.1) and the GlobalEconomy data set (Section 5.2). The manuscript was jointly written by the two main authors Martha Krakowski (née Tatusch) and Gerhard Klassen under supervision of Jun.-Prof. Dr. Marcus Bravidor and Prof. Dr. Stefan Conrad.

Status: published

[R7] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. Predicting Erroneous Financial Statements Using a Density-Based Clustering Approach. In *Proceedings of the 4th International Conference on Business and Information Management*, ICBIM '20, New York, NY, USA, 2020. Association for Computing Machinery.

Contributions: The main idea for this work was developed by Martha Krakowski (née Tatusch) and Gerhard Klassen . Both main authors implemented the described approach and the associated experiments in equal parts. Jun.-Prof. Dr. Marcus Bravidor motivated the work and provided the business context. Prof. Dr. Stefan Conrad supervised the work.

Status: published

[R8] Martha Tatusch, Gerhard Klassen, and Stefan Conrad. Behave or Be Detected! Identifying Outlier Sequences by Their Group Cohesion. In *Big Data Analytics and Knowledge Discovery*, Lecture Notes in Computer Science, pages 333–347. Springer International Publishing, Cham, 2020.

Contributions: The idea and the conception of the presented paper were mainly developed and implemented by Martha Krakowski (née Tatusch) . Gerhard Klassen was responsible for obtaining and preprocessing the real world data sets and conducting the related experiments. The manuscript was prepared in equal parts by the two main authors under the supervision of Prof. Dr. Stefan Conrad.

Status: published

[R9] Martha Tatusch, Gerhard Klassen, and Stefan Conrad. Loners stand out. Identification of anomalous subsequences based on group performance. In *Advanced Data Mining and Applications - 16th International Conference, ADMA 2020, Foshan, China, November 12-14, 2020, Proceedings*, Lecture Notes in Computer Science. Springer, 2020.

Contributions: The concept and main idea of the paper was developed jointly by Martha Krakowski (née Tatusch) and Gerhard Klassen. Martha Krakowski (née Tatusch) implemented the proposed method and wrote the major part of the paper. Gerhard Klassen mainly had an advisory role in the implementation and writing of the paper. Prof. Dr. Stefan Conrad supervised the work.

Status: published

6.2 Further Publications

[F1] Kirill Bogomasov, Ludmila Himmelspach, Gerhard Klassen, Martha Tatusch, and Stefan Conrad. Feature-based approach for severity scoring of lung tuberculosis from ct images. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018*, CEUR Workshop Proceedings. CEUR-WS.org, 2018.

[F2] Stefan Conrad, Martha Tatusch, Kirill Bogomasov, and Gerhard Klassen. Bilddaten in den digitalen geisteswissenschaften. In *Bilddaten in den Digitalen Geisteswissenschaften*, Episteme in Bewegung, pages 85–99. Harrassowitz Verlag, Wiesbaden, 2020.

[F3] Gerhard Klassen and Stefan Conrad, editors. *Proceedings of the 30th GI-Workshop Grundlagen von Datenbanken, Wuppertal, Germany, May 22-25, 2018*, CEUR Workshop Proceedings. CEUR-WS.org, 2018.

[F4] Gerhard Klassen and Michael Singhof. Shape based outlier detection in slic superpixels. In *Proceedings of the 29th GI-Workshop Grundlagen von Datenbanken, Blankenburg/Harz, Germany, May 30 - June 02, 2017*, CEUR Workshop Proceedings, pages 60–65. CEUR-WS.org, 2017.

[F5] Michael Singhof, Gerhard Klassen, Daniel Braun, and Stefan Conrad. Detection and implicit classification of outliers via different feature sets in polygonal chains. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme (DBIS), 6.-10. März 2017, Stuttgart, Germany, Proceedings*, LNI, pages 237–246. GI, 2017.

References

[1] Evolutionary clustering. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 554–560, 2006.

[2] Charu C Aggarwal and Chandan K Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 1st edition, 2013.

[3] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering - A decade review. *Information Systems*, 53, 2015.

[4] Dibya Jyoti and Bora and Dr. Anil Kumar Gupta. A Comparative study Between Fuzzy Clustering Algorithm and Hard Clustering Algorithm. *International Journal of Computer Trends and Technology*, 10(2):108–113, April 2014.

[5] Mihael Ankerst, Markus M. Breunig, Hans Peter Kriegel, and Jörg Sander. OPTICS: Ordering Points to Identify the Clustering Structure. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(2):49–60, June 1999.

[6] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 07-09-Janu of *SODA '07*, pages 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[7] Michael W. Berry, Azlinah Mohamed, and Bee Wah Yap. *Supervised and unsupervised learning*. Unsupervised and Semi-Supervised Learning. Springer International Publishing, Cham, 2020.

[8] James C. Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, (2-3):191–203, January 1984.

[9] Gert Böhme. *Fuzzy-Logik*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[10] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215, 2001.

[11] Andriy Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

[12] Waqqasur Rehman Butt and Martin Servin. Static object detection and segmentation in videos based on dual foregrounds difference with noise filtering. *Computing Research Repository (CoRR)*, abs/2012.10708, 2020.

[13] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 785–794, New York, NY, USA, August 2016. ACM.

[14] Xiang Chen, Dun Zhang, Yingquan Zhao, Zhanqi Cui, and Chao Ni. Software defect number prediction: Unsupervised vs supervised methods. *Information and Software Technology*, 106:161–181, 2019.

[15] Monica Chiş, Soumya Banerjee, and Aboul Ella Hassanien. Clustering time series data: An evolutionary approach. In *Studies in Computational Intelligence*, pages 193–207. 2009.

[16] T. Conlon, H. J. Ruskin, and M. Crane. Cross-correlation dynamics in financial time series. *Physica A: Statistical Mechanics and its Applications*, 388(5):705–714, March 2009.

[17] PATRICIA M. DECHOW, WEILI GE, CHAD R. LARSON, and RICHARD G. SLOAN. Predicting Material Accounting Misstatements*. *Contemporary Accounting Research*, 28(1):17–82, March 2011.

[18] Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5):533–534, May 2020.

[19] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, (3):32–57, 1973.

[20] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[21] Vladimir Estivill-Castro. Why so many clustering algorithms. *ACM SIGKDD Explorations Newsletter*, 4(1):65–75, June 2002.

[22] Brian Everitt. Cluster analysis. *Quality and Quantity*, 14(1):7–9, January 1980.

[23] Cristiano Hora Fontes, Izete Celestina Santos, Marcelo Embiruçu, and Pedro Aragão. Pattern reconciliation: A new approach involving constrained clustering of time series. *Comput. Chem. Eng.*, 145:107–169, 2021.

[24] Limin Fu and Enzo Medico. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinformatics*, (1):3, 2007.

[25] J. E. Gentle, L. Kaufman, and P. J. Rousseuw. *Finding Groups in Data: An Introduction to Cluster Analysis.*, volume 47. John Wiley & Sons, 1991.

[26] Fred H.M. Gertsen, Cees B.M. van Riel, and Guido Berens. Avoiding Reputation Damage in Financial Restatements. *Long Range Planning*, 39(4):429–456, August 2006.

[27] Michael Geurts, George E. P. Box, and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*, volume 14. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[28] C. W. J. Granger. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*, 37(3):424, August 1969.

[29] Jan Hauke and Tomasz Kossowski. Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data. *Quaestiones Geographicae*, 30(2):87–93, June 2011.

[30] Martin Heckmann. Supervised vs. unsupervised learning of spectro temporal speech features. In *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audition (SAPA)*, pages 1–6. ISCA, 2010.

[31] D. Horvatic, H. E. Stanley, and B. Podobnik. Detrended cross-correlation analysis for non-stationary time series with periodic trends. *EPL (Europhysics Letters)*, 94(1):18007, April 2011.

[32] Eyke Hullermeier, Maria Rifqi, Sascha Henzgen, and Robin Senge. Comparing Fuzzy Partitions: A Generalization of the Rand Index and Related Measures. *IEEE Transactions on Fuzzy Systems*, (3):546–556, 2012.

[33] Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 115–122. IEEE Comput. Soc, 2003.

[34] Ronald S King. *Cluster analysis and data mining: an introduction.* Mercury Learning and Information, Dulles, Virginia ; Boston, Massachusetts ; New Delhi, 2015.

[35] Gerhard Klassen and Michael Singhof. Shape based outlier detection in slic superpixels. In *Proceedings of the 29th GI-Workshop Grundlagen von Datenbanken, Blankenburg/Harz, Germany, May 30 - June 02, 2017*, CEUR Workshop Proceedings, pages 60–65. CEUR-WS.org, 2017.

[36] Duo Li, Yifei Zhao, and Yan Li. Time-series representation and clustering approaches for sharing bike usage mining. *IEEE Access*, 7:177856–177863, 2019.

[37] Jiamin Li and Harold W. Lewis. Fuzzy Clustering Algorithms - Review of the Applications. In *Proceedings - 2016 IEEE International Conference on Smart Cloud, SmartCloud 2016*, pages 282–288. IEEE, 2016.

[38] Jinbo Li, Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, 100:106919, March 2021.

[39] Taoying Li, Xu Wu, and Junhe Zhang. Time Series Clustering Model based on DTW for Classifying Car Parks. *Algorithms*, 13(3):57, March 2020.

[40] Taoying Li, Xu Wu, and Junhe Zhang. Time series clustering model based on DTW for classifying car parks. *Algorithms*, 13(3):57, 2020.

[41] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[42] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[43] Shie Mannor, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, and Xinhua Zhang. K-Means Clustering. In *Encyclopedia of Machine Learning*, pages 563–564. Springer US, Boston, MA, 2011.

[44] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with Naive Bayes - Which Naive Bayes? In *3rd Conference on Email and Anti-Spam - Proceedings, CEAS 2006*, 2006.

[45] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[46] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[47] Thomson Reuters. Eikon financial analysis and trading software.

[48] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[49] Gil Sadka. The economic consequences of accounting Fraud in Product Markets: Theory and a case from the U.S. telecommunications industry (WorldCom). *American Law and Economics Review*, 8(3):439–475, 2006.

[50] Warren S. Sarle, Anil K. Jain, and Richard C. Dubes. *Algorithms for Clustering Data*, volume 32. Prentice-Hall, Inc., USA, 1988.

[51] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[52] William W S Wei. *Time Series Analysis - Univariate and Multivariate Methods*. Pearson Addison Wesley, Boston, 2006.

[53] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, Amsterdam, 3 edition, 2016.

[54] Liying Zhang, Tao Pei, Bin Meng, Yuanfeng Lian, and Zhou Jin. Two-Phase Multivariate Time Series Clustering to Classify Urban Rail Transit Stations. *IEEE Access*, 8:167998–168007, 2020.

[55] Yuchao Zhang, Hongfu Liu, and Bo Deng. Evolutionary clustering with DBSCAN. In *2013 Ninth International Conference on Natural Computation (ICNC)*, pages 923–928. IEEE, 2013.