A Framework for Anomaly Detection
with Applications to Machine-Generated Data

ANDRÉ ERIKSSON

Abstract

Anomaly detection is an important issue in data mining and analysis, with applications in almost every area in science, technology and business that involves data collection. The development of general anomaly detection techniques can therefore have a large impact on data analysis across many domains. In spite of this, little work has been done to consolidate the different approaches to the subject.

In this report, this deficiency is addressed in the target domain of temporal machine-generated data. To this end, new theory for comparing and reasoning about anomaly detection tasks and methods is introduced, which facilitates a problem-oriented rather than a method-oriented approach to the subject. Using this theory as a basis, the possible approaches to anomaly detection in the target domain are discussed, and a set of interesting anomaly detection tasks is highlighted.

One of these tasks is selected for further study: *the detection of subsequences that are anomalous with regards to their context within long univariate real-valued sequences*. A framework for relating methods derived from this task is developed, and is used to derive new methods and an algorithm for solving a large class of derived problems. Finally, a software implementation of this framework along with a set of evaluation utilities is discussed and demonstrated.

Contents

1 INTRODUCTION

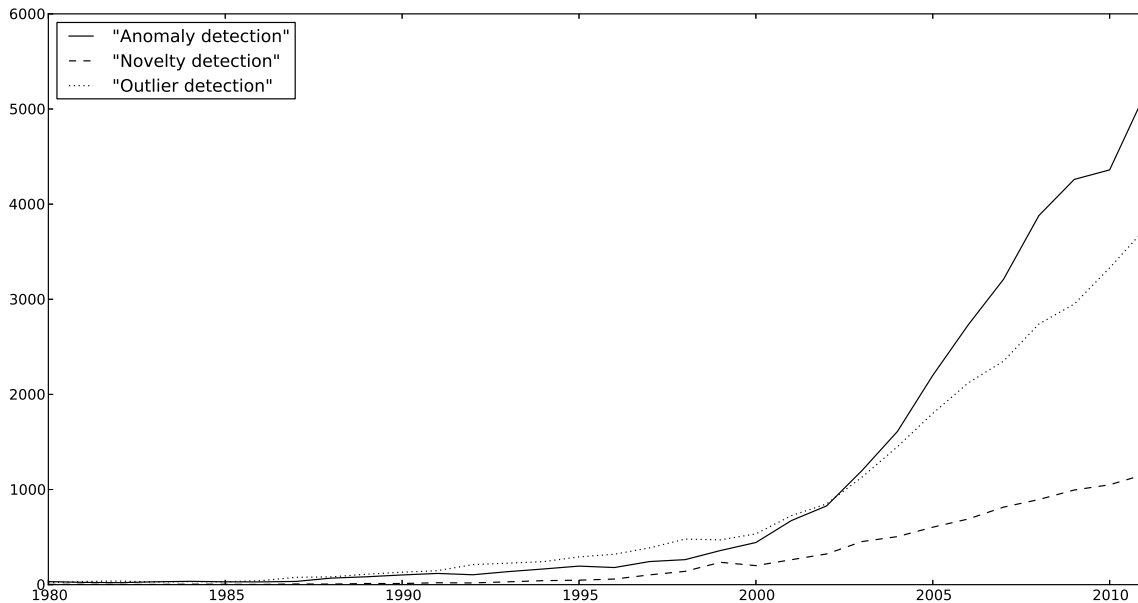Figure 1.1: Approximate number of papers (by year) published between 1980 and 2011 containing the terms "anomaly detection", "outlier detection" and "novelty detection". All three terms exhibit strong upward trends in recent years. Source: Google Scholar.

This report is the result of a master's thesis project at the KTH Royal Institute of Technology, performed partly in conjunction with an internship at Splunk Inc., based in San Francisco, California, USA. The goal of the project was to develop efficient and general methods of anomaly detection suitable for real-valued time series data. The main contributions of this thesis are:

1. A general framework for comparing and relating anomaly detection problems and methods.

2. An application of this framework to anomaly detection in real-valued time series.

3. Software for automatically discovering optimal anomaly detection methods for real-valued time series datasets.

Splunk is essentially a database and tool for storing and analyzing very large sets of sequential machine-generated data (i.e. machine-generated data consisting of discrete events with associated time stamps). The term *machine-generated data* refers to any data consisting of discrete events that have been created automatically from a computer process, application, or other machine without the intervention of a human. Common types of machine-generated data include computer, network, or other equipment logs; environmental or other types of sensor readings; or other miscellaneous data, such as location information [1]. Splunk is designed for this type of data, especially datasets where each event has an associated time stamp.

In recent years, anomaly detection has become increasingly important in a variety of domains in business, science and technology. Roughly defined as the automated detection within datasets of elements that are somehow abnormal, anomaly detection encompasses a broad set of techniques and problems. In

part due to the emergence of new application domains, and in part due to the evolving nature of many traditional domains, new applications of and approaches to anomaly detection and related subjects are being developed at an increasing rate, as indicated in Figure 1.1.

Since anomaly detection is an important and common problem in the domains in which Splunk is used, it was determined that Splunk could benefit for efficient, general anomaly detection methods. Furthermore, since continuous time series are easy to form from sequential machine-generated data, and readily amenable to analysis, these were selected as a focus.

In Chapter 2, various background information pertinent to the rest of the report is presented. Specifically, the subject of anomaly detection is presented in more depth, along with some background on the issues which are relevant to the rest of the report.

Due to a lack of previous research on general (i.e. non-application specific) anomaly detection methods for continuous time series, new theory was required. As part of the project, a framework was developed for interrelating about anomaly detection problems. This framework can be used to consolidate different approaches to anomaly detection and to simplify reasoning about and comparing anomaly detection problems and methods systematically. Chapter **??** introduces this framework.

Next, the framework was applied to anomaly detection in real-valued time series. As part of this, a general algorithm for solving a large class of time series anomaly detection problems was introduced. This algorithm can then be used together with an optimization strategy to automate the process of finding methods suitable for new datasets.

A software implementation of this algorithm was then developed. Called `ad-eval`, the implementation was designed as a minimalistic Python toolkit. It has been released under an open source license at `http://github.com/aeriksson/ad-eval`, in order to facilitate reproducibility in anomaly detection research. Both the application of the framework to time series, and `ad-eval` are treated in Chapter **??**.

Finally, a simple evaluation of a few implemented anomaly detection methods was performed using `ad-eval`, in order to demonstrate its use. The source code used in this evaluation was designed to be highly modular and was made available as part of the `ad-eval` source code repository, to guarantee the reproducibility of the results and to facilitate more thorough evaluations once adequate data becomes available. The results are presented in Chapter 5.

The report is concluded in Chapter 6 with a summary of the project and a few possible directions for future work.

2   BACKGROUND

This chapter gives a brief introduction to the subject of anomaly detection. The framework of tasks and problems used throughout the paper is presented and justified.

2.1   ANOMALY DETECTION

In essence, anomaly detection is the task of automatically detecting items (*anomalies*) in datasets that in some sense do not fit in with the rest of those datasets (i.e. are *anomalous* with regard to the rest of the data). The nature of both the datasets and anomalies are dependent on the specific application in which anomaly detection is applied, and vary drastically between application domains. As an illustration of this, consider the two datasets shown in Figures 2.2 and 2.2. While these are similar in the sense that they both involve sequences, they differ in the type of data points (real-valued vs. symbolic), the structure of the dataset (one long sequence vs. several sequences), as well as the nature of the anomalies (a subseqeuence vs. one sequence out of many). Several surveys [6] [11] [3] [7] and books [8] [9] [10] have been published which treat various anomaly detection applications in greater depth.
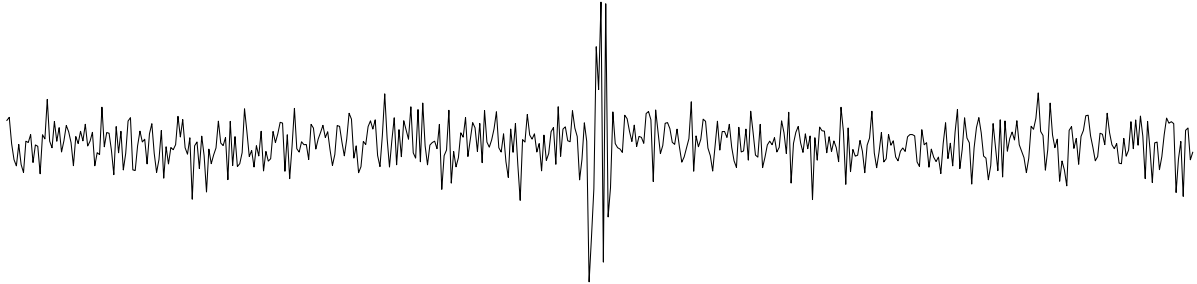

Figure 2.1: Real-valued sequence with an anomaly at the center.

Like many other concepts in machine learning and data science, the term 'anomaly detection' does not refer to any single well-defined problem. Rather, it is an umbrella term encompassing a collection of loosely related techniques and problems. Anomaly detection problems are encountered in nearly every domain in business and science in which data is collected for analysis. Naturally, this leads to a great diversity in the applications and implications of anomaly detection techniques. Due to this wide scope, anomaly detection is continuously being applied to new domains despite having been researched for decades.

S_1	login	passwd	mail	ssh	...	mail	web	logout
S_2	login	passwd	mail	web	...	web	web	logout
S_3	login	passwd	mail	ssh	...	web	web	logout
S_4	login	passwd	web	mail	...	web	mail	logout
S_5	login	passwd	login	passwd	login	passwd	...	logout

Figure 2.2: Several sequences of user commands. The bottom sequence is anomalous compared to the others.

In other words, anomaly detection as a subject encompasses a diverse set of problems, methods, and applications. Different anomaly detection problems and methods often have few similarities, and no unifying theory exists. Indeed, the eventual discovery of such a theory seems highly unlikely, considering the subjectivity inherent to most anomaly detection problems. Even the term 'anomaly detection' itself has evaded any widely accepted definition [6] in spite of multiple attempts.

Despite this diversity, anomaly detection problems from different domains often share some structure, and studying anomaly detection as a subject can be useful as a means of understanding and exploiting such common structure. Anomaly detection methods are vital analysis tools in a wide variety of domains, and the set of scientific and commercial domains which could benefit from improved anomaly detection methods is huge. Indeed, due to increasing data volumes, exhaustive manual analysis is (or will soon be) prohibitively expensive in many domains, rendering effective automated anomaly detection critical to future development.

2.2 ON ANOMALY DETECTION RESEARCH

Most anomaly detection research work consists of either taking existing methods and applying them to new applications (i.e. on new types of data), or investigating new methods for previously studied applications. In order to handle the increasing need for effective anomaly detection in many areas of business and science it is vital that these activities can be performed in a highly automated and straightforward manner. However, there are a few issues with the current state of the subject, which make anomaly detection research needlessly complicated.

Firstly, comparing different anomaly detection methods found in the literature is difficult, since even though it might not appear so at first glance, papers on anomaly detection often target subtly different problems. For instance, TODO. This renders direct comparisons problematic and makes it hard to assess which methods are appropriate to use in new applications. A systematic way of comparing anomaly detection methods would be helpful in mitigating this problem.

The second problem is that there is often a lack of reproducibility of produced results. Due in part to the subjective nature of the subject, and in part to a historical lack of freely available datasets, new methods are often not adequately compared to previous methods. Furthermore, the performance of many anomaly detection methods is often highly dependent on parameter choices, and only the results for the best parameter values (which might be difficult to find) are often presented [30]. Finally, source code is often unavailable, which makes verification a tedious process. These issues, when taken together, make it hard to reproduce results, which in turn makes anomaly detection research needlessly difficult.

This work attempts to simplify anomaly detection research by addressing the above issues. First, a general framework for systematically comparing anomaly detection problem formulations is presented, the purpose of which is to help highlight similarities and differences between problems, and thereby simplify the application of existing methods to new domains by mitigating the first problem above.

This framework is then used to formalise the subject of anomaly detection in the domain of real-valued

time series, and to reformulate the activity of finding appropriate methods for specific datasets in this domain as an optimization problem over the set of possible algorithms. It is then shown that solutions to this optimization problem can be algorithmically approximated for a large class of algorithms (including most previously published methods).

Finally, a software implementation of this optimization problem is presented, along with some preliminary performance results. This mitigates the second problem above for anomaly detection in real-valued time series, by providing an environment in which previous methods can be easily replicated and compared on arbitrary datasets.

One distinction that is important to make is between problems and methods. Informally, the process of finding an appropriate anomaly detection method for some application consists of two steps. First, an appropriate problem formulation must be found, which accurately captures intuitive notions of what constitutes an anomaly in the specific application. Next, a method of solving or finding approximate solutions to this problem must be constructed.

Due to the subjective nature of anomaly detection, radically different problem formulations might be appropriate for applications that are superficially very similar. Furthermore, there is often no obvious connection between the intuitive notion of what constitutes an anomaly in some application and the problem formulations which most accurately capture that notion, so prospective problem formulations must themselves be empirically evaluated.

This means that unless specific information is available on what problems are appropriate for a given application, finding the correct problem formulations should take priority over formulating methods. Finding efficient methods should be done only after it has been shown that the problem the methods are solving is relevant to the application. In the literature, methods are often emphasised at the expense of problems. Since the goal of this project was to find methods of automating the research process for arbitrary applications, there is instead a heavy emphasis on finding problems in this report.

2.3   Problem formulation

As a first step towards automating anomaly detection research, the problem of finding anomaly detection methods for some application must be formalised. In this section, the first step of the anomaly detection research process—finding an appropriate problem—is formulated as an optimization problem. The remainder of this report deals with this problem.

Informally, the idea is to search the set of possible problem formulations for the specific problem formulation with solutions (i.e. anomaly scores) that come the closest to the solutions which would be produced by a domain expert.

To begin with, the sets of valid problem inputs (datasets) and outputs (solutions) must be defined. Here, we simply assume that some set D has been defined consisting of all the valid datasets for the current application (such as the set of all finite collections of images of some specific size and bit depth), and

that some set S has been defined consisting of all valid solutions (such as the indices of the two most anomalous images in a set).

TODO: replace S with $\forall d \in D : S_d$?

To begin with, a formal description of this problem requires that a set of all possible problem formulations is constructed. Here, it is simply assume that this set has been defined, and that it consists of the set of valid formulae in some logic sufficiently expressive to capture all relevant problem formulations. Let us call this set P.

Next, an objective function must be formulated, which associates with each problem formulation $p \in P$ how well p captures the anomalies of our hypothetical domain expert. Since how the objective function is computed is not relevant to the optimization problem formulation, it is helpful to introduce an oracle $O(p, i) : P \times D \to S$, which takes a problem $p \in P$ and an input dataset $i \in D$, and computes the associated solution $s_{p,i}$. The success of p in capturing the anomalies in i can then be stated as $\epsilon(i, O(p, i))$, where $\epsilon : D \times S \to \mathbb{R}^+$ is an error function, consistent with the assessments of our hypothetical domain expert, that assigns a value to each $s \in S$ according to how well it captures the anomalies in i.

Finally, since the goal is to minimise the expected error for datasets sampled from the given application, a random variable I over D must be introduced that models the probability of generating any given $d \in D$. A suitable objective function would then be $\mathbb{E}_I[\epsilon(i, O(p, i))]$.

The goal then, is to find

$$p_{opt} = \operatorname*{argmin}_{p \in P} \mathbb{E}_I[\epsilon(i, O(p, i))].$$

Here, p_{opt} corresponds to the best possible problem formulation for the specific ϵ, and $O'(i) = O(p_{opt}, i)$ to the algorithm which computes the answer. Of course, computing either of these is impossible, for a few reasons. The biggest theoretical hurdle is that, for any logic P^* sufficiently complicated to encompass non-trivial problems, the problem of producing a solution S for an arbitrary problem definition P with input data I is uncomputable (TODO: verify), so the oracle O described above can not exist. For the optimization problem to be of any real-world use, P must be restricted to a set of problems that can be solved by some computable oracle O.

Another problem is that it is generally not possible to directly formulate either X or ϵ. To appropriately select X would require a steady stream of incoming datasets, something that is typically not available for most applications. Similarly, since ϵ essentially represents an idealised, objective domain expert's opinion of what parts of X constitute anomalies, it can not be computed without (implicit) knowledge of P^o. To get around these issues, X and ϵ must be replaced with a set of training data $T \subset D$. Unless specific knowledge of the domain is available from which ϵ can be reconstructed, this data must also be *labeled*, i.e. with each $t_i \in T$ must be associated a $s_i \in S$.

The rest of this report builds upon this problem formulation. In chapter **??**, an algorithm is constructed which can solve a large class anomaly detection problems for time series. This algorithm is then taken to be a partial representation O^* of O. Correspondingly P is replaced by the set $P^* \subset P$ of problems which can be solved by O^*. Finally, ϵ can be approximated by $\epsilon^*(s_a, s_b) = \delta(s_a, s_b)$ for some suitable distance

measure δ. This enables the computation of the following estimate of p_{opt}:

$$p_{opt}^* = \underset{p \in P^*}{\operatorname{argmin}} \sum_{(i_j, s_j) \in T} \epsilon^*(s_j, O(p, i_j)).$$

With this approach, finding the most accurate anomaly detection algorithm (supported by O^*) for T becomes an optimization problem over P^*, where the objective function is $\sum_{(i_j, s_j) \in T} \epsilon^*(s_j, O(p, i_j))$.

The main purpose of the framework presented in the next chapter is to simplify the formulation of a suitable restricted oracle O^*, by providing a means of systematically constraining P^*. This approach is taken in Chapter ?? to construct a oracle that can solve most of the time series anomaly detection problems found in the literature, while remaining tractable.

$$p_{opt}^* = \underset{p \in P^*}{\operatorname{argmin}} \sum_{(i_j, s_j) \in T} \epsilon^*(s_j, O(p, i_j)).$$

3   A Framework for Anomaly Detection

In this chapter, a framework for reasoning about anomaly detection problems is presented. In this report, the framework is mainly utilised as a means of limiting the scope of the optimisation problem outlined in the previous chapter, but it is also useful as a means of relating different anomaly detection methods and finding new methods for specific applications.

A core idea of the framework is to classify and interrelate anomaly detection methods based on a few key factors. In Section 3.1, the basic principles and motivations behind the framework are presented. In sections 3.2 through 3.6, the individual factors, as well as common choices for them, are presented.

Finally, in Section **??**, a few problems that are typically considered related to anomaly detection are presented in relation to the framework.

3.1   Description

As previously mentioned, the optimisation problem presented in section 2.3 is intractable since to the set of possible problem formulations is so large. In order to successfully apply the optimisation problem to some specific application domain, the set of problems under consideration must be restricted. However, appropriate restricted set of problems can be problematic. Ideally, such a problem set should contain all problems previously used for the specific domain, while remaining small enough to be tractable. Without some method of systematically relating problems and approaches, however, finding such a set can be complicated.

We now present a framework, using which finding an appropriate problem set can be done systematically. The key idea behind this framework is that if anomaly detection problems can be described using a few choices of standardised factors, then comparing problems, as well as restricting the problem set and finding gaps in the methods researched for applications becomes much simpler.

We here propose using the following factors to classify anomaly detection problems:

Dataset format How the datasets (method inputs) are structured.

Solution format How the solutions (method outputs) should be structured.

Training data What type of training data is available to methods.

Anomaly type Which structural properties of the data should be considered.

Anomaly measure The heuristic used to assess how anomalous items are.

As mentioned previously, one major advantage of an approach based on factors is that it facilitates comparisons of methods. Studying common factor choices between problems can be useful in illuminating differences and similarities. Furthermore, as we will see, specific factor choices often generalise other factor choices. Recognising this is crucial to finding general methods. Thus, the framework can be utilized to recognize when problems are similar or (partially) generalise other problems.

The remainder of this chapter deals with each of the factors presented above in turn, listing choices treated by existing methods, and introducing a few new concepts along the way.

3.2 DATASET FORMAT

Obviously, the available choices of dataset format are typically heavily restricted by the application under consideration. However, transforming source datasets to some more appropriate format is a common and important task. Since the format of the data also limits the set of suitable problem definitions, selecting an appropriate format is important.

In the literature, various classifications are used to distinguish different dataset formats. In this section, a few of these are presented.

To begin with, assuming that input is given as a dataset $D = (d_1, d_2, \ldots, d_n)$, where the d_i belong to some set X, the structure of X is of interest. A distinction is typically made between categorical, discrete, and real-valued datasets based on the properties of X. A dataset is said to be *categorical* (or *symbolic*) if X is finite, *discrete* if X is countable and *real-valued* if $X \subseteq \mathbb{R}^n$ for some n. It is also frequently the case that X consists of some combination of categorical, discrete and real-valued data. In this case, D is referred to as *mixed*.

Categorical data arises in many contexts and is comparatively easy to handle. In many cases, methods for handling categorical data are relatively mature. For instance, categorical sequences are exhaustively treated in [?]. Often, there is no ordering or other relation between the elements in the underlying set, which makes the analysis simpler.

Many techniques used for categorical data (such as information-theoretic measures and certain probabilistic models) are not applicable to discrete or real-valued data. In such cases, a suitable discretisation of the underlying set might be useful.

Real-valued datasets encountered in applications are often samples of processes that are assumed to be continuous. When the ordering of these samples is reflected in the dataset, the dataset is sometimes referred to as *continuous*.

Anomaly detection in mixed datasets is relatively poorly understood; typically, such data is split into several separate datasets that are either categorical, discrete or real-valued before analysis.

It is fairly common to use *numerosity reduction* techniques to convert discrete data to categorical data, or to compress categorical data. An example of numerosity reduction in time series is shown in figure 4.1.

Another important classification considers the dimensionality of the data points. A distinction is typically made between univariate and multivariate data. A dataset $D = (d_1, d_2, \ldots, d_n)$ is said to be *univariate* if the d_i are scalars, and *multivariate* if the d_i are vectors of length greater than one. An illustration of uni- and multivariate time series is shown in figure 3.1.
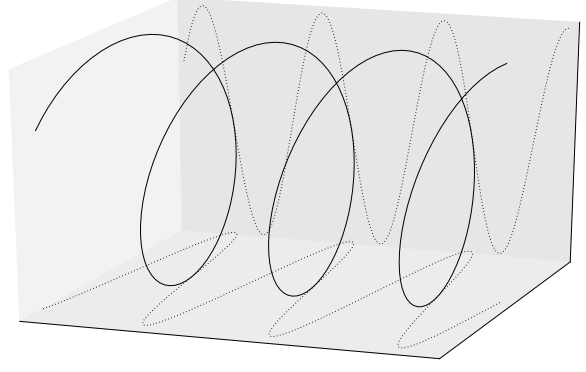
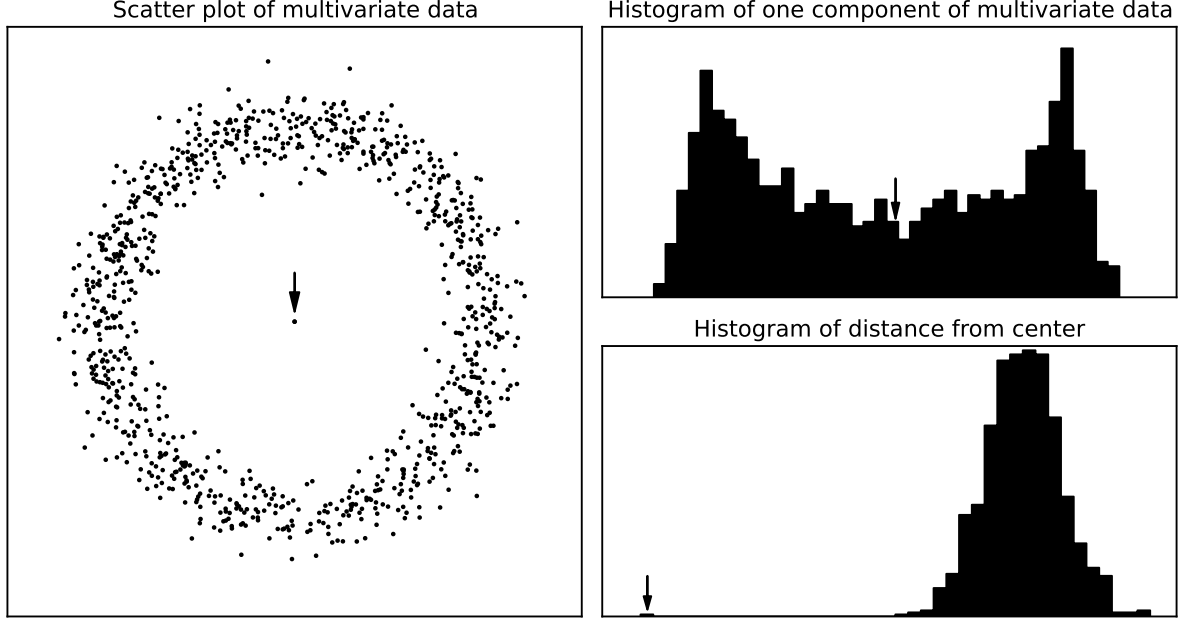Figure 3.1: Two sine curves regarded as two separate

Figure 3.2: An example of dimensionality reduction in a point anomaly detection problem in R^2. The left figure shows a set of 500 data points (x_i, y_i) containing one anomaly. The top right figure shows a histogram of the x_i, while the bottom right figure shows a histogram of the distance from the center point. In each figure, the location of the anomalous point is marked by an arrow. While the anomaly is easy to detect in the left and bottom right figures, it can not be seen in the top right figure. This is due to the linear inseparability of the data, and illustrates how dimensionality reduction can lead to information losses if not performed properly.

While the distinction between uni- and multivariate data might seem superfluous, it proves important in applications. Most machine learning methods take significantly longer to learn (both in terms of time and convergence) as the dimensions of the data increase. Furthermore, many methods are not applicable to multivariate data at all. Any multivariate dataset may be trivially split into a set of univariate datasets, something that should reasonably always be done unless there are anomalies in the multivariate dataset which are not reflected in the individual dimensions (for an example of this, see figure 3.2).

To mitigate the difficulties of analysing high-dimensional data, *dimensionality reduction* is often performed as a pre-analysis step on multidimensional datasets. Essentially, dimensionality reduction refers to the process of transforming multidimensional data to a lower-dimensional representation, such that the aspects most pertinent to the analysis remain.

Many techniques have been designed with this goal in mind. A distinction is typically made between *feature selection* and *feature extraction* approaches. Feature selection approaches try to select a subset

of the dimensions present in the original data. Feature extraction approaches, in contrast, transform the data into some new space, in which the relevant features are hopefully more apparent. Feature extraction methods are commonly employed as a pre-processing step in anomaly detection algorithms.

Common feature extraction methods for data in $\mathbb{R}^n$ include *principle component analysis* [?] (PCA), *semidefinite embedding* [?], *partial least-squares regression* [?], and *independent component analysis* [?]. Which methods are appropriate to use depends heavily on the application. An example of feature extraction is shown in figure 3.2.

In applications there is usually additional structure present, such as orderings or other relations between elements within datasets, that can be utilised to improve the analysis. Such additional structure is discussed in Sections 3.4 and 3.5.

3.3 TRAINING DATA

As is customary in most areas of machine learning, anomaly detection problems are classified as either *supervised*, *semi-supervised* or *unsupervised* [1] based on the availability of *training* (or *training*) data. In contrast to the input dataset, the training data acts as a baseline, defining what constitutes normal and anomalous.

In *supervised* anomaly detection, training data containing both normal and anomalous items is available. In essence, this constitutes a traditional supervised classification problem. As such, it can be handled by any two-class classifier, such as regular support vector machines. Unfortunately, supervised approaches are usually not suitable for anomaly detection, for a few reasons. To begin with, anomalous training data is almost always relatively scarce, potentially leading to skewed classes (described in [12] and [13]). Secondly, supervised anomaly detection methods are by definition unable to detect types on anomalies that are not represented in the training data, and so can not be used to find *novel* anomalies. This is problematic as it is often not possible to obtain training data containing all possible anomalies.

Figure 3.3: Euler diagram of the available training data for the four types of supervision.

Semi-supervised anomaly detection, on the other hand, assumes the availability of only one class of training data. While anomaly detection with only anomalous training data has been discussed (for
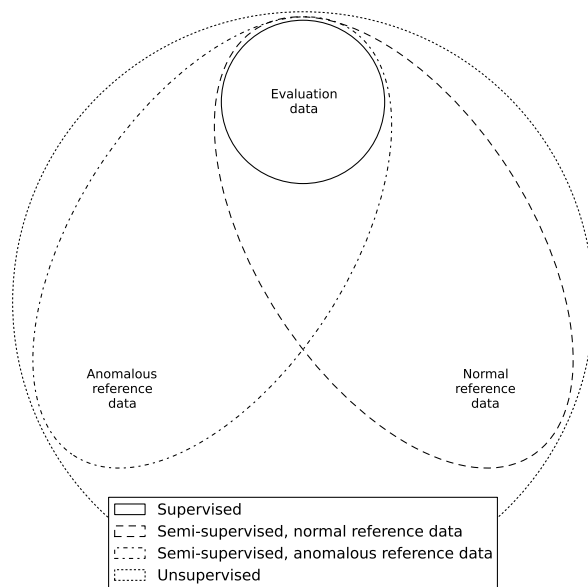
---

[1]Note that we here use the convention of [3], and take supervised learning to mean that both classes of training data are available, and semi-supervised to mean that only one class of training data is available. Conventially, supervised learning is taken to mean any learning from training data, and semi-supervised learning is taken to mean that both labeled and unlabeled data is available [?].

instance in [14]), the vast majority of semi-supervised methods assume that normal training data is available. Considering the difficulties involved in obtaining anomalous training data mentioned above, this should not be surprising. Semi-supervised methods are used more frequently than supervised methods in part due to the relative ease of producing normal training data to anomalous training data.

Finally, *unsupervised* anomaly detection requires no training dataset. Since training data is not always available, unsupervised methods are typically considered to be of wider applicability than both supervised and semi-supervised methods [3]. However, unsupervised methods are unsuitable for certain tasks. Since training data can not be manually specified, it is more difficult to sift out uncommon but uninteresting items in unsupervised anomaly detection than in semi-supervised anomaly detection. Furthermore, unsupervised methods will not detect anomalies that are common but unexpected (although such items are arguably not anomalies by definition).

It is useful to note that unsupervised anomaly detection problems can often be reduced to semi-supervised anomaly detection problems by letting the input dataset serve as normal training data and modifying the anomaly measure such that all elements are judged as dissimilar to themselves.

Of course, it is sometimes not feasible or desirable to compare items with the entirety of the training dataset. This is mainly the case when the dataset supports additional structure, such an ordering or metric, which gives rise to a natural concept of locality within the dataset. As a concrete example of such an application, consider unsupervised anomaly detection in a long sequence: often how an item compares to those items 'closest' to it (in the ordering) is much more relevant to whether or not that item should be considered an anomaly than how it compares to the rest of the sequence.

In such cases, it is reasonable to associate with each individual data item a subset of the training data, and let this subset constitute the training data for that item. Based on the discussion in [3], we refer to such subsets as the *contexts* of the individual data items in this report. To formalise the concept of contexts, we also introduce the concept of *context functions*. For some dataset D, a context function is a function $C : 2^D \to 2^D$ that associates with each $D' \subset D$ some set $C(D') \subset D$ (the context of D') such that $D' \cap C(D') = \emptyset^2$. As will be seen, context functions can be used to generalise many anomaly detection concepts.

Consider again the example of a long sequence. Writing this sequence as $S = (s_1, s_2, \ldots, s_n)$, a reasonable context function defined for individual points could be the following:

$$C(s_i) = \{s_{i-w}, s_{i-w+1}, \ldots, s_{i-1}, s_{i+1}, s_{i+2}, \ldots, s_{i+w}\}.$$

When using this context, which is referred to as the *symmetric local context*, the local characteristics of the sequence around s_i are taken into account, while the rest of the sequence is ignored.

Context functions $C(d)$ defined on individual elements $d \in D$ (such as the one above) can be naturally extended to subsets $D' \subseteq D$ of the data by defining $C(D') = \bigcup_{d \in D'} C(d) \setminus D'$. The context functions encountered in this report are all on this form. For this reason, we do not make any distinction between

---

[2]The requirement that $D' \cap C(D') = \emptyset$ is necessary to keep elements from interfering with the computation of their own anomaly scores when performing unsupervised anomaly detection.
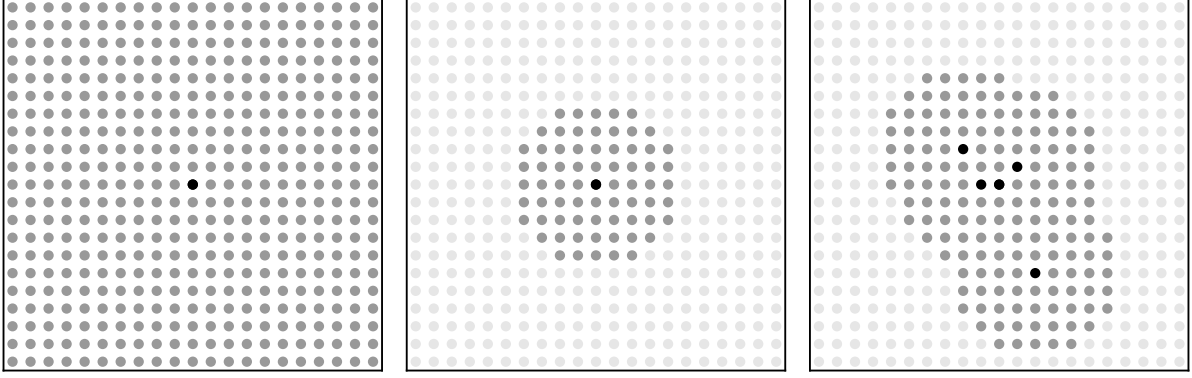
Figure 3.4: Schematic view of a dataset illustrating a few contexts. In each panel, the black dots represent selected items, the dark grey dots represent items in the context of the selected items, and the light grey dots indicate items not in the context of the selected items. The left panel shows the trivial context—all items are part of the context. The middle panel shows a local context of a single item. The right panel shows a local context of a subset of the dataset.

contexts defined for single elements and contexts defined for sets.

Note that contexts can be seen as a generalization of the concept of training data. For instance, the *trivial context*, given by, for $d \in D$; $C(d) = D \setminus d$, corresponds to traditional unsupervised anomaly detection. It is obtained when the scope of any local context grows large enough. Context functions also generalise anomaly detection problems to various tasks that have traditionally been considered separate from anomaly detection. For instance, *novelty detection*, novelty detection [3], which refers to the detection of novel, or previously unseen, items or subsequences in a sequence [3], is really just the use of a one-sided context $C(s_i) = \{s_{i-w}, s_{i-w+1}, \dots, s_{i-1}\}$ in an anomaly detection problem.

Figure 3.4 shows a schematic view of a dataset, along with three contexts. The leftmost panel illustrates the trivial context of a single element, the middle panel illustrates a local context of a single element, and the rightmost panel illustrates the natural extension of this local context to subsets.

3.4 ANOMALY TYPES

An important aspect of any problem is which subsets of the dataset D to consider as potential anomalies; i.e. which subsets of D should constitute the *evaluation set E*. If all subsets are considered, the size $|E|$ of E is $2^{|D|}$. This number is obviously too large to handle effectively, and the evaluation set must somehow be limited.

Fortunately, only a small fraction of all possible subsets is typically of interest in any given application. Precisely which subsets are interesting depends on the structure of $D = \{d_1, d_2, \dots, d_n\}$. If D lacks additional structure (such as an ordering or metric) inducing a concept of locality, then it is reasonable to consider only the singleton sets, i.e. $E = \{\{d_i\}|d_i \in D\}$. When such additional structure exists (and is pertinent to the analysis), it is reasonable to let E consist of subsets in which all elements are 'close' (with regards to this additional structure).

---

[3]It should be noted that the term 'novelty detection' is occasionally used in the literature to refer to semi-supervised anomaly detection.
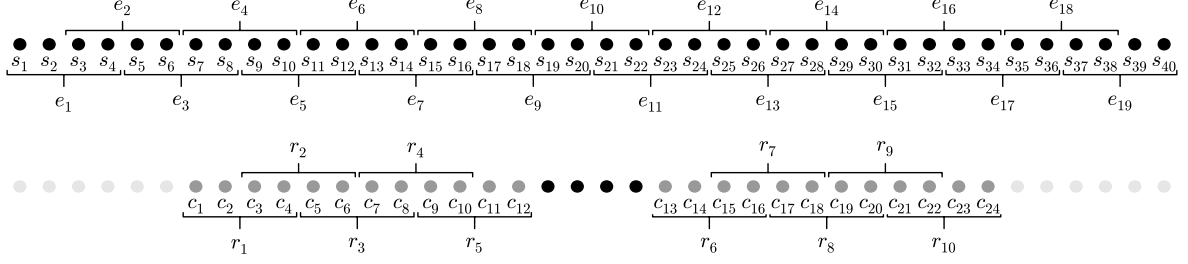
Figure 3.5: Schematic illustration of filters and contexts acting on an evaluation sequence $S = (s_1, s_2, \ldots, s_{40})$. The top panel shows the evaluation set $E = \mathcal{F}(S) = \{e_1, e_2, \ldots, e_{19}\}$ extracted by a sliding window filter with width 4 and step 2. The bottom panel shows the local symmetric context of e_{10} with width $w = 12$: $C(e_{10}) = \{c_1, c_2, \ldots, c_{24}\}$, as well as the training dataset $R_{e_{10}} = \mathcal{F}_R(e_{10}) = \{r_1, r_2, \ldots, r_{10}\}$ extracted by an analogous sliding window training filter.

As an example, consider a sequence $S = (s_1, s_2, \ldots, s_n)$. As mentioned in the previous section, a locality concept is naturally induced by the sequence ordering, and it reasonable to let E consist of contiguous subsequences of S:

$$E = \{(s_{a_1}, s_{a_1+1}, \ldots, s_{b_1}), (s_{a_2}, s_{a_2+1}, \ldots, s_{b_2}), \ldots, (s_{a_k}, s_{a_k+1}, \ldots, s_{b_k})\}.$$

For such E, it is the case that $|E| \in O(|D|^2)$. Furthermore, it is often the case that not all contiguous subsequences must be evaluated—for instance it may suffice to treat only subsequences of some specific length, leading to $|E| \in O(|D|)$. Finally, if the ordering is not relevant to the analysis, then E should be the singleton sets of S, and $|E| = |D|$. Thus, placing reasonable restrictions (based on the structure of the dataset) on E can render the analysis much more manageable.

As a way of formalising the construction of E, we propose that the concepts of training and evaluation *filters* be introduced. Informally, these are functions from some set X to some subset of the power set 2^X of possible subsets of X. An *evaluation filter* is a function $\mathcal{F}_E(D) : D \mapsto E \subset 2^D$ that constructs the evaluation set. One evaluation filter for sequences used later in this report is the *sliding window filter*:

$$\mathcal{F}_E(S) = \{(s_1, s_2, \ldots, s_w), (s_{s+1}, s_{s+2}, \ldots, s_{s+w}), \ldots, (s_{n-w}, s_{n-w+1}, \ldots, c_n)\}^4$$

with width w and step s. This filter is the most reasonable choice for sequences when all items in E must be of the same length (as is typically the case).

It is further useful to, in addition to the evaluation set E, also construct a training set with regards to which to compare the elements in E. If this training set is taken to be fixed, then it can be seen as the training dataset used in a semi-supervised problem. However, we can generalise this concept to other setups by using a context function to associate different training sets with different $e_i \in E$. In this case, with each element $e_i \in E$ should be associated one such set R_{e_i}, consisting of subsets of the context $C(e_i)$. Analogously with evaluation filters, *training filters* can be introduced, which simplify the construction of such R_{e_i}. Since the context is a set of sets, these should have the form $\mathcal{F}(e_i) : C(e_i) \mapsto R_{e_i} \subset 2^D$. As an example of a training filter, consider the sliding window training filter for sequences with length w and

---

[4]It is here assumed that $(s+w)|n$. If this is not the case, the last element extracted might be a bit different.

step s:

$$\mathcal{F}_R(e_i) = \bigcup_{(c_1, c_2, \ldots, c_n) \in C(e_i)} \{(c_1, c_2, \ldots, c_w), (c_{s+1}, c_{s+2}, \ldots, c_{s+w}), \ldots, (c_{n-w}, c_{n-w+1}, \ldots, c_n)\}.$$

A schematic illustration of the operation on a sequence of sliding window filters and a local context is shown in Figure 3.5. Here, an evaluation set consisting of 19 subsequences of a sequence of length 40 is constructed. With each element $e_i \in E$ is associated a training set R_{e_i} (as is seen in the figure, $R_{e_{10}} = 10$). An anomaly detection algorithm could compare each of the e_i to the corresponding R_{e_i} in turn in order to detect contextual collective anomalies (defined below).
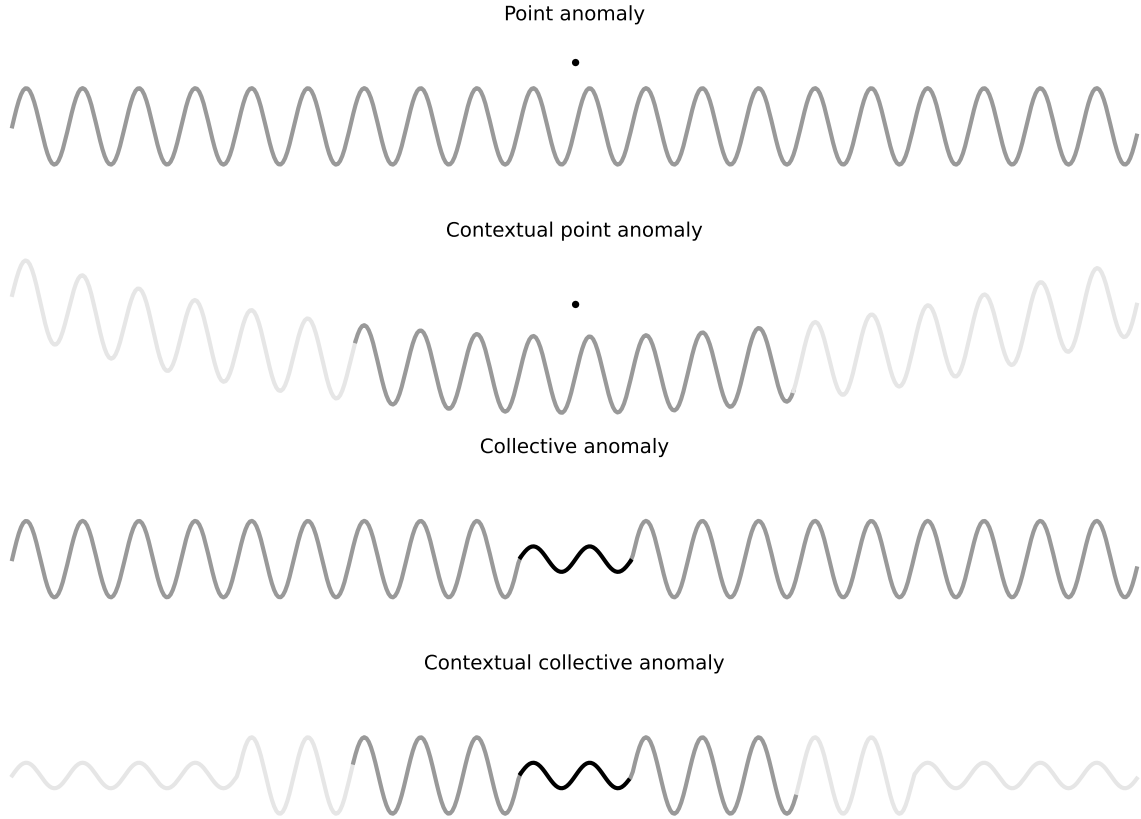

Figure 3.6: Different types of anomalies in a real-valued continuous sequence. In the middle of each series is an aberration—shaded black—corresponding to a specific type of anomaly. Appropriate contexts for these anomalies are shaded dark grey, while items not part of the contexts are shaded light grey. The top panel contains a point anomaly—a point anomalous with regard to all other points in the series. The second panel contains a contextual anomaly—a point anomalous with regard to its context (in this case, the few points preceding and succeeding it), but not necessarily to the entire series. The third panel contains a collective anomaly—a subsequence anomalous with regard to the rest of the time series. The fourth contains a contextual collective anomaly—a subsequence anomalous with regard to its context.

To simplify the discussion of contexts and evaluation/training sets, we will now introduce a few different *anomaly types*, inspired by the concepts of contextual and collective anomalies discussed in [3]. In order of increasing generality, these are *point anomalies*, *contextual point anomalies*, *collective anomalies*, and *contextual collective anomalies*. An illustration of these anomaly types in the context of real-valued

sequences is shown in Figure 3.6.

Point anomalies are arguably the simplest out of these anomaly types. They correspond to single points in the dataset (i.e. E consists of the singleton sets of D) that are considered anomalous with regard to the entire training set (i.e. a trivial context is appropriate). Point anomalies are often referred to as *outliers* and arise in many domains [16]. Their detection is often relatively straightforward. Statistical anomaly measures have been shown to be well suited for handling point anomalies, and are often used. For certain applications, distance-based anomaly measures, such as the local outlier factor [?] are useful.

Of course, point anomalies are not often not sufficient to describe all anomalies when D admits a concept of locality. In this case, *contextual point anomalies* can capture a more general class of anomalies. Contextual anomalies are individual items that are anomalous with regards to their context (as given by some non-trivial context function); i.e. while they might seem normal when compared with all elements in the training data, they are anomalous when compared to the other items in their context. Formally, contextual point anomalies can be defined as, given a dataset D and a context function $C(d)$, the contextual point anomalies of D are the elements $d \in D$ that are point anomalies in $C(d)$. Thus, contextual point anomalies are a generalisation of point anomalies, in the sense that a point anomaly is a contextual point anomaly with regard to the trivial context $C(d) = D \setminus d$.

Sometimes detecting individual anomalous points $d \in D$ might not always suffice, and *collective anomalies* might be required to capture relevant anomalies. Collective anomalies correspond to contiguous sets of non-anomalous points that, when taken as a whole, are anomalous with regards to the entire training set. The task of detecting such anomalies can be formulated using filters. Formally, given a set D and a filter $\mathcal{F}$, the collective anomalies of D are the point anomalies of $\mathcal{F}(D)$. Of course, point anomalies are a special case of collective anomalies, corresponding to the case where $\mathcal{F}(D) = D$.

Finally, *contextual collective anomalies* are the most general class of anomalies, and correspond to contiguous sets of non-anomalous points that are anomalous with regard to a specific context but not to the entire training set. Formally, given a dataset D, a filter $\mathcal{F}$, and a context function C, the contextual collective anomalies of D are the elements of $X \in \mathcal{F}(D)$ that are point anomalies in $C(X)$. As expected, all of the three previous anomaly types can be considered special cases of contextual collective anomalies.

An illustration of the above concepts in real-valued sequences is shown in Figure 3.6. Assuming that unsupervised anomaly detection is used, Detecting point anomalies amounts to disregarding the information provided by the ordering and detecting only 'rare' items. While the task can capture the aberration in the first sequence in Figure 3.6, none of the aberrations in the other sequences would be considered point anomalies.

While the value at the aberrant point at the center of the second sequence occurs elsewhere in that sequence, it is anomalous with regards to its local context, and as such, should be considered a contextual point anomaly and can be captured by problem formulations that use contextual point anomalies.

Since the third time series is continuous, the aberration present at its center can not be a (contextual) point anomaly. It is, however, a collective anomaly, and can be accurately captured by problem formulations that use collective anomalies.

Finally, neither of the first three types of anomalies can capture the aberration in the fourth sequence, as it is both continuous and occurs elsewhere in the sequence. However, with an appropriate choice of (local) context, it can be deemed a contextual collective anomaly, and can be captured by problem formulations that use contextual collective anomalies.

It should be noted that while contextual point anomalies, collective anomalies, and contextual collective anomalies are all generalisations of point anomalies, it is often possible to reduce each of these anomaly types to of point anomalies, as well. As outlined above, each of these anomaly types can be defined using point anomalies. Furthermore, data normalisation be utilized to solve some contextual anomaly detection problems using point anomaly detection (see [17], for instance).

3.5 ANOMALY MEASURES

Arguably the most significant aspect of an anomaly detection problem is the measure used to decide if items are anomalous or not. This factor defines (often in unpredictable ways) what types of features will be considered anomalous, so it is vital to choose it appropriately. Formally, an anomaly measure for some dataset D, given some (potentially trivial) context function C, can be seen as a function from some evaluation set $E \subset 2^D$ to R^+, that associates with each $e_i \in E$ a score according to how anomalous it is with regards to $C(e_i)$.

Many different types of anomaly measures have been used, with varying degrees of justification and success. No exhaustive presentation of these is given here; instead a selection of some of the more common approaches is presented. Two approaches that are especially interesting are statistical and information theoretic anomaly measures, since unlike most other measures, these measure admit convenient theoretical justifications.

Statistical measures usually operate under the assumption that $C(e_i)$ has been generated from some underlying distribution or stochastic process, and associates an anomaly score with e_i based on how likely it is to have been generated by the same distribution or process. Typically, statistical measures work by using some standard inference method, coupled with a few assumptions about the dataset, to estimate some simple distribution underlying the $C(e_i)$. Statistical measures have been applied to a wide range of domains, often with good results. Several books and surveys have been published on the subject of anomaly detection using statistical methods [8] [11] [10] [9].

Statistical measures are usually classified as either parametric or non-parametric. *Parametric statistical measures* assume that distribution underlying $C(e_i)$ is known, but has unknown parameters (for instance, it might be assumed that the data is $N(\mu, \sigma^2)$, where μ and σ are unknown). *Non-parametric statistical measures*, on the other hand, do not assume that the distribution is known and instead try to estimate the distribution itself by assigning weights to a set of basis functions.

While non-parametric approaches are more widely applicable (the distribution of data is usually not known), the extra information provided to parametric methods mean that they converge faster and are more accurate (as long as the given assumptions are correct). Of course, parametric methods are also

less widely applicable, since the underlying distribution is often not known.

For datasets that can be modeled by stochastic processes, *predictive models*, such as Markov chains [**?**], hidden Markov models [**?**], and autoregressive models [**?**] are frequently used as anomaly measures. It should be noted that most predictive models presuppose an ordering and a one-sided context.

Due to the relatively high computational cost of density estimation, statistical methods are mainly used to find point anomalies. Since contextual anomalies require different training sets for each $e_i \in E$, detecting contextual anomalies requires $|E|$ density estimations (unless some clever optimisation is employed), which is typically prohibitively expensive. Since most density estimation methods scale poorly with increasing dimensionality, collective anomalies can also be prohibitively expensive to detect using statistical methods.

A relatively novel and interesting class of anomaly measures is *information theoretic measures*. Mainly used for symbolic datasets, these measures judge similarity by estimating how much information is shared between items or subsets of items (i.e. by computing measures of shared information between elements). Like statistics, information theory can be used as a theoretical basis for anomaly detection.

Several different measures of shared information have been suggested, such as the compressive-based dissimilarity measure (CDM) [27] and (relative) conditional entropy [41]. While information theoretic approaches are mainly useful for symbolic data, they have shown promise for describing anomalies in continuous data when combined with a discretization and numerosity reduction [27].

Anomaly measures inspired by traditional machine learning methods are also common and have been extensively researched in various contexts. For instance, classifier-based methods such as support vector machines are commonly used (TODO: add citation here). While classifiers only produce as many distinct outputs as there are classes, ensembles or weighing schemes can be utilized to produce finer grained output. Like statistical anomaly measures, classifier-based anomaly measures are relatively expensive to train, so they are typically not suitable for non-trivial contexts.

Distance-based anomaly measures are also commonly used. These assign anomaly scores to elements by means of some local point density estimate. Examples include k-nearest neighbors (TODO: cite) and local outlier factor (TODO: cite). Distance-based typically measures scale well with increasing dimensionality, and are appropriate for non-trivial contexts since they are often simple to compute.

3.6 SOLUTION FORMAT

As can be expected, the expected format of solutions to an anomaly detection problem affects both the difficulty of the analysis and the usefulness of its results. As such, the choice of this factor must depend on the target presentation format and performance requirements. In this section, the most common solution formats are presented. To this end, we will consider a set of possible anomalies A. Depending on the filter (see Section 3.4) is used, each element in A can either be a subset or an element of the original (input) dataset.

In many applications, relative anomaly rankings are not essential, and a list of anomalous elements might suffice. In such cases, specifying a *set of anomalous elements* as the output format might be appropriate.

Essentially, this output format turns the problem of anomaly detection into a two-class classification problem, so techniques traditionally used for classification (such as support vector machines) are naturally well suited to it.

Producing a set of anomalous elements is typically appropriate when the goal of the analysis is to highlight potential anomalies to an analyst. It can also be problematic, however, since it requires some sort of threshold to be used to determine whether items are marked as anomalous or not.

A solution format that has received a lot of attention in recent years ([26], [33], [34], [32], [35]) is *discords*, or the set of the k most anomalous elements in A. Since discords are less computationally intensive to produce than other solution formats, they are appropriate for analysis of very large datasets. However, discords might not be appropriate for anomaly detection in large amounts of small datasets, or for monitoring applications.

The most general solution format is to associate *anomaly scores* with the elements of A. Typically, anomaly scores are positive real-valued numbers, and higher anomaly scores signify more anomalous elements. Of course, the fact that anomaly scores outputs have size $|A|$ means that they are comparatively computationally intensive to produce, especially if the size of A is larger than the size of the original dataset. If this is the case, anomaly scores can be produced for the elements of the input dataset by weighing the anomaly scores for A.

While the anomaly scores of non-anomalous elements might be of little interest, producing them can still be useful for visualisation purposes. Furthermore, both of the previously mentioned solution formats can be constructed from anomaly scores, which makes anomaly scores especially useful when the output requirements are not clear.

4 AN APPLICATION TO TIME SERIES

In this chapter, the framework presented in the previous chapter is applied to univariate real-valued time series, in order to derive an instance of the omptimisation problem defined in Section 2.3 appropriate for univariate, real-valued time series. As mentioned, the optimisation problem can be stated as

$$p_{opt}^* = \operatorname*{argmin}_{p \in P^*} \sum_{(i_j, s_j) \in T} \epsilon^*(s_j, O(p, i_j)),$$

where the objects that need to defined are the problem set P^*, the test data T, the error function ϵ^*, and the oracle O.

In Section 4.1, some terminology related to time series is presented, along with a brief summary of previous research on time series anomaly detection. The construction of regular time series from irregular data is also discussed.

In Section 4.3, a problem set P^* is constructed, which generalises many of the approaches presented in the previous section. An oracle O for problems in this set is then presented.

Next, in Section ??, issues related to the test data T, as well as a few possible choices of ϵ^*, are discussed.

Finally, Section ?? details an implementation of the optimisation problem using the objects from the previous sections.

4.1 BACKGROUND

In this section, various previous research on time series is presented, using the framework from the previous chapter, along with some of the terminology and definitions which are used later in the report.

4.1.1 TERMINOLOGY

Since various incompatible definitions of sequences and time series are used in the literature, we begin by defining what we mean when we use these concepts.

From here on, a *sequence* will be taken to mean a progression $S = (s_1, s_2, \dots)$, where $\forall i : s_i \in X$ for some set X. Furthermore, a *time series* is taken to be any sequence $T = ((s_1, t_1), (s_2, t_2), \dots)$, where $\forall i : (s_i, t_i) \in X \times \mathbb{R}^+$ for some set X and $\forall i, j : i > j \to t_i \geq t_j$. In other words, a time series is any sequence in which each item is associated with a point in time. We refer to sequences and time series as symbolic/categorical, discrete, real-valued, vector-valued et cetera based on the characteristics of X.

When a time series is sampled at regular intervals in time (i.e. $\forall i, j : t_{i+1} - t_i = t_{j+1} - t_j$), it is said to be *regular*. We will treat regular time series as sequences, suppressing the t_i and writing $T = (s_1, s_2, \dots)$. Henceforth all time series will be taken to be regular unless explicitly stated[1].

---

[1]As mentioned above, there is some confusion surrounding these terms in the literature. Specifically, what we would

4.1.2 Previous research

Anomaly detection in sequence is an important and active area of research, and plenty of problems related to anomaly detection in sequences have been studied over the years. In this section, a selection of previously researched problems are presented, arranged according to the framework presented in the previous chapter. More detailed surveys of anomaly detection in sequences are available in [3] and [?].

Dataset format

Naturally, categorical, discrete, and real-valued sequences have all been studied extensively. Categorical sequences arise naturally in bioinfomatics [?] and intrusion detection [?] applications. Discrete sequences are typically encountered when monitoring the frequency of events over time. Finally, real-valued sequences are encountered in any application that involves measuring physical phenomena (such as audio, video and other sensor-based applications).

Essentially, the dataset formats in sequences can be classified into two main categories: *Detecting anomalous sequences in a set of sequences*, and *detecting anomalous subsequences in a long sequence.*

Detecting anomalous sequences in a set of sequences is mainly interesting when the dataset consists of large amounts of similar sequences, for instance when analyzing user command records (as in Figure 4.2). Many methods dealing with such applications have been published [23][37][22][38][24][25]. More thorough reviews are found in [4] and [5].

Detecting anomalous subsequences in a long sequence has not been as extensively researched, and is mainly interesting for monitoring and diagnostics applications [?]. TODO: write something more here

Feature extraction is commonly performed to reduce the dimensionality of sequences, and especially of real-valued time series. In this context, the task of feature extraction can be rephrased as follows: Given a sequence $S = (s_1, s_2, \ldots, s_n)$ where $s_i \in \mathbb{R}$, find a set of basis functions $\{\phi_1, \phi_2, \ldots, \phi_m\}$ where $m < n$ that T can be projected onto, such that T can be recovered with little error. Many different methods for obtaining such bases have been proposed, including the discrete Fourier transform [19], discrete wavelet transforms [21] [32], various piecewise linear and piecewise constant functions [28] [31], and singular value decomposition [28]. An overview of different representations is provided in [40].

Arguably the simplest of these bases are piecewise constant functions $(\phi_1, \phi_2, \ldots, \phi_n)$:

$$
\phi_i(t) = \begin{cases} 1 & \text{if } \tau_i < t < \tau_{i+1} \\ 0 & \text{otherwise.} \end{cases}
$$

where $(\tau_1, \tau_2, \ldots \tau_n)$ is a partition of $[t_1, t_n]$.

Different piecewise constant representations have been proposed, corresponding to different partitions. The simplest of these, corresponding to a partition with constant $\tau_{i+1} - \tau_i$ is proposed in [29] and [20]

---

refer to as 'symbolic sequences' and 'regular time series' are often simply referred to as 'sequences' or 'time series'. In such contexts, other types of sequences (and time series) are usually ignored
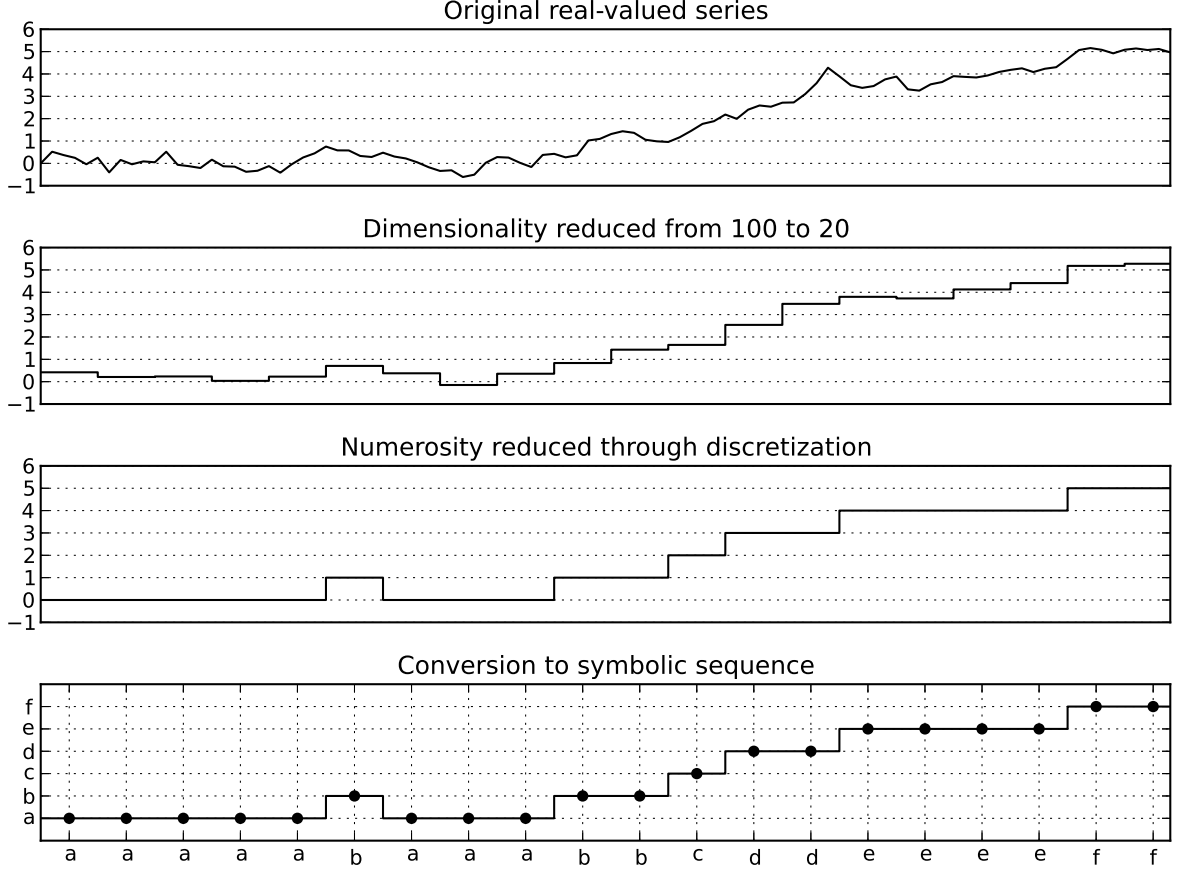
Figure 4.1: Illustration of numerosity and dimensionality reduction in a conversion of a real-valued sequence to a symbolic sequence. The top frame shows a real-valued time series sampled from a random walk. The second frame shows the resulting series after a (piecewise constant) dimensionality reduction has been performed. In the third frame, the series from the second frame has been numerosity-reduced through rounding. The bottom frame shows how a conversion to a symbolic sequence might work; the elements from the third series is mapped to the set $\{a, b, c, d, e, f\}$.

and is usually referred to as *piecewise aggregate approximation (PAA)*. As shown in [30], [28] and [20], PAA rivals the more sophisticated representations listed above.

Numerosity reduction is also commonly utilised in analysis of real-valued sequences. One scheme that combines numerosity and dimensionality reduction in order to give real-valued sequences into a categorical representation is *symbolic aggregate approximation* (SAX) [39]. This representation has been used to apply categorical anomaly measures to real-valued data with good results [?]. A simplified variant of SAX is shown in figure 4.1.

In general, sequences are much easier to deal with than irregular time series. For this reason, irregular time series are commonly transformed to form regular time series, which can be treated as sequences. Formally, such transformations transform a time series $((t_1, x_1), (t_2, x_2), \ldots, (t_n, x_n))$ into some sequence $(s_1, s_2, \ldots, s_m)$.

The simplest such transformation involves simply dropping the t_i to form the sequence $(x_1, x_2, \ldots, x_n)$. This is useful when only the order of items is important, as is often the case when dealing with categorical sequences. An example of such an application is shown in figure 4.2.

$\mathbf{S_1}$	login	passwd	mail	ssh	...	mail	web	logout
$\mathbf{S_2}$	login	passwd	mail	web	...	web	web	logout
$\mathbf{S_3}$	login	passwd	mail	ssh	...	web	web	logout
$\mathbf{S_4}$	login	passwd	web	mail	...	web	mail	logout
$\mathbf{S_5}$	login	passwd	login	passwd	login	passwd	...	logout

Figure 4.2: Multiple symbolic sequences consisting of user commands. In this context, anomaly detection tasks involving finding individual anomalous sequences are interesting. Arguably, problems based on such tasks should capture $\mathbf{S_5}$ as an anomaly due to its divergence from the other sequences. Based on a figure in [4].

Another common class of transformations involves estimating the (weighted) frequency of events. This is useful in many scenarios, especially in applications involving machine-generated data.

Several methods can be used to generate sequences appropriate for this task from time series, such as histograms, sliding averages, etc. These can be generalised as the following transformation:

Given a time series $T = ((s_1, t_1), (s_2, t_2), \dots)$ where $\forall i : s_i \in X$, with associated weights w_i and some envelope function $e(s,t) : X \times \mathbb{R} \to X$ as well as a spacing and offset $\Delta, t_0 \in \mathbb{R}^+$, a sequence $S' = ((s_1', \tau_1), (s_2', \tau_2), \dots)$ is constructed, where $\tau_i = t_0 + \Delta \cdot i$ and $s_i' = \sum_{(s_j, t_j) \in S} s_i w_i e(t_j - \tau_i)$.

The τ_i can then be discarded and the regular time series treated as a sequence. Histograms are recovered if $e(s,t) = 1$ when $|t| < \Delta/2$ and $e(x,t) = 0$ otherwise. Note that this method requires multiplication and addition to be defined for X, and is thus not applicable to most symbolic/categorical data. Also note that S' is really just a sequence of samples of the convolution $f_S * e$ where $f_S = \sum_i \delta(t_i) s_i w_i$.

How this aggregation is performed has a large and often poorly understood impact on the resulting sequence. As an example, when constructing histograms, the bin width and offset have implications for the speed and accuracy of the analysis. A small bin width leads to both small features and noise being more pronounced, while a large bin width might obscure smaller features. Similarly, the offset can greatly affect the appearance of the histograms, especially if the bin width is large. There is no 'optimal' way to select these parameters, and various rules of thumb are typically used [36].

Finally, irregular or noisy data is often resampled to form regular time series. In this case, any of a number of resampling methods from the digital signal processing literature /citeTODO may be employed.

Training data

As in most other machine learning applications, problems involving various degrees of supervision have been researched.

The detection of point anomalies in sequences is a well-researched problem, and has mainly been researched in conjunction with statistical anomaly measures [?]. Of course, detecting point anomalies in sequences is rather uninteresting since it amounts to disregarding the extra information provided by the sequence ordering.

The ordering present in sequences naturally give rise to a few interesting contexts when dealing with anomalous subsequences. A few interesting such contexts are now presented using context functions

(assuming a sequence $S = (s_1, s_2, \ldots, s_n)$).

As mentioned in the previous chapter, contexts can be used to generalise the concept of training data. Semi-supervised anomaly detection corresponds to the *semi-supervised context* $C((s_i, s_{i+1}, \ldots, s_j)) = T$, where T is some training set data. Many semi-supervised sequence and time series anomaly detection problems data have been studied [**?**].

Likewise, traditional unsupervised anomaly detection for subsequences can be formulated using the *trivial context* $C((s_i, s_{i+1}, \ldots, s_j)) = \{(s_1, s_2, \ldots, s_{i-1}), (s_{j+1}, s_{j+2}, \ldots, s_n)\}$. This corresponds to finding either point anomalies or collective anomalies in a sequence, and is studied in [**?**].

Another interesting context is the *novelty context* $C((s_i, s_{i+1}, \ldots, s_j)) = \{(s_1, s_2, \ldots, s_{i-1})\}$. This context captures the task of novelty detection in sequences, which has been researched in [**?**].

Finally, a family of *local contexts* $C_{n,m}((s_i, s_{i+1}, \ldots, s_j)) = \{(s_{i-m}, s_{i-m+1}, \ldots, s_{i-1}), (s_{j+1}, s_{j+2}, \ldots s_{j+n})\}$ may be defined, in order to handle anomalies such as the one in the last sequence of figure 3.6.

It should finally be noted that contextual collective anomalies in sequences do not appear to have been researched.

For the problem of finding anomalous sequences in a set of sequences, there are fewer interesting contexts. Given a set of sequences $S = \{s_a, s_b, \ldots, s_m\} = \{(s_1^a, s_2^a, \ldots, s_{n_a}^a), (s_1^b, s_2^b, \ldots, s_{n_b}^b), \ldots, (s_1^m, s_2^m, \ldots, s_{n_m}^m)\}$, the main contexts of interest are the *semi-supervised context* $C(s_a) = \{t_a, t_b, \ldots, t_o\}$ (corresponding to semi-supervised anomaly detection) and the *trivial context* $C(s_a) = S \setminus s_a$ (corresponding to traditional unsupervised anomaly detection).

Anomaly types

Just as the sequence ordering induces natural contexts, it also naturally induces filters which produce subsequences. Since most anomaly measures defined on sequences require all selected subsequences to have the same length, sliding window approaches are by far the most commonly used. Using the concept of filters, such approaches correspond to sliding window filters, as described in Section 3.4.

For anomaly measures that do not require all sequences to be of the same length, such as some information-theoretic measures, filters that select elements of non-uniform size can be useful. In [**?**], a subdivision approach is taken in which a binary search is performed over the sequence in order to find anomalous substrings of some specific length. No training filter is required, since the anomaly measure used can compare substrings of different size.

Anomaly measures

As is typically the case, the anomaly measure is the most important aspect of any anomaly detection problem for sequences. Formally, any anomaly measure that takes an evaluation vector and a set of reference vectors as inputs and returns a real value is a candidate for $\mathcal{M}$. We now discuss a few such anomaly measures, following the classification in Section 3.5.

As previously mentioned, *statistical measures* are attractive due to the theoretical justification they provide for anomaly detection. However, there are certain factors which render their use problematic for general applications. To begin with, it can generally not be assumed that the data belongs to any particular distribution, and parametric statistical measures are only appropriate in specific circumstances. Nevertheless, parametric statistical methods in sequences are an active area of research [?].

Non-parametric methods are more widely applicable.

Since few non-parametric methods for anomaly detection in sequential data can take into account either collective anomalies or context, and since naive approaches are likely to suffer from convergence issues, [2] suggesting appropriate non-parametric methods for Task 4.3 is difficult. However, in the case of Task 4.1, point anomalies (for which statistical methods have been extensively researched) are more interesting than collective anomalies, and statistical methods are likely to be applicable.

Information theoretical measures are especially interesting for anomaly detection in categorical sequences. However, most such methods are essentially distance-based anomaly measures equipped with information theoretical distance measures. For this reason, we do not cover information theoretical anomaly measures separately from distance-based measures.

Classifier-based measures have also shown promise, especially for the task of finding anomalous sequences in a set of sequences [5]. Generally, any one-class classifier is potentially suitable for the task; see [48] for an exhaustive discussion of this topic. While one-class classifiers produce binary output, appropriate anomaly vectors can still be produced through a suitable weighting scheme.

Predictive model-based measures are also potentially interesting, since are naturally well suited for dealing with the novelty context. However, existing predictive model-based approaches seem to be lacking for the task at hand. In [5], a leading model-based novelty detection method [18] which uses an autoregressive model was shown to perform relatively poorly.

Distance-based measures are especially interesting, due to their flexibility and scalability. A few kNN-based anomaly measures were shown to perform very well for detecting anomalous sequences in sets of sequences in [5].

When dealing with distance-based problems, the choice of distance measure has a profound impact on which anomalies are detected. As with other aspects of anomaly measures, however, drawing conclusions about method efficacy through theory alone is difficult; implementing, evaluating, and comparing several measures is likely to be more useful.

Possible interesting measures include the *Euclidean distance* or the more general *Minkowski distance*; measures focused on time series, such as *dynamic time warping* [50], *autocorrelation measures* [49], or the *Linear Predictive Coding cepstrum* [47]; or measures developed for other types of series (accessible through transforms), such as the *compression-based dissimilarity measure* [27].

Additionally, the choice of distance measure affects how well methods can be optimized. Naive approaches to distance-based problems typically scale prohibitively slowly, and are not suitable for large amounts of

---

[2]For instance, the expectation maximization algorithm for Gaussian mixture models has convergence issues in high dimensions with low sample sizes [?].

data. Optimizations typically involve exploiting properties of the distance measure in order to reduce the number of distance computations (for instance, the commonly used k-d tree nearest neighbor algorithm requires the distance to be a Minkowski metric).

Solution format

Since any solution format can be used with any problem, and since the solution format has little impact on the analysis (except performance-wise), it is not treated in depth here. As noted in the previous chapter, all common output formats can be produced from anomaly scores, so other output formats are interesting mainly as optimisations.

4.1.3 SEQUENCE EXTRACTION

In light of the discussion of uni- and multivariate data above, it seems reasonable to emphasize extracting univariate series from the long multivariate sequences encountered in temporal machine datasets (i.e. Task **??**). Indeed, the provision of tools for manually or automatically extracting interesting time series is likely necessary for the uptake of large-scale anomaly detection (and indeed several other related tasks) in databases such as Splunk. Ideally, methods should be available that automatically examine all potentially pertinent time series and report on those in which anomalies are found. However, the total number of such time series is likely to grow quickly with the number of fields and distinct values of categorical variables in the data datasets, possibly rendering such methods intractable.

In order to estimate the feasibility of these methods, it can be helpful to consider the size of the search space—i.e., the number of possible interesting time series that can be constructed from a typical Splunk index. For example, consider an index consisting of web logs for a single web server. Assume that a log message is generated for every new HTTP request, containing the following fields: time stamp; host name and ident of the user; HTTP request type, URL, status and request/response size/time/latency; and the user agent and referrer. All of these factors are categorical, apart from the HTTP request/response size/time/latency, which ought to be considered real-valued. Let us denote the number of categories encountered in each categorical factor as $n_1, n_2, \ldots, n_k$. For most websites, each of the n_i ought to lie between a few hundred and a few million per day.

We first consider the size of the set of potentially interesting univariate sequences. Interesting univariate sequences could be constructed from at least the frequencies of every value of every categorical factor; the continuous factors over time; or any factor conditioned by any single value of any categorical factor (such as the sequence of URLs or the latency over time for any single user). These combinations alone lead to a number of $N = \sum_i n_i(1 + k + \sum_{j \neq i} n_j)$ potentially interesting time series. If the n_i are sufficiently small, the quadratic nature of this expression is workable. However, if the n_i are large (as is the case in our example, and is likely to be the case in most Splunk applications), a very large number of time series would have to be extracted and analyzed. Of course, detecting anomalies in series conditioned by two or more variables could also be interesting—causing the addition of higher-degree terms to the expression of N. While linearity could be achieved by disregarding conditioned series, this would lead to a significant reduction in analysis power.

Considering this, and the fact that anomaly detection is typically a computationally expensive task (especially when performed on the large datasets), completely automated series extraction is not likely to be feasible except on small indexes. However, if combined with manual pruning of the search space, automated extraction might be feasible. For instance, if in our example we were only interested in detecting anomalous user patterns, the analysis could be limited to a number of $n_u(k + \sum_{i \neq u} n_i)$ sequences (where n_u is the number of distinct users), or perhaps as few as $n_u \cdot k$.

Developing the tools to perform this type of extraction and analysis in Splunk should prove relatively straightforward, and this would be the logical next step once appropriate anomaly detection methods have been implemented.

4.2 IMPORTANT TASKS

Having discussed principal factors and transformations in depth, we are finally in a position to reason about which tasks might be the most relevant to anomaly detection in the target domain.

This chapter begins in Section 4.2.1 with a presentation and discussion of objectives in designing anomaly detection methods for machine data.

In Section 4.2.2, the principal factors and high level tasks presented in Chapter 3 are considered, from the perspective of their applicability to temporal machine data and how well they match the chosen objectives.

Finally, the chapter is concluded in Section 4.2.3 with a summary of the anomaly detection tasks most relevant to the target domain.

4.2.1 OBJECTIVES

A set of objectives for anomaly detection methods dealing with temporal machine data are now presented, which will be used to reason about which tasks and methods are appropriate for such data.

The most fundamental objective is *accuracy*. Methods must detect anomalies as accurately as possible. For obvious reasons, a high rate of false negatives will render the analysis useless, while a high false positive rate will require a high degree of manual analysis of the results—something that is is problematic since such an analysis might be hard to perform and can be sensitive to bias on part of the analyst. This objective can be stated as follows:

Methods should be as accurate as possible. They should have low misclassification rates.

Another important property is *generality*. Many methods are accurate only on few sets of data. For instance, parametric statistical methods typically rely on the assumption that the data is sampled from a specific distribution. When this assumption fails, the results are usually poor. As the target domain is arbitrary sets of machine data, methods should require as few assumptions as possible. This objective is summarized as follows:

Methods should perform well on as many datasets as possible. They should make as few assumptions as possible about the data.

A related objective is that methods should have as *few parameters* as possible. The goal is to provide tools that facilitate the monitoring and analysis of very large sets of data, and methods that require a lot of parameter tweaking in order to work properly will likely hamper this effort, since appropriately configuring such methods requires knowledge of the intricacies of both methods and data. Furthermore, methods with many parameters can easily lead to misinterpreted results [27]. Finally, the amount of 'peripheral' configuration (selection of training data, output format, etc.) should be kept to a minimum. This leads to the following objective:

Methods should require as few parameters as possible. Configuration should be kept to a minimum.

Finally, it is vital that methods be *scalable.* Since the hope is to provide methods that can be useful in monitoring and diagnosis scenarios, methods that can handle large sets of data are required. Specifically, this means that methods should have low complexity in terms of both time and memory, and be able to give accurate results even for large datasets. Furthermore, it is preferable for methods can be distributable and work in a streaming fashion, e.g. not require random disk accesses or multiple passes over the data. This objective can be stated as follows:

Methods should be scalable: they should be able to handle very large amounts of data without long computation times or disk issues.

4.2.2   High-level tasks

Which high level tasks are relevant for temporal machine data is now discussed. The principal factors are treated sequentially, according to the order they were encountered in Chapter 3.

This Section is also based in large part on the discussions in the previous Section and those of Chapter **??**.

Data format

As per the discussion in Sections **??** and 4.1.3, it is appropriate to focus on extracting sequences from indexes for use with Tasks **??** or **??**.

What remains is to settle on a specific presentation of these sequences. While all three aggregation tasks presented in Section **??** are potentially useful for temporal machine data, their relative utility to the stated objectives can still be judged. We will now discuss each of the three tasks in turn, and estimate their utility.

First, consider Task **??** (sequential anomaly detection). While interesting in certain contexts (such as system call or web session traces), it is not likely to be able to capture a majority of interesting sequences in temporal machine datasets, since it is not appropriate for non-categorical sequences. While there are certainly interesting categorical sequences that can be analyzed in many temporal machine datasets, it can be expected that these constitute a low share of the total number of interesting sequences in Splunk

applications. For this reason, it ought to be argued that this task, while interesting in some cases, should have lower priority than the other two aggregation tasks.

Task **??** (frequential anomaly detection) is more interesting. Since it enables anomaly detection to be applied to the frequency of any recurring event or value, it should be applicable to a wider range of datasets. However, it is limited in the sense that it can not accurately capture continuous quantities or other types of quantities that vary over time. This limits its applicability to temporal machine datasets, since a large share of the interesting quantities in such data (at least in Splunk applications) vary over time.

In such cases, Task **??** is more appropriate, as it allows for correct estimates of evolving quantities. As pointed out in Section **??**, the aggregation transformations involved in the Tasks **??** and **??** are very similar, and implementing them both should not be difficult. However, they are likely to require different anomaly measures; to limit the scope of the project, it was decided that the focus would be placed on the latter.

The next issue to settle is whether uni- or multivariate sequences should be the focus. As discussed in Section **??** multivariate sequences are more difficult to analyze than univariate sequences, and can often be broken down into univariate sequences without significant losses in analysis power. Therefore, it is reasonable to (at least initially) focus solely on univariate sequences.

Finally, it must be decided which of the Tasks **??** (detecting anomalous subsequences) and **??** (detecting anomalous sequences in a set of sequences) to focus on. While the latter task is interesting and occurs naturally in many contexts, e.g. system call traces (figure 4.2), it is more reasonable to focus on the former since temporal machine datasets typically feature long and largely dissimilar sequences.

Based on the above, the following two tasks ought to be promoted as important, with the latter being preferential as a first step:

Task 4.1 *Detect anomalous subsequences in real-valued univariate sequences containing the frequency of some event over time.*

Task 4.2 *Detect anomalous subsequences in real-valued univariate sequences sampled from a continuous process.*

Reference data

One of the four objectives stated in Section 4.2.1 is that methods should be as generally applicable and require as little configuration as possible. Obviously, the reference data tasks presented in Section **??** meet these demands to varying degrees. Since unsupervised anomaly detection (task **??**) does not require the user to provide reference data, it is superior to supervised and semi-supervised anomaly detection in this regard.

However, it is not clear whether unsupervised methods can scale adequately; there seems to be a trade-off between the amount of required supervision and the scalability of algorithms. Most naive methods

scale as $O(f(R \cdot N))$, where $f(n) \in \Omega(n)$, R is the number of reference items, and N is the number of items to be evaluated. When performing semi-supervised anomaly detection, it is usually the case that $R < N$. In unsupervised anomaly detection, however, $N = R$, and so methods scale as $O(f(N^2))$. This is problematic, and some form of pruning is required to allow large-scale analysis. One form of pruning that should perform relatively well is the use of some form of constant-size context (e.g., the local context). Nevertheless, it was decided that unsupervised methods should be the focus of this project.

Anomaly types

The choice of both anomaly types and anomaly measure together define which anomalies will be captured by problem formulations.

Reviewing Section 3.4, and especially Figure 3.6, can help when reasoning about anomaly types. *Point anomalies* are obviously not appropriate, as they can not capture most anomalies in continuous sequences (to see this, consider that none of the aberrations in the second, third, or fourth panels of Figure 3.6 are point anomalies).

Contextual point anomalies are more relevant, especially in the context of Task 4.1. Here, each individual data point in the resulting data series encodes the frequency of some event at some specific time. Assuming that these frequencies among points sharing context are independent, contextual anomaly methods should be sufficient to find temporary anomalous frequencies.

However, contextual point anomalies do not occur in continuous sequences, such as those encountered in Task 4.2 or in the third and fourth panels of Figure 3.6.

Collective anomalies, on the other hand, are very interesting in continuous time series, since they can cover all anomalies that arise in such series, as long as the anomalies are global (i.e., they do not constitute normal behaviour elsewhere in the series).

However, non-global anomalies do occur, especially in long time series such as those encountered in monitoring applications. With this in mind, it is reasonable to conclude that being able to take *contextual collective anomalies* into account is highly desirable. Further, based on the fact that all the previous anomaly types are special cases of contextual collective anomalies, it can be concluded that finding this type of anomaly should be sufficient[3].

As with any task involving contextual anomalies, a *context function* must be specified before this method can be useful. Natural choices of sequence context functions include the *symmetric local context*, which considers k items before and after the subsequence in question, and the *asymmetric local context*, which considers k items before and j items after the subsequence.

Since which context functions are appropriate depends heavily on the dataset, context functions should be evaluated empirically.

---

[3]Note that while this is certainly true, it does not mean that methods optimized for finding contextual anomalies (for instance) would not be better at that specific task than methods for finding contextual collective anomalies. A thorough evaluation of all methods—for both tasks—would have to be performed to answer this question

Anomaly measures

As stressed in Section 3.5, the anomaly measure is probably the most important part of any anomaly detection problem. However, due to the unpredictability of how they affect results, there is not much that can said about which anomaly measures are the most appropriate for the target domain. For this reason, we will leave this factor unspecified, and instead focus on evaluating as many different choices of it as possible.

Output format

In one sense, the choice of output format is cosmetic, as it depends entirely on the context in which the results should be presented.

However, as noted in Section 3.6, most other output format tasks can be reduced to Task **??** (anomaly scores). In order to reduce the risk of focusing on an inappropriate output format, this task was chosen as a focus.

High-level tasks related to anomaly detection

Most of these tasks are not immediately useful for anomaly detection in Splunk. For instance, signal recovery is a complex task that is unlikely to be required for most machine data and for which using specialized software is more appropriate.

The task of *novelty detection* (task **??**), however, has potential in both monitoring and diagnosis applications. In the former case, anomalies that occur periodically will be diagnosed every time they occur, possibly leading to an unnecessarily large number of discovered anomalies. In the latter application, anomalies that occur periodically will be completely ignored (i.e. treated as normal if they occur many times), which can be problematic. Novelty detection methods avoid these problems.

Note that (as stated in Section **??**) novelty detection is really just contextual (or contextual collective) anomaly detection with a specific context function. For this reason, we do not treat novelty detection on sequences as a task separate from anomaly detection, but as a specific choice of context, which we refer to as the *novelty context*. Finally, the problems in monitoring and diagnosis applications described above can also be mitigated by other context functions.

4.2.3 SUGGESTED TASKS

The tasks selected in the previous Section into complete tasks in which all principal factors are fixed.

Based on the discussion in Section 4.2.2, the following task ought to be given the highest priority:

Task 4.3 *Detect contextual collective anomalies in, and assign anomaly scores to, the elements of a single univariate real-valued sequence sampled from a continuous process.*

The remainder of this report will focus on this task. In the next chapter, the component framework, which simplifies the specification of remaining factors for this task, is proposed. Both ad-eval and the evaluation performed as part of this project focus on this task as well.

Of course, there are other, similar tasks that could also be very important to machine data anomaly detection. A set of tasks which naturally complement Task 4.3 in monitoring and diagnosing of sets of machine data are now presented.:

To begin with, as discussed in Section 4.2.2, frequential anomaly detection is very similar to continuous anomaly detection and is likely to be relevant to Splunk applications. We present this as the following task:

Task 4.4 *Detect contextual collective anomalies in, and assign anomaly scores to the elements of, a single univariate real-valued sequence containing the frequency of some event over time.*

Furthermore, symbolic time series are potentially interesting in several applications, suggesting the following task (which is purposely vague, since it requires a different choice of principal factors):

Task 4.5 *Detect anomalous subsequences in long symbolic sequences.*

Of course, applying Task **??** (detecting anomalous sequences in a set of sequences) to the above tasks can also be interesting. The resulting tasks can be briefly stated as follows:

Task 4.6 *Detect anomalous sequences in a set of univariate real-valued time series sampled from continuous processes.*

Task 4.7 *Detect anomalies sequences in a set of univariate real-valued time series containing the frequency of some event over time.*

Task 4.8 *Detect anomalous sequences in a set of similar symbolic sequences.*

Since all of the tasks presented in this section are complementary, and since they share many similarities, they should ideally be implemented and evaluated in tandem. However, due to time and space constraints, it was not possible to study most of them in sufficient depth this project (although the component framework and `ad-eval` can both support these tasks).

4.3 COMPONENT FRAMEWORK

Before Task 4.3 can be evaluated, an understanding of the set of problems that can be derived from it must be acquired. In this chapter, the *component framework* is proposed, which facilitates a structured approach to this problem.

First, Section 4.3.1 introduces the component framework. In Section 4.3.2, modifying the component framework to handle various related tasks is discussed. Finally, the component framework is used in Section 4.3.3 to discuss and derive interesting problems from Task 4.3.

4.3.1   The framework

While Task 4.3 has not been previously studied, several papers exist covering special cases of it (typically corresponding to certain combinations of context functions, filters and anomaly measures). A key realization in the development of the component framework was that almost all these methods could be specified using a set of independent components, used by a common algorithm to find anomalies. If the components could be formalized and abstracted to pertain to Task 4.3, then an understanding of the set of problems derived from that task could be obtained through the study of the individual components.

The first step towards this goal was to consider which factors are needed to specify a problem from Task 4.3. We note that, at minimum, the following factors must be specified:

The context function. Since we are dealing with contextual collective anomalies, the context must be defined.

The filters. Filters for extracting both *evaluation subsequences* (from the input sequence) and *reference subsequences* (from the context) must be specified.

The anomaly measure. Arguably the most significant factor still unspecified.

Transformations. Which transformations, if any, are to be applied to the data before analysis.

The aggregation method. A method for aggregating individual anomaly scores into an anomaly vector must be provided.

Next, these five *components* were defined mathematically, in order to facilitate their standardization and use in algorithms. Specifically, the tuple of components $(\mathcal{F}_E, \mathcal{C}, \mathcal{F}_R, \mathcal{M}, \mathcal{A})$ was defined, where (letting $T \in X^n$ be the sequence under consideration and X be some arbitrary set, S_T be the set of subsequences of T, and $2^{\mathbb{Z}_n}$ be the set of corresponding indices in T):

The evaluation filter $\mathcal{F}_E$ selects which subsequences of T are to be evaluated by the anomaly measure. We let $\mathcal{F}_E(T) = \{(T_1, I_1), (T_2, I_2), \ldots, (T_k, I_k)\} \subseteq \mathcal{S}_T \times 2^{\mathbb{Z}_n}$ be a set of tuples of subsequences of T and their corresponding indices.

The context function $\mathcal{C}$ is responsible for selecting the appropriate context for subsequences. Formally, for $(T_i, I_i) \in \mathcal{F}_e(T)$, $\mathcal{C}(I_i) \subset \mathcal{S}_T$ gives one or more subsequences constituting the context.

The reference filter $\mathcal{F}_E$ extracts subsequences from the context to produce a set of reference items.

The anomaly measure $\mathcal{M}$ assigns an anomaly score to each of the selected items with regard to their reference set: $\mathcal{M}(T_i, \mathcal{F}_r \mathcal{C}(I_i)) \in \mathbb{R}$.

The aggregation function $\mathcal{A}$ aggregates the anomaly scores to produce an anomaly vector; i.e. if $S = \{(a_i, I_i)\}_{i \in \mathbb{Z}_k}$ is a vector of anomaly scores and indices for individual elements, then $\mathcal{A}(S) \in \mathbb{R}^n$.

Finally, an algorithm could be formulated that solves an arbitrary problem specified using this component framework, as follows:

Require: A sequence T and a tuple $(\mathcal{F}_E, \mathcal{C}, \mathcal{F}_R, \mathcal{M}, \mathcal{A})$.

$\quad S \leftarrow \emptyset$ $\hfill \triangleright$ initialize anomaly score container

$\quad$ **for** $(T_i, I_i) \in \mathcal{F}_E(T)$ **do** $\hfill \triangleright$ iterate over subsequences (and indexes) selected by filter

$\quad\quad C \leftarrow \mathcal{C}(I_i)$ $\hfill \triangleright$ acquire context

$\quad\quad R \leftarrow \mathcal{F}_R(C)$ $\hfill \triangleright$ extract reference set from context

$\quad\quad S \leftarrow S \cup (\mathcal{M}(T_i, R), I_i)$ $\hfill \triangleright$ save anomaly score with indexes of subsequence

$\quad$ **end for**

$\quad$ **return** $\mathcal{A}(S)$ $\hfill \triangleright$ aggregate scores to form anomaly vector

Restating existing problems within this framework provides several advantages. Firstly, it facilitates comparison of methods, making it easier to see their similarities and differences. Secondly, it makes it easy to overview the entire set of problems and propose novel problems based on combinations of existing components. Finally, the components can be easily interchanged, simplifying method evaluations.

The fact that a single algorithm can be used for all problems suggests the possibility of automated, accurate handling of diverse datasets. Typically, the extent to which problems can capture anomalies varies drastically between series from different sources. If this ability could be related to underlying characteristics of the series, a preliminary analysis could guide the choice of components. Since the proposed framework simplifies automated analysis and allows for problems to be applied interchangeably on the same data, it could help both in finding these underlying characteristics and in applying them to anomaly detection.

As an example of such a characteristic, in [5], various anomaly measures are compared on a task derived From **??** (finding anomalous real-valued sequences in a set of similar sequences) and a correlation is noted for some problems between the share of anomalies captured and the periodicity of the sequences. Based on this, investigating links between problem accuracy and characteristics of series in the frequency domain may prove fruitful.

4.3.2 RELATED TASKS

It is important to note that the component framework is not limited to Task 4.3, but can be modified to handle several related tasks as well. For instance:

Semi-supervised tasks are retrieved by specifying a reference sequence R in addition to the evaluation sequence T, and letting $\mathcal{C}(T_i) \equiv R$.

Collective anomaly detection tasks are retrieved by specifying $\mathcal{C}(T_i) = T \setminus T_i$.

Point anomaly and contextual anomaly detection tasks can be retrieved by specifying $\mathcal{F}_{E,R}(T) = \{(t_i)\}_{t_i \in T}$ and selecting an appropriate context function.

Sequential and frequential tasks need no modifications; they merely require different types of pre-processing.

Alternative anomaly scores can be retrieved through trivial modifications of the aggregation method $\mathcal{A}$.

Finding anomalous sequences among a set of sequences mainly requires simplification: $\mathcal{F}_E$, $\mathcal{C}$, $\mathcal{F}_R$ and $\mathcal{A}$ can all be ignored for such tasks. The only relevant part is the anomaly measure.

Based on this, the component framework could help elucidate how different sequential anomaly detection tasks are related. Furthermore, since all derived problems can be solved using a single algorithm when formulated using the component framework, the implementation and evaluation of a large class of anomaly detection tasks could be performed relatively easily. Indeed, it would be simple to modify `ad-eval` (which implements the component framework) to handle any of the above tasks.

4.3.3 COMPONENTS

We now discuss possible choices of the components $\mathcal{F}_E$, $\mathcal{C}$, $\mathcal{F}_R$, $\mathcal{M}$ and $\mathcal{A}$ and the implications of these choices.

While Task 4.3 has not been previously studied, components based on existing methods for other tasks can be used with it. In the following discussion, several such component choices, as well as a few new components, are mentioned.

The evaluation filter

The evaluation filter takes a sequence and extracts subsequences from it to be used in the analysis. In the literature, *sliding window* approaches are almost always taken. In these methods, evenly spaced subsequences of equal length are extracted from the series:

$$\mathcal{F}_E((t_1, t_2, \ldots, t_n)) = \{(t_1, t_2, \ldots, t_w), (t_{1+s}, t_{2+s}, \ldots, t_{w+s}), \ldots, (t_{\alpha-s}, t_{\alpha-s+1}, \ldots, t_\alpha)\},$$

where w is the window length, s is the displacement, and $\alpha = \lfloor \frac{n}{s} \rfloor \times w$. A constant value of w is almost always used, since most anomaly measures can not deal with items of varying dimensionality. The value of s is typically taken to be $1 \leq s \leq w$. The total number of elements is $\lfloor \frac{n-w}{s} \rfloor + 1$, and thus decreases with larger s.

Other approaches might be appropriate for anomaly measures which support variable-size elements. In [27], a binary search is performed on T to find discords. Note that such an approach requires a slight modification of the algorithm given above.

The context function

As mentioned before, several different context functions are plausible for use in sequential anomaly detection. We have previously mentioned the context (assuming that $T = (t_1, t_2, \ldots, t_n)$ is the series under consideration):

$$\mathcal{C}((t_i, t_{i+1}, \ldots, t_j)) = \{(t_{i-m}, t_{i-m+1}, \ldots t_{i-1}), (t_{j+1}, t_{j+2}, \ldots, t_{j+n})\},$$

which we refer to as the *symmetric local context* if $m = n$ and the *asymmetric local context* otherwise; the *novelty context*:

$$\mathcal{C}((t_i, t_{i+1}, \ldots, t_j)) = \{(t_1, t_2, \ldots, t_{i-1})\};$$

and the *trivial context* (for detecting collective or point anomalies):

$$\mathcal{C}((t_i, t_{i+1}, \ldots, t_j)) = \{(t_1, t_2, \ldots, t_{i-1}), (t_{j+1}, t_{j+2}, \ldots, t_n)\}.$$

We also introduce the *semi-supervised context*, in which $\mathcal{C}(T_i) \equiv X$ for some reference sequence X.

Note that the number of elements in the context will typically have a significant impact on analysis time. While the sizes of the local and semi-supervised contexts are both constant as functions of the sequence length, the novelty and trivial contexts grow linearly, which is likely to cause problems in large datasets.

The reference filter

The reference filter works like the evaluation filter, but extracts subsequences from the context instead of from the evaluation series. As with the evaluation filter, it is usually best to use a sliding window as the reference filter. If the anomaly measure does not require items of equal length, the reference filter can be ignored, i.e. $\mathcal{F}_R(T) = T$.

The anomaly measure

As is typically the case, the anomaly measure is the central part of any problem derived from Task 4.3. Formally, any anomaly measure that takes an evaluation vector and a set of reference vectors as inputs and returns a real value is a candidate for $\mathcal{M}$. We now discuss a few such anomaly measures, following the classification in Section 3.5.

As previously mentioned, *statistical measures* are attractive due to the theoretical justification they provide for anomaly detection. However, there are certain factors which render their use problematic for the task at hand. Firstly, as discussed in Section 4.2.1 we can not assume that the data belongs to any particular distribution, so parametric problem formulations are not appropriate.

Since few non-parametric methods for anomaly detection in sequential data take into account either collective anomalies or context, and since naive approaches are likely to suffer from convergence issues, [4]

---

[4] For instance, the expectation maximization algorithm for Gaussian mixture models has convergence issues in high dimensions with low sample sizes.

suggesting appropriate non-parametric methods for Task 4.3 is difficult. However, in the case of Task 4.1, point anomalies (for which statistical methods have been extensively researched) are more interesting than collective anomalies, and statistical methods are likely to be applicable.

Information theoretical measures are potentially interesting for the chosen task. However, most such methods are essentially distance-based anomaly measures equipped with information theoretical distance measures. For this reason, we do not cover information theoretical anomaly measures separately from distance-based measures.

Instead, we emphasize *distance-based measures*. Since they encompass a wide variety of methods, which have been shown to be successful for related problems [5], we dedicate Section 4.3.3 to discussing them.

Classifier-based measures have also shown promise for related problems [5]. Generally, any *one*-class classifier is potentially suitable for the task; see [48] for an exhaustive discussion on this topic. While one-class classifiers produce binary output, appropriate anomaly vectors can still be produced through a suitable choice of $\mathcal{A}$.

Predictive model-based measures are also potentially interesting. They are naturally well suited for novelty detection, since they necessarily implement a novelty context. However, existing predictive model-based approaches seem to be lacking for the task at hand. In [5], a leading model-based novelty detection method [18] which uses an autoregressive model was shown to perform poorly on a related task. While models are important as a means of reducing the computational complexity of methods, they appear appropriate to consider only as approximations to other anomaly measures.

Distance measures   When dealing with distance-based problems, the choice of distance measure has a profound impact on which anomalies are detected. As with other aspects of anomaly measures, however, drawing conclusions about method efficacy through theory alone is difficult; implementing, evaluating, and comparing several measures will likely prove more instructive.

Possible interesting measures include the *Euclidean distance* or the more general *Minkowski distance*; measures focused on time series, such as *dynamic time warping* [50], *autocorrelation measures* [49], or the *Linear Predictive Coding cepstrum* [47]; or measures developed for other types of series (accessible through transforms), such as the *compression-based dissimilarity measure* [27].

Additionally, the choice of distance measure affects how well methods can be optimized. Naive approaches to distance-based problems typically scale prohibitively slowly, and are not suitable for large amounts of data. Optimizations typically involve exploiting properties of the distance measure in order to reduce the number of distance computations (for instance, the commonly used k-d trees algorithm requires the distance to be a Minkowski metric).

Transformations   Anomaly measures can be combined with one or more transformations of the data extracted by the filters to speed up the analysis or to facilitate the detection of certain types of anomalies.

One commonly used data transformation is the Z-normalization transform, which modifies a sequence to exhibit zero empirical mean and unit variance. It has been argued that comparing time series is

meaningless unless the Z-normalization transform is used [30]. However, this is doubtful, as the transform masks anomalies consisting of subsequences similar to their surrounding data but with different variance or mean.

Transformations that transform the data into some alternative domain can also be useful. For example, transformations based on the *discrete Fourier transform* (DFT) and *discrete wavelet transform* (DWT) [32] have shown promise for sequential anomaly detection.

Furthermore, transformations for real-valued data which produce symbolic sequences are very important, since they enable the application of symbolic approaches (studied in bioinfomatics, for instance) to real-valued sequences. While several such transformations exist, *symbolic aggregate approximation* (SAX) [39] is by far the most commonly used.

The aggregation method

Once the extracted subsequences have been assigned anomaly scores, they must be aggregated before an anomaly vector can be formed. While little has been written on this subject, it is not difficult to describe reasonable aggregation methods. It seems clear that all aggregation functions, or *aggregators*, ought to have the form

$$\mathcal{A}(\{(a_1, I_1), (a_2, I_2), \ldots, (a_k, I_k)\}) = (f(\{a_i : 1 \in I_i\}), f(\{a_i : 2 \in I_i\}), \ldots, f(\{a_i : n \in I_i\})),$$

where I_i are intervals, a_i are assigned anomaly scores, and f is some aggregator-specific function acting on a vector $\mathbb{R}^n$ and producing an aggregate score in $\mathbb{R}$. The *maximum*, *minimum*, *median* and *mean* of the values in S all constitute reasonable choices of $f(S)$. We will refer to aggregators using any of these four functions as maximum, minimum, median, and mean aggregators, respectively.

4.4 Implementation

As stated in the abstract, the development of a software framework for the evaluation of anomaly detection methods, called `ad-eval` and available at `http://github.com/aeriksson/ad-eval`, was a significant part of the project. In this chapter, the design, development process, and features of `ad-eval` are discussed.

Essentially, `ad-eval` consists of three separate parts: a library implementing the component framework, a comprehensive set of utilities for evaluating the performance of methods and problem, and an executable leveraging the library. The entire project is written in Python.

In this chapter, `ad-eval` is described in detail. The three parts are described in Sections 4.4.1, 4.4.2, and 4.4.3, and some of the design choices made in the development of `ad-eval` are discussed in Section 4.4.4.

4.4.1 IMPLEMENTED COMPONENTS

The anomaly detection part of `ad-eval` (given the Python package name `anomaly_detection`) is a faithful implementation of the component framework, including the algorithm proposed in Section 4.3.1. In order to preserve the modular nature of this framework, the individual components are implemented as autonomous modules, described below.

As there are no real alternatives found in the literature, the sliding window filter is the only implemented evaluation filter. Since the optimal window width w and step length s depend on the application, both of these parameters were left to be user-specified.

Since new new context functions can be implemented relatively easily, and since they have a relatively major impact on the analysis, all previously discussed contexts (specifically, the asymmetric and symmetric local contexts, the novelty context, the trivial context, and the semi-supervised context) were implemented.

As reference filters, a sliding window filter and the identity filter $\mathcal{F}_E(X) = X$ were implemented, the latter because it is a better fit for dimension-independent distance measures.

Due to the limited scope of the project, the only implemented anomaly measures (henceforth referred to as *evaluators*) were a variant of k-Nearest Neighbors (kNN), in which the distance to the k'th nearest element is considered, and a one-class support vector machine (SVM). For the kNN evaluator, the Euclidean distance as well as the compression-based dissimilarity measure [27] and dynamic time warping [50] distances were made available. Furthermore, the symbolic aggregate approximation (SAX) and discrete Fourier transform (DFT) transformations were added as an optional pre-anomaly measure transformations. All parameters of the evaluators, distances, and transformations were left for the user to specify.

Finally, the mean, median, maximum, and minimum aggregators were implemented.

4.4.2 EVALUATION UTILITIES

The set of possible interesting tests that could be run on problems derived from Task 4.3 is considerable. A large number of component combinations can be used; most components have many possible parameter values, and it is important to assess how these affect the results; and a large set of methods with various optimizations and approximations can be proposed. For all of these choices, it is important that accuracy and performance are properly evaluated.

However, the performance of methods is highly dependent on the characteristics of the datasets to which they are applied. As mentioned in Section 4.5.1, there is no hope to exhaustively cover the space of possible evaluation sets. Instead, sample data from the target application domain must be obtained before any tests are performed. Furthermore, obtaining adequate labeled test data is often difficult, and artificial anomaly generation must be considered as an option.

With this in mind, it was decided that an evaluation framework should be added to `ad-eval` to help

facilitate the implementation, standardization, and duplication of accuracy and performance evaluations. Due to the variety of interesting tests highlighted above, an approach focused on the provision of tools that assist in scripting custom tests was deemed preferable to one focused on the construction of a single configurable testing program. To this end, utilities were developed for:

- Saving and loading time series to/from file, with or without reference anomaly vectors.

- Pseudorandomly selecting and manipularing subsequences of series (e.g. for adding anomalies).

- Facilitating the testing of large numbers of parameter values.

- Generating various types of artificial anomalies and superpositioning them onto sequences.

- Calculating the anomaly vector distance measures ϵ_E, ϵ_{ES} and ϵ_{FS} discussed in Section 4.5.2.

- Facilitating the automated comparison of several problems and methods on individual datasets.

- Automating the collection of performance metrics.

- Reporting results.

- Generating various types of custom plots from results.

With these tools in place, it is simple to write scripts that, for instance, generate large amounts of similar series containing random anomalies and evaluate the performance of several problems on this data in various ways.

The tools were included in `ad-eval` as a separate Python module (called `eval_utils` and located in the `evaluation` directory of the repository). This module was used to perform all tests in the evaluation phase of this project.

4.4.3 EXECUTABLE

To enable the stand-alone use of the anomaly_detection package, an executable was added to the `ad-eval` repository (called `anomaly_detector` and located in the `bin` directory of the repository). This executable is used through a command-line interface and a `key:value` style configuration file, can perform supervised or semi-supervised anomaly detection on sequences from files or standard input, and can use any of the components implemented in `anomaly_detection`.

To avoid having to modify this program every time a component in `anomaly_detection` was changed, the executable was made unaware of all internal details of that package. Consequently, a command-line interface capable of configuring the components could not be implemented; instead, a configuration file parser is used to read and pass the component configuration to `anomaly_detection`.

4.4.4 DESIGN

The development of `ad-eval` began at the start of the project. Initially, development efforts focused on the implementation of a few optimized methods found in the literature, to produce a Splunk app. However, as the project progressed and the issues discussed in Section 2.1 (that most methods were targeted at subtly different tasks, and that due to lacking evaluations, assessing which methods are really the 'best' is not possible), this approach was recognized as fruitless, and abandoned.

Development then shifted towards an implementation of the component framework, with the goals of maximizing the ease of implementing and evaluating large amounts of components.

Consequently, modularity was a major focus throughout the development process, achieved through various means. As mentioned previously, the individual components were separated into independednt modules. Additionally, the evaluation utilities and the executable were decoupled from the component framework implementation, interfacing with it through only two method calls. Finally, the decision was made to distribute the configuration of the component framework implementation, letting each component handle its own configuration and making the rest of the package configuration-agnostic.

It was a natural choice to write the entire implementation in Python, for several reasons. First, Python is well suited for small, flexible projects such as `ad-eval`, thanks to its simplicity and flexibility. Furthermore, a number of great libraries for data mining and machine learning exist for Python, which were used to accelerate the development. Finally, if `ad-eval` becomes adopted for real-world use, Python's good C integration could be leveraged to write optimized code.

Finally, the evaluation utilities were designed with ease of use and flexibility in mind. For instance, while classes facilitating test data generation, evaluation, and reporting are provided, their use is optional. As a result, evaluation scripts could be short and simple—the scripts used in the next chapter are all 30 to 70 lines long—without sacrificing flexibility.

4.5 EVALUATION

As mentioned throughout this report, thorough empirical evaluations on real-world datasets are pivotal to elucidating the performance characteristics of problems, methods and tasks. In this chapter, two factors important when performing evaluations are discussed: evaluation data and error measures.

4.5.1 EVALUATION DATA

For obvious reasons, the choice of evaluation data largely determines the outcome of the evaluation. Since the spaces of possible evaluation items for most anomaly detection methods are enormous (often $\mathbb{R}^n$ for very large n), there is little hope of performing exhaustive analyses. Furthermore, while only relatively small subsets of these spaces are typically interesting for any given application, discovering and delimiting these subsets is usually impossible, due to a limited understanding of the processes underlying the data.

For this reason, it is critical that a set of evaluation items is used, containing as representative a selection of relevant phenomena exhibited by data items in the application domain as possible. Since such sets can not be obtained through purely synthetic means (arguably, the existence of a model that could accurately create such sets would presuppose a level of understanding of the application domain that would render anomaly detection unnecessary), real-world data must be used.

However, simply obtaining real-world data is *not* sufficient; if the anomalies in the data are not labeled, no baseline can be established, and the evaluation is useless. In the context of Task 4.3, this means that, along with each sequence $S_i = (s_1, s_2, \ldots, s_n)$ in the evaluation set, a reference anomaly vector $A_i = (a_1, a_2, \ldots, a_n)$ must be provided, where each a_i indicates how anomalous s_i is.

When appropriate data is unavailable, there are two possible options. The first is to generate both series and anomalies, in order to create completely artificial sequences. The second is to take unlabeled real-world datasets and superposition artificially generated anomalies onto them.

Due to how greatly the dataset affects the outcome of anomaly detection analyses, the first approach is not recommended, other than as a means of gaining a qualitative understanding of tasks, problems or methods. Even if the artificial data appears highly similar to data from the target domain, it is very unlikely that the "best" methods for such data and real-world data will be the same. The second approach, while also problematic, can at least suggest how well methods are able to recognize normal data from the target domain from anomalous data.

In the initial stages of this project, it was hoped that sufficient data would be obtained by searching the internet and existing papers on the subject, or by examining sequences of machine data with a domain expert. However, enough suitable series could not be obtained through these means, and it became clear that completely artificial data would have to be used in the evaluation. Because such data can not adequately capture the intricacies of the target domain, a qualitative evaluation of implemented methods was performed instead. Still, since a major goal of the project was to facilitate evaluations using real-world data, utilities for performing appropriate evaluations were added to `ad-eval`.

4.5.2 ERROR MEASURES

For any evaluation, some way of assessing the accuracy of the produced anomaly vectors is required. Formally, given a sequence $S = (s_1, s_2, \ldots, s_n)$, a reference anomaly vector $R = (r_1, r_2, \ldots, r_n) \in \mathbb{R}^n$ and a method that takes a sequence as input and produces an anomaly vector $A = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$, a function $\epsilon(A, R) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ is required that assigns a similarity score, or *error measure* to A based on how well it approximates R.

Considering the difficulties involved in obtaining any kind of labeled sequences, and the fact that ranking elements based on relative degrees of anomalousness is even more difficult, it is reasonable to assume that all reference anomaly vectors are binary: $R = (r_1, r_2, \ldots, r_n) \in \{0, 1\}^n$. Since any method derived from Task 4.3 produces real-valued output, this means that error measures should work by comparing one binary and one real-valued vector; i.e. $\epsilon(A, R) : \mathbb{R}^n \times \{0, 1\}^n \to \mathbb{R}^+$.

Since constant factors do not affect how accurate an anomaly vector appears, any $\epsilon(A, A_R)$ should be invariant under uniform translations and scalings of A. This means that regular real-valued distance measures (such as the Euclidean distance) are not applicable. This can be avoided by normalizing A—by scaling and translating it such that all its elements lie in $[0, 1]$—before computing the distance. For instance, the *normalized Euclidean error* ϵ_E, given by

$$\epsilon_E = \sqrt{\sum_{i=1}^{n} \left(\frac{a_i - a_{\min}}{a_{\max} - a_{\min}} - r_i \right)^2}$$

ought to constitute a reasonable choice of error measure. However, since ϵ_E is equally sensitive to the behaviour of A on normal elements as on anomalous elements, how A treats normal elements (which should not matter unless they have high anomaly scores) can significantly harm the accuracy of this measure, as we will see in 5.4.

This pitfall can be avoided by converting A to a binary string A_B as well, and using a binary distance between A_B and R as the error measure. Since this is equivalent to selecting a set S_A of indexes of elements from A and comparing it to the set S_R of indexes of nonzero elements of R, such error measures can be seen as evaluating the performance of A_B on Task **??**, which is reasonable considering the binary nature of R.

What remains is to decide how A_B should be constructed from A and what binary distance measure to use. Since the individual elements of S_R have equal weight, and since the locations of discrepancies between A_B and R are not important, the *Hamming distance* δ_H constitutes a suitable choice of distance measure. Furthermore, A_B should be constructed such that $\forall i, j \leq n : a_i < a_j \wedge i \in S_A \Rightarrow j \in S_A$, which is equivalent to setting a threshold $a_{\min} \leq \tau \leq a_{\max}$ and letting A_B be given by

$$A_B = (a_{b,1}, a_{b,2}, \ldots, a_{b,n}), \quad \text{where } a_{b,i} = \begin{cases} 0 & \text{if } a_i < \tau \\ 1 & \text{if } a_i \geq \tau \end{cases}.$$

This gives rise to a number of possible A_B equal to the number of distinct values of A. We here propose two error measures corresponding to different choices of τ, both based on δ_H. The *equal support error* ϵ_{ES} corresponds to setting τ such that $|S_A| = |S_R|$, while the *full support error* ϵ_{FS} corresponds to choosing the largest τ such that $\forall i : i \in S_R \Rightarrow i \in S_A$. While other choices (such as choosing the τ that minimizes δ_H) might also be appropriate, only these two were selected in order to limit the scope of the discussion.

Since the three error measures proposed above are expected to produce different results, and since they have not been (to the author's knowledge) used in the context of series anomaly detection, they should be evaluated empirically before being used. Such an evaluation is performed in Section 5.4.

5 RESULTS

Due to the lack of appropriate evaluation data mentioned in Chapter 4.5, and in order to limit the scope of this report, a comprehensive evaluation of the implemented problems could not be performed. Instead, a preliminary, qualitative evaluation was performed, with the goal of gaining some insight into the relative performance of problem formulations derived from Task 4.3, and demonstrating how `ad-eval` can be used to simplify and standardize the process of evaluating anomaly detection problems and methods.

In this chapter, the results of this evaluation are presented. Since distance-based evaluators can be combined with a wider set of other components than classifier-based or other types of evaluators, and to keep this report from becoming overly long, the focus was placed entirely on the kNN evaluator implemented in `ad-eval`. The performance of this evaluator was investigated through the identification of all remaining unspecified parameters of the components $(\mathcal{F}_E, \mathcal{C}, \mathcal{F}_R, \mathcal{M}, \mathcal{A})$, and individual investigations of how these parameters affect the analysis.

Of course, all graphs and data in this chapter were obtained using `ad-eval`. Since one of the main goals of the evaluation was to demonstrate how `ad-eval` can be used to perform standardized evaluations, all scripts used to obtain the figures and results in this chapter are available in the `ad-eval` source code repository. Modifying these scripts to use other datasets or to evaluate other components (such as the SVM evaluator implemented in `ad-eval`) is trivial.

5.1 EVALUATION APPROACH

Since the number of possible combinations of components and parameter values is enormous (even with the limited number of components implemented in `ad-eval`), it was not feasible to include a comprehensive evaluation of all these combinations in this report. Instead, the components were studied individually, using a preset configuration and varying individual parameter values. Note that this approach only allows for the assessment of local characteristics of the parameter space around the specific configuration.

As a second limitation, it was decided that the analysis would be focused on a single artificial sequence, owing to the lack of proper datasets to evaluate. While a large artificial dataset could have been generated for the evaluation, this would have been counterproductive for several reasons. Specifically, as discussed in Chapter 4.5, generating dataset sufficiently diverse to accurately reflect the characteristics of real-world datasets in target domain is practically impossible, and creating even a rough artificial approximation would require substantial effort. Any such dataset that could fit within the scope of this project would thus be severely limited, and would not be appropriate for assessing real-world performance. To highlight these deficiencies, and to simplify the exposition, it was decided that the focus would be placed on examining the performance characteristics of problems on a single artificial sequence. Rather than performing evaluations with different types of artificial sequences, all scripts used in the evaluation were added to the `ad-eval` source repository, and were constructed such that performing similar evaluations on new sequences would be trivial.
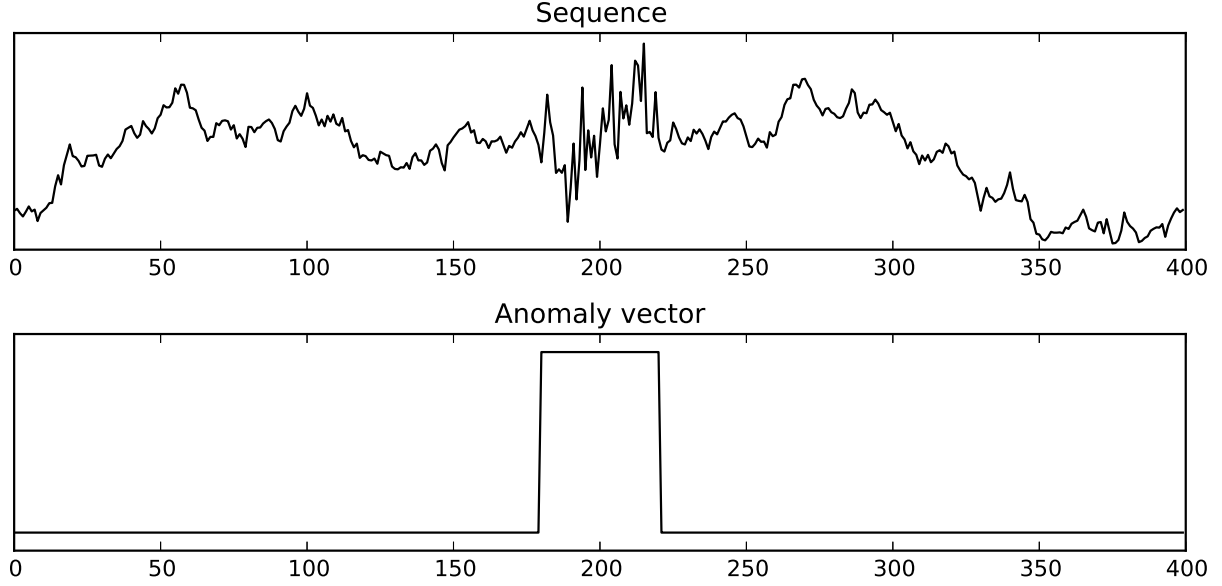
Figure 5.1: The standard sequence with corresponding reference anomaly vector.

Before choosing a sequence, there were a few things to consider. Since evaluating a large number of problem should be possible on modest hardware in short time spans, and since anomaly detection methods are typically rather slow, the sequence should be short. However, the sequence should still contain both normal data, homogeneous enough to establish a baseline of 'normal' behaviour, as well as an anomaly that deviates appropriately from this baseline. "Appropriate", in this case, means that it should be relatively easy to detect with most reasonable parameter choices, but difficult enough to be detectable regardless of the parameter choices. A sequence of length 400 was settled on, generated by a random walk, with added noise in the range 180 to 220. This sequence, referred to in the remainder of this chapter as the *standard sequence* or s^*, is shown in Figure 5.1 along with the corresponding reference anomaly vector a^*.

5.2 PARAMETER SPACE

When considering the set of problems derived from some task, it can be helpful to regard the set of variables necessary to fully specify a problem from it as the *parameter space* of that task. Individual variables correspond to dimensions in the space, while problems correspond to points. General tasks are associated with large, high-dimensional parameter spaces while more specific tasks have smaller, more manageable spaces. Searching for an optimal problem derived from some task for some dataset, then, means searching for a point in the parameter space of the task at which the corresponding problem can most efficiently find the anomalies in the dataset. Equivalently, this can be thought of as an attempt to minimize some error function over the parameter space.

In this case, the parameter space corresponds to the free parameters of the components $C = (\mathcal{F}_E, \mathcal{C}, \mathcal{F}_R, \mathcal{M}, \mathcal{A})$. We will denote these by Θ. Assuming that some function $A(\Theta, s)$ is provided that solves the problem corresponding to Θ for a sequence s (or equivalently uses the anomaly detector corresponding to Θ to evaluate s) and outputs an anomaly vector, the task of finding an optimal problem formulation for some

dataset S can be seen as the task of finding $\text{argmin}_\Theta E(\Theta, S)$ for some error function E that evaluates the error of $A(\Theta, s)$ for each $s \in S$. Assuming that this error function is a linear combination of the errors according to some error measure δ of the elements in S, this can be written as

$$E_{S,\delta}(\Theta) = \sum_{(s_i, a_i) \in (S,A)} \delta(A(\Theta, s_i), a_i).$$

Given a dataset and a set of possible components, this task is relatively straight-forward. While filters, contexts and aggregators with infinite parameter spaces are possible, the components discussed in this report have relatively small, finite parameter spaces. This means that an exhaustive search would be possible in theory.

However, most choices of C will typically have a large number of free parameters, resulting in a high-dimensional parameter space. This, in combination with the fact that $E_{S,\delta}(\Theta)$ typically takes a long time to evaluate, even for small S, renders such exhaustive searches prohibitively computationally expensive in practice.

5.3 STANDARD CONFIGURATION

Due to the limited computational resources available when performing the evaluation, and in order to simplify the presentation, the parameters had to be studied in isolation. This amounts to studying the behaviour of $E_{S,\delta}$ near a single point in the parameter space by considering its behaviour along orthogonal lines meeting at this point. This is not sufficient to draw any strong conclusions about global minima of the error over the parameter space, unless the parameters influence E independently, which is certainly not the case for Task 4.3. However, this is not problematic; the aim of the evaluation was to gain insight into how the individual parameters affect the outcome, not to find global minima.

The fixed point in the parameter space will be referred to as the *standard configuration*, and denoted by $\Theta^* = (\theta_1^*, \theta_2^*, \ldots, \theta_n^*)$. The choice of Θ^* is now described.

As mentioned previously, $\mathcal{E}$ was fixed as a kNN evaluator. This evaluator has the free parameters k (which of the nearest neighbors to use when calculating the anomaly score—set to 1 by default), δ (the distance function—the standard Euclidean distance δ_E, by default) and an optional transformation $t : \mathbb{R}^n \to \mathbb{R}^m$ to apply to the items in the reference set before the evaluation. By default, no transformation is used.

Because the distance measures implemented in `ad-eval` (except the compression-based dissimilarity measure) require extracted elements to be of the same length, sliding window filters were the only reasonable choice. It was thus decided that sliding window filters would be used for both the evaluation and reference filters, with the free parameters w (window width—10 by default) and s (step length—1 by default).

Since the evaluation was performed on artificial data, there was no reason to use either the novelty or asymmetric local context. Furthermore, since the trivial context is just a special case of the local symmetric context (with the context width m set to a maximum), the trivial context was selected to be the default, with the single free parameter m, set to 400.

Finally, since the four aggregators implemented in `ad-eval` are all parameter-free, it was decided that the aggregator itself would be treated as a parameter. Since it was assumed that the mean aggregator would give the most accurate results, it was selected as the default.

In summary, then, the parameter space for this evaluation is parametrized by $(k, \delta, t, w, s, m, \mathcal{A})$. To simplify the following discussion, we will take Θ^* to be the point where all parameters take on the values specified above, and let $\Theta^*_{\alpha_1, \alpha_2, \ldots, \alpha_n}$ be the set of points where all parameters except $\alpha_1, \alpha_2, \ldots, \alpha_n$ take on these values (e.g., Θ^*_k corresponds to the set of points where all parameters other than k agree with the standard configuration). We will further denote the set of corresponding anomaly vectors on s^* as $A_{\alpha_1, \alpha_2, \ldots, \alpha_n}$. The A_α for $\alpha = k, \delta, t, w, s, m$, and $\mathcal{A}$ are examined in Section 5.5.

5.4   ERROR MEASURES

(a) Normalized Euclidean error   (b) Equal support error   (c) Full support error
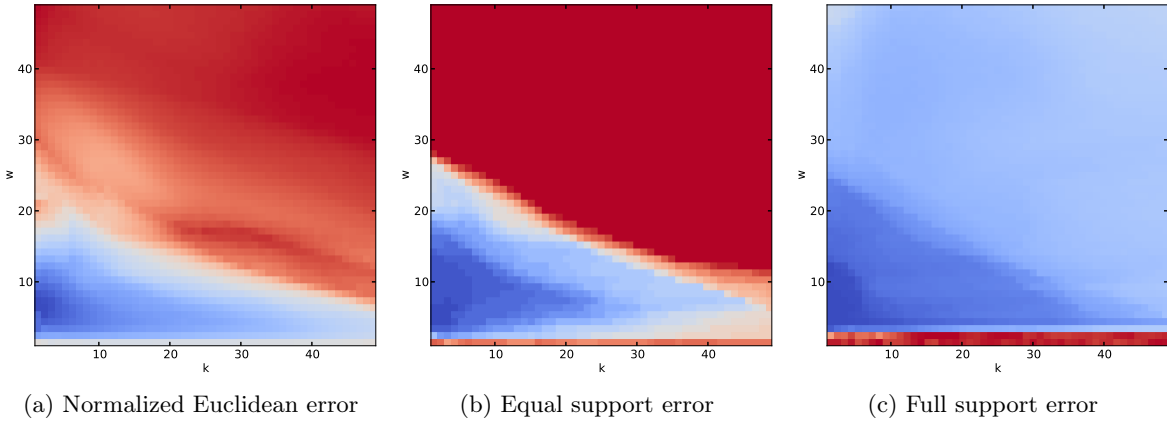
Figure 5.2: Heat maps showing the normalized values of $\epsilon(A(\Theta^*_{k,w}, s^*), a^*)$ for the three errors and $k, w = 1, 2, \ldots, 50$. Blue and red correspond to low and high error values, respectively.

In Section 4.5.2, three error measures for anomaly vectors were introduced: the normalized Euclidean error ϵ_E, the equal support error ϵ_{ES}, and the full support error ϵ_{FS}. Since these methods have not been previously studied with regards to sequential anomaly detection, how well they capture the intuitive notions of accuracy must be assessed before they can be used to evaluate problem formulations.

Such an assessment was performed by computing and graphing the errors $\epsilon(A(\Theta^*_{k,w}, s^*), a^*)$ for $\epsilon = \epsilon_E, \epsilon_{ES}$ and ϵ_{FS} and $(k, w) \in \{1, 2, \ldots, 50\}^2$. Heat maps of these values are shown in Figure 5.2. A few of the $A_{k,w}$ given the lowest values by each of the error measures were also graphed (figure 5.3).

As shown in the heat maps, the three error measures give similar results, attaining minima and maxima in the same regions. Since ϵ_{ES} and ϵ_{FS} operate on binary strings and thus have discrete domains, they often assign identical errors to nearby points. This is the cause of the relatively jagged appearance in the plots of these errors compared to the smoother appearance of the ϵ_E plot.

Figure 5.3 shows the anomaly vectors with the nth lowest errors for the three distance measures. All three measures give similar anomaly vectors for $n = 1$ and $n = 10$, with the normalized Euclidean and full support errors giving the same anomaly vector in both cases. For $n = 50$ and $n = 100$, however,
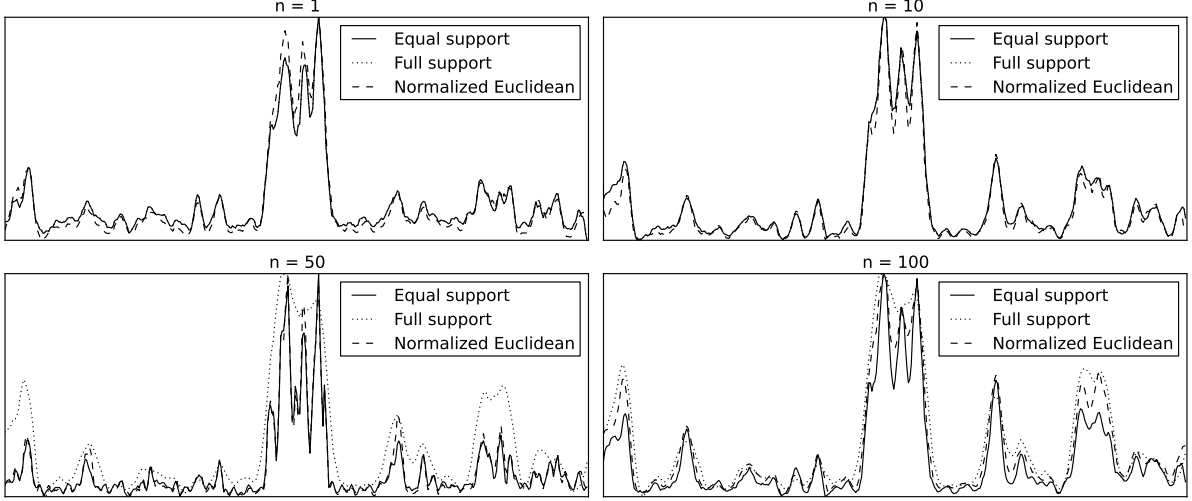
Figure 5.3: The nth best $A_{k,w}$ according to the three error measures for $n = 1, 10, 50$ and 100.

the full support error seems to prioritize smooth anomaly vectors, while the other two anomaly measures prioritize anomaly vectors with few false positives.

One interesting aspect evidenced in the heat map plot is that while ϵ_{ES} and ϵ_{FS} are both very large for $A_{k,w}$ with small w, this tendency is not shared by the Euclidean error. As seen in Section 5.5.4, anomalies significantly larger than w will not be detected by kNN methods, which means that assigning a large value to these anomaly vectors is reasonable. Since the normalized Euclidean error (unlike the other two distances) gives equal weight to every component, it will assign relatively low values to anomaly vectors that only partially capture anomalies as long as most of their elements are close to zero. Indeed, this is the case for the $A_{k,w}$ with small w since these anomaly vectors are close to constant everywhere except for a few spikes.

As an illustration of the potential problems this could cause, see Figure 5.4, which shows one reference anomaly vector and two candidate anomaly vectors for a long sequence. Here, the first anomaly vector, while noisy, accurately captures the anomaly while the second not only misses the anomaly, but also introduces two false positives. While ϵ_{FS} and ϵ_{ES} are significantly smaller for the first candidate than for the second, the reverse is true for ϵ_E. This problem is amplified as the sequence length grows. These results indicate that ϵ_E should be used with caution, and that since other two error measures are preferable since they were defined specifically to avoid problems such as this.
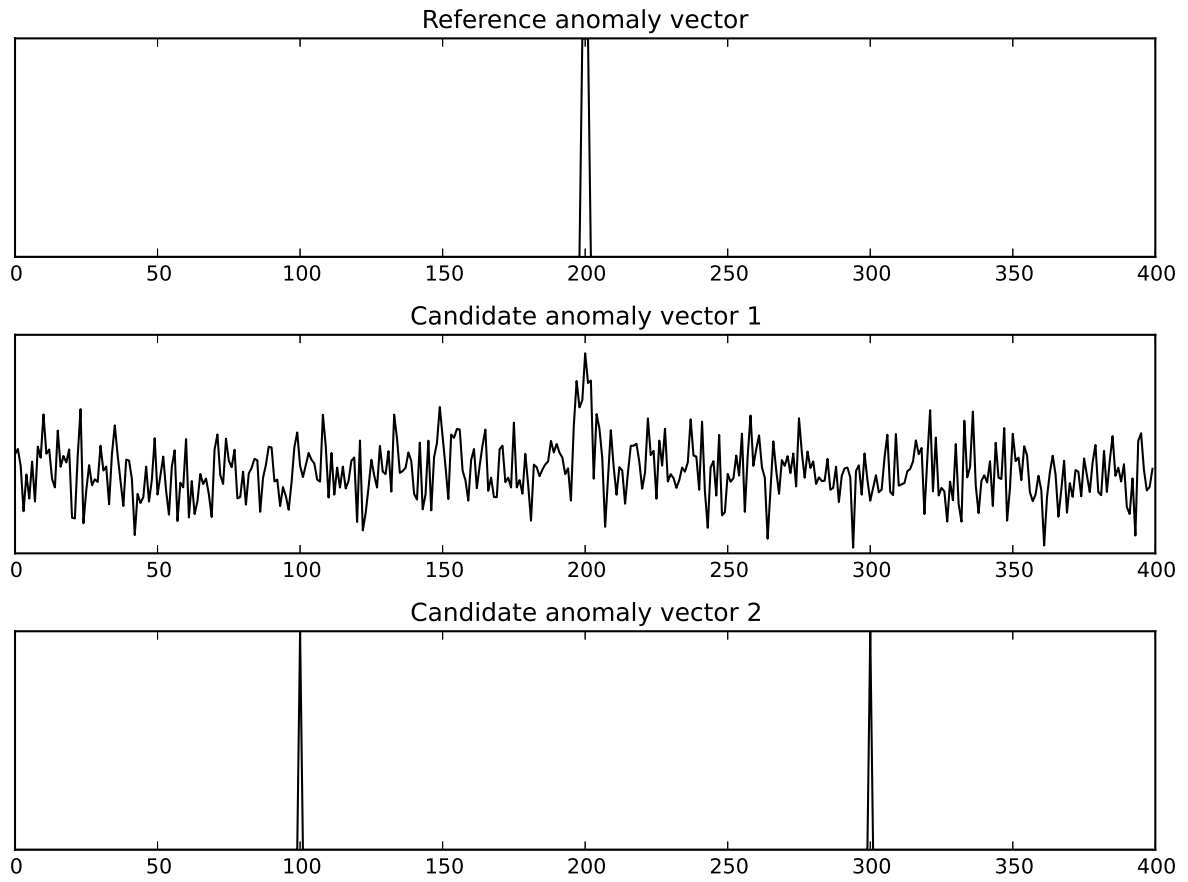
Figure 5.4: A reference anomaly vector for a long sequence and two corresponding candidate anomaly vectors. The first candidate vector, while noisy, correctly marks the anomaly. The second candidate does not mark the anomaly and marks two false anomalies. ϵ_E, ϵ_{ES} and ϵ_{EF} for the two sequences are 8.3 and 2.2; 0.010 and 0.99; and 0.0050 and 0.99, respectively.

5.5   Parameter values

Each of the free parameters $(k, \delta, t, w, s, m, \mathcal{A})$ described in Section 5.3 are now covered in detail, by means of studying their anomaly vectors on the standard sequence A_α as well as the corresponding errors and evaluation times as α varies.

As mentioned previously, since the analysis in this Section is based on studying the Θ_α^* separately on the sequence s^*, it is not sufficient for any conclusions to be drawn either about global minima of $E_{\delta,S}(\Theta)$ or about how well the results might extend to other sequences. Instead, the analysis in this Section should be considered a first step towards establishing a broader understanding of how the $E_{\delta,S}(\Theta)$ vary over the parameter spaces of distance-based problems, and as an introduction to `ad-eval`, including some useful ways to explore performance characteristics.

5.5.1   The k value
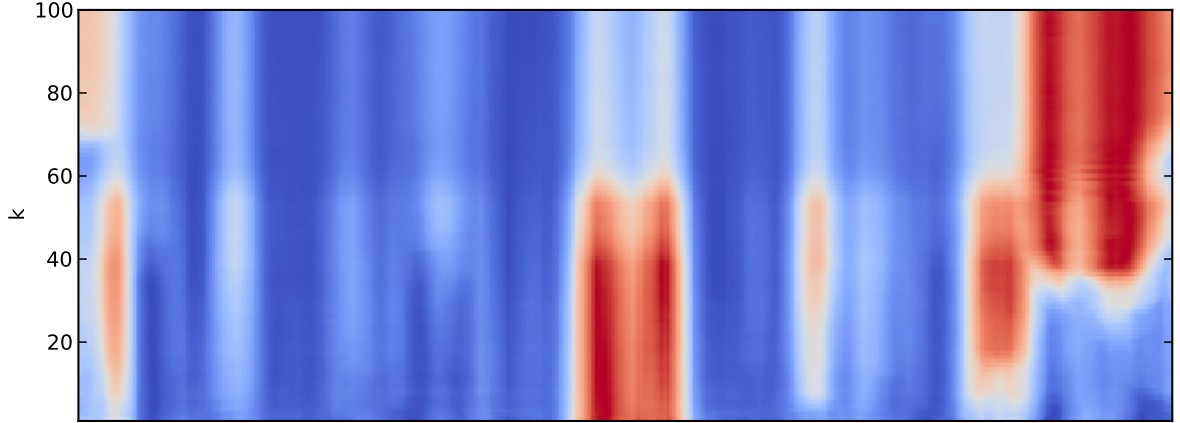
Figure 5.5: Heat map showing A_k for $k = 1, 2, \ldots, 100$. Red and blue indicate high and low anomaly scores, respectively.

It is important to study how the anomaly vectors vary with k; first, because the choice of k is likely to have a large impact on the appearance of the anomaly vector, regardless of the dataset; and second, because the kNN evaluator only operates on a single k value at a time, it is in a sense the simplest distance-based evaluator, and thus an ideal tool for better understanding how the choice of k impacts the analysis. This understanding is crucial in effectively designing other types of distance-based evaluators.
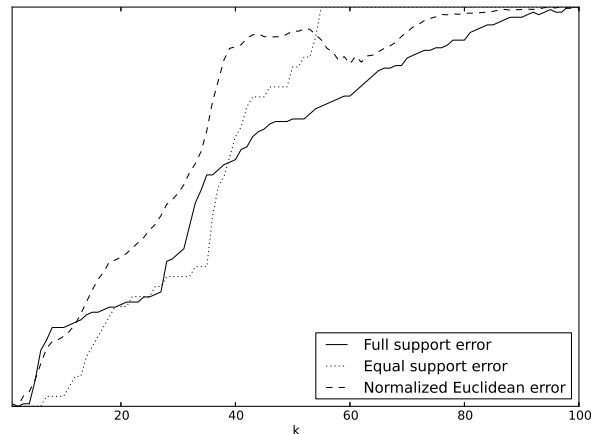
Figure 5.6: Errors as a function of k for the standard sequence.

In order to understand how the k value affects the resulting anomaly vectors, the anomaly vectors A_k for $k = 1, 2, \ldots, 100$ were calculated. Figure 5.5 shows the resulting anomaly vectors, displayed as a heat map in which all anomaly vectors have been individually normalized to lie in the unit interval. Corresponding values of the three error measures are

shown in Figure 5.6. Note that this plot shows only relative errors, as the three error graphs have been individually normalized to the unit interval.
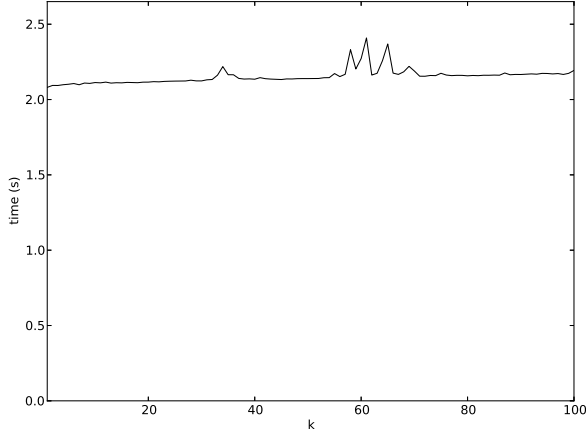
Figure 5.7: Evaluation times when varying k on the standard sequence.

Finally, Figure 5.7 shows the computation times for calculating the anomaly vectors A_k. Since the implemented kNN method operates by brute force, the entire reference set must be evaluated regardless of k, so the constant evaluation time exhibited in this figure is expected. For any distance measure that is a metric, such as the Euclidean distance, more efficient methods exist.

The smoothness with which the A_k vary with k indicate that using several nearby k in distance-based evaluators is not likely to significantly improve accuracy. Furthermore, at least in this case, $k = 1$ minimizes all three error measures, and there is no indication that considering additional k might help. While higher k do lead to other regions being marked as anomalous, these regions do not correspond to relevant features. If this holds in general, there is no need to consider k higher than 1, and using linear combinations of several k is not likely to lead to any significant increase in accuracy. However, a much more thorough evaluation is required before any conclusions can be drawn.

5.5.2   THE DISTANCE FUNCTION

Figure 5.8: Heat maps showing $A_{k,\delta}$ for the Euclidean and DTW distances.

For obvious reasons, the choice of the distance function δ can have a great impact on the anomaly vectors when using distance-based methods. The distance measures implemented in `ad-eval` are the Euclidean distance, the dynamic time warp (DTW) distance, and the compression-based dissimilarity measure (CDM). To investigate the relative performance of these, the anomaly vectors $A_{k,\delta}$ were examined. Note that since A_δ consists of only one value per distance measure, calculating only A_δ would have yielded

insufficient data.

The `ad-eval` implementation of the CDM performed poorly. To begin with, it ran significantly slower than the other methods, rendering any comprehensive analysis impossible. Furthermore, it produced poor anomaly vectors. There are a few possible explanations for this. First, the z-normalization step of the SAX transformation (in which each extracted subsequence is given zero empirical mean and unit variance) leads to poor results on random data regardless of the distance measure. Secondly, the window width of 10 used in the standard configuration means that the extracted sequences are short and can not be efficiently compressed, leading to a roughly constant distance value. While the CDM will likely perform better and with other parameters, it was decided that the CDM would not be investigated further due to its slowness.

Instead, the focus was placed on comparing the Euclidean and DTW distances. Heat maps of the resulting anomaly vectors are shown in Figure 5.8 and a plot of the corresponding errors is shown in Figure 5.9. As is seen in the heat map, there is generally little difference between the outcomes of the two distance measures; the DTW distance gives slightly 'cleaner' (i.e., with non-anomalous regions closer to 0) anomaly vectors for very low values of k, while the Euclidean distance assigns a slightly lower score to the false anomalies encoun-

Figure 5.9: Errors for $A_{k,\delta}$.

tered at high values of k. While there are some differences in the obtained errors—the DTW distance gives a better normalized Euclidean error, while the Euclidean distance generally gives better values of the other two errors—the evaluation is not sufficient to draw any conclusions about the relative merits of the two measures.

However, the fact that the DTW distance does not perform worse than the Euclidean distance in this evaluation is interesting. Since the DTW was designed to recognize long, shifted but relatively similar continuous sequences, it might be expected to perform poorly on other types of data, such as the short, random data used in this evaluation. The fact that this is not the case is a positive indication.
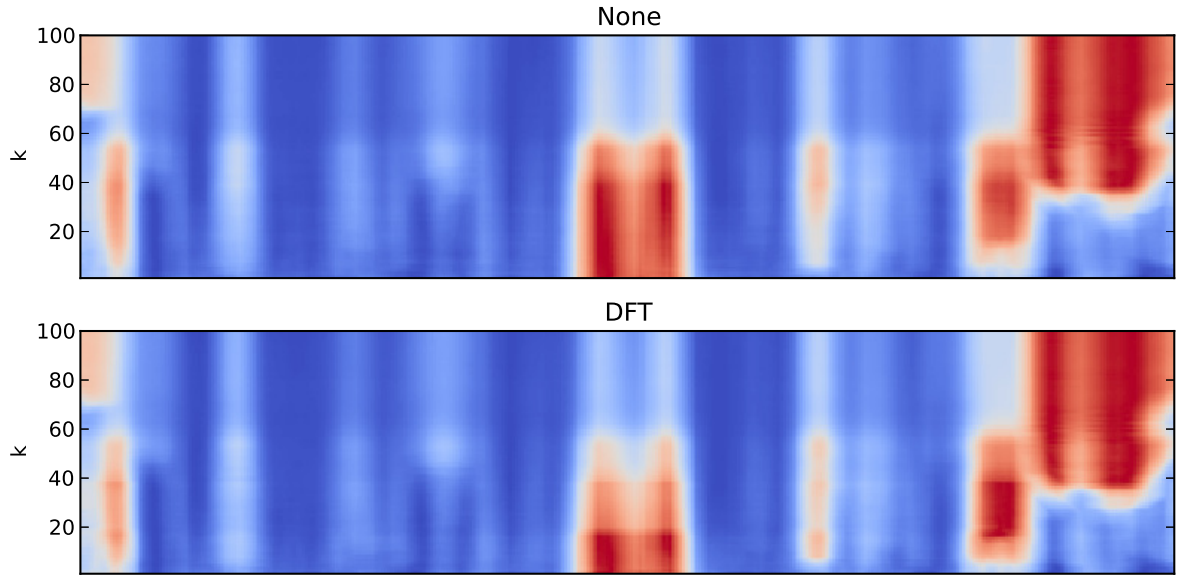
5.5.3 Transformations

Figure 5.10: Heat maps of the $A_{k,t}$ for $k = 1, 2, \ldots, 100$ with and without the discrete Fourier transform.

As discussed in Chapter **??**, applying transformations to extracted subsequences prior to evaluation, such as to perform dimensionality reduction, might assist in discovering certain types of anomalies. While a large number of compressions and other transformations deserving investigation have been proposed, due to time constraints, only the discrete Fourier transform (DFT) was implemented in `ad-eval`.

The performance of the DFT was investigated by evaluating the standard sequence for $k = 1, 2, \ldots, n$ with and without the DFT. A heat map of the results is shown in Figure 5.10, and a plot of the corresponding errors is shown in Figure 5.11.

While the DFT gave fairly accurate anomaly vectors for low values of k, it performed poorly overall, returning less accurate anomaly vectors and higher error values over all k. This is reasonable: the DFT is not expected to perform well on random data. A proper evaluation of the performance characteristics of kNN methods using the DFT would require a more diverse dataset.
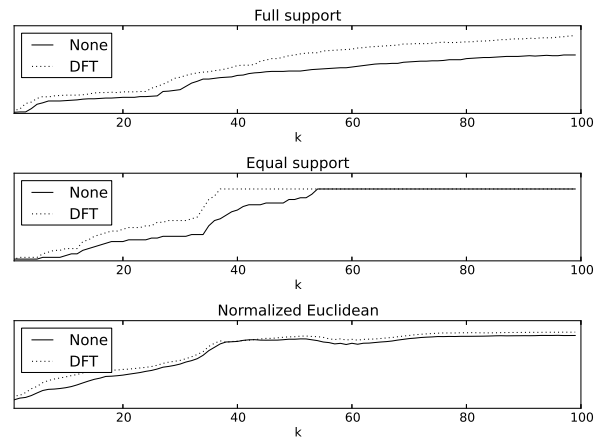
Figure 5.11: Errors of the $A_{k,t}$.

5.5.4 THE SLIDING WINDOW WIDTH
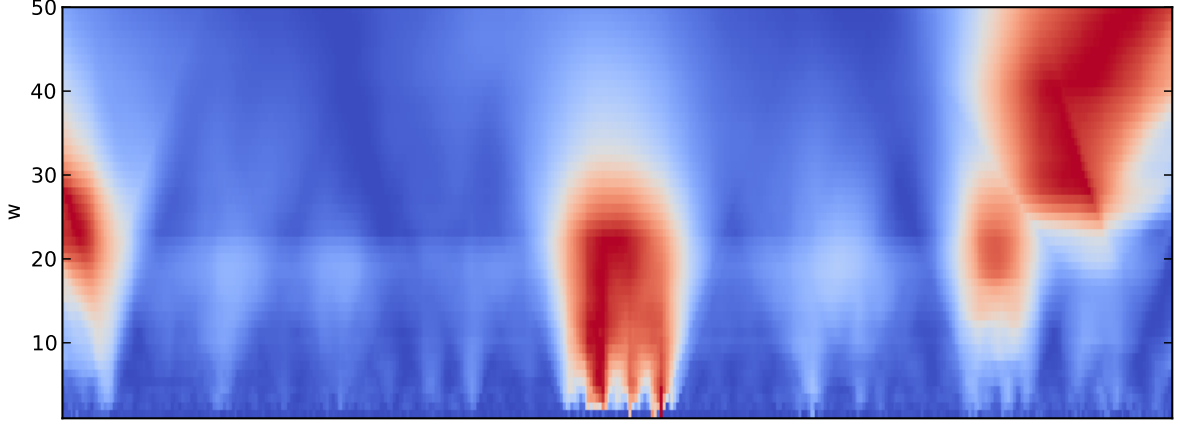
Figure 5.12: Heat map of the A_w for $w = 1, 2, \ldots, 50$.

Since w, the sliding window width, determines the size of the elements used by the evaluator, it should have a significant impact on the size of detected features. To determine if this was the case, the anomaly vectors A_w for $w = 1, 2, \ldots, 50$ were computed and examined. The results are shown in Figures 5.12, 5.13, and 5.14.

As seen in the figures, very low values of this parameter are associated with a very high error. This is expected, since as w tends to 1, the target anomaly type is reduced to point anomalies. Fur-

Figure 5.13: Errors for the anomaly vectors A_w.

thermore, all errors increase sharply as w nears 20, indicating that large values of w lead to inaccurate results.
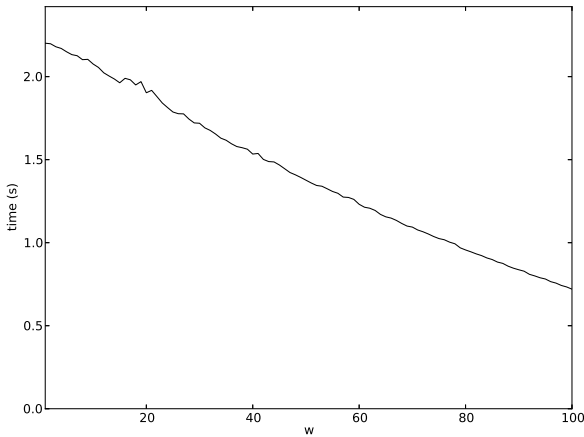
Figure 5.14: Evaluation times for the anomaly vectors A_w.

Interestingly, the plot in Figure 5.12 shows that beyond $w \approx 3$, increasing w essentially amounts to smoothing the resulting anomaly vectors. Since the anomaly in the standard sequence has a relatively small width of 40, and since its surroundings have low anomaly values for low values of w, this could help explain why the anomaly is not detected after $w \approx 40$.

It is further interesting to note that while the errors are at a minimum when $w \approx 5$, the anomaly vectors in this area contain three separate spikes in the vicinity of the anomaly, rather than a single smooth bump. Arguably, the anomaly vectors at $w \approx 10$ are preferable, since they more clearly mark the anomaly. This suggests that the error measures may need refinement.

Finally, while the evaluation time ought to be roughly independent of w (or proportional to the evaluation

time of the distance metric with vectors of length w), Figure 5.14 shows a decrease in the evaluation time as w grows. This is likely due to the fact that the relatively small width of the evaluation sequence means fewer elements are evaluated as w grows. An evaluation performed on a long sequence, in which the evaluation filter operates on the middle of the sequence while the reference filter operates on the entire sequence, could be used to confirm this.
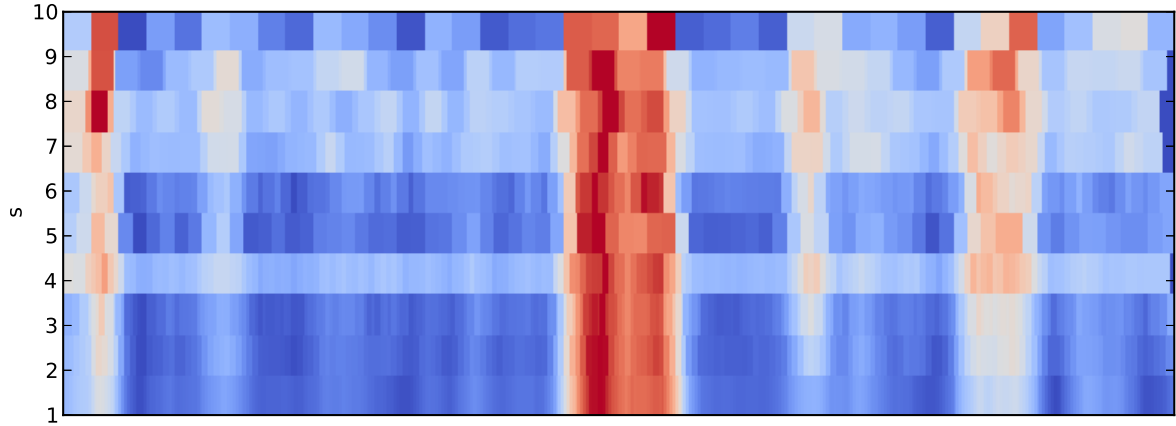
5.5.5 The sliding window step

Figure 5.15: Heat map of the anomaly vectors A_s for $s = 1, 2, \ldots, 10$. Note that no major false anomalies occur for $s < 8$.
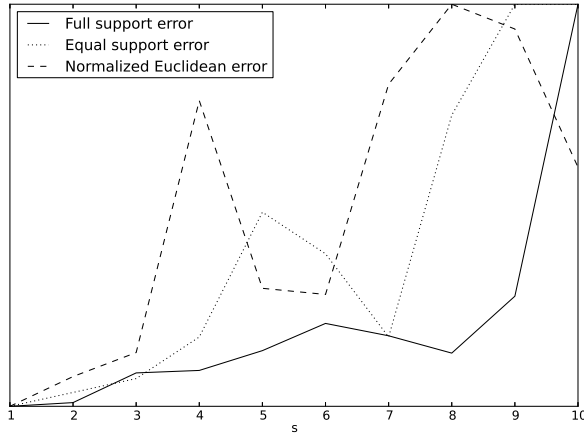
Figure 5.16: Errors of the anomaly vectors A_s.
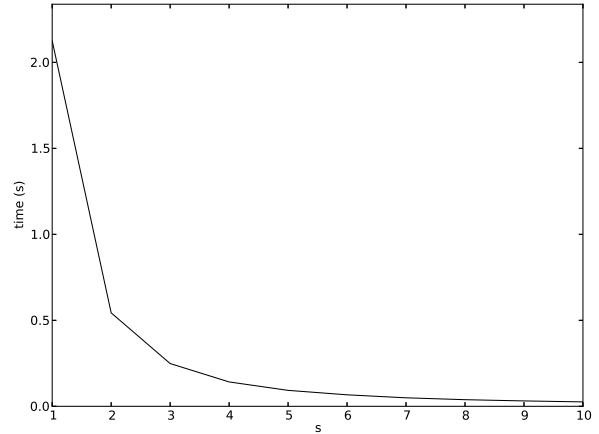
Figure 5.17: Evaluation times for the A_s. As expected, the graph indicates that the times are $O(1/s^2)$.

The sliding window step, s, is interesting mainly for the large effect it has on the execution time. For a brute-force kNN evaluator with the trivial context and sliding window filters, the number of comparisons performed on a sequence of length L is $\Theta((L/s)^2)$. It is therefor desirable to choose a value of s that is as large as possible. However, it is likely that all three errors increase with s for all sequences, and large s values might lead to poor results.

To gain some insight into the performance of kNN methods for higher s, the anomaly vectors A_s were computed for $s = 1$ to 10 (the value of w is 10 in the default configuration). The results are shown in Figures 5.15, 5.16, and 5.17.

As seen in Figure 5.15, the anomaly vectors are fairly accurate for all s. No major false anomalies are

exhibited for $s < 8$, and the actual anomaly is still clearly detected over all s. This is reflected in the errors in Figure 5.17: all errors are low until $s \geq 8$. Additionally, the evaluation time plot follows the expected $O(1/s^2)$ trend.

In light of these results, perhaps a multi-resolution scheme should be considered, in which a preliminary, 'coarse' evaluation (corresponding to high s), and a 'fine' evaluation (corresponding to low s) is performed only on those subsequences which are given the highest anomaly scores in the coarse evaluation. Depending on how the subsequences for the fine evaluation are selected, and on the context type, such an algorithm could achieve either lower computational complexity or an evaluation time reduction by a constant factor. If, as indicated in this evaluation, false positives but no false negatives are introduced as s increases, fine evaluation would only rule out false anomalies, and there would be no loss of analytical power.
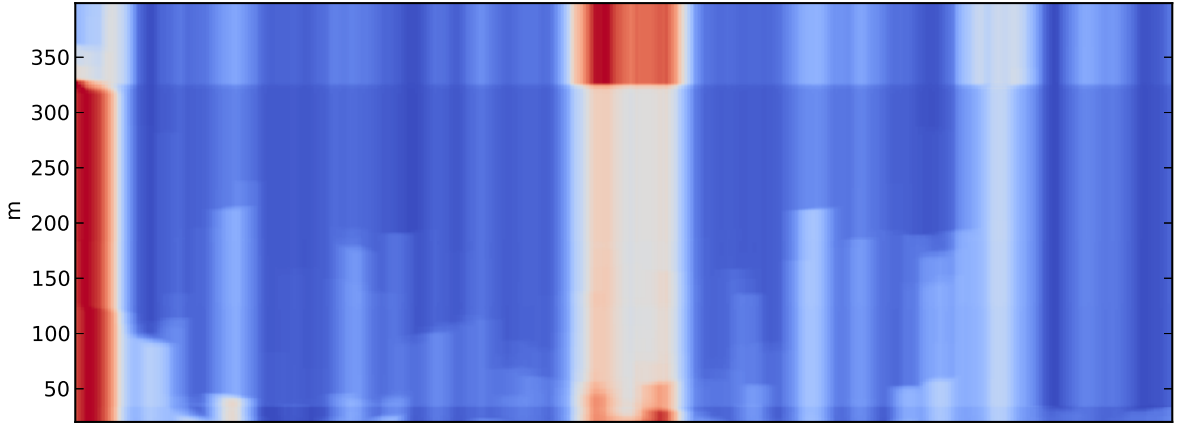
5.5.6   THE CONTEXT WIDTH

Figure 5.18: Heat map of the A_m for $m = 20, 21, \ldots, 400$. Note the false anomaly present at the left end of the anomaly vectors until $m \approx 330$.

Which values of the context width m are appropriate depends heavily on the application domain and on the types of anomalies present in the data. Ideally, the importance of the context width should be evaluated by considering several sequences with a natural context concept, such as the bottom series in Figure 3.6. Constructing representative artificial datasets of such sequences is likely to be difficult, so real-world series should be used for such an evaluation.

While such datasets are not available, a simple evaluation on the available data can still prove illuminating. The standard sequence is highly homogeneous and has no natural contexts. Thus, all errors should be expected to decrease monotonically with increasing m. To confirm this, the anomaly vectors A_m were computed for $m = 20$ to 400. The results of this evaluation are shown in Figures 5.18, 5.19, and 5.20.
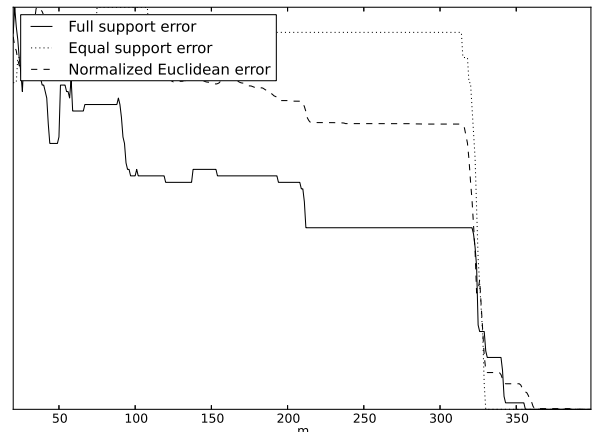
Figure 5.19: Errors of the anomaly vectors A_m.

As these figures demonstrate, the anomaly vectors identify a false anomaly at the left end until $m \approx$ 330, at which point the false anomaly disappears and the errors decline sharply. That this false anomaly appears for small context widths is understandable since, as seen in Figure 5.1, the sequence includes values at its left end that are not seen again until the right end. As expected, the error is minimized when the trivial context (corresponding to $m > 390$) is incorporated.

Finally, it should be noted that while the size of the reference set, and consequently the evaluation time, grows linearly with the size of the context, the average context size only grows linearly with m when m is much smaller than the sequence length. When m is close to the sequence length, the context size for a large portion of the subsequences extracted by the evaluation filter will be limited by the sequence edges. This leads to the curve in Figure 5.20.
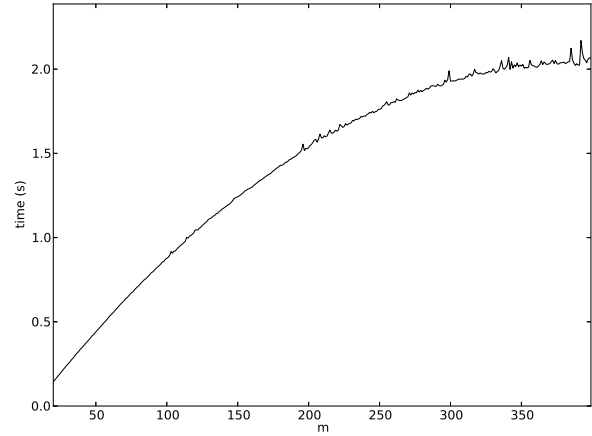

Figure 5.20: Evaluation times of the anomaly vectors A_m.

5.5.7 THE AGGREGATOR

To get an idea of how the choice of aggregator affects the analysis, the anomaly vectors $A_{k,\mathcal{A}}$ were computed and analyzed for the minimum, maximum, median and mean aggregators, with $k = 1, 2, \ldots, 100$. Heat map plots of the results are shown in Figure 5.21, and plots of the corresponding error measures are shown in Figure 5.22. Single anomaly vectors for $k = 1$ are shown in Figure 5.23.

As seen in Figures 5.22 and 5.23, the min and max aggregators produce blockly, piecewice constant anomaly vectors, while the mean aggregator (and, to a lesser extent, the median aggregator) produces
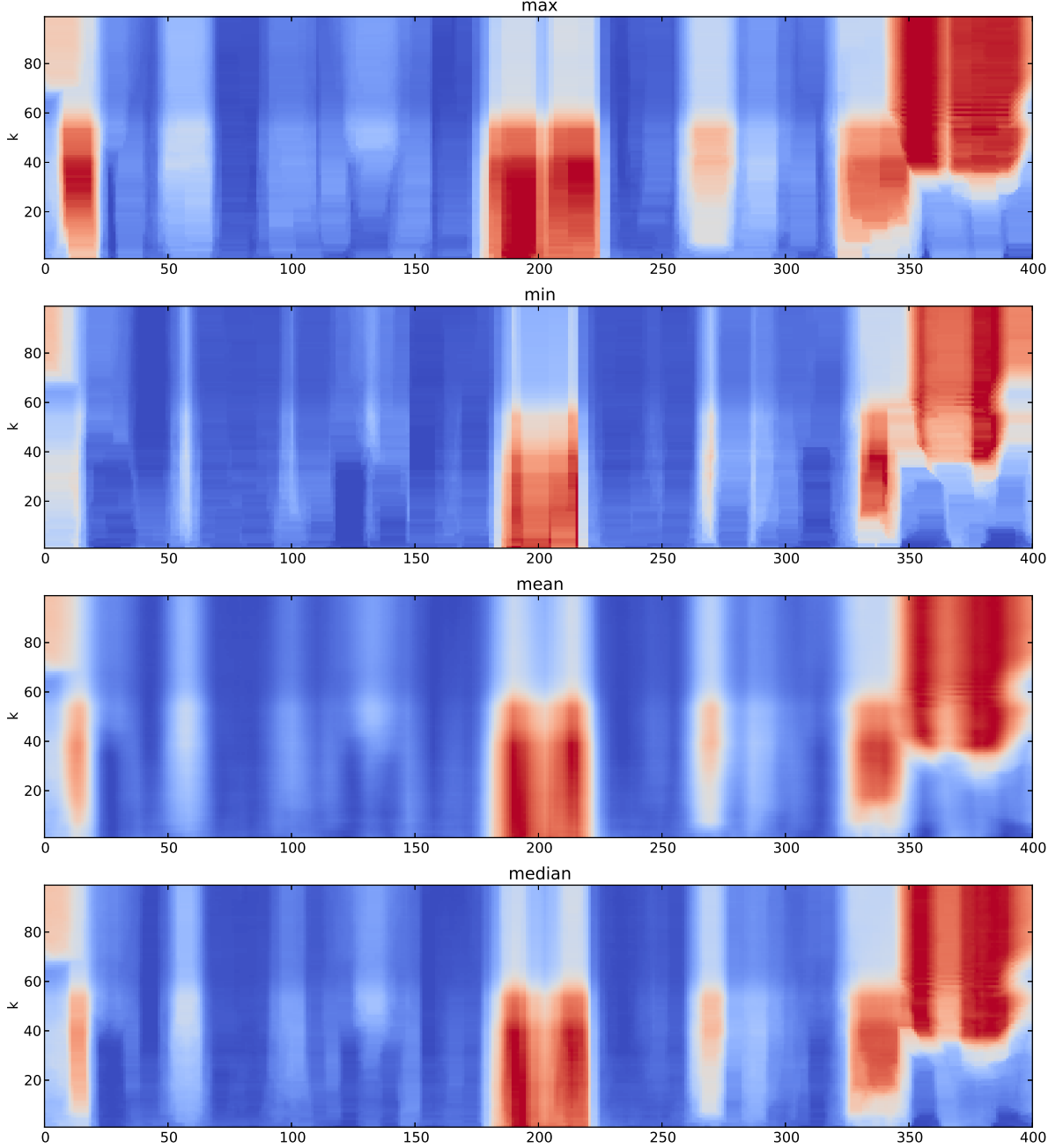

Figure 5.21: Heat maps showing $A_{k,\mathcal{A}}$ for the four aggregators.

As could be expected, the minimum aggregator consistently led to the highest values of ϵ_{FS}. It is likely to give a low score to a point if a single element containing that point has a low anomaly score, which effectively means that parts of anomalies will tend to be undervalued—something the full support error is

sensitive to. In contrast, the maximum aggregator consistently led to the lowest support error values. This is also as expected, since max will assign high values to any point contained in an anomalous subsequence. The median and mean aggregators performed roughly equally well—while the mean performed better for higher k, this is not relevant; both aggregators were very far off for higher k.
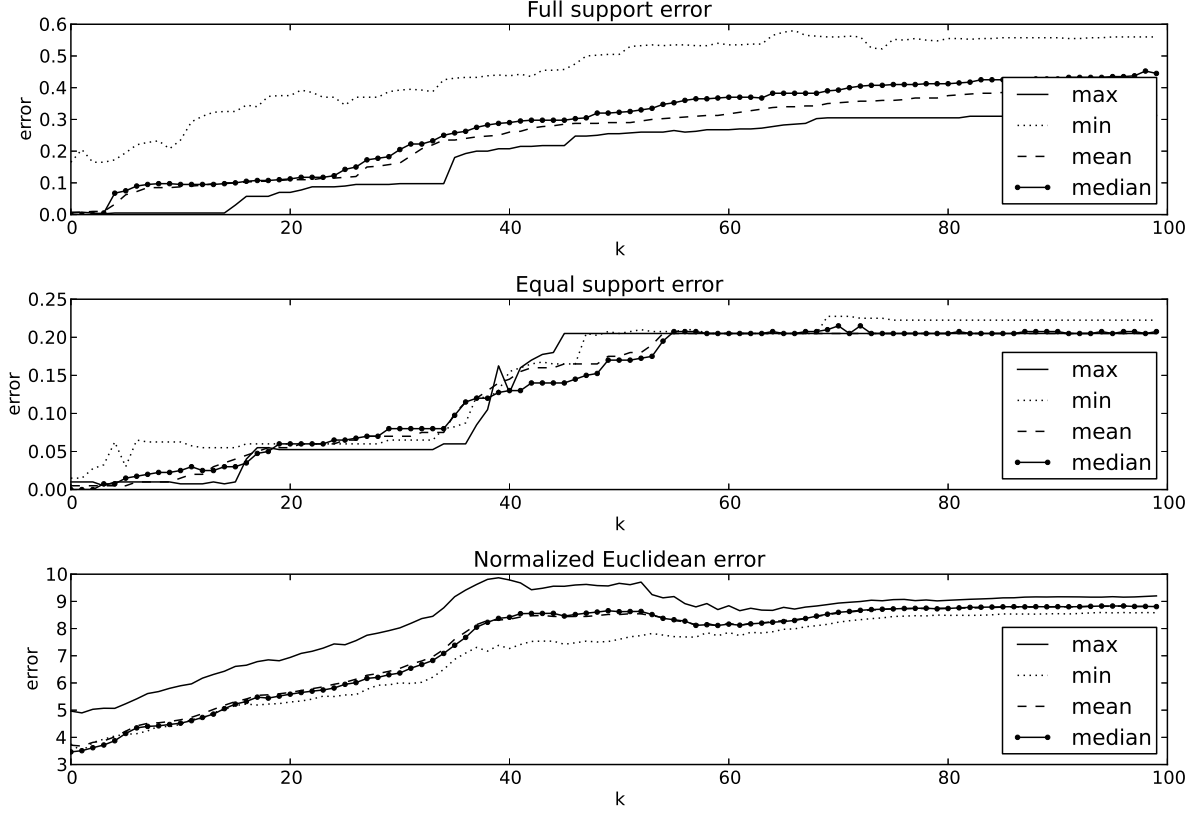

Figure 5.22: Errors of the anomaly vectors $A_{k,\mathcal{A}}$.

Similar, but less clear, results were obtained for the equal support errors. The minimum aggregator consistently performed the worst with low k, while the maximum aggregator performed the best, on average, with k up to 40. Again, the mean and median aggregators performed too similarly for any conclusions to be drawn on their relative merits.

Finally, the normalized Euclidean error gives almost indentical values to the mean and median aggregators, but exhibits a clear preference for the minimum aggregator over the maximum aggregator. This is likely a consequence of the fact fact that the minimum aggregator tends to assign scores close to zero to all elements except for a few, while the maximum aggregator tends to assign scores close to zero to only a few elements. As discussed in Section 5.4, the normalized Euclidean error has a bias in favor of anomaly vectors where most elements are close to zero, unlike the type of anomaly vectors produced by the maximum aggregator.
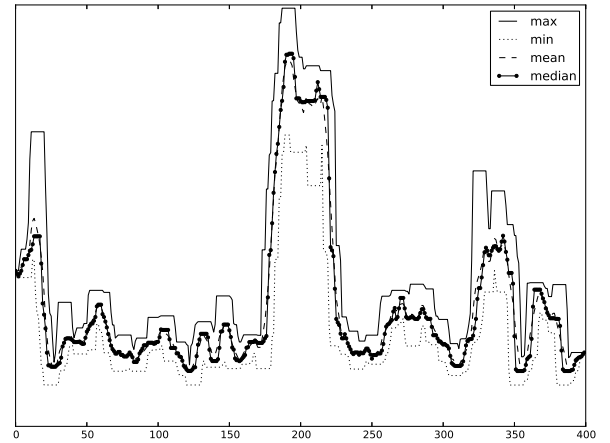

Figure 5.23: Plot of the $A_{k,\mathcal{A}}$ for $k = 1$.

61

In conclusion, all aggregators performed roughly equally well on s^* (arguably, the minimum aggregator performed slightly worse than the others). If this holds in general, then it appears that the choice of aggregator is mainly one of aesthetics.

6   DISCUSSION

We conclude this report with a short post-mortem and a discussion of future work.

6.1   POST-MORTEM

Overall, the project was successful. The new theory introduced in the form of the tasks and problems and component frameworks has made reasoning about and evaluating problems, as well as proposing novel problems and methods, significantly easier. Additionally, `ad-eval` has shown that the component framework can be effectively implemented and used to easily compare problems and methods. Furthermore, the evaluation utilities and evaluation scripts in `ad-eval` have shown that performing objective, reproducible method evaluations that can be reused with different datasets need not be difficult. Finally, as summarized in the next section, the project illuminated several new frontiers for future work.

However, there were some shortcomings. Since the initial focus on the implementation and evaluation of a few specific methods was not recognized as inappropriate until these methods had been partially implemented and large sections of the report had been written, and since the subsequent way forward was initially unclear, much of the work performed for the project was ultimately discarded.

Moreover, a proper evaluation of the methods implemented in `ad-eval` was beyond the scope of the project. While mitigated by the fact that the qualitative evaluation performed in Chapter 4.5 and the reusable evaluation scripts and utilities added to the repository will greatly facilitate such an evaluation once appropriate datasets are obtained, this rendered the project goal of finding the problem formulations most appropriate for the target domain only partially achievable.

6.2   FUTURE WORK

As repeated throughout this paper, there remains much work to be done on Task 4.3 and related tasks, as well as on `ad-eval`. A few potential areas in which such work would be useful are now highlighted.

6.2.1   EVALUATION

As mentioned several times in this paper, the evaluation performed as part of this project, alone, cannot conclusively answer which methods are appropriate for the target domain. As indicated in Section 4.5.1, preferably labeled data from the target domain should be used in the evaluation. However, such data could not be obtained, so a qualitative evaluation of how parameter and component choices affect the output anomaly vectors was performed. Once appropriate evaluation data is obtained, several interesting questions could be answered.

First, the tests performed as part of the evaluation in this project should be re-run on a more diverse dataset, to see if the conclusions made in Chapter 4.5 hold in general. Second, a larger portion of the parameter space should be evaluated. To mitigate the difficulties caused by the very long evaluation times required for such evaluations, a few modifications should be made to `ad-eval`. Methods for caching the results obtained in evaluations, along with tools for parallelizing and distributing evaluations, should be implemented. Tools for more effectively searching parameter spaces for minima should also be provided. The relative smoothness with which the output anomaly vectors seem to vary with parameter choices (at least for the kNN evaluator) could be exploited to avoid a search of the entire parameter space; methods such as discrete gradient descent combined with random restarts could significantly reduce the computation time as opposed to a brute-force search, likely without significant loss of accuracy. Finally, optimizations to the `anomaly_detection` module could potentially lead to large, constant-factor evaluation speedups.

6.2.2 Performance

Throughout the implementation of `ad-eval` and the evaluation, performance (in terms of computation time and memory usage) was deemphasized in favor of accuracy. This was done consciously, in order to limit the scope of the paper and to avoid the excessive focus on methods and optimizations found in much of the literature.

Several interesting questions regarding performance warrant investigation. Several types of both pure optimizations and approximations could be applied to the implemented problem formulations. For instance, multi-resolution algorithms, such as the one suggested in Section 5.5.5, could potentially lead to methods that are both fast and accurate.

Fortunately, the modular nature of `ad-eval` facilitates the evaluation of optimized methods. Just as a suite of unit, integration, and performance tests are often run after additions to commercial software projects, running performance and accuracy tests to evaluate how optimizations affect performance would be trivial in `ad-eval`.

6.2.3 Components

Several tasks and problems within the component framework would be interesting to study in more depth. As shown in Chapter **??**, there is large potential in suggesting and implementing several new methods within the component framework. For instance, there exist several anomaly measures and other components in the literature (including several model-based, artificial neural network-based, and statistical measures, as well as other distance-based methods and classifiers) that have not been implemented in `ad-eval`, and it is likely that some of these will perform better than the implemented components on certain datasets. Furthermore, several types of transformations should be evaluated (such as the discrete wavelet transform).

And of course, the parameter spaces of the implemented methods should also be studied in more depth. It would be especially interesting to examine how the error functions $E_{S,\delta}(\Theta)$ vary with the evaluation

data S. If the minima were to be computed for a large number of sequences from the target domain, then which problem formulations best suit which kinds of data could be studied from several angles.

6.2.4 RELATED TASKS

As indicated in Section 4.3.2, the components framework could easily be adapted to cover a large number of related tasks, including the tasks suggested in Section 4.2.3. Evaluations like the one in this paper could then be performed on these related tasks. It would be especially interesting to compare how individual components fare on different tasks.

Of course, other tasks would require other types of data, which might be difficult to obtain. However, the amount of development time required to adapt `ad-eval` should be minimal.

6.2.5 DEPLOYMENT

As indicated in Section 4.4.4, `ad-eval` was designed for eventual use in real-world applications. Due to its architecture, integration with software such as Splunk would be trivial. Then, the combination of `ad-eval` with a good user interface could prove an invaluable tool to monitoring and diagnosis through anomaly detection in a wide range of application domains.

6.2.6 ENSEMBLES

The components framework might also be used to find correlations between the accuracy of problem formulations and the underlying characteristics of the data being analyzed, as mentioned in Section 4.3.1. If such correlations exist, `ad-eval` could prove instrumental in their discovery due to the ease with which multiple problem formulations can be applied to, and evaluated in relation to, datasets.

Ideally, this could lead to an ensemble-style approach, in which several methods are combined and weighed based on their relative suitability to the characteristics of the data.

Acknowledgements

Bibliography

[1] Curt Monash "Three broad categories of data." http://www.dbms2.com/2010/01/17/three-broad-categories-of-data/

[2] Splunk, Inc. "Big Data Analytics." http://www.splunk.com/view/big-data/SP-CAAAGFH

[3] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." *ACM Computing Surveys (CSUR)* 41.3 (2009): 15.

[4] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection for discrete sequences: A survey." *Knowledge and Data Engineering, IEEE Transactions on* 24.5 (2012): 823-839.

[5] Chandola, Varun. "Anomaly detection for symbolic sequences and time series data." *Dissertation.* University of Minnesota, 2009.

[6] Hodge, Victoria, and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22.2 (2004): 85-126.

[7] Agyemang, Malik, Ken Barker, and Rada Alhajj. "A comprehensive survey of numeric and symbolic outlier mining techniques." *Intelligent Data Analysis* 10.6 (2006): 521-538.

[8] Barnett, Vic, and Toby Lewis. "Outliers in statistical data." *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, Chichester: Wiley, 1984, 3nd* (1984).

[9] Hawkins, D. M. "Identification of outliers." *Monographs on Applied Probability and Statistics*, (1980).

[10] Leroy, Annick M., and Peter J. Rousseeuw. "Robust regression and outlier detection." *Wiley Series in Probability and Mathematical Statistics, New York: Wiley*, (1987).

[11] Bakar, Zuriana Abu, et al. "A comparative study for outlier detection techniques in data mining." *Cybernetics and Intelligent Systems, 2006 IEEE Conference on.* IEEE, 2006.

[12] Phua, Clifton, Damminda Alahakoon, and Vincent Lee. "Minority report in fraud detection: classification of skewed data." *ACM SIGKDD Explorations Newsletter* 6.1 (2004): 50-59.

[13] Joshi, Mahesh V., Ramesh C. Agarwal, and Vipin Kumar. "Predicting rare classes: Can boosting make any weak learner strong?." *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2002.

[14] Dasgupta, Dipankar, and Fernando Nino. "A comparison of negative and positive selection algorithms in novel pattern detection." *Systems, Man, and Cybernetics, 2000 IEEE International Conference on.* Vol. 1. IEEE, 2000.

[15] Song, Xiuyao, et al. "Conditional anomaly detection." *Knowledge and Data Engineering, IEEE Transactions on* 19.5 (2007): 631-645.

[16] ESKIN, ELEAZAR, ET AL. "A GEOMETRIC FRAMEWORK FOR UNSUPERVISED ANOMALY DETECTION: DETECTING INTRUSIONS IN UNLABELED DATA." (2002).

[17] BASU, SABYASACHI, AND MARTIN MECKESHEIMER. "AUTOMATIC OUTLIER DETECTION FOR TIME SERIES: AN APPLICATION TO SENSOR DATA." *Knowledge and Information Systems* 11.2 (2007): 137-154.

[18] MA, JUNSHUI, AND SIMON PERKINS. "ONLINE NOVELTY DETECTION ON TEMPORAL SEQUENCES." *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2003.

[19] *Christos Faloutsos, M. Ranganathan and Yannis Manolopoulos, "Fast Subsequence Matching in Time-Series Databases."* Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data. *ACM, 1994*

[20] YI, BYOUNG-KEE, AND CHRISTOS FALOUTSOS. "FAST TIME SEQUENCE INDEXING FOR ARBITRARY LP NORMS." *Proceedings of the 26th international conference on very large databases*, 2000.

[21] *Chan, Kin-Pong, and Ada Wai-Chee Fu. "Efficient time series matching by wavelets."* Data Engineering, 1999. Proceedings., 15th International Conference on. *IEEE, 1999.*

[22] YE, NONG. "A MARKOV CHAIN MODEL OF TEMPORAL BEHAVIOR FOR ANOMALY DETECTION." *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop.* VOL. 166. OAKLAND: IEEE, 2000.

[23] BLENDER, R., K. FRAEDRICH, AND F. LUNKEIT. "IDENTIFICATION OF CYCLONE-TRACK REGIMES IN THE NORTH ATLANTIC." *Quarterly Journal of the Royal Meteorological Society* 123.539 (1997): 727-741.

[24] SEKAR, R., ET AL. "A FAST AUTOMATON-BASED METHOD FOR DETECTING ANOMALOUS PROGRAM BEHAVIORS." *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on.* IEEE, 2001.

[25] SEKAR, R., ET AL. "SPECIFICATION-BASED ANOMALY DETECTION: A NEW APPROACH FOR DETECTING NETWORK INTRUSIONS." *Proceedings of the 9th ACM conference on Computer and communications security.* ACM, 2002.

[26] KEOGH, EAMONN, ET AL. "FINDING THE MOST UNUSUAL TIME SERIES SUBSEQUENCE: ALGORITHMS AND APPLICATIONS." *Knowledge and Information Systems* 11.1 (2007): 1-27.

[27] KEOGH, EAMONN, STEFANO LONARDI, AND CHOTIRAT ANN RATANAMAHATANA. "TOWARDS PARAMETER-FREE DATA MINING." *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2004.

[28] KEOGH, EAMONN, ET AL. "LOCALLY ADAPTIVE DIMENSIONALITY REDUCTION FOR INDEXING LARGE TIME SERIES DATABASES." *ACM SIGMOD Record.* VOL. 30. NO. 2. ACM, 2001.

[29] KEOGH, EAMONN, ET AL. "DIMENSIONALITY REDUCTION FOR FAST SIMILARITY SEARCH IN LARGE TIME SERIES DATABASES." *Knowledge and information Systems* 3.3 (2001): 263-286.

[30] KEOGH, EAMONN, AND SHRUTI KASETTY. "ON THE NEED FOR TIME SERIES DATA MINING BENCHMARKS: A SURVEY AND EMPIRICAL DEMONSTRATION." *Data Mining and Knowledge Discovery* 7.4 (2003): 349-371.

[31] GEURTS, PIERRE. "PATTERN EXTRACTION FOR TIME SERIES CLASSIFICATION." *Principles of Data Mining and Knowledge Discovery* (2001): 115-127.

[32] FU, ADA, ET AL. "FINDING TIME SERIES DISCORDS BASED ON HAAR TRANSFORM." *Advanced Data Mining and Applications* (2006): 31-41.

[33] BU, YINGYI, ET AL. "WAT: FINDING TOP-K DISCORDS IN TIME SERIES DATABASE." *SDM*, 2007.

[34] YANKOV, DRAGOMIR, EAMONN KEOGH, AND UMAA REBBAPRAGADA. "DISK AWARE DISCORD DISCOVERY: FINDING UNUSUAL TIME SERIES IN TERABYTE SIZED DATASETS." *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on.* IEEE, 2007.

[35] LIN, JESSICA, ET AL. "APPROXIMATIONS TO MAGIC: FINDING UNUSUAL MEDICAL TIME SERIES." *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on.* IEEE, 2005.

[36] VENABLES, WILLIAM N., AND BRIAN D. RIPLEY. CH. 5.6 "DENSITY ESTIMATION" *Modern applied statistics with S.* SPRINGER, 2002.

[37] CHAN, PHILIP K., AND MATTHEW V. MAHONEY. "MODELING MULTIPLE TIME SERIES FOR ANOMALY DETECTION." *Data Mining, Fifth IEEE International Conference on.* IEEE, 2005.

[38] WARRENDER, CHRISTINA, STEPHANIE FORREST, AND BARAK PEARLMUTTER. "DETECTING INTRUSIONS USING SYSTEM CALLS: ALTERNATIVE DATA MODELS." *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on.* IEEE, 1999.

[39] LIN, JESSICA, ET AL. "EXPERIENCING SAX: A NOVEL SYMBOLIC REPRESENTATION OF TIME SERIES." *Data Mining and Knowledge Discovery* 15.2 (2007): 107-144.

[40] MÖRCHEN, FABIAN. "TIME SERIES KNOWLEDGE MINING." *Dissertation.* 2006, PHILIPPS-UNIVERSITÄT MARBURG.

[41] LEE, WENKE, AND DONG XIANG. "INFORMATION-THEORETIC MEASURES FOR ANOMALY DETECTION." *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on.* IEEE, 2001.

[42] CHEN, SCOTT, AND PONANI GOPALAKRISHNAN. "SPEAKER, ENVIRONMENT AND CHANNEL CHANGE DETECTION AND CLUSTERING VIA THE BAYESIAN INFORMATION CRITERION." *Proc. DARPA Broadcast News Transcription and Understanding Workshop.* 1998.

[43] RADKE, RICHARD J., ET AL. "IMAGE CHANGE DETECTION ALGORITHMS: A SYSTEMATIC SURVEY." *Image Processing, IEEE Transactions on* 14.3 (2005): 294-307.

[44] JAIN, ANIL K., M. NARASIMHA MURTY, AND PATRICK J. FLYNN. "DATA CLUSTERING: A REVIEW." *PACM computing surveys (CSUR)* 31.3 (1999): 264-323.

[45] Sandve, Geir Kjetil, and Finn Drablos. "A survey of motif discovery methods in an integrated framework." *Biol Direct* 1.11 (2006).

[46] Tanaka, Yoshiki, Kazuhisa Iwamoto, and Kuniaki Uehara. "Discovery of time-series motif from multi-dimensional data based on mdl principle." *Machine Learning* 58.2 (2005): 269-300.

[47] Kalpakis, Konstantinos, Dhiral Gada, and Vasundhara Puttagunta. "Distance measures for effective clustering of ARIMA time-series." *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on.* IEEE, 2001.

[48] Tax, David MJ. "One-class classification." *Dissertation.* University of Delft, 2001.

[49] Wang, Changzhou, and X. Sean Wang. "Supporting content-based searches on time series via approximation." *Scientific and Statistical Database Management, 2000. Proceedings. 12th International Conference on.* IEEE, 2000.

[50] Berndt, D., and James Clifford. "Using dynamic time warping to find patterns in time series." *KDD workshop.* Vol. 10. No. 16. 1994.