KTH Computer Science
and Communication

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

ANDRÉ ERIKSSON

Master's Thesis at NADA
Supervisor: Hedvig Kjellström
Examiner: TODO

TRITA xxx yyyy-nn

Abstract

Anomaly detection is an important issue in data mining and analysis, with applications in almost every area of science, technology and business that involves data collection. The development of generally applicable anomaly detection methods can therefore have a large impact on data analysis across many domains. However, due to the highly subjective nature of anomaly detection, there are no generally applicable methods, and for each new application a large number of possible methods must be evaluated. In spite of this, little work has been done to automate the process of anomaly detection research for new applications.

In this report, a novel approach to anomaly detection research is presented, in which the task of finding appropriate anomaly detection methods for some specific application is formulated as an optimisation problem over a set of possible problem formulations. In order to facilitate the application of this optimisation problem to applications, a high-level framework for classifying and reasoning about anomaly detection problems is also introduced.

An application of this optimisation problem to anomaly detection in sequences is also presented; algorithms for solving general anomaly detection problems in sequences are given, along with tractable formulations of the optimisation problem for the main anomaly detection tasks in sequences.

Finally, a software implementation of the optimisation problem for detecting anomalous subsequences in long real-valued sequences is presented, along with some preliminary performance results.

Contents

Chapter 1

Introduction

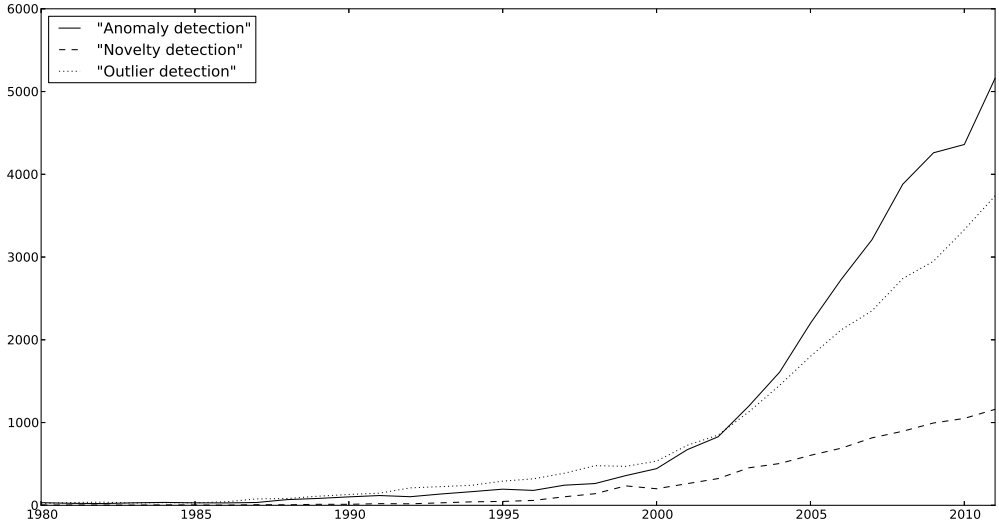


Figure 1.1. Approximate number of papers (by year) published between 1980 and 2011 containing the terms "anomaly detection", "outlier detection" and "novelty detection". All three terms exhibit strong upward trends in recent years. Source: Google Scholar.

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 sequences (and especially real-valued continuous time series).

Splunk is essentially a database and tool for storing and analyzing very large sets of machine-generated data. 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 equip-

ment 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.

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 recent years, anomaly detection has become increasingly important in a variety of domains in business, science and technology. 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 can be expected that efficient and general anomaly detection tools could be of great benefit to Splunk. Furthermore, since real-valued time series are easy to form from machine-generated data with timestamps, and are relatively amenable to analysis, anomaly detection methods for real-valued time series can be expected to be especially useful.

Typically, finding appropriate anomaly detection methods for a given application is a laborious process that requires expertise both in data analysis and in the specific application and involves extensive trial and error. One key of the key challenges in providing general anomaly detection tools is to streamline and simplify this process.

With the above in mind, it was decided that the aim of this thesis should be to investigate automated methods of finding appropriate anomaly detection methods for arbitrary sets of real-valued sequences. To this end, the task of finding such methods was formalised as an optimisation problem, which was then studied in depth. The main contributions of the thesis are:

1. A search problem formulation of the task of finding appropriate anomaly detection methods.

2. A framework for comparing and reasoning about anomaly detection problems.

3. An application of the optimisation problem and framework to anomaly detection in sequences.

4. A software implementation of the optimisation problem for real-valued sequences.

In Chapter 2, various background information useful to the rest of the report is presented. Specifically, the subject of anomaly detection is presented in more depth, along with some background on some of the problems faced in anomaly detection research. Finally, the optimisation problem approach is introduced. The main barriers to practical applications of the optimisation problem—finding an appropriate tractable set of problems over which to optimise, and finding an oracle for solving arbitrary problems in that problem set—are discussed.

As a means of overcoming these hurdles, in Chapter 3, a framework for reasoning about and comparing anomaly detection problems is introduced. As part of the framework, a few novel concepts and generalisations of existing concepts are introduced.

Next, in Chapter 4, the framework is applied to find tractable problem sets and corresponding oracles for two anomaly detection tasks commonly encountered in applications involving sequences. In conjunction with this, a thorough survey of previous research on anomaly detection in sequences is presented. Finally, a software implementation, called `ad-eval`, of the optimisation problem applied to the task of finding anomalous subsequences in real-valued univariate sequences is presented.

In Chapter **??**, some preliminary performance results of optimisation using `ad-eval` are presented. TODO: finish this paragraph once the results chapter is done.

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

Chapter 2

Background

In this chapter the subject of anomaly detection is briefly introduced, along with a discussion of some of the major challenges in anomaly detection research. Finally, the task of finding appropriate anomaly detection methods for a given application is formulated as an optimisation problem.

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. categorical), the structure of the dataset (a long sequence vs. several sequences), as well as the nature of the anomalies (a subsequence vs. one sequence out of many).

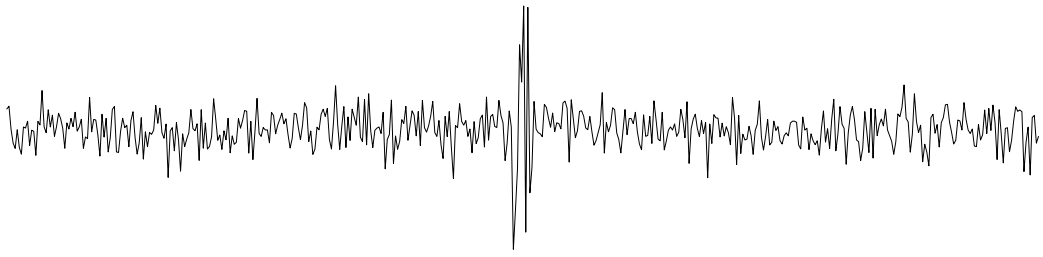


Figure 2.1. Real-valued sequence with an anomaly at the center.

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 um-

brella 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.

$\mathbf{S_1}$	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 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.

As a consequence of the above, a thorough survey of the subject could not fit within the scope of this report. The interested reader is instead referred to any of several published surveys [6] [11] [3] [7] and books [8] [9] [10] have been published which treat various anomaly detection applications in greater depth.

We now present a few classifications which are useful in reasoning about anomaly detection problems.

2.1.1 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 data.

[1] Note that we here adopt the convention used in [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. Conventionally, supervised learning is taken to mean any learning from

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.

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 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.

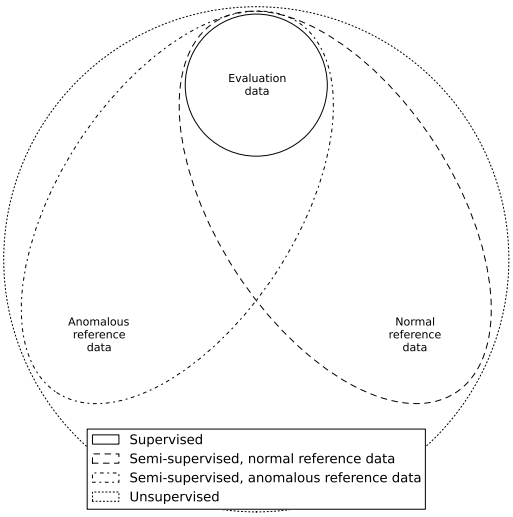


Figure 2.3. Euler diagram of the available training data for the four types of supervision.

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).

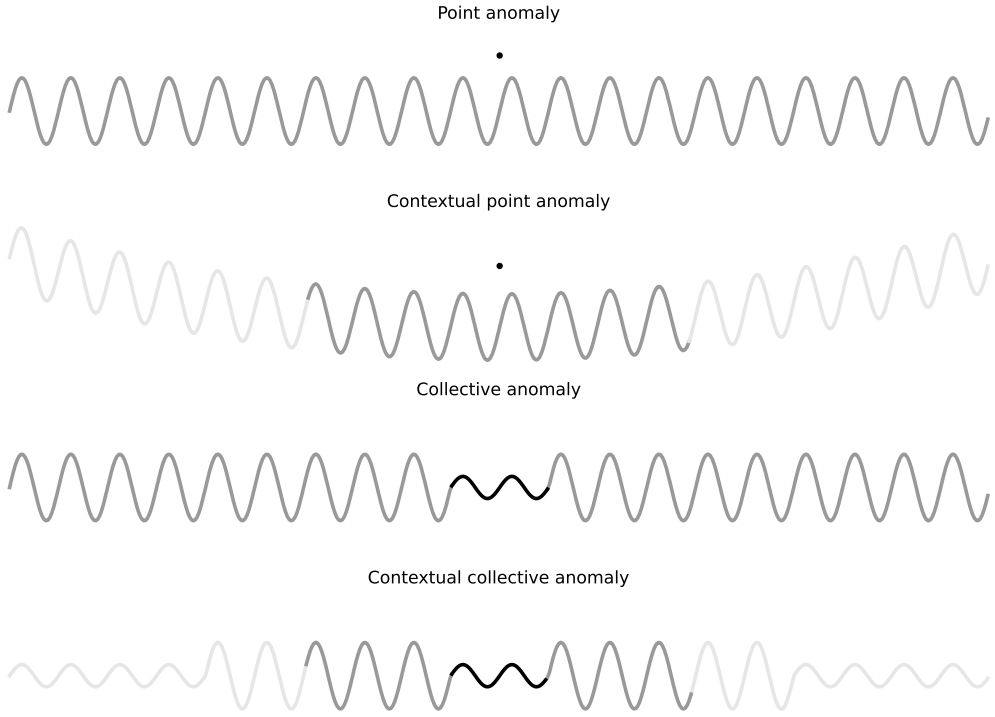


Figure 2.4. Different types of anomalies in a real-valued continuous sequence. In the middle of each series is an aberration—shaded black—corresponding to a specific type of anomaly. Appropriate contexts for these anomalies are shaded dark grey, while items not part of the contexts are shaded light grey. 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.

2.1.2 Anomaly types

It is very useful to classify problems based on what types of anomalies they can detect. To this end, we now describe four *anomaly types* [2], which can be used to classify the types of anomalies handled by problems. In order of increasing generality, these are *point anomalies*, *contextual point anomalies*, *collective anomalies*,

training data, and semi-supervised learning is taken to mean that both labeled and unlabeled data is available [**?**].

[2]While the concept of an anomaly type as defined here is novel, it is based on the concepts of contextual and collective anomalies discussed in [3].

and *contextual collective anomalies*. An illustration of these anomaly types in the context of real-valued sequences is shown in Figure 2.4.

Point anomalies is the simplest of the anomaly types. These correspond to single points in the dataset that are considered anomalous with regard to the entire training set. Point anomalies are often referred to as *outliers* and arise in many domains [16]. Compared to the other anomaly types, detecting point anomalies is 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 [**?**] can be useful. Essentially, point anomalies are the only anomaly type it makes sense to look for when the individual elements of the input dataset are unrelated.

When the individual elements of the input dataset *are* related (for instance, through an ordering or a metric), however, not all interesting anomalies will be point anomalies. The concept of *contextual point anomalies* generalises point anomalies to take context into account, which makes it more suitable to such cases. We here take the context to be the set of items with which an item is compared; when the input dataset admits a concept of proximity, the context of an item is usually those items which are closest to that item. Contextual anomalies are defined as individual items that are anomalous with regards to their context; 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 follows: Given a dataset D and a context function $C(d)$ which associates a context with each $d \in D$, a contextual point anomaly d is a point anomaly 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 $C(d) = D \setminus d$.

Of course, detecting individual anomalous points $d \in D$ might not always suffice, and the concept of *collective anomalies* might be required to capture relevant anomalies. Collective anomalies correspond to subsets of the input data that, when taken as a whole, are anomalous with regards to the entire training set. The task of detecting such anomalies can be formulated with the help of filter functions, which are map an input dataset D to a set of candidate anomalies $F(D)$ (where $\forall f_i \in F(D) : f_i \subset D$). Formally, given a set D and a filter F, the collective anomalies of D are the point anomalies of $F(D)$. Of course, point anomalies are a special case of collective anomalies, corresponding to the case where $F(D) = \{D\}$.

Finally, the concept of *contextual collective anomalies*, which generalises contextual point anomalies and collective anomalies, can be introduced. Contextual collective anomalies correspond to subsets of the input dataset that are anomalous with regard to their context. Formally, given a dataset D, a filter F, and a context function C, the contextual collective anomalies of D are the elements of $X \in F(D)$ 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 2.4. 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 2.4, none of the aberrations in the other sequences would be considered point anomalies.

While the value at the anomalous point at the center of the second sequence occurs elsewhere in that sequence, it is anomalous with regards to its 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 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).

2.2 On Anomaly Detection Research

Most anomaly detection research involves either applying existing methods to new applications (i.e. on new types of data) or investigating new methods in the context of 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 manner. However, little work has been done on developing automated methods and tools for anomaly detection research.

There are a few difficulties which complicate research into anomaly detection for new applications. For one, 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. This renders direct comparisons problematic and makes it hard to assess which methods might be appropriate to use in new applications. A systematic way of comparing anomaly detection methods would be helpful in mitigating this problem.

Furthermore, reproducing existing results as well as applying existing methods to new datasets is often difficult. 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, there is a lack of freely available software implementation of most methods.

An important distinction to make is that between problems and methods. Informally, the process of finding an appropriate anomaly detection method for some application can be described as a two-step process. As a first step, an appropriate problem is formulated, which accurately captures intuitive notions of what constitutes an anomaly in the specific application. Once such a problem formulation has been found, an efficient method of solving or finding approximate solutions to the problem is 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, rather than problems, are often emphasised, and it can often be unclear exactly what problem a given method is meant to solve. In this report, the focus is instead placed almost entirely on problems.

This work attempts to simplify the research process by providing tools which help address the hurdles described above. As a means of mitigating the first problem, a general framework for systematically comparing anomaly detection problem formulations is presented, the purpose of which is to facilitate high-level reasoning about anomaly detection problems, and which thereby can help simplify the application of existing methods to new domains.

Furthermore, the process of finding appropriate anomaly detection methods for a given application is studied as an optimisation problem. A software implementation of this optimisation problem for anomaly detection in real-valued sequences is presented, which can help mitigate the reproducibility issues outlined above, as well as streamline the research process by enabling researchers to automatically evaluate a large amount of problem formulations.

2.3 Problem formulation

As a first step towards the goal of automated tools for anomaly detection research, the task we are trying to automate—-that of finding appropriate anomaly detection methods for some given 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.

To motivate our optimisation problem formulation, one can consider a stylized variant of the typical anomaly detection research process, in which a researcher equipped with a working hypothesis (in the form of a problem formulation, which associates a set of anomaly scores with each possible input from the application) is given a dataset sampled from the target application, for which she constructs a set of anomaly scores in line with her hypothesis (i.e. a solution to her problem formulation). She then shows her results to a domain expert, who rates them based on how well aligned he deems them to be with his notion of what is anomalous and not in the specific application. This is repeated, with the researcher successively improving her problem formulation until the domain expert tends to agree with sufficiently well with its solutions.

A significant share of the work involved in finding appropriate methods could be avoided if this process could be automated, and one way to automate it is to formulate it as an optimisation problem that can be algorithmically solved. To do this, the concepts presented above must be formalised.

To begin with, the sets of valid problem inputs (datasets) and outputs (solutions) must be defined. Here, we simply assume that some set $\mathcal{D}$ has been defined containing all possible datasets for the application, along with some set $\mathcal{S}$ consisting of all valid corresponding solutions.

Next, a formal description of all allowed problem formulations must be constructed. Here, we simply assume that this description consists of a set of formulae in some logic sufficiently expressive to capture all relevant problem formulation. Let us call this set $\mathcal{P}$.

The role of the domain expert can be modeled by means of an error function $\epsilon(D, S) : \mathcal{D} \times \mathcal{S} \to \mathbb{R}^+$, which associates a score to any solution S based on how accurately it captures the anomalies in the data D.

The researcher, on the other hand, really has two roles; finding a new P based on the feedback from the domain expert, and computing a solution S given some dataset D and problem P. The former role corresponds to the heuristic driving the optimisation—searching the problem set $\mathcal{P}$ for an appropriate problem—and does not need be formalised yet. The latter role can be formalised as an oracle $O(P, D) : \mathcal{P} \times \mathcal{D} \to \mathcal{S}$, which takes a problem $P \in \mathcal{P}$ and an input dataset $D \in \mathcal{D}$, and computes the associated solution $S \in \mathcal{S}$.

The success of P in capturing the anomalies in D can then be stated as $\epsilon(D, O(P, D))$. Finally, since the goal is to minimise the *expected* error for datasets sampled from the given application, a random variable X over $\mathcal{D}$ ought to be introduced, that models the probability of encountering any given $D \in \mathcal{D}$. A suitable objective function would then be $\mathbb{E}_X[\epsilon(D, O(P, D))]$.

The optimisation problem then becomes:

$$P_{opt} = \operatorname*{argmin}_{P \in \mathcal{P}} \mathbb{E}_X[\epsilon(D, O(P, D))].$$

Here, P_{opt} corresponds to the best possible problem formulation, and $O(P_{opt}, D)$ to the solution of this problem formulation.

Of course, this optimisation problem is not tractable unless heavy restrictions are placed on the problem set $\mathcal{P}$, since any logic sufficiently complicated to encompass non-trivial problems is undecidable (TODO: provide source?). Thus, no oracle can not exist when $\mathcal{P}$ is the set of all formulae in such a logic. A major challenge is thus to find a reduced problem set $\mathcal{P}^*$ which has a tractable oracle O^* and which contains a majority of interesting problem formulations.

Another problem is that it is generally not possible to compute ϵ or X. Indeed, an algorithmic formulation of ϵ presupposes knowledge of the optimal problem formulation and would consequently render the optimisation process redundant. Likewise, the generation of a stream of data in accordance with X would require an exact model of the underlying process, which could just as well be used directly to detect anomalies. Even if an external stream of datasets would be available, an actual domain expert would be required to represent ϵ.

To get around these issues, the random vairable X can be replaced with a set of labeled training data, i.e. a set $\mathcal{T} \subset \mathcal{D}$ in which each $T_i \in \mathcal{T}$ has an associated $s(T_i) \in \mathcal{S}$. Correspondingly, $\epsilon(D, S)$ can then be replaced with some $\epsilon^*(T_i, S) = \delta(s(T_i), S)$, where $\delta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+$ is some distance measure. This approach leads to the following estimate of P_{opt}:

$$P^*_{opt} = \operatorname*{argmin}_{P \in \mathcal{P}^*} \sum_{T_i \in \mathcal{T}} \delta(s(T_i), O^*(P, T_i)).$$

A major focus of this report is the construction of restricted problem sets P^* and corresponding tractable oracles O^*. In Chapter 3, a framework for reasoning about anomaly detection problems is outlined, which can be used to construct appropriate problem sets for specific applications. This framework is then applied to sequences in Chapter 4, in order to construct a problem sets that generalise a majority of previously studied problem formulations while admitting a simple oracle.

Chapter 3

A Framework for Anomaly Detection

In this chapter, a framework for reasoning about anomaly detection problem formulations is presented. This framework can be utilised in order to limit the scope of the optimisation problem outlined in the previous chapter by enabling the systematic construction of tractable problem sets.

The core idea of the framework is that anomaly detection problems can be almost exhaustively classified based on a few independent factors, and that by studying the factor choices handled in the anomaly detection literature, insights may be obtained into what problem formulations are appropriate for specific applications as well as how to formulate algorithms which solve these problem formulations.

As mentioned in Section 2.3, a problem formulation is a specification that associates with each element in the set $\mathcal{D}$ of possible datasets a unique element of the set $\mathcal{S}$ of possible solutions. In other words, problem formulations can be seen as functions $P : \mathcal{D} \to \mathcal{S}$, and the problem set $\mathcal{P}$ can be seen as the set of all such functions. Correspondingly, the task of selecting an appropriate restricted problem set $\mathcal{P}^*$ is equivalent to the task of finding an appropriate restricted subset of such functions.

One interesting aspect of anomaly detection is that almost all problem formulations found in the literature share common structure, in that they involve selecting a set of subsets of the input data, comparing each element of this set to some set of reference elements to produce a set of anomaly scores, and aggregating these anomaly scores into a result.

If this common structure could be captured by decomposing P into a collection of functions between sets, then the task of finding an appropriate $\mathcal{P}^*$ could be simplified to the task of placing appropriate restrictions on these individual functions. Correspondingly, formulating an oracle which can solve arbitrary $P \in \mathcal{P}^*$ could be simplified to the task of computing each individual function.

3.1 The problem decomposition

Our framework involves a specific such decomposition of P, which covers almost all previously studied problem formulations by capturing the common structure mentioned above. This decomposition is now presented in detail, assuming that $\mathcal{D}$ and $\mathcal{S}$ are fixed by the application. In parallel, illustrations are presented of a decomposition of an actual anomaly detection problem.

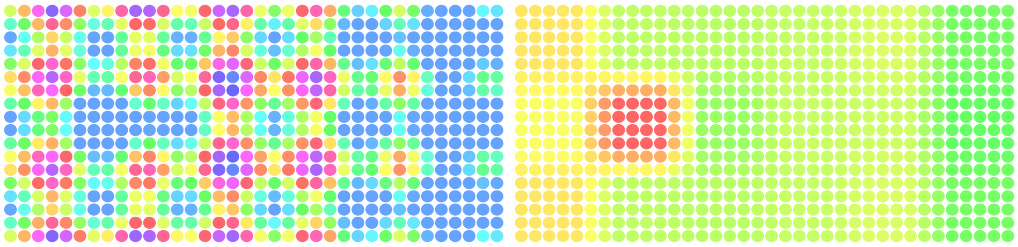


Figure 3.1. The example input data $D \in \mathcal{D}$ and the corresponding solution $P(D) = S \in \mathcal{S}$.

Roughly, our example problem associates an anomaly score with each element in a grid of colour values. These anomaly scores are also colours; red and green signify high and low anomaly scores, respectively. Specifically, the problem involves finding contextual collective anomalies—i.e. contiguous subsets of the data which are anomalous with regards to their surroundings—in such grids. To illustrate this problem, we will use the dataset shown in figure 3.1. This dataset contains an interesting anomaly to the left; a blue region that is larger than and has a different shape than nearby blue regions. The problem P we will decompose can be used to identify this anomaly, and the corresponding solution is shown to the right in the figure.

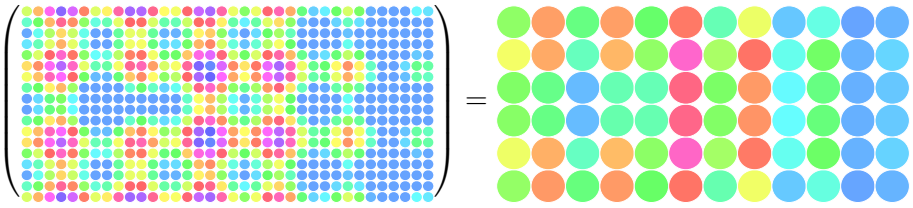
We make the assumption that the input data is an ordered collection, i.e. a list [1], of homogeneous items, which we denote by $\mathcal{D} = [D]$ for some arbitrary set D. Likewise, we take the solution format to be a list as well: $\mathcal{S} = [S]$ for some arbitrary set S. This means that we consider the set of problems to be equivalent to the set of all functions $P : [D] \to [S]$.

The proposed decomposition splits each such P into a composition of the following functions:

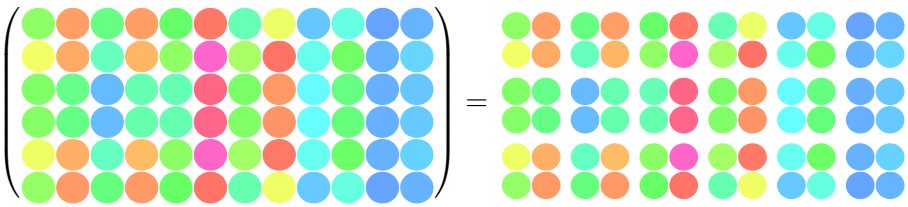
1. A transformation $T_D : [D] \to [D']$,which transforms a list of input data with elements in some set D to a list of elements in some other set D'. Typically,

[1] We will denote a list containing the items a, b and c by $[a, b, c]$, and we will denote the set of all lists with items in some set X by X. We will also assume that items in lists implicitly carry indices, and that any function $f : [X] \to [X]$ that maps a list to one of its sub-lists will preserve these indices, i.e. if $f([a, b, c]) = [b]$, then it is apparent that $f([a, b, c])$ is the second element of $[a, b, c]$, even if $b = a$ or $b = c$.

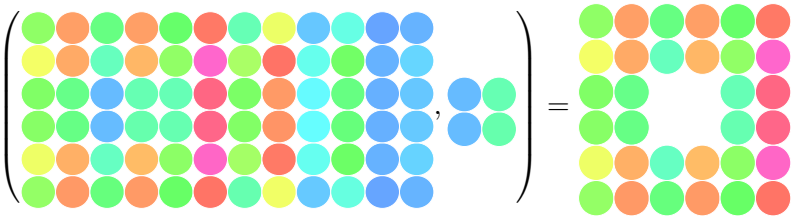
this is done in order to speed up or simplify the analysis. In our example [2], this transformation reduces the dimensionality of the dataset, by averaging the values of adjacent elements:

T_D  =

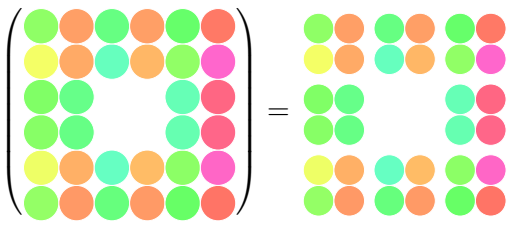
2. A evaluation filter $F_E : [D'] \to [[D']]$,which maps the transformed data to an *evaluation set*—a list of subsets of the transformed data, corresponding to potential anomalies. In our example, F_E simply partitions its input into collections of four elements:

F_E  =

3. A context function $C : ([D'], [D']) \to [D']$,which takes a dataset and a corresponding candidate anomaly (i.e. a sublist of the dataset), and produces an associated *context*. In our example, the context function $C(X, Y)$ produces a set of elements in X adjacent to Y:

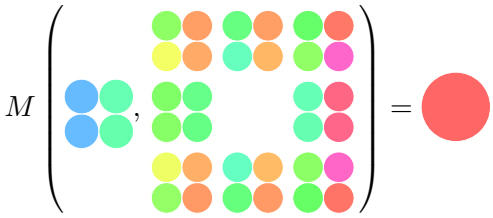
C  =

4. A reference filter $F_R : [D'] \mapsto [[D']]$,which works analogously to the evaluation filter, but operates on contexts instead of input data. In our case, F_R is identical to F_E; it partitions the context into subsets of four items:

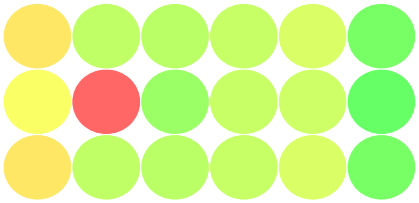
F_R  =

[2]Note that we here implicitly assume that a well-defined ordering of the elements in the example is provided, such that the data can be treated as a list.

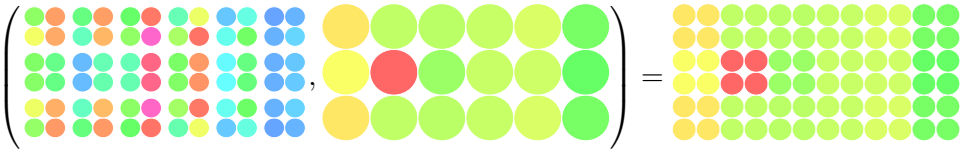
5. An anomaly measure $M : ([D'], [[D']]) \to A$, which takes an item $x \in [D']$ and a list $[x_1, x_2, \ldots, x_n] \in [[D']]$ of reference items, and computes an anomaly score (in some arbitrary set A) based on how anomalous x is with regards to $[x_1, x_2, \ldots, x_n]$. In our example, M works by computing an average distance between x and the x_i, which is mapped to a colour:



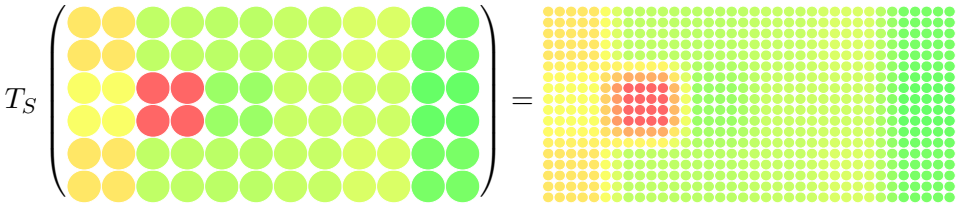
   Computing $M(e, F_R(C(e)))$ The result of applying C, F_R and M to each element of the evaluation set is:



6. An aggregation function $\Sigma : ([[X']], [A]) \to [S']$, which aggregates the anomaly scores for the elements of the evaluation set to form a 'preliminary' solution for elements of $[D']$. In our example, the aggregation function associates with each element the anomaly score of the candidate anomaly to which that element belongs:



7. A transformation $T_S : [S'] \to [S]$, which transforms the preliminary solution into an actual solution. In our example, T_S uses bilinear interpolation to produce anomaly scores for all elements of the input data:



The process of computing a solution to a problem P with associated T_D, F_E, C, F_R, M, Σ, and T_S for an input dataset $d \in [D]$ can be seen as a series of transformations on d.

Now, the sets $\mathcal{D}$ and $\mathcal{S}$, as well as the functions T_D, F_E, C, F_R, M, Σ, and T_S are discussed in detail.

3.2 The input data format $\mathcal{D}$

As mentioned above, we represent the set of possible input datasets by means of a set $\mathcal{D} \subset [D]$, i.e. a set of lists over some application-specific set D. A problem formulation associates with each element of $\mathcal{D}$ a solution in the set of solutions $\mathcal{S}$.

Methods are commonly classified based on characteristics of D. For instance, a distinction is typically made between categorical, discrete, and real-valued data based on the cardinality of D. The input data is said to be *categorical* (or *symbolic*) if D is finite, *discrete* if D is countable and *real-valued* if $D \subseteq \mathbb{R}^n$ for some n (other uncountable sets are typically not encountered). It is also frequently the case that D consists of some combination of categorical, discrete and real-valued data, in which case the input data is referred to as *mixed*[**?**].

Another classification, also based on characteristics of D, is that between uni- and multivariate data. If $D = X^n$ for some set X, the input data is called *multivariate*; otherwise it is called *univariate*. An illustration of uni- and multivariate time series is shown in figure 3.2.

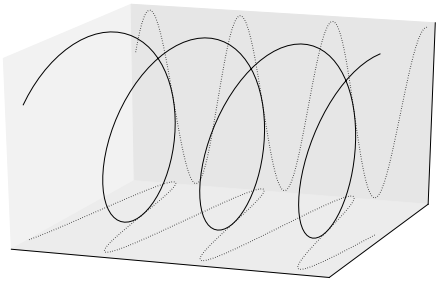


Figure 3.2. Two sine curves regarded as two separate univariate time series (dotted lines) and as one multivariate time series (solid lines).

Characteristics such as the dimensionality of the data typically prove important in applications. For instance, categorical data is typically both computationally and conceptually easier to handle than either discrete or real-valued data. Likewise, univariate data is typically much easier to handle than multivariate data.

3.3 The set of solutions $\mathcal{S}$

The collection of possible solutions is modelled as a set $\mathcal{S} \subset [S]$, i.e. some set of lists over some application-specific set S. In this section, typical choices of $\mathcal{S}$—which will henceforth be referred to as the *solution format*—are discussed. In practice, only a few distinct solution formats are used.

One common solution format involves associating *anomaly scores* with each element of the input dataset. In other words, for some input data $[d_1, d_2, \ldots, d_n]$, the corresponding solution will have the format $[s_1, s_2, \ldots, s_n] \in [S]$, where each s_i indicates how anomalous the corresponding d_i is. Typically, $S = \mathbb{R}^+$.

Another common approach is to, for some input data $[d_1, d_2, \ldots, d_n]$, let S contain the indices of the most anomalous elements of D, i.e. $S = \mathbb{Z}_n$. Two approaches can be distinguished here. Either, the solution has a fixed size, in which case the solution format is typically referred to as *discords* [26] [33] [34] [32] [35]., or the

solution will contain the indices of all elements which are considered sufficiently anomalous.

Finally, it might be interesting to let the solution consist of a list of anomalous subsets of the input data, i.e. $S = [D]$.

Which of these solution formats is used depends on the performance requirements of the application. For instance, while any set of anomalous indices can be constructed from a real-valued anomaly score vector, producing a set of anomalous indices (and especially discords) directly can be far less computationally intensive.

3.4 The transformations T_D and T_S

It is common for the input data to be preprocessed to make it more amenable to analysis. To account for this, two transformations $T_D : [D] \to [D']$ and $T_S : [S'] \to [S]$ are included in the framework. These transformations are complementary, in the sense that T_D maps the input data to some set $[D']$, while T_S takes a solution $[S']$ to some problem defined on elements in $[D']$ and maps it to a solution in $[S]$ for the data fed into T_D.

Typically, T_D involves either dimensionality or numerosity reduction. *Dimensionality reduction* involves reducing the dimensionality in the individual elements of the input dataset; i.e. a transformation $T_D : [D] \to [D']$ is a dimensionality reduction transformation if D' is of lower dimensionality than D.

Such transformations invariably involve some degree of information loss. Ideally, the information which they retain should be that which is most relevant to the analysis. Many methods have been designed with this goal in mind. A distinction is typically made between feature selection and feature extraction methods. *Feature selection* methods select a subset of the features present in the original data, while *feature extraction* methods create new features from the original data. An example of feature extraction is shown in figure 3.3.

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.

Numerosity reduction, on the other hand, serves to reduce the cardinality of the data, either by converting real-valued data to discrete or categorical data, by converting discrete data to categorical data, or by compressing categorical data. An example of numerosity reduction in time series is shown in figure 4.1.

3.5 The evaluation filter F_E

An important aspect of any problem is which subsets of the transformed data are considered candidate anomalies; i.e. which sublists of the transformed data $[d'_1, d'_2, \ldots, d'_n] \in [D']$ constitute the *evaluation set* $E \in [[D']]$. Letting the evaluation set consist of all sublists is not computationally feasible, and considering only

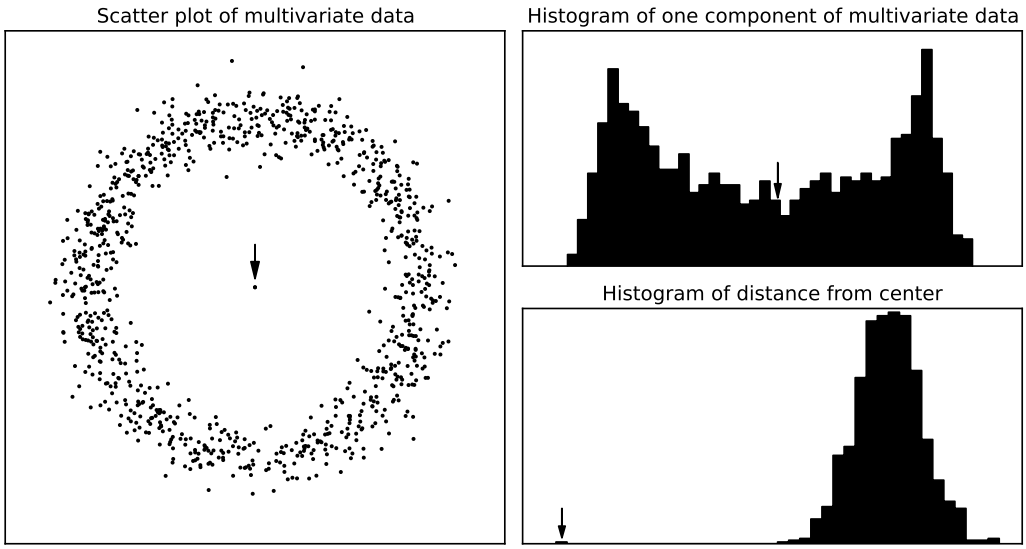


Figure 3.3. An example of dimensionality reduction in a point anomaly detection problem in R^2. The left figure shows a set of 500 data points (x_i, y_i) containing one anomaly. The top right figure shows a histogram of the x_i, while the bottom right figure shows a histogram of the distance from the 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.

single element lists is likely to be overly limiting for many applications. To allow for greater flexibility in the choice of evaluation set, the framework includes a function, the *evaluation filter* $F_E : [D'] \to [[D']]$, which includes the choice of evaluation set as part of the problem formulation.

What F_E is appropriate depends on whether or not there is any structure that relates the elements of the input data. If no such structure is present, allowing candidate anomalies with more than one element is not meaningful, and F_E should be given by $F_E([d'_1, d'_2, \ldots, d'_n]) = [[d'_1], [d'_2], \ldots, [d'_n]]$. On the other hand, if any such structure (such as an ordering of or distance between the elements) exists (and is pertinent to the analysis), then F_E ought to take that structure into account.

As an example, consider the case where the input elements $X = [d'_1, d'_2, \ldots, d'_k]$ constitute a sequence. Here, a concept of locality is naturally induced by the sequence ordering, and it is reasonable that $F_E(X)$ consist of contiguous sublists of X:

$$F_E([d'_1, d'_2, \ldots, d'_k]) = [[d'_{a_1}, d'_{a_1+1}, \ldots, d'_{b_1}], [d'_{a_2}, d'_{a_2+1}, \ldots, d'_{b_2}], \ldots, [d'_{a_n}, d'_{a_n+1}, \ldots, d'_{b_n}]]$$

for some arbitrary n, $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$, where $\forall i : b_i > a_i$.

An evaluation filter which extracts contiguous sublists of a fixed length is illustrated in Figure 3.4.

3.6 The context function C

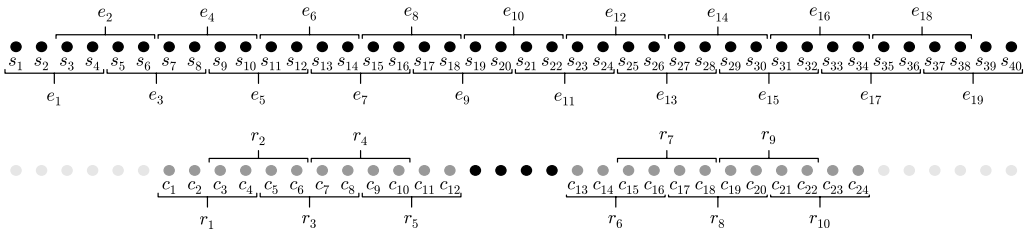


Figure 3.4. Schematic illustration of filters and contexts acting on a sequence $\mathbf{s} = [s_1, s_2, \ldots, s_{40}]$. The top panel shows the evaluation set $E = F_E(\mathbf{s}) = [e_1, e_2, \ldots, e_{19}]$ extracted by an evaluation filter which selects contiguous subsequences of fixed length. The bottom panel shows the context of e_{10} given by $C(\mathbf{s}, e_{10}) = [c_1, c_2, \ldots, c_{24}]$, as well as the corresponding reference set $F_R(e_{10}) = [t_1, t_2, \ldots, t_{10}]$ extracted by a reference filter identical to the evaluation filter.

Once the candidate anomalies have been identified, a *context* must be associated with each candidate anomaly. The context for a candidate anomaly represents the set of elements with which that candidate anomaly is to be compared. To this end, the decomposition associates with each problem a *context function* $C : ([D'], [D']) \to [D']$, which takes the transformed input data and a list of candidate anomalies, and associates with each candidate anomaly a context (in $[D']$).

As with the evaluation filter, what constitutes an appropriate context function is entirely dependent on what structure is present in the input data. If no such structure is present, the only reasonable choices of C are

$$C([d'_1, d'_2, \ldots, d'_k], [d'_{a_1}, d'_{a_2}, \ldots, d'_{a_l}]) = [d'_{b_1}, d'_{b_2}, \ldots, d'_{b_m}]$$

where $\{b_1, b_2, \ldots, b_m\} = \mathbb{Z}_k \setminus \{a_1, a_2, \ldots, a_l\}$, or $C(X, Y) = T$ for some fixed data $T \in [D']$. If, on the other hand, there is such structure present, a context function which takes it into account might be preferable. As a concrete example of such an application with such structure, consider a long sequence: often how an item compares to those items 'closest' to it (in the ordering) is more relevant to the analysis than how it compares to the rest of the sequence.

Context functions generalise several of the concepts discussed in the previous chapter; semi-supervised anomaly detection with normal training data and unsupervised anomaly detection correspond to $C(X, Y) = T$ and $C(X, Y) \subset X$ respectively, while contextual and point anomalies are just special cases of $C(X, Y) \subset X$.

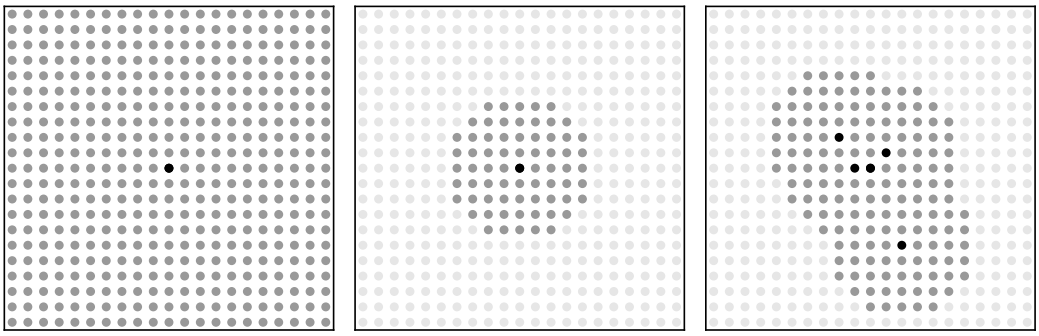


Figure 3.5. Schematic view of a dataset illustrating a few contexts. In each panel, the black dots represent selected items, the dark grey dots represent items in the context of the selected items, and the light grey dots indicate items not in the context of the selected items. The left panel shows the 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.

Context functions also generalise anomaly detection problems to various tasks that have traditionally been considered separate from anomaly detection. For instance, *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

$$C([s_1, s_2, \ldots, s_n], [s_i, s_{i+1}, \ldots s_j]) = [s_{i-w}, s_{i-w+1}, \ldots, s_{i-1}],$$

for some w, in an anomaly detection problem.

A few possible context functions are shown in Figures 3.4 and 3.5.

3.7 The reference filter F_R

Once a set of candidate anomalies has been produced and a context has been associated with each candidate anomaly, all that remains is to assign anomaly scores to each candidate anomaly (and then combine the results of this computation into a solution). However, a candidate anomaly and its context are likely to be of different sizes, which can be inconvenient, since most anomaly measures only work on elements of the same size. To get around this, the context can be split into chunks of the same size as the candidate anomaly.

To accommodate for this step, the framework includes a *reference filter* $F_R : [D'] \to [[D']]$, which (analogously to the evaluation filter) maps a context to a *reference set* of items with which the candidate anomaly can be effectively compared. A possible reference filter for sequences is illustrated in Figure 3.4. If the anomaly measure can operate on elements of different sizes, then $F_R(X) = [X]$ might be a good choice.

[3] It should be noted that the term 'novelty detection' is occasionally used in the literature to refer to what is referred to as semi-supervised anomaly detection in this report.

3.8 The anomaly measure M

The anomaly measure $M : ([D'], [[D']]) \to A$ takes a candidate anomaly and a reference set and produces an anomaly score in some set A. Realistically, letting $A = \mathbb{R}+$ ought to be sufficient in most cases. The anomaly measure essentially defines (often in unpredictable ways) what types of features will be considered anomalous, so it is vital that it is chosen appropriately.

A large number of different anomaly measures are used in the literature, and listing them all is not possible within the scope of this report. Methods are often categorised based on what type of anomaly measure they employ; for instance, a method might be classified as statistical, classifier-based, distance-based, or information-theoretic based on the anomaly measure. A good presentation of these categories is given in [3].

3.9 The aggregation function Σ

The aggregation function $\Sigma : ([[D']], [A]) \to [S']$ takes a list of candidate anomalies and a list of corresponding anomaly scores, and produces a "preliminary" solution in $[S']$. As mentioned previously, this solution is preliminary in the sense that it corresponds to a solution to a problem $P' : [D'] \to [S']$. If a problem does not involve any transformation, i.e. if $D' = D$ and T_D is the identity transformation, then T_S must also be the identity transformation, and Σ will produce the actual solution.

How this is done depends on the nature of both $\mathcal{S}$ and the specific problem formulation. If the solution is expected to be a list of anomalous subsets, then Σ ought to simply select a subset of the candidate anomalies, possibly along with the corresponding A_j. The same goes if the evaluation set contains only single element lists (i.e. if only point anomalies are sought).

If, on the other hand, the evaluation set contains non-singleton subsets of the input data, and the sought solution format is not a list of anomalous subsets, then the anomaly scores A must be weighted together to form a list in $[S']$. Since all other solution formats can be retrieved from a list of anomaly scores, it seems reasonable to in these cases only consider aggregation functions which produce a list of anomaly scores.

Reasonably, such an aggregation function ought to be on the form

$$\Sigma([e_1, e_2, \ldots, e_n], [a_1, a_2, \ldots, a_n]) = [\sigma(\{a_i : d'_1 \in e_i\}), \sigma(\{a_i : d'_2 \in e_i\}), \ldots, \sigma(\{a_i : d'_n \in e_i\})],$$

where σ is some function that maps a set of elements in A to an element of S'.

3.10 Constructing an oracle

Assuming that each of the functions in some problem formulation $P = (T_D, F_E, C, F_R, M, \Sigma, T_S)$ can be computed, the construction of an oracle O that computes the solution to P

is an easy task. Specifically, the following algorithm solves P:

Require: Some $X \in [D]$.
  $X' \leftarrow T_D(X)$
  $A \leftarrow []$ ▷ initialize anomaly scores to empty list
  for $E \in F_E(X')$ **do** ▷ iterate over the evaluation set
    $R_E \leftarrow F_R(C(X', E))$ ▷ compute a reference set
    $append(A, M(E, R_E))$ ▷ compute and store anomaly scores
  end for
  return $F_S(\Sigma(F_E(X'), A))$ ▷ aggregate scores to form anomaly vector

The filters F_E and F_R, the context function C and the aggregation function Σ can all be expected to be simple to compute, since they really just involve selecting subsets of or aggregating their input data; given a description of any of these functions, producing an algorithm that computes it is trivial.

On the other hand, the transformations T_D and T_S, as well as the anomaly measure M can in principle be arbitrarily difficult to compute. Consequently, special attention must be paid to ensure that these functions are dealt with properly.

3.11 Constructing a problem set

Using the framework presented in this chapter, the construction of an appropriate problem set for some specific application can be approached in a systematic manner.

First, the format of the input and solution elements D and S must be formally defined. The set of all possible problems then corresponds to the set of functions $P : [D] \to [S]$. Assuming that all interesting problems can be decomposed as a combination of the components T_D, F_E, C, F_R, M, Σ, and T_S, an appropriate restricted problem set can be constructed by individually restricting the components.

As has been previously pointed out, what restrictions should reasonably be placed on the filters and context function F_E, F_R and C depends entirely on the presence of structure in the input data $D \in \mathcal{D}$ which relates the $d_i \in D$. If the d_i are completely unrelated, then it is reasonable to bypass these components by restricting them to only consider point anomalies. If, on the other hand, there is a relation between the $d_i \in D$, then it is reasonable to restrict these components to take this structure into account.

The aggregation function Σ is similarly limited. For problems involving point anomalies, all of the solution formats presented above can be produced directly from the output of the anomaly measure, so Σ is naturally constrained. For problems involving collective anomalies, anomaly scores for several candidate anomalies might have to be weighted together to produce a solution. Depending on the application, it might be appropriate to constrain the possible Σ to a single weighting method or to a set of different weighting methods.

Restricting T_D and T_S is more difficult, since it is typically not known what types of transformations are appropriate for a given application. With this in mind, it might be appropriate to simply restrict T_D and T_S to some set of transformations

which have been used in the specific application or similar applications. That way, the problem set can be said to generalise previously studied methods while remaining tractable.

Finally, the anomaly measure M must be appropriately restricted. Since M can not be appropriately restricted based on information about the application, it is reasonably to take a similar approach as with $R_{\mathcal{D}}$ and restrict M to some set of anomaly measures which have been previously studied in conjunction with the specific application.

While restricting the components individually simplifies problem set construction, it might lead to a problem set that is overly permissive, in the sense that some combinations of components might not be useful. In this case, it is reasonable to place additional restrictions on the problem set based on combinations of components.

3.12 Evaluation

TODO: move Section 4.4 here.

3.12.1 Training data

TODO: move Section 4.4.1 here.

3.12.2 Error measures

TODO: move Section 4.4.2 here.

Chapter 4

An application to sequences

In this chapter, the framework presented in the previous chapter is applied to anomaly detection in sequences. As a reminder, the optimisation problem can be stated as

$$P^*_{opt} = \underset{P \in \mathcal{P}^*}{\operatorname{argmin}} \sum_{T_i \in \mathcal{T}} \delta(s(T_i), O^*(P, T_i)),$$

where the objects that need to defined are the problem set $\mathcal{P}^*$, the test data T, the error function ϵ^*, and the oracle O. The framework presented in the previous chapter makes the assumption that all relevant problems P can be decomposed into a set of functions $P = (T_{\mathcal{D}}, F_E, C, F_R, M, \Sigma, T_{\mathcal{S}})$.

In Section **??**, some terminology related to sequences and time series is presented.

Section 4.2 contains a survey of previous research on anomaly detection in sequences. The components $T_{\mathcal{D}}$, F_E, C, F_R, M, Σ, and $T_{\mathcal{S}}$ are studied individually.

In Section **??**, appropriate problem sets and oracles are presented for the two major tasks encountered in anomaly detection in seqeuences.

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 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 $\mathbf{s} = (s_1, s_2, \dots)$, where $\forall i : s_i \in X$ for some set X. Furthermore, a *time series* is taken to be any sequence $\mathbf{t} = ((s_1, t_1), (s_2, t_2), \dots)$, where $\forall i : (s_i, t_i) \in X \times \mathbb{R}^+$ for some set X and $\forall i, j : i > j \rightarrow t_i \geq t_j$. 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 $\mathbf{t} = (s_1, s_2, \dots)$. Henceforth all time series will be taken to be regular unless explicitly stated[1].

Two main tasks can be distinguished in anomaly detection in sequences; *finding anomalous sequences in a set of sequences*, and *finding anomalous subsequences in a long sequence* [3]. The former task can be formulated as the detection of point anomalies in an unstructured set of sequences. The latter corresponds to finding contextual anomalies in a set equipped with a total ordering.

4.2 Previous research

TODO: follow the organization of the previous chapter here

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 [**?**].

4.2.1 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

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

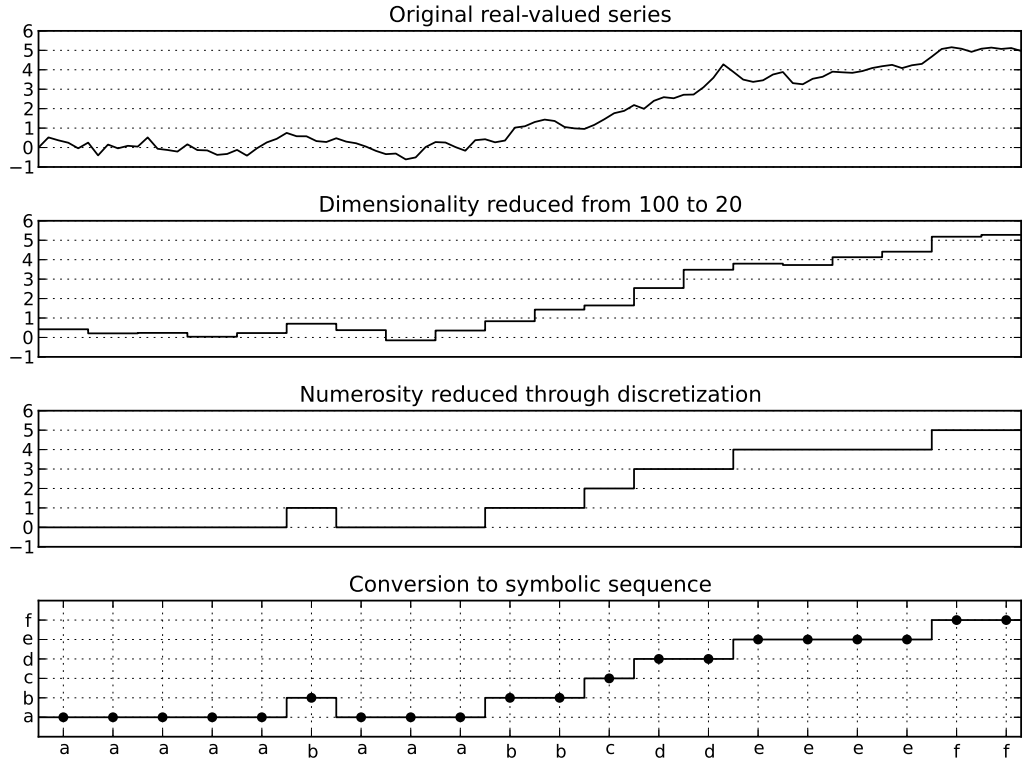


Figure 4.1. Illustration of numerosity and dimensionality reduction in a conversion of a real-valued sequence to a symbolic sequence. The top frame shows a real-valued time series sampled from a random walk. The second frame shows the resulting series after a (piecewise constant) dimensionality reduction 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 $\{a, b, c, d, e, f\}$.

extraction can be rephrased as follows: Given a sequence $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ where $s_i \in \mathbb{R}$, find a set of basis functions $\{\phi_1, \phi_2, \ldots, \phi_m\}$ where $m < n$ that $\mathbf{s}$ can be projected onto, such that $\mathbf{s}$ 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 $(\tau_1, \tau_2, \dots \tau_n)$ 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] 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.

$\mathbf{S_1}$	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].

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 $((t_1, x_1), (t_2, x_2), \dots, (t_n, x_n))$ into some sequence $(s_1, s_2, \dots, s_m)$.

The simplest such transformation involves simply dropping the t_i to form the sequence $(x_1, x_2, \dots, x_n)$. 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.

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 $\mathbf{t} = ((s_1, t_1), (s_2, t_2), \dots)$ where $\forall i : (s_i, t_i) \in X \times \mathbb{R}$, 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 $\mathbf{s'} = ((s_1', \tau_1), (s_2', \tau_2), \dots)$ 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 $\mathbf{s}'$ 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 [**?**] may be employed.

4.2.2 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 $\mathbf{s} = (s_1, s_2, \ldots, s_n)$).

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 2.4.

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 $\mathbf{s} = \{s_a, s_b, \ldots, s_m\} = \{(s_1^a, s_2^a, \ldots, s_{n_a}^a), (s_1^b, s_2^b, \ldots, s_{n_b}^b), \ldots, (s_1^m, s_2^m, \ldots, s_{n_m}^m)\}$, 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).

4.2.3 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 2.1.2.

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.

4.2.4 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 **??**.

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 **??** is difficult. However, in the case of Task **??**, 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 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).

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

4.2.5 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.

4.3 Optimisation problem

The construction of appropriate problem sets for the tasks of detecting anomalous sequences and subsequences, as well as corresponding oracles, is now discussed.

First, in Section 4.3.1, a few reasonable restrictions on the problem sets, as well a suggested set of components useful in parametrising the problem sets, are introduced.

Next, in Sections 4.3.2 and 4.3.3, the components are formally defined, and oracles which operate on them are presented.

Finally, in Section 4.3.4, a few component choices are presented along with the parameters they take.

4.3.1 Problem set

As mentioned in the section 2.3, the problem set must be limited in order for it to admit an oracle. The goal of this section is to present a problem set that is general enough to admit most of the problems while remaining limited enough to be useful in practice.

Based on the previous discussion, a few sensible limitations can be derived. Since other solution formats can be constructed from anomaly scores, anomaly scores are a reasonable solution format. Furthermore, since contextual collective anomalies generalise all other mentioned anomaly types (point anomalies, contextual anomalies, and collective anomalies), as well as the most commonly used forms of supervision (semi-supervised and unsupervised), limiting the problem set to this anomaly type is reasonable. While further limiting the problem set is possible, in the interest of generality, this is not done here.

In order for a problem set to be tractable, it needs to be parametrisable. To this end, it is useful to, if possible, try to define the set in terms of a few components which can be chosen independently (and then parametrised and optimised individually).

Since which factors remain to be specified depends on the task being studied, we will treat the tasks of detecting anomalous subsequences and detecting anomalous sequences in a set of sequences separately, beginning with the former.

For the task of detecting anomalous subsequences, we propose that the following set of components be used:

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

The filters. Filters for extracting sequences from the input data and context must be specified.

The anomaly measure. The measure by which extracted sequences are judged as anomalous or normal must be specified.

Transformations. Which transformations (including discretisation, numerosity reduction and dimensionality reduction transformations), if any, are to be applied to the extracted data items before the anomaly measure is applied.

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

Finding anomalous sequences in a set of sequences is really a (contextual) point anomaly detection; it corresponds to finding anomalous sequences (points) in a set of sequences, either with regard to the rest of that set (unsupervised context) or to some other set of sequences (semi-supervised context). As such, there is no need for either filters or an aggregation method, and the task can be specified using the following components:

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

The anomaly measure. The measure by which extracted sequences are judged as anomalous or normal must be specified.

Transformations. Which transformations (including discretisation, numerosity reduction and dimensionality reduction transformations), if any, are to be applied to the individual vectors.

While searching over all possible choices of any of these components is still not feasible, this is not really problematic since a vast majority of the possible choices can be expected to perform poorly. Instead, a constructive approach can be taken in which the problem set is built up from component choices which have already been shown to be effective. Indeed, searching over most or all of the previously studied choices for each individual component is likely to be feasible. If the components can chosen and optimised somewhat independently, the development of efficient optimisation heuristics should be possible.

Taking a constructive approach has the additional advantage that proposed new components (such as novel anomaly measures) can be efficiently compared to existing methods. Another interesting advantage is that the approach could facilitate research into possible connections between characteristics of the input datasets and appropriate choices of components. 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 dataset analysis could be used to seed the optimisation. As an example of such a characteristic, in [5], various anomaly measures compared on the task of finding anomalous real-valued sequences in a set of similar sequences, and it is concluded that different anomaly measures are appropriate depending on the periodicity of the sequences.

4.3.2 An oracle for anomalous subsequence problems

Before the oracle can be specified, the dataset and solution formats, as well as the unspecified components specified above, must be formally defined.

For the task of detecting anomalous subsequences in a long sequence, the input dataset consists of a sequence $\mathbf{x} \in X^n$, for some set X. The corresponding anomaly scores are a vector $\mathbf{a} \in \mathbb{R}^n$.

The components can then be defined as (where $S(\mathbf{x})$ is the set of indexed subsequences of $\mathbf{x}$, e.g. $S((x_1, x_2)) = \{((x_1), (1)), ((x_2), (2)), ((x_1, x_2), (1, 2))\}$):

The evaluation filter F_E maps a sequence $\mathbf{x}$ to a subset of $S(\mathbf{x})$ to be evaluated.

The context function C maps a sequence $\mathbf{x}$ and a vector of indices $\mathbf{i} \in \mathbb{Z}_n^m$ (representing a subsequence) to a subset of $S(\mathbf{x})$ (representing the context of $\mathbf{i}$).

The training filter F_T**,** like the evaluation filter, is a function that takes a sequence $\mathbf{x}$ (from the output of C) to some subset of $S(\mathbf{x})$.

The transformation T maps a sequence $\mathbf{x}$ to some other sequence $\mathbf{y} \in Y^l$ for some set Y and some $l \in \mathbb{N}$.

The anomaly measure M maps a sequence $\mathbf{y}$ and a set of sequences $\{\mathbf{y}_i\}_{i \in \mathbb{Z}_n}$ to an anomaly score in $\mathbb{R}$.

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

Given these definitions, the oracle can be formulated as follows:

Require: A sequence $\mathbf{x}$ and a tuple (F_E, C, F_T, T, M, A).
  $S \leftarrow \emptyset$ ▷ initialize anomaly score container
  for $(\mathbf{x}_i, \mathbf{i}_i) \in F_E(\mathbf{x})$ **do** ▷ iterate over subsequences (and indexes) selected by filter
    $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_j\} \leftarrow C(\mathbf{i}_i)$ ▷ compute context
    $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k\} \leftarrow \bigcup_{\mathbf{c} \in \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_j\}} F_T(\mathbf{c})$ ▷ extract training set from context
    $S \leftarrow S \cup (M(T(\mathbf{x}_i), \{T(\mathbf{t}_1), T(\mathbf{t}_2), \ldots, T(\mathbf{t}_k)\}), \mathbf{i}_i)$ ▷ save transformed anomaly score with indexes
  end for
  return $A(S)$ ▷ aggregate scores to form anomaly vector

The corresponding optimisation problem is to find

$$\underset{(F_E,C,F_T,T,M,A)}{\operatorname{argmin}} \sum_i \epsilon^*(\mathbf{s}_i, O(\mathbf{x}_i, (F_E, C, F_T, T, M, A))),$$

where $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \ldots, (\mathbf{x}_k, \mathbf{s}_k)\}$ is some set of labeled training data.

4.3.3 An oracle for anomalous sequence problems

For the chosen task of detecting anomalous sequences in a set of sequences, the input dataset consists of a collection $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in X^n$, for some arbitrary set X (typically X is the set all vectors of some specific length over some set Y). The corresponding anomaly scores are a vector $\mathbf{a} \in \mathbb{R}^n$.

The components can then be defined as:

The context function C maps a sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in X^n$ and an index $i \in \mathbb{Z}_n$ (representing one of the sequences) to a subset of $\{\mathbf{x}_j\}_{j \in \mathbb{Z}_n} \backslash \mathbf{x}_i$ (representing the context of $\mathbf{x}_k$).

The transformation T maps a sequence $\mathbf{x}$ and to some other sequence $\mathbf{y} \in Y^o$ for some set Y and some $o \in \mathbb{N}$.

The anomaly measure M maps a sequence $\mathbf{y}$ and a set of sequences $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ (representing the context) to an anomaly score in $\mathbb{R}$.

Given these components, a simple oracle can be formulated as follows:

Require: A vector of sequences $\mathcal{X} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m})$ and a tuple (C, M).
  for $\mathbf{x}_i \in S(\mathcal{X})$ **do** ▷ iterate over sequences
    $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_j\} \leftarrow C(\mathcal{X}, i)$ ▷ compute context
    $a_i \leftarrow M(T(\mathbf{x}_i), \{T(\mathbf{c}_1), T(\mathbf{c}_2), \ldots, T(\mathbf{c}_j)\})$ ▷ save anomaly score
  end for
  return $(a_1, a_2, \ldots, a_n)$ ▷ aggregate scores to form anomaly vector

The corresponding optimisation problem is to find

$$\underset{(C,T,M)}{\operatorname{argmin}} \sum_i \epsilon^*(\mathcal{S}_i, O(\mathcal{X}_i, (C, T, M))),$$

where $\{(\mathcal{X}_1, \mathcal{S}_1), (\mathcal{X}_2, \mathcal{S}_2), \ldots, (\mathcal{X}_k, \mathcal{S}_k)\}$ is some set of labeled training data.

4.3.4 Components

Possible choices of the individual components above are now discussed. Before the optimisation problem can be implemented, a set of possible parameters must be defined for each component. With this in mind, the component choices discussed in this section include parameter descriptions.

The filters

The evaluation filter takes a sequence and extracts subsequences from it to be used in the analysis. In the literature, *sliding window* approaches, as described in section 4.2 are almost always taken.

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, there is typically little reason to use any filter other than a sliding window filter as the reference filter. If the anomaly measure does not require items of equal length, the reference filter can be ignored, i.e. $F_R(T) = T$.

Sliding windows are parametrised by the *step length* $s \in \mathbb{N}$ and *window width* $w \in \mathbb{N}$; i.e. any sliding window filter will have the form

$$F(\mathbf{x}) = \{(x_1, x_2, \ldots, x_w), (x_{s+1}, x_{s+2}, \ldots, x_{s+w}), \ldots, (x_{n-w}, x_{n-w+1}, \ldots, x_n)\}.$$

The context function

For the task of finding anomalous subsequences, all of the context functions described in section 4.2 are potentially interesting.

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.

For the task of detecting anomalous sequences, contexts other than the semi-supervised and unsupervised contexts are not likely to be interesting.

Different contexts admit different numbers of parameters. The trivial context and the novelty context are both parameter-free, while the local contexts admit two integral parameters.

The transformation

Anomaly measures can be combined with one or more transformations of the data extracted by the filters, in order speed up or otherwise improve the analysis.

One commonly used transformation for real-valued data is the Z-normalization transform, which modifies a sequence to exhibit zero empirical mean and unit variance.[3]

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. The DFT is parameter-free, while the DWT can be said to be parametrised due to the variety of possible wavelet transforms.

[3]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 sequences that are anomalous because they are displaced or scaled relative to other sequences.

Furthermore, transformations for real-valued data which produce symbolic sequences are very important, since they enable the application of symbolic approaches to real-valued sequences. While several such transformations exist, *symbolic aggregate approximation* (SAX) [39] is by far the most commonly used. This transformation takes two integral parameters which control the degrees of numerosity and dimensionality reduction, respectively.

The anomaly measure

As mentioned previously, the anomaly measure is likely the most important component. It is also the component with the largest number of interesting choices. Properly defining the required parameters for all of the anomaly measures discussed in Section 4.2 is not possible within the scope of this report, so we only discuss the most interesting anomaly measures here as indicated in [5].

Distance-based anomaly measures, and especially kNN-based anomaly measures are among these. Essentially, the kNN anomaly measure takes two parameters: a distance measure (defined on the specific type of sequences under consideration) δ, and a k-value $k \in \mathbb{N}$, and given a sequence $\mathbf{x}$ and a set of context sequences $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, computes the anomaly score by taking the average of the k smallest $\delta(\mathbf{x}, \mathbf{x}_i)$.

Thus, the kNN anomaly measure has the two parameters k and δ, where δ may (for instance) be any of the distance measures discussed in Section 4.2. Note that the distance measure, in turn, may be parametrised (for instance, the Minkowski measure has an order parameter $p \in \mathbb{R}^+$).

TODO: discuss parameters for distance measures?

Classifier-based anomaly measures are also interesting. A support vector machine-based anomaly measure is shown to perform especially well in [5]. Support vector machine-based anomaly measures take a few parameters: a kernel, eventual kernel parameters, and a soft margin parameter C. For a more thorough discussion of support vector machines and their parameters, see [**?**].

Note that further anomaly measures can be constructed by chopping up the input sequences $\mathbf{x}$ and $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ into smaller sequences using filters before applying the distance measure, and then aggregating the result into an anomaly score using some aggregation function. This is done for the support vector machine anomaly measure in **??**.

TODO: maybe mention a few other anomaly measures

The aggregation method

Once the extracted subsequences have been assigned anomaly scores, they must be aggregated into an anomaly vector. We here suggest a few aggregation functions, on the form

$$A(\{(\mathbf{a}_1, \mathbf{i}_1), (\mathbf{a}_2, \mathbf{i}_2), \ldots, (\mathbf{a}_n, \mathbf{i}_n)\}) = (f(\{\mathbf{a}_i : 1 \in \mathbf{i}_i\}), f(\{\mathbf{a}_i : 2 \in \mathbf{i}_i\}), \ldots, f(\{\mathbf{a}_i : n \in \mathbf{i}_i\})),$$

where I_i are intervals, a_i are assigned anomaly scores, and f is some aggregation method-specific function that produced a real-valued aggregate score from a set of real-valued element-wise scores. The *maximum*, *minimum*, *median* and *mean* of the values in S all constitute reasonable choices of $f(S)$. Aggregation functions using any of these four functions will be referred to as maximum, minimum, median, and mean aggregators, respectively. Each of these is parameter-free.

4.4 Evaluation

Two aspects of the optimisation problem which have have not yet been discussed are the training data and error measure. Since these aspects essentially define the objective function used to guide the optimisation, it is important that they are chosen appropriately. In this section, they are discussed in detail.

4.4.1 Training data

As has been previously mentioned, while the optimisation should ideally operate on an expected error over some stream of data, this is typically infeasible. In practice, the optimisation must be performed over some pre-selected set of training data. If the optimisation is to be truly useful, this training data must also somehow be *labeled* (otherwise, the optimisation would hinge on some built-in assumption of what constitutes an anomaly and what does not, which is problematic concerning the subjective nature of anomaly detection in applications).

For the task of finding anomalous subsequences, this means that the training data should ideally consist of a set of sequences and corresponding label vectors $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \ldots, (\mathbf{x}_k, \mathbf{s}_k)\}$, where $\mathbf{x}_i \in X^{n_i}$, $\mathbf{s}_i \in \mathbb{R}^{n_i}$.

However, obtaining real-valued anomaly scores that are meaningful objective is typically not possible (again, due to the subjective nature of the subject). A more realistic scenario would be for the $\mathbf{s}_i$ to instead be binary vectors, i.e. $\mathbf{s}_i \in \{0, 1\}^n$. Such vectors can typically readily be constructed for any dataset by a domain expert.

If labeled training data can not be provided, an interesting alternative is to superimpose artificial anomalies onto a set of unlabeled training sequences. With this approach, the objective function measures how well problem formulations discern regular data from the application from similar data that contains (specific, artificial) anomalies. Of course, unlike with true labeled training data, there is no guarantee that the superimposed anomalies are relevant to the application, so the optimisation might still not lead to an accurate problem formulation.

4.4.2 Error measures

As mentioned in section 2.3, an error measure must be defined, which given a sequence and an anomaly vector judges how accurately the anomaly vector captures anomalies in the sequence. Ideally, this error measure should consist of an objective and infallible domain expert, who carefully judges each sequence $\mathbf{x}_i$ and anomaly

vector $\mathbf{a}_i \in \mathbb{R}^n$. Failing this, the next best thing would be to have a reference anomaly vector $\mathbf{r}_i$, with which $\mathbf{a}_i$ can be compared.

As mentioned above, realistically such reference anomaly vectors would have to be binary. Since we have made the assumption that problems have real-valued vectors as solutions, our error measures thus have to operate on one binary and one real-valued vector, i.e. they must be functions $\epsilon : \mathbb{R}^n \times \{0,1\}^n \to \mathbb{R}^+$. To our knowledge, such error methods have not been previously studied, so we now suggest a few such error measures.

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

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

which ought to constitute a reasonable choice of error measure. However, since ϵ_E is as sensitive to the accuracy of $\mathbf{a}$ on normal elements as on anomalous elements, comparatively high or low values of $\mathbf{a}$ for non-anomalous elements may distort this error measure.

This pitfall can be avoided by converting $\mathbf{a}$ to a binary string $\mathbf{a}_B \in \{0,1\}^n$ as well, and using a binary distance between $\mathbf{a}_B$ and $\mathbf{r}$ as the error measure. Since this is equivalent to selecting a set $S_\mathbf{a}$ of indexes of elements from $\mathbf{a}$ and comparing it to the set $S_\mathbf{r}$ of indexes of nonzero elements of $\mathbf{r}$, this can be seen as converting the anomaly score to a set of anomalous elements, and comparing this set to a reference set.

What remains is to decide how $\mathbf{a}_B$ should be constructed from $\mathbf{a}$ and what binary distance measure to use. If all elements in the sequence are to be assigned equal importance, the natural choice of distance measure is the *Hamming distance* δ_H [**?**]. Similarly, for $\mathbf{a}_B$ it ought to hold that $\forall i, j \leq n : a_i < a_j \wedge i \in S_A \Rightarrow j \in S_\mathbf{a}$, which is equivalent to setting a threshold $a_{\min} \leq \tau \leq a_{\max}$ and letting $\mathbf{a}_B$ be given by:

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

This gives rise to a number of possible error measures, based on how the threshold value τ is set. We define the following measures:

The equal support error ϵ_{ES}, which corresponds to setting τ such that $|S_\mathbf{a}| = |S_\mathbf{r}|$.

The full support error ϵ_{FS}, which corresponds to using the largest τ for which $\forall i : i \in S_\mathbf{r} \Rightarrow i \in S_\mathbf{a}$.

The optimal support error ϵ_{BS}, which corresponds to using the τ that gives the smallest error value.

Since the error measures discussed above have not previously been studied in the context of series anomaly detection, an empirical evaluation of their performance is performed in Section 5.4.

4.5 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 4.5.1, 4.5.2, and 4.5.3, and some of the design choices made in the development of `ad-eval` are discussed in Section 4.5.4.

4.5.1 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 **??**. 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 $F_E(X) = X$ 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.

4.5.2 Evaluation utilities

The set of possible interesting tests that could be run on problems derived from Task **??** 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 4.4.1, 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 4.4.2.
- 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.

4.5.3 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`.

4.5.4 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.

Chapter 5

Results

Due to the lack of appropriate evaluation data mentioned in Chapter 4.4, 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 **??**, 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 4.4, 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.

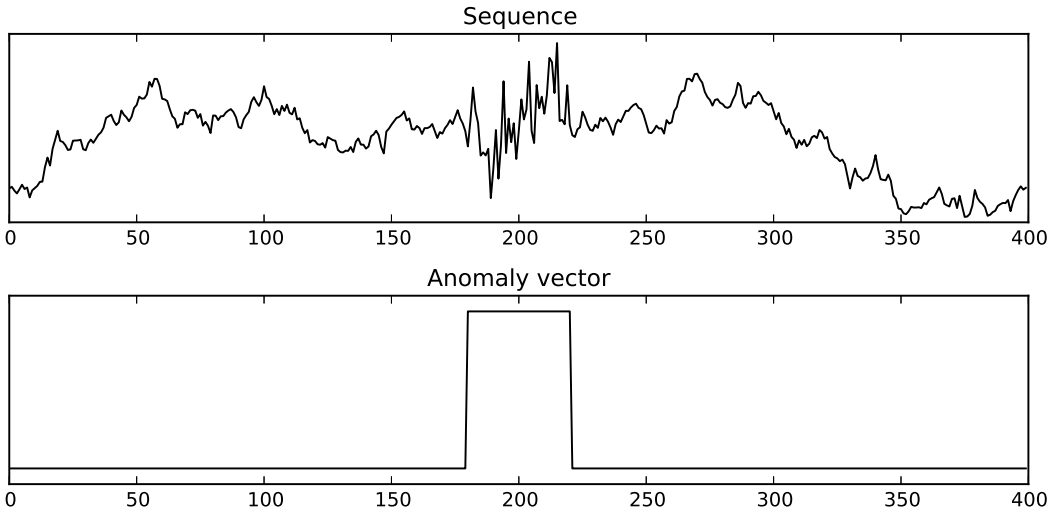


Figure 5.1. The standard sequence with corresponding reference anomaly vector.

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 $\operatorname{argmin}_\Theta E(\Theta, S)$ 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 straightforward. 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 **??**. 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

In Section 4.4.2, 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 **??**. A few of the $A_{k,w}$ given the lowest values by each of the error measures were also graphed (figure 5.2).

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.

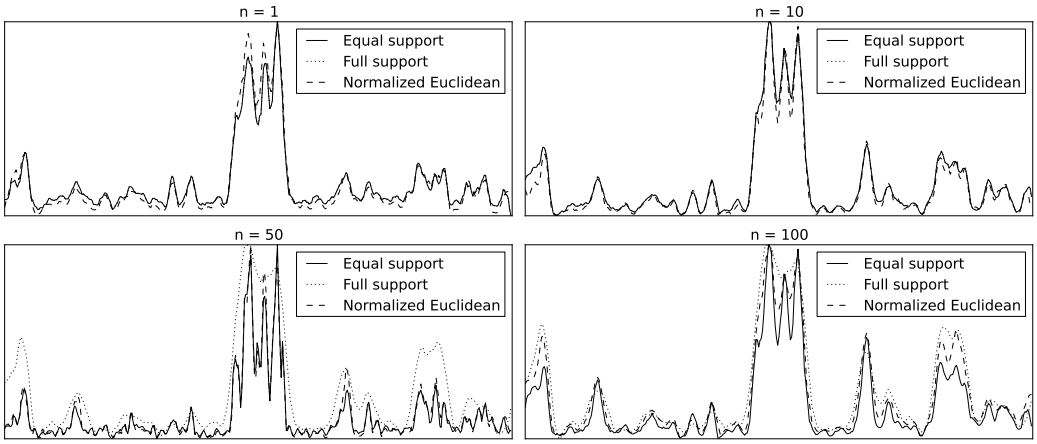


Figure 5.2. The nth best $A_{k,w}$ according to the three error measures for $n = 1, 10, 50$ and 100.

Figure 5.2 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, 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 5.3, 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

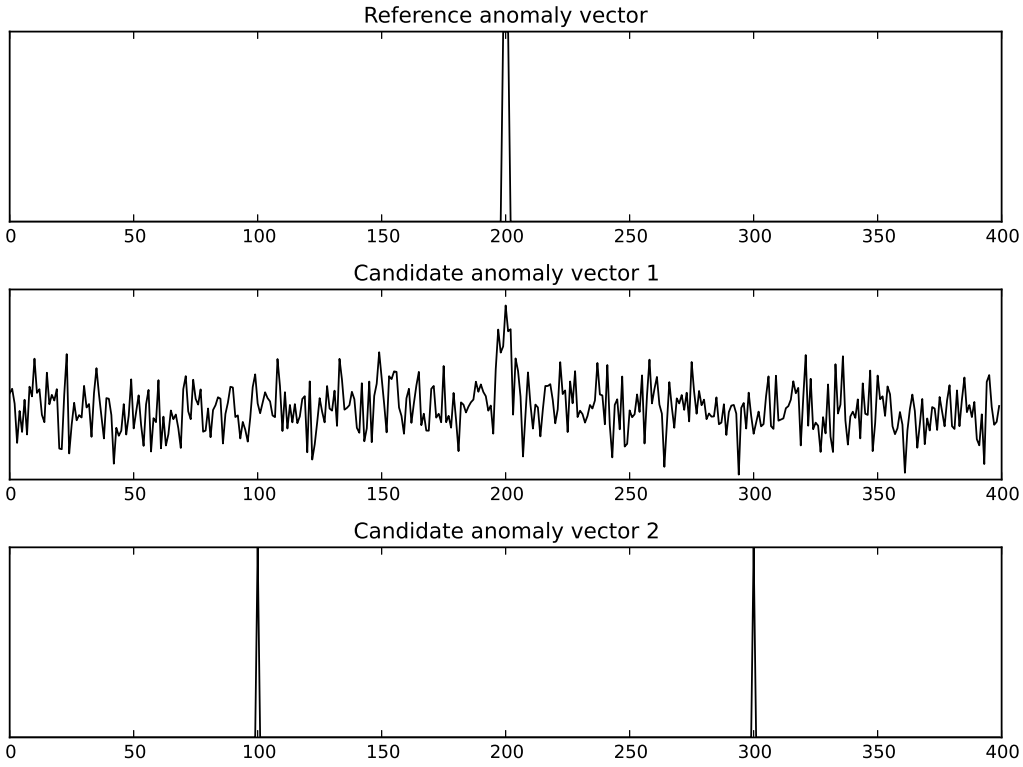


Figure 5.3. A reference anomaly vector for a long sequence and two corresponding candidate anomaly vectors. The first candidate vector, while noisy, correctly marks the anomaly. The second candidate does not mark the anomaly and marks two false anomalies. ϵ_E, ϵ_{ES} and ϵ_{EF} for the two sequences are 8.3 and 2.2; 0.010 and 0.99; and 0.0050 and 0.99, respectively.

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.

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

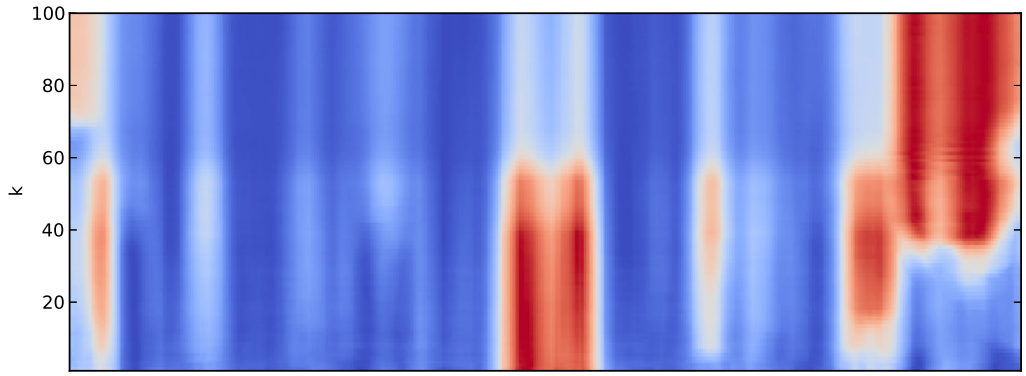


Figure 5.4. Heat map showing A_k for $k = 1, 2, \ldots, 100$. Red and blue indicate high and low anomaly scores, respectively.

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.

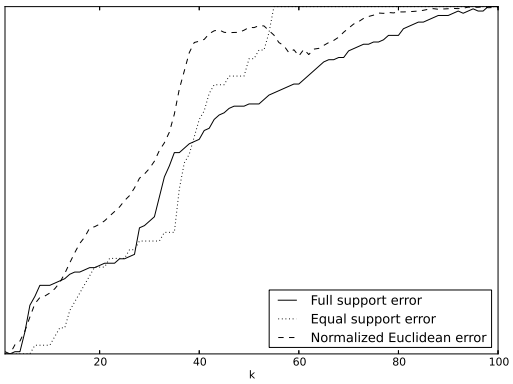


Figure 5.5. Errors as a function of k for the standard sequen

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 5.4 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 5.5. Note that this plot shows only relative errors, as the three error graphs have been individually normalized to the unit interval.

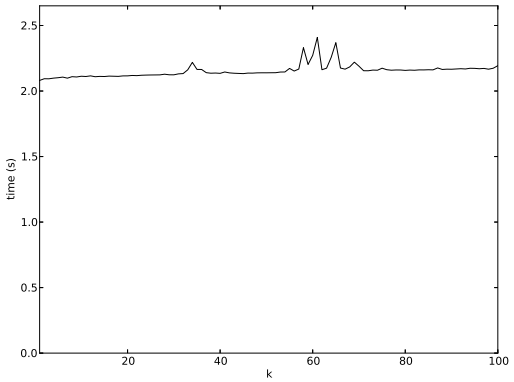


Figure 5.6. Evaluation times when varying k on the standard sequence.

Finally, Figure 5.6 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

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

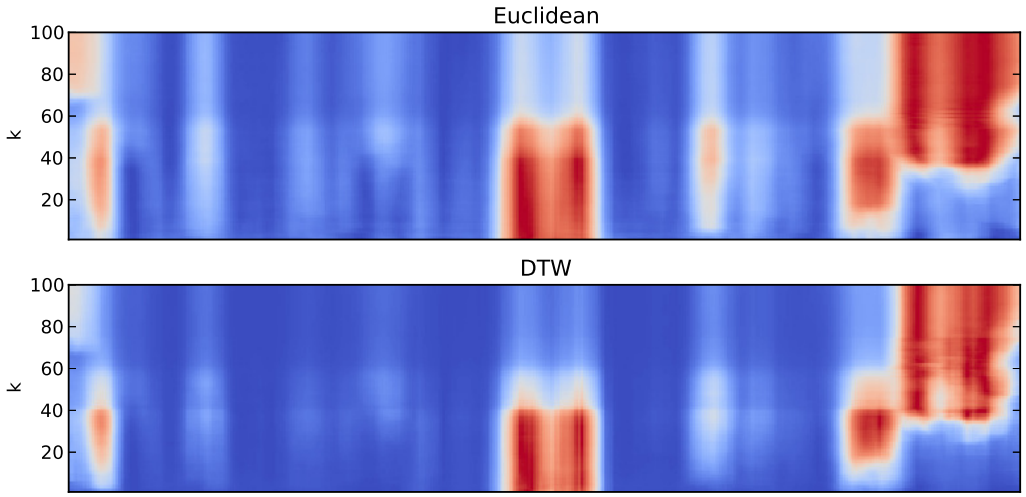


Figure 5.7. Heat maps showing $A_{k,\delta}$ for the Euclidean and DTW distances.

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 5.7 and a plot of the corresponding errors is shown in Figure 5.8. 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 encountered 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.

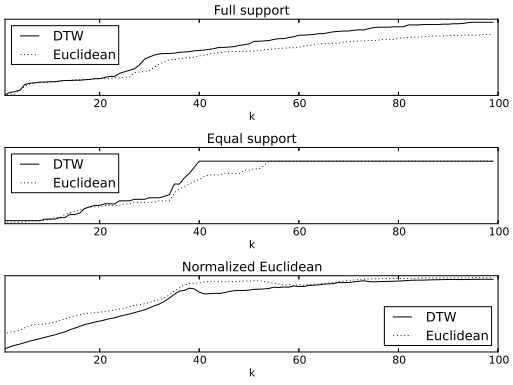


Figure 5.8. Errors for $A_{k,\delta}$.

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

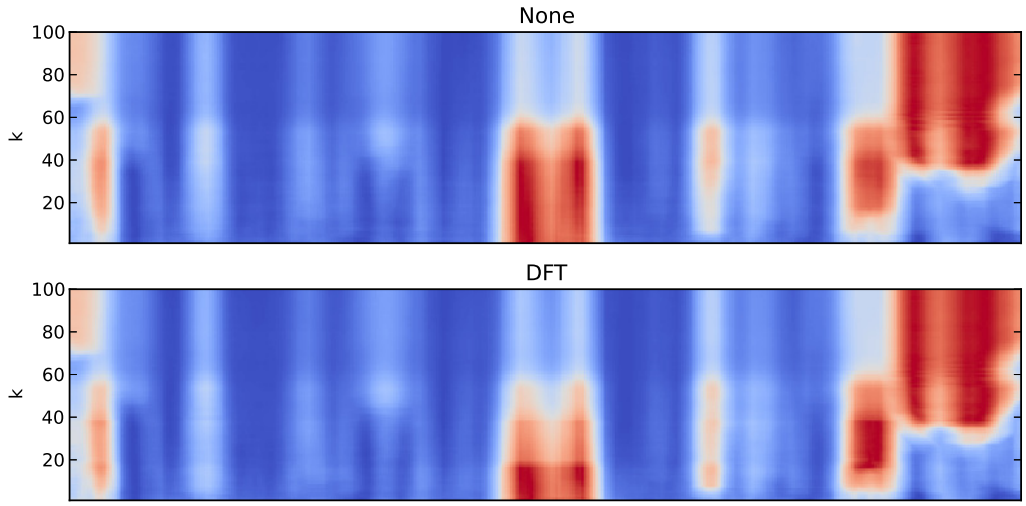


Figure 5.9. Heat maps of the $A_{k,t}$ for $k = 1, 2, \ldots, 100$ with and without the discrete Fourier transform.

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 5.9, and a plot of the corresponding errors is shown in Figure 5.10.

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.

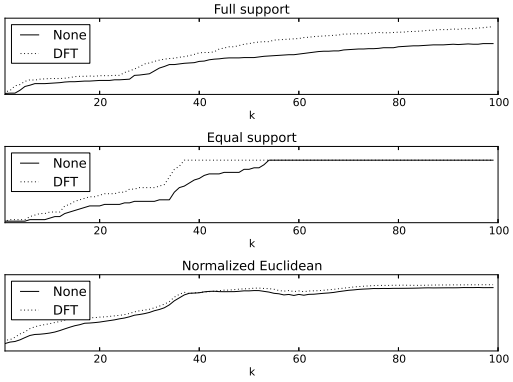


Figure 5.10. Errors of the $A_{k,t}$.

5.5.4 The sliding window width

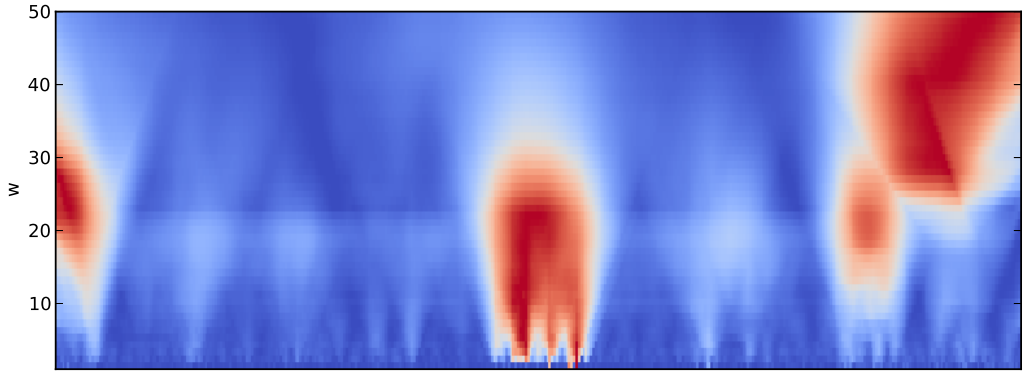


Figure 5.11. Heat map of the A_w for $w = 1, 2, \ldots, 50$.

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, ..., 50 were computed and examined. The results are shown in Figures 5.11, 5.12, and 5.13.

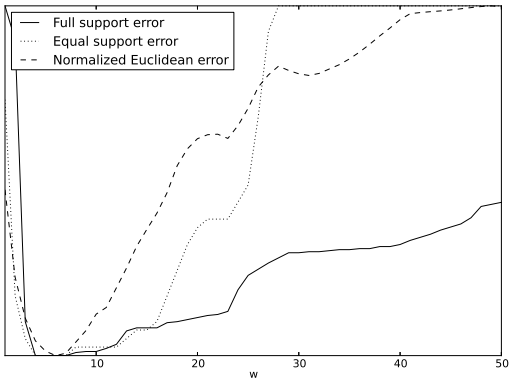


Figure 5.12. Errors for the anomaly vectors A_w.

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. Furthermore, all errors increase sharply as w nears 20, indicating that large values of w lead to inaccurate results.

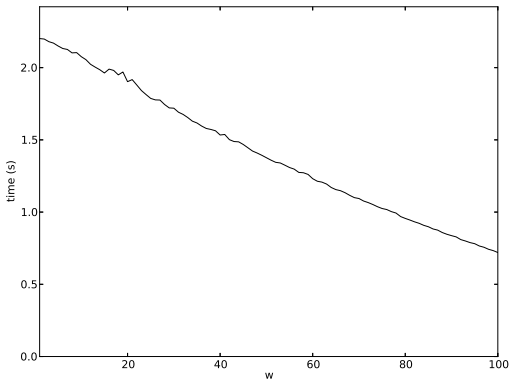


Figure 5.13. Evaluation times for the anomaly vectors A_w.

Interestingly, the plot in Figure 5.11 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 5.13 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

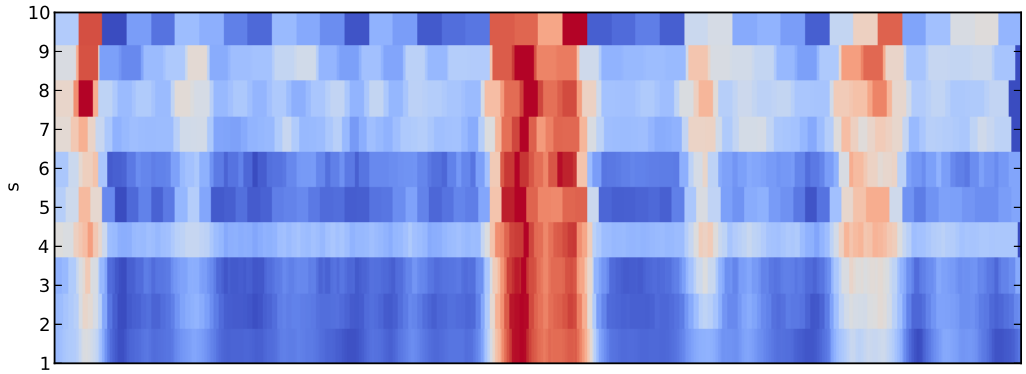


Figure 5.14. Heat map of the anomaly vectors A_s for $s = 1, 2, \ldots, 10$. Note that no major false anomalies occur for $s < 8$.

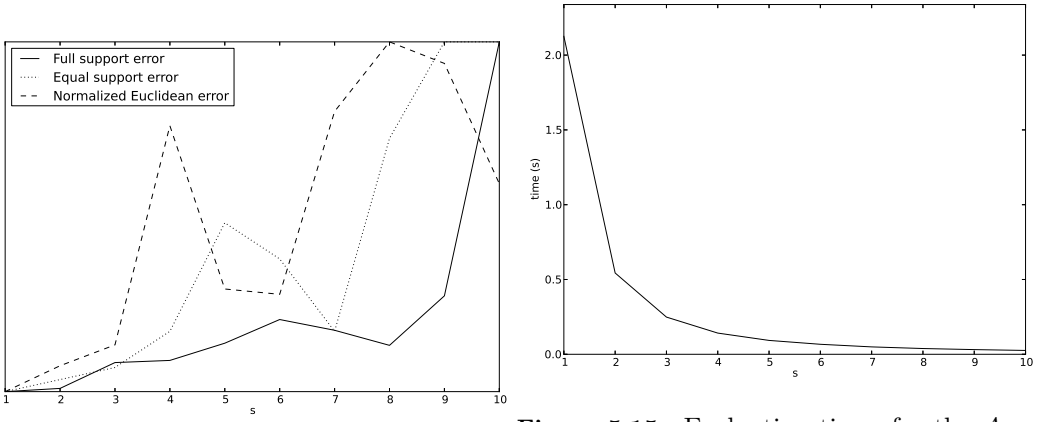


Figure 5.15. Evaluation times for the A_s. As expected, the graph that the times are $O(1/s^2)$.

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 5.14, 5.5.5, and 5.15.

As seen in Figure 5.14, 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 5.15: 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

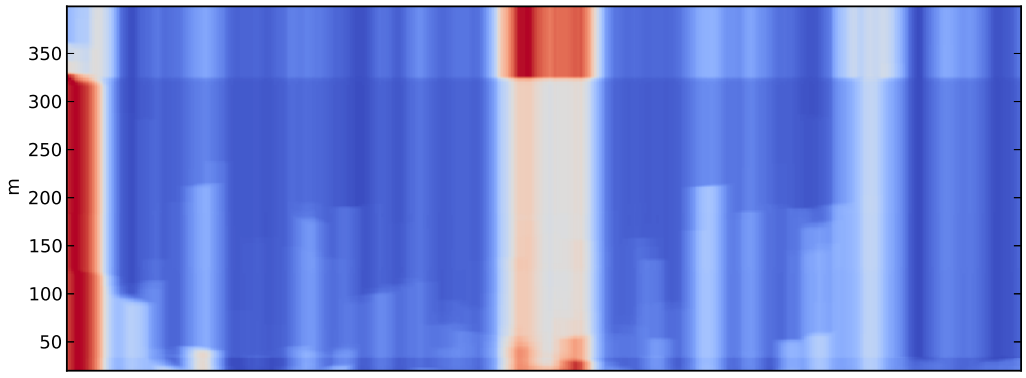


Figure 5.16. Heat map of the A_m for $m = 20, 21, \ldots, 400$. Note the false anomaly present at the left end of the anomaly vectors until $m \approx 330$.

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 2.4. 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.

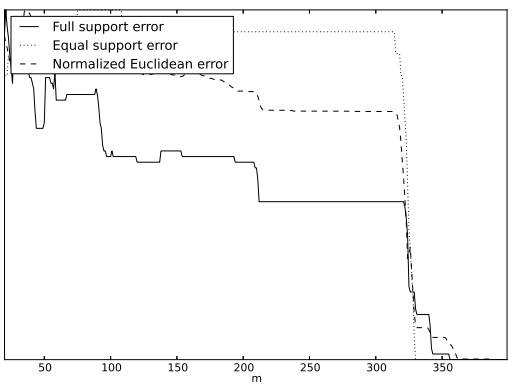


Figure 5.17. Errors of the anomaly vectors A

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 5.16, 5.17, and 5.18.

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 5.18.

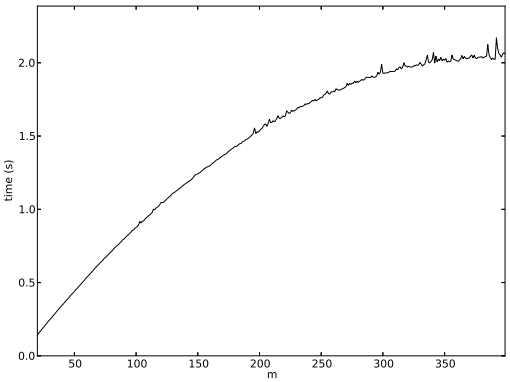


Figure 5.18. Evaluation times of the anomaly vectors A_m

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 5.19, and plots of the corresponding error measures are shown in Figure 5.20. Single anomaly vectors for $k = 1$ are shown in Figure 5.21.

As seen in Figures 5.20 and 5.21, the min and max aggregators produce blockly, piecewice constant anomaly vectors, while the mean aggregator (and, to a lesser extent, the median aggregator) produces smooth, continuous anomaly vectors.

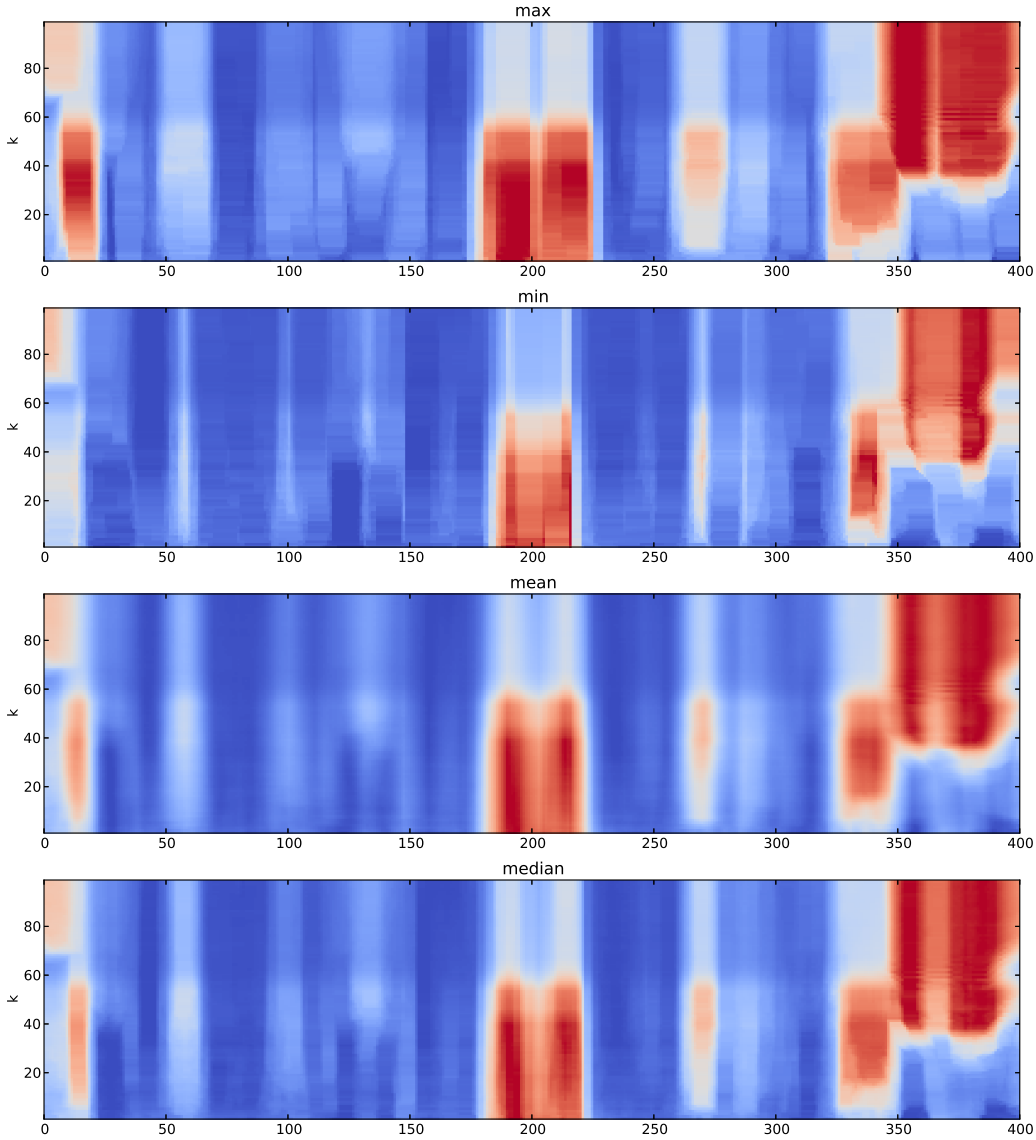


Figure 5.19. Heat maps showing $A_{k,\mathcal{A}}$ for the four aggregators.

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.

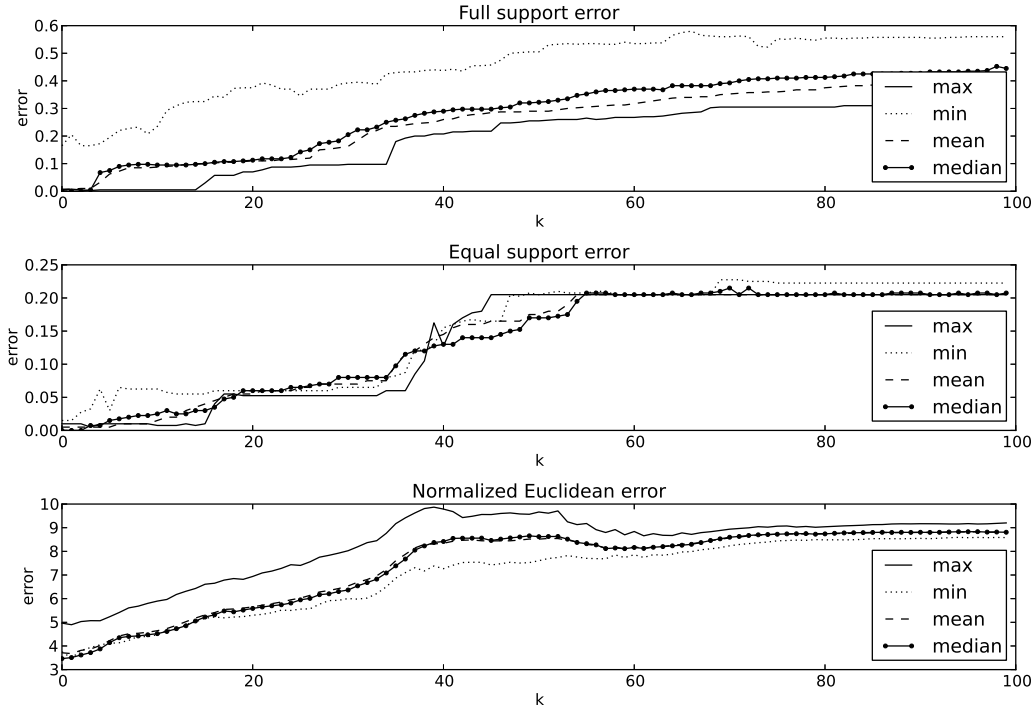


Figure 5.20. Errors of the anomaly vectors $A_{k,\mathcal{A}}$.

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

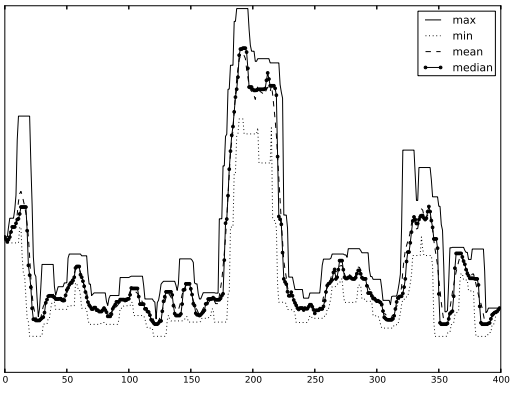


Figure 5.21. Plot of the $A_{k,\mathcal{A}}$ for $k = 1$.

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.

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.

Chapter 6

Conclusions

We conclude this report with a short summary and a discussion of a few possible directions for future work.

6.1 Summary

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 4.4 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 **??** 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 4.4.1, 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 4.4 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 **??**, the components framework could easily be adapted to cover a large number of related tasks, including the tasks suggested in Section **??**. 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 4.5.4, `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 **??**. 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

I would like to thank Splunk for giving me the inspiration and resources to complete this project; Boris Chen for his support throughout my internship at Splunk and this thesis; and Konrad Rzezniczak for his suggestions and help.

I would further like to thank Timo Koski for his help in supervising the project, as well as Chris Conley for his assistance with the proof-reading of this report.

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.