Aalto University
School of Science
Degree Programme in Life Science Techonologies

Inka Saarinen

Adaptive real-time anomaly detection for multi-dimensional streaming data

Master's Thesis
Espoo, February 22, 2017

Supervisor:	Professor Samuel Kaski
Advisor:	Yrjö Häme D.Sc. (Tech.)
	Timo Similä D.Sc. (Tech.)

Aalto University
School of Science

Aalto University
School of Science
Degree Programme in Life Science Techonologies

ABSTRACT OF
MASTER'S THESIS

Author:	Inka Saarinen		
Title: Adaptive real-time anomaly detection for multi-dimensional streaming data			
Date:	February 22, 2017	**Pages:**	vi + 85
Major:	Bioinformatics		
Supervisor:	Professor Samuel Kaski		
Advisor:	Yrjö Häme D.Sc. (Tech.) Timo Similä D.Sc. (Tech.)		

Data volumes are growing at a high speed as data emerges from millions of devices. This brings an increasing need for streaming analytics, processing and analysing the data in a record-by-record manner.

In this work a comprehensive literature review on streaming analytics is presented, focusing on detecting anomalous behaviour. Challenges and approaches for streaming analytics are discussed. Different ways of determining and identifying anomalies are shown and a large number of anomaly detection methods for streaming data are presented. Also, existing software platforms and solutions for streaming analytics are presented.

Based on the literature survey I chose one method for further investigation, namely Lightweight on-line detector of anomalies (LODA). LODA is designed to detect anomalies in real time from even high-dimensional data. In addition, it is an adaptive method and updates the model on-line.

LODA was tested both on synthetic and real data sets. This work shows how to define the parameters used with LODA. I present a couple of improvement ideas to LODA and show that three of them bring important benefits. First, I show a simple addition to handle special cases such that it allows computing an anomaly score for all data points. Second, I show cases where LODA fails due to lack of data preprocessing. I suggest preprocessing schemes for streaming data and show that using them improves the results significantly, and they require only a small subset of the data for determining preprocessing parameters. Third, since LODA only gives anomaly scores, I suggest thresholding techniques to define anomalies. This work shows that the suggested techniques work fairly well compared to theoretical best performance. This makes it possible to use LODA in real streaming analytics situations.

Keywords: anomaly detection, machine learning, on-line learning, streaming analytics

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Life Science Technologies -koulutusohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Inka Saarinen

Työn nimi:
Mukautuva moniulotteisten poikkeavuuksien tunnistaminen reaaliaikaisesti

Päiväys: 22. helmikuuta 2017 **Sivumäärä:** vi + 85

Pääaine: Bioinformatiikka

Valvoja: Professori Samuel Kaski

Ohjaaja: Tekniikan tohtori Yrjö Häme
Tekniikan tohtori Timo Similä

Datan määrä kasvaa kovaa vauhtia miljoonien laitteiden tuottaessa dataa. Tämä luo kasvavan tarpeen datan prosessoinnille ja analysoinnille reaaliaikaisesti.

Tässä työssä esitetään kattava kirjallisuuskatsaus reaaliaikaisesta analytiikasta keskittyen anomalioiden tunnistukseen. Työssä pohditaan reaaliaikaiseen analytiikkaan liittyviä haasteita ja lähestymistapoja. Työssä näytetään erilaisia tapoja määrittää ja tunnistaa anomalioita sekä esitetään iso joukko menetelmiä reaaliaikaiseen anomalioiden tunnistukseen. Työssä esitetään myös reaaliaika-analytiikkaan tarkoitettuja ohjelmistoalustoja ja -ratkaisuja.

Kirjallisuuskatsauksen perusteella työssä on valittu yksi menetelmä lähempään tutkimukseen, nimeltään Lightweight on-line detector of anomalies (LODA). LODA on suunniteltu tunnistamaan anomalioita reaaliaikaisesti jopa korkeaulotteisesta datasta. Lisäksi se on adaptiivinen menetelmä ja päivittää mallia reaaliaikaisesti.

Työssä testattiin LODAa sekä synteettisellä että oikealla datalla. Työssä näytetään, miten LODAa käytettäessä kannattaa valita mallin parametrit. Työssä esitetään muutama kehitysehdotus LODAlle ja näytetään kolmen kehitysehdotuksen merkittävä hyöty. Ensinnäkin, näytetään erikoistapauksia varten yksinkertainen lisäys, joka mahdollistaa anomaliapisteytyksen laskemisen jokaiselle datapisteelle. Toiseksi, työssä näytetään tapauksia, joissa LODA epäonnistuu, kun dataa ei ole esikäsitelty. Työssä ehdotetaan reaaliaikaisesti prosessoitavalle datalle soveltuvia esikäsittelymenetelmiä ja osoitetaan, että niiden käyttö parantaa tuloksia merkittävästi, samalla käyttäen vain pientä osaa datasta esikäsittelyparametrien määrittämiseen. Kolmanneksi, koska LODA antaa datapisteille vain anomaliapisteytyksen, työssä on ehdotettu, miten sopivat raja-arvot anomalioiden tunnistukseen voitaisiin määrittää. Työssä on osoitettu, että nämä ehdotukset toimivat melko hyvin verrattuna teoreettisesti parhaaseen mahdolliseen tulokseen. Tämä mahdollistaa LODAn käytön oikeissa reaaliaika-analytiikkatapauksissa.

Asiasanat: anomalioiden tunnistus, koneoppiminen, reaaliaikainen analytiikka, reaaliaikainen oppiminen

Kieli: Englanti

Acknowledgements

I wish to thank my instructors Yrjö Häme and Timo Similä for their guidance and support. They were of great help especially in the first half of this process when the scope and direction of the work were still under design and evolving. In addition, they have given feedback and guided me in the right direction throughout the process. I thank my supervisor Samuel Kaski for valuable feedback on how to convey the main messages of the thesis. I thank Comptel Corporation for the facilities and resources that have enabled this work. Finally, I thank my co-workers, friends and family for supporting me throughout this journey.

Espoo, February 22, 2017

Inka Saarinen

Contents

Chapter 1

Introduction

Large amounts of data are generated constantly around the globe. For instance, in 2011, 1.8 zetabytes of data were generated in just two days, the expected number of RFID tags in 2021 is 209 billion and Facebook generates hundreds of terabytes of log and image data daily [47, 71]. There is a growing need and interest to take advantage of that data and turn it into information that can provide new business opportunities and value. Thus, there is a need for techniques that can process and analyse large amounts of data.

Streaming analytics means processing and analysing high-volume data in real-time or near-real-time. Traditional analytics and machine learning methods are designed to be run in batch mode: the model is trained with a finite amount, a batch, of data [58]. Making predictions on new data points using the trained model is often fast and can be applied either on a batch of data or on single data points one at a time. This allows making predictions in streaming mode, that is, the data can arrive constantly from different sources and predictions are made on each source independently based on the trained model.

If the incoming data distribution changes over time, a model learned in batch mode does not remain accurate. Then there is a need to do model training also on-line. This is referred to as on-line learning or incremental learning. The model is adaptive as it is updated constantly as data arrives. Sometimes it would be sufficient to learn a new model on a new batch of data at regular time intervals but the adaptation is not fast enough in all cases and retraining the model requires time and space. Streaming and batch processing can also be combined in a lambda architecture, which can be used to answer queries on big historical data with some delay and smaller amount of recent data faster [14]. Table 1.1 shows use cases for batch and on-line model learning and applying.

Streaming analytics has many applications. Malicious web-usage can be

Table 1.1: Use cases of batch and on-line learning and applying

	Batch applying	On-line applying
Batch learning	No real-time processing needed, all the data can be stored	Real-time prediction needed, data distribution does not change significantly between model trainings
On-line learning	Data distribution changes over time or only recent values are interesting but prediction depends also on some future values or predictions are needed only periodically	Real-time prediction needed and data distribution may change over time

identified and blocked by fraud detection [12, 61]. Social network analytics recognizes trends from users' activities in social media [65] and popular news topics can be identified [73]. Energy usage can be analyzed to gain energy cost savings for households [65] or energy suppliers can adjust their energy generation [61] or energy companies can decide when to sell or buy energy [56]. The same applies for water and gas consumption [56].

Processing and analysing streaming data requires both a suitable platform and good methods. The platform should allow for scalable, often distributed computing and handle data coming from different sources, as well as transfer and store the data with high speed. The methods should turn the data in the system into meaningful information and actions. There are two main issues concerning streaming data processing and analytics: space and time. When the amount of data is huge, it limits the possibilities of storing the data. Usually it is not possible to store all the data but only a relatively small subset.

Analysing streaming data should be near-real-time and the time spent on a single event needs to be minimal. The methods need to adapt to high throughput of the data such that they keep updating in the same speed as data arrives, otherwise an increasing backlog of data that waits for processing and analysis is formed. This often means that only main memory can be used, disk accesses would be too slow [38]. The methods need to be one-pass, that is, they are only allowed one look at each element in the data because the data is not stored in memory after being processed [38, 61].

In streaming analytics there is often a trade-off between time, space and accuracy [61]. Using more space often allows for faster processing. Sometimes it might be required that accuracy is high and exact algorithms are needed

Figure 1.1: Options for streaming analytics methods

- Batch learning vs. on-line learning
- Batch applying vs. on-line applying
- Exact vs. approximate
- Centralized vs. decentralized
- Historical data used vs. only recent data used

but often the requirements can be relaxed and approximate algorithms used to gain in using less space and processing time [38, 57, 65]. There are two ways to decrease accuracy: approximation and randomization. Approximation refers to having a small error bound and randomization refers to a small probability that the result is incorrect [57]. These can be combined to obtain an algorithm that provides with a high probability a result that has a small error. Raghavan [87] presents comparison of memory usage for exact and approximate algorithms on streaming data.

Heintz et al. [65] discuss how systems may have different requirements at different situations. For some cases, the queries may need to be processed fast, whereas in some other cases slower processing is sufficient. The same applies to accuracy. Sometimes there are large amounts of data, sometimes only small. Whether computation should be done in a centralized or non-centralized manner depends on the conditions. Since the conditions may vary over time and situation, designing the system is not easy. Some use cases require historical data, whereas some only recent data. Figure 1.1 summarizes different options of streaming analytics methods.

Gaber et al. [55] and Gama et al. [58] state that real-time models still lack good evaluation measures. Cross-validation for instance is non-feasible for streaming data due to the requirement that every element is only seen once. This also affects parameter tuning since it is not possible to use cross-validation or similar techniques to choose optimal parameters [75].

Krempl et al. [75] state that there are no good preprocessing methods for streaming data. For instance, zero-mean unit-variance normalization is difficult since data is constantly arriving. They also state that missing value handling is lacking for streaming data. Some of the data labels might also come with a delay and are thus not available to correct predictions or there is no recent data available for predictions.

Streaming data generated by devices often constructs one or several time series. Time series data contains a time stamp and a value and the temporal behaviour of the value is of interest. Time series analysis can be for instance tracking variables and their change over time. One application of time series analysis is anomaly detection, detection of unexpected or atypical behaviour. Detecting anomalies can then trigger alerts or certain actions. Anomaly detection performed in streaming mode on time series data allows to react fast to changes and odd behaviour.

Application areas of anomaly detection include fraud detection in credit card usage or insurance, intrusion detection or detecting novelties in surveillance images [45, 68]. Anomaly detection can also be used to identify floods from environmental sensor data or anomalous change in flight height or angle [64]. There are applications in the area of smart home or healthcare, for instance identifying that a person has turned on bath water but has not turned it off before going to bed or detecting diseases or other medical conditions [64]. In finance, anomalies in share prices or other market events can indicate buying or selling opportunities [68]. Some of these require very fast reactions and therefore there is a need for real-time anomaly detection methods.

In addition to containing exceptional values or patterns, the data distribution may change over time. This is called concept drift [60]. Detecting and adapting to concept drift is important in order to maintain accurate models. Gama et al. [60] present some applications where concept drift occurs. The first is predicting mass flow in an industrial boiler. Prediction improves operation and control of the boiler. Fuel feeding is manual and thus unstable and can result in concept drift. Another example is that wind power production and usage can be better predicted when taking concept drift into account instead of using a stationary model. Also recommendation systems should take concept drift into account. The users' interests may change over time but also depend on the context or have seasonal patterns. The methods should adapt to these kinds of shifts. Adaptive methods are useful also in self-driving cars where the concept drift can indicate change in the environment.

In this thesis challenges and approaches for streaming analytics are discussed. A literature survey on streaming analytics is presented, focusing on detecting anomalous behaviour, and commercially available software solutions for streaming analytics are presented. Test results on both synthetic and real anomaly detection data are shown for one real-time anomaly detection method. The thesis is structured as follows. In section 2, background on streaming analytics, time series analysis and anomaly detection is presented. In section 3, methods for streaming anomaly detection as well as available commercial solutions are presented. Also, the method chosen for further investigation is discussed in more detail in section 3. In section 4, the synthetic and real data sets used for the experiments are presented. In section 5, experiments and their results are discussed and in section 6 the work is concluded by discussing its impact and future directions.

Chapter 2

Background

2.1 Streaming analytics in general

Importance of streaming has increased in the past years with exponential growth in the amount of produced data and new application fields. However, streaming analytics is not a new field but has existed for decades. For instance, Page [83] presented a method for detecting change of distribution in 1954, Flajolet and Martin [53] presented a method for approximating count of distinct elements for data streams in 1985 and Alon, Matias and Szegedy [36] presented algorithms to estimate frequency moments on data streams. In the recent years, algorithms on streaming data for different purposes and applications have been developed. Data streams are different from batch mode data in several aspects. This section introduces some key specialities of streaming data and analytics performed on it.

Babcock et al. [38] present the notion of a data stream management system (DSMS) as opposed to the traditional data base management system (DBMS). A DSMS is a system that tries to handle the specific requirements of streaming data with respect to data management and querying. Babcock et al. [38] and Golab et al. [61] state that traditional data base management systems are not well suited for streaming data because they are not designed for fast and continuously arriving data and do not support continuous queries. Continuous queries are queries that are evaluated continuously as new elements arrive from the data [38, 61]. The opposite is one-time queries that are evaluated at a given time point from a snapshot of the database [38].

Another distinction in queries can be done between predefined and ad hoc queries [38]. Predefined queries can be optimized and it is known beforehand which attributes of the data should be stored. Ad hoc queries, on the other hand, which are not known at the start of processing a data stream, are more difficult for streaming data since they may refer to historical data that has already been discarded. It may be impossible to answer ad hoc queries on streaming data.

In streaming analytics recent events are usually more interesting than events that happened a long time ago. There are two popular approaches to take this into account in the methods: windows and weighting.

In windowing techniques only a set of subsequent samples, a window, is inspected at each time point. The window shifts with time, including more recent and disregarding older data. Windowing techniques include:

- Sliding windows where either N latest events or all events in the last t time units are kept and analytics is performed on those [57].
- Landmark windows where one endpoint, either beginning or end, of the window is fixed and the other endpoint is moving, thus either increasing or decreasing the window size [61].
- Tumbling windows where the whole window expires at the same time. For instance, data from each hour is gathered and processed separately. When the next hour starts, all analytics on the previous hour should be finished and that data is then discarded.
- Tilted windows where most recent data is stored with higher granularity than older data [57].
- Jumping windows which operate on a batch of data at once and are used in batch mode evaluation [61].

Choosing the size of the window can be difficult, too small window captures noise easily but too large window does not take change into account [56]. In some cases adaptive size windows can be used [56]. Wider windows show longer trends whereas narrower windows reveal the current state. These can be combined to get optimal results [59].

Another possibility to emphasize recent data is to use weighting on the data such that older data points have smaller weights [58]. This allows most recent data to have most influence but also to take into account older data to diminish the effect of short-term variations. Weighting can be implemented using fading factors. Then, the effect of each data point decreases as each new data point arrives. [56]

Weighting can also be performed within a sliding window, combining the two approaches [56]. In terms of forgetting, windows perform abrupt forgetting whereas weighting corresponds to gradual forgetting [56].

2.2 Time series

A time series represents sequential measurements over time, it is a collection of chronologically ordered data points. Data points in a time series are viewed as part of the whole time series and they are typically dependent on the previous values, therefore they cannot be looked at as individual data points [54]. A data stream forms a time series and thus time series analytics is often relevant for streaming analytics, although the exact order or time stamp of the data points is not always important. This section addresses analysis of time series.

Time series can have a trend which describes where the series is going. The trend can be described in terms of an average. A popular choice is the moving average which indicates the average of previous n points. Variations are cumulative moving average, weighted moving average and exponential moving average. Time series can also have a seasonality which describes seasonal, such as weekly or daily, patterns of the time series. Seasonality can be discovered by autocorrelation of the series. [56]

Typical analytics tasks for time series are classification and clustering of time series, finding similar time series, prediction of future values and detecting motifs and anomalies in time series. Motif discovery means identifyng repeating patterns from time series [51]. The patterns can be of various lengths and may overlap with each other.

Predicting values for future time points is a broadly studied area in time series analysis. Autoregressive models are often used in prediction. The simplest autoregressive model can be defined as $z_t = \alpha_1 + \alpha_2 z_{t-1} + \epsilon_t$ where z_t is a data point at time t, α are coefficients and ϵ_t is an error term at time t [56]. Another popular method is Kalman filter which implements a hidden first order Markov model [56]. A subcategory of prediction is recursive prediction where predicted values are used as model inputs. This may occur if true values are not immediately available and therefore cannot be used for predicting the next values. However, this makes the task more difficult as there is more uncertainty in predicted than observed data [51].

Finding similar time series requires defining a similarity or a distance measure between two time series. The most simple is the Euclidean distance, $d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$, where x and y are two time series of length n. However, the problem with this measure is that time series are not always the same length, they may have a lag or the may have different amplitude, period or scale. Therefore, more adjustable methods are needed. Dynamic time warping uses dynamic programming to find the optimal alignment between two time series and distance can be measured on the aligned series. [56]

2.3 Data reduction

In streaming analytics, the volume of data is typically very high. Often storing all the data is not possible but only a relatively small subset of the data can be stored. Space can be saved by storing only a subset of the samples or performing dimensionality reduction or constructing some kind of summary of the data, using e.g. averages and variances. This section presents several ways to perform data reduction.

One approach is to store only a subset or a sample of the objects that arrive, often keeping only or giving more weight to most recent objects [38]. A challenge in sampling is to maintain a representative sample of the data. Although this issue has been tackled [39], for some applications, such as anomaly detection for time series data, sampling is not appropriate since identifying anomalies is dependent on certain (previous) data points [55, 57]. A similar technique to sampling is load shedding. Instead of choosing randomly the set of points, in load shedding a sequence of data points is dropped [55]. Windowing techniques also store only a subset of the samples. Sampling can be combined with weighting such that older data points have lower probability of being in the sample.

Dimensionality reduction techniques can also be used to represent the data. One approach is sketching. Sketching means constructing a summary of the data stream which takes less space than the data itself and using this summary it is possible to approximately answer typical queries [38]. Sketching uses random projections of the features [57]. Another approach is wavelets which project the data stream onto an orthogonal set of basis vectors [38].

Histograms summarize the data by dividing it into bins. The data can also be aggregated, storing sums or variances of the data instead of the raw values [55]. Aggregation can also be combined with sampling [38]. Hashing, i.e. mapping the data from an arbitrarily-sized representation to a fixed-sized representation, can be used, often combined with some aggregation or sampling [61].

In time series analysis, reducing the number of points is called segmentation. Segmentation reduces granularity such that the reconstruction error is minimized or the shape of the series resembles the original as much as possible [51]. Segmentation is often done as a preprocessing step before other analytical tasks since many methods perform more accurately when the number of data points is smaller. Time series can have different representations and one goal in transforming the series into another representation is having a more compact representation that allows for fast computation. For instance, Symbolic Aggregate approXimation (SAX) transforms the time series into a compact string representation [56].

Even if the amount of data is reduced by sampling or dimensionality reduction, it might still be too much to be kept in memory. This poses new challenges to the methods as the models must be updated to remove or diminish effects of older data which can no longer be accessed. This often results in approximation algorithms.

2.4 Anomaly detection

Anomaly detection identifies unexpected events or patterns in the data. Anomalies can be also referred to as outliers, exceptions, aberrations, peculiarities or surprises [45]. Anomaly detection is trying to detect events which are interesting to the user and perhaps can be used to trigger some actions [45]. In this section ways to define anomalies are discussed and different approaches for anomaly detection are presented.

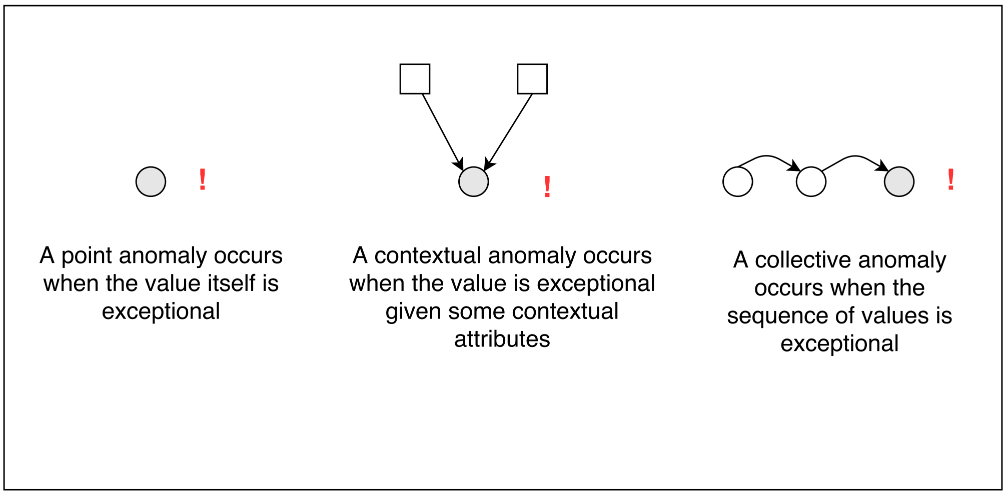
One challenge in anomaly detection is that normal behavior can be changing and thus it should be possible to adapt the models [45]. This is especially true for streaming data as huge amounts of data arrive to the system and old behavior, e.g. what happened the previous year, might be outdated. How big of a deviance from normal is considered an anomaly depends on the application field [45]. Optimally, the level can be easily tuned. This is possible for instance if the level is based on confidence intervals. The models used for anomaly detection should also be able to identify noise [45] but this is a difficult task because real noise is rarely normally distributed as assumed by many models.

Detecting anomalies from sensor networks poses some additional challenges [45]. There are typically noise and missing values and the data is distributed across the sensors. In addition, the cause of anomaly can be a faulty device or an actual exceptional event or a malicious attack [64]. In that case, it would be useful to be able to classify the cause of anomalies.

Chandola et al. [45] propose a classification of anomalies into three categories: point anomalies, contextual anomalies and collective anomalies as shown in Figure 2.1. Point anomalies are the simplest type of anomalies, the value itself is exceptional compared to all other values in the data and can be considered an anomaly, independent of other factors.

Contextual anomalies depend on the context, meaning other attributes affect whether the value is abnormal, e.g. whether temperature is normal is dependent on the context of the time of the year. Chandola et al. [45] divide the detection methods of contextual anomalies in two categories. In the first,

Figure 2.1: Anomaly types


!
A point anomaly occurs when the value itself is exceptional
!
A contextual anomaly occurs when the value is exceptional given some contextual attributes
!
A collective anomaly occurs when the sequence of values is exceptional

the context is defined for each event and the anomaly detection is done in the context. This reduces to point anomaly detection within the context. In the second approach, the contextual attributes are used as parameters in a model which is trained on the training data. Then, for a test data point, a prediction is computed and it is compared to the observed value. If the observed value deviates significantly from the prediction, it is considered an anomaly. Song et al. [91] propose a framework where the attributes are divided into environmental attributes that define the context and indicator attributes that are the values of interest. The probability density distribution of the indicator attributes is then conditioned on the environmental attributes to induce the context in the model.

Collective anomalies, on the other hand, depend on a consecutive series of values: a single value is not exceptional but the set of values appearing consecutively deviate from a normal set of consecutive values. Collective anomaly detection aims to detect exceptional subsequences in the data. Some methods map the subsequences into another feature space and perform point anomaly detection in that space. Some methods, on the other hand, define models for the sequence probabilities, such as Markov models, and label the subsequence as an anomaly if the sequence is very unlikely according to the model. One approach is to compute the expected frequency of a subsequence and compare the observed frequency to it. There are also methods which compare subsequences to the entire sequence, assuming that normal behaviour follows a certain pattern [45]. Detecting collective anomalies is significantly more difficult than detecting point anomalies [64].

Obtaining labeled data, that is, information on which events are normal and which are anomalous is often not possible, and anomalous events are much more rare than normal ones. Therefore, using supervised methods is often not feasible but unsupervised methods are typically used in anomaly detection [45]. If labels for some of the data points are available, semi-supervised methods can be used. One typical case is that labels for normal data points are known but the number of anomalous data points is insufficient or they are missing altogether. In that case, the method can try to learn boundaries for the normal class and classify whatever is outside those boundaries as anomalous [68]. This requires that the normal training data covers the whole region of normal data, otherwise many false alarms will appear [68]. Another example where some of the labels are known, is if the user gives feedback for events classified as anomalous. However, these labels may come with significant delays.

Events can be given either binary labels, whether they are anomalous or not, or scores indicating how likely they are anomalous [45]. Scoring can be used to choose e.g. n highest scoring events as anomalies. If scores are used, the user can typically tune the definition and threshold for an anomaly. However, sometimes a simple list of anomalies, which binary labels give clearly, is desired. Another possibility is to discretize the scores into anomaly levels, such as minor, major and critical.

Techniques to detect anomalies can be divided into different categories: classification based, clustering based, nearest neighbour based, neural network based, decision tree based, statistical, information theoretic and spectral methods [45, 68]. Classification based algorithms learn a model based on training data and predict the class (normal / anomaly) of each event. They are often not feasible because labels are not available. Clustering based methods find clusters and define as anomalies those events that are outside the normal clusters. These clusters might be very different in terms of e.g. density, and sometimes anomalies can form their own clusters whereas sometimes they appear as individual outlier points. Therefore it might be difficult to detect anomalies. [45]. Nearest neighbour based algorithms - also referred to as distance based algorithms [68] - classify the events based on their neighbourhood. They often have quadratic complexity which is unfeasible for large data sets. There exist some modifications which compute distances to only a subset of the points by dividing the points into e.g. pre-defined cells to reduce the computational complexity [68].

Neural network based models can operate either in a supervised or unsupervised manner [68]. In the first setting, a new data point is classified either normal or anomalous based on the learned classes from the network whereas in the latter case the network is trained on the normal data and if a test data point does not lie close to the normal classes, it is classified as anomalous [68]. Decision tree or rule based techniques can in a similar manner be used in supervised or unsupervised setting [34].

Statistical algorithms define a statistical model and classify the data points based on their probability according to the model [45]. They assume the data to be generated by some distribution and this assumption may not always be valid. In non-parametric approaches, density estimation is performed and points in low-density regions are classified as anomalies [68]. Information theoretic methods are based on the assumption that anomalies induce irregularities in the information content of the data. They try to maximize the difference between complexity of the whole data set and the complexity of a data set where anomalies are removed. They often require a large amount of anomalies to be present in the data. Spectral algorithms try to map the data to lower-dimensional space where normal and anomalous events are well separable. However, such a space does not always exist and even if it does, it might be difficult to find it [45].

Detecting anomalies from high-dimensional data creates additional challenges [33]. Estimating distributions for high-dimensional data is difficult and distances between points tend to be similar when the points are spread over a high-dimensional space and the data points lie sparsely in the space. Sometimes anomalies might be normal in many of the dimensions and only exceptional in a small set of the dimensions and those are difficult to detect when trying to detect from the entire space. In addition, high dimensionality introduces additional computational cost [33].

Therefore, dimensionality reduction techniques become important in outlier detection. However, many dimensionality reduction techniques lose interpretability of the result, i.e. after dimensionality reduction is performed it is difficult to identify the cause of an anomaly. Feature selection preserves interpretability but finding an optimal subspace among the features is a computationally expensive task. In addition, different outliers might appear in different subspaces and thresholds for low- or high-density regions or long distances may vary in different subspaces, especially if they are of different dimensions [33]. Furthermore, with e.g. 1000-dimensional data, each point is in fact likely to be an outlier in at least one of the dimensions [101].

There are different ways to define anomalies and different types of anomalies are interesting for different applications. There is a variety of approaches to detect anomalies and which one should be chosen depends on the data, application requirements and types of anomalies that should be detected.

2.5 Concept drift

It is often unrealistic to assume that data coming from a data stream would follow a stationary probability distribution. Instead, it can change and evolve over time. This is called concept drift [57]. Detecting change thus becomes important. The models need to take concept drift into account and adapt when concept drift is detected. In that case, in addition to incremental learning, decremental unlearning, i.e. removing effect of older samples from the model, is required [57].

Concept drift has several variations. It can be sudden or gradual. Sudden changes are easier to detect than gradual changes. Gradual change is also sometimes referred to as concept shift. An old concept may reappear later on and occurrences of concept drifts may or may not follow a pattern [75]. The changes can be of different nature, e.g. change in the mean or change in the variance.

Concept drift is typically unexpected and unpredictable [60]. In the case of a classification or regression task, concept drift can be drift in the distribution of input or output variables or drift in how the output variables depend on the input variables [60].

Sometimes a valid assumption is that the stream contains subsequences, each generated by a different distribution. Then the task is to detect change of distribution. However, sometimes there is noise in the data which should not be confused with a new distribution. Therefore, the methods need to also identify noise. [56]

Several methods have been proposed to identify concept drift. One approach is to compare data from a recent period of time to data from a longer period of time. Significant change in the means of the two windows is an indication of concept drift. Also other measures than mean can be compared when comparing a reference window and a recent window. For instance, it can be estimated whether they are generated by the same distribution [60]. To estimate the distributions, sometimes different methods can be applied to the longer and shorter time window [60]. The two windows can also be used to make predictions separately. Then change occurs when the accuracy of the recent window goes above the accuracy of the longer one since the longer one should be more accurate if the concept stays the same [60]. Comparing distributions of two windows of course requires more time and space than more simple methods but may give more accurate results. Change detection methods can also monitor performance of predictors by some predictor indicators and say a change has occurred if there is a change in the performance [56].

Change detection methods usually take a threshold parameter that governs how big deviations are noted as change. Small values cause more false alarms whereas large values do not detect changes at very early stages [60].

Sometimes change itself is interesting and then the output of the model could simply be a notification whenever a change is detected. In that case, also questions on what kind of changes there are and which changes are interesting are relevant [50]. However, often simply detecting change is not enough but the model also needs to adapt when a change is detected. This occurs when the model is used for prediction or anomaly detection.

In the case of using sliding windows or fading factors, the model naturally adapts to changes although change itself is not explicitly monitored or observed. This is sometimes referred to as blind adaptation [60]. This approach reacts well to gradual changes but is not very fast in reacting to sudden changes. The reaction speed can be tuned by modifying the window size or rate of fading but that also results in less robust models if they only depend on a small set of recent data points. The window size can also be tuned: it can be shortened when a change is detected and widened when the distribution is stationary [60]. Sampling can also be used instead of windowing [60]. Especially for re-occurring patterns, window or fading factor based approaches are not optimal since they forget behaviour that has occurred in the past.

Sometimes the model is updated based on the outcome of the latest data point: if the prediction differs enough from the true value, the model is updated to the direction of that value [60]. This approach does not imply forgetting unless fading factors are used and therefore cannot respond to concept drift. Also, sometimes the feedback, i.e. the true values or labels, are not available immediately or even in a short interval after the prediction and therefore, the model cannot be updated real-time.

If the model is not updated incrementally, then it must be retrained when a change is detected. This requires always keeping a certain amount of recent data which can be used to train the new model. Also, the computation takes some time. With some models, like decision trees, it is possible to adjust the model instead of training it again from scratch [60].

One option is to use an ensemble of learners and use e.g. majority voting to determine the final output [60]. This might be useful for reoccurring concepts: learners can also be activated and deactivated, then retraining does not take time but old patterns can be activated fast if the correct model is found.

Evaluating metrics for detecting concept drift are proportion of true detections and false detections and how much delay there is in detection [56]. In addition, deviation from true change point can be measured [60]. Related to concept drift is novelty detection where new patterns must be recognized [57]. A novelty can indicate for instance a new class [56].

2.6 Distributed settings

In this work inspection is restricted to systems where the streams are coming from a single source. Thus, it is assumed that the streams are coming from one system and are not distributed, or the distributed data sources send their data to a centralized location real-time with low latency. However, in many real-life situations, e.g. sensor networks, the data are produced in different locations in a distributed manner and data transfer might be a bottleneck for computation [56]. Therefore, it would be wise to perform some of the computation locally and transfer less data. The question is partially architectural but methods to tackle distributed streaming data have also been proposed. There has been research on investigating best approaches to deal with distributed data: how to transfer as little data as possible and still cope with the limited available memory in the devices producing the data [64]. This could also be scaled according to the amount of available memory [56]. This is a wide area of research and out of scope of this work. Therefore, distributed settings are left for future work.

Chapter 3

Methods

3.1 Available software platforms and solutions

This section presents a study of available software platforms and solutions for streaming analytics. There are several streaming platforms available for commercial use or as open source. Many of them are able to process millions of events per second [1, 9, 10, 15, 16, 24, 26, 28, 31]. Many of the solutions allow for combining historical and real-time data. Table 3.1 shows an overview of the solutions.

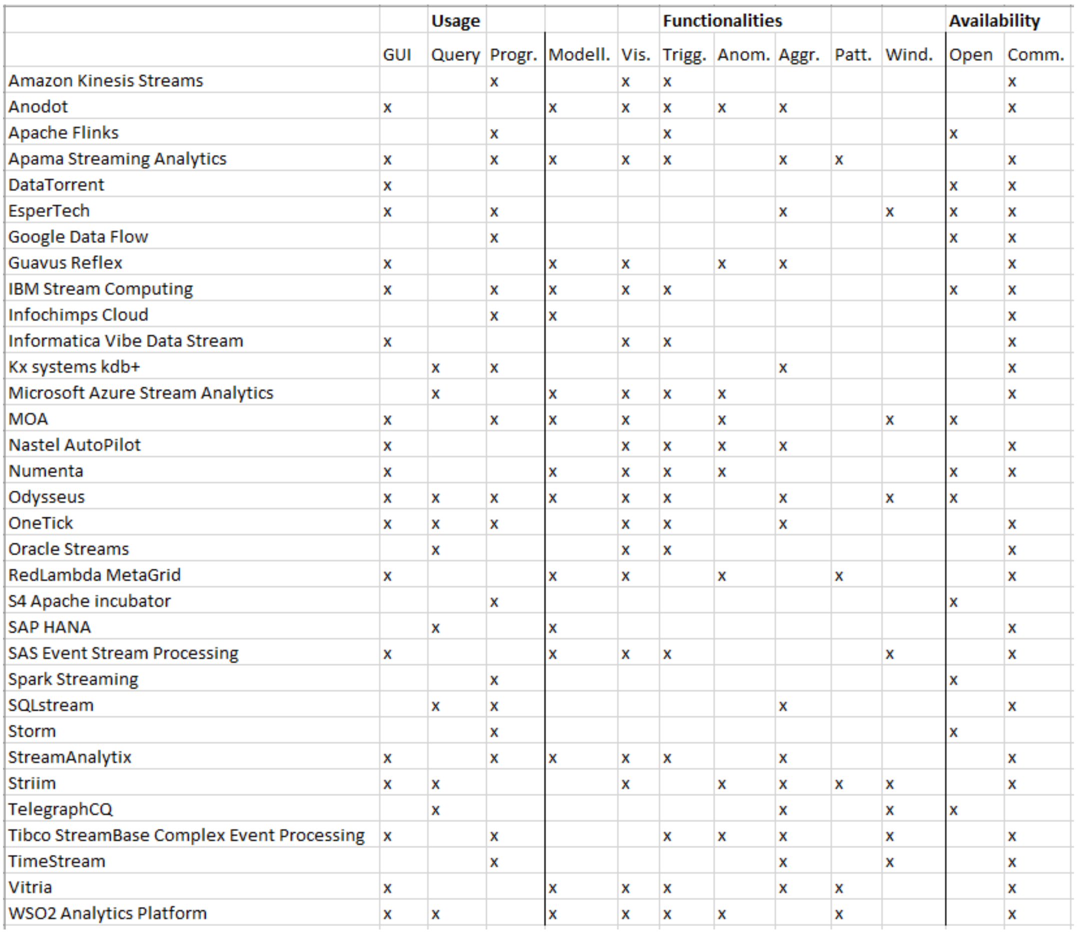
The platforms have been divided in three categories: those that contain only a platform for developers for stream processing, more advanced platforms that do not include any modelling and those that include modelling. Some of the solutions are a combination of these and they are discussed in the category where they fit best.

3.1.1 General development platforms

Some of the products are merely platforms that take care of the data flow and allow building applications on top of the platform. No event processing is included in the platform and it has to be programmed by the application developers. The platform can include libraries that ease the development, for instance aggregation or visualization libraries. Amazon Kinesis Streams [1], Apache Flink [3], Spark Streaming [25], Infochimps Cloud [10], S4 Apache incubator [22], Storm [27], TimeStream [86], Kx systems kdb+ [13], TelegraphCQ [46], SQLstream [26] and Google Data Flow [7] are such platforms.

Apache Flink, Spark Streaming, S4 Apache incubator and Storm are open source platforms. Apache Flink and Spark contain machine learning libraries. Apache Flink also includes some triggering options, that is, options to trigger some commands or events when certain condition occurs. Spark Streaming

Figure 3.1: Platforms and solutions for streaming analytics. Abbreviations: GUI = graphical user interface, query = query language, progr. = programming, modell. = modelling, vis. = visualization, trigg. = triggers, anom. = anomaly detection, aggr. = aggregation, patt. = pattern detection, wind. = windows, open = open source, comm. = commercial. Some of the platforms are open source, some available commercially. Some have both open source and commercial versions and some contain some open source features, are built on top of other open source platforms or contain open source platforms that allow further development but do not reveal the source code.



		Usage				Functionalities					Availability	
	GUI	Query	Progr.	Modell.	Vis.	Trigg.	Anom.	Aggr.	Patt.	Wind.	Open	Comm.
Amazon Kinesis Streams			x		x	x						x
Anodot	x			x	x	x	x	x				x
Apache Flinks			x			x					x	
Apama Streaming Analytics	x		x	x	x	x		x	x			x
DataTorrent	x										x	x
EsperTech	x		x					x		x	x	x
Google Data Flow			x								x	x
Guavus Reflex	x			x	x		x	x				x
IBM Stream Computing	x		x	x	x	x					x	x
Infochimps Cloud			x	x								x
Informatica Vibe Data Stream	x				x	x						x
Kx systems kdb+		x	x					x				x
Microsoft Azure Stream Analytics		x		x	x	x	x					x
MOA	x		x	x	x		x			x	x	
Nastel AutoPilot	x				x	x	x	x				x
Numenta	x			x	x	x	x				x	x
Odysseus	x	x	x	x	x	x		x		x	x	
OneTick	x	x	x		x	x		x				x
Oracle Streams		x			x	x						x
RedLambda MetaGrid	x			x	x		x		x			x
S4 Apache incubator			x								x	
SAP HANA		x		x								x
SAS Event Stream Processing	x			x	x	x				x		x
Spark Streaming			x								x	
SQLstream		x	x					x				x
Storm			x								x	
StreamAnalytix	x		x	x	x	x		x				x
Striim	x	x			x		x	x	x	x		x
TelegraphCQ		x						x		x	x	
Tibco StreamBase Complex Event Processing	x		x			x	x	x		x		x
TimeStream			x					x		x		x
Vitria	x			x	x	x		x	x			x
WSO2 Analytics Platform	x	x		x	x	x	x		x			x

is not an actual streaming platform but it operates in micro-batch mode, processing data in very small batches with high speed near-real time which makes the outcome resemble streaming.

Amazon Kinesis Streams can be combined with Amazon Kinesis Aggregators to ease aggregation of data. Infochimps Cloud provides a cloud platform that allows for both stream and batch processing. It contains tool-kits for developing real-time analytics. TimeStream also allows the user to query the data continuously using windows. Possible use cases are network monitoring, audience monitoring, sentiment analysis, finding top elements (on some metric) and counting distinct elements.

Kx systems kdb+ is a database system that can be used for stream processing, combined with historical data. It can be used for time series analysis. TelegraphCQ and SQLstream operate SQL queries on streaming data. They both allow using windows. SQLstream contains aggregation functions such as average and standard deviation, more advanced analytics functions can be imported using other programming languages.

3.1.2 More developed platforms with no modelling included

Here I discuss platforms and solutions that have ready applications or applications can be easily built via a graphical user interface (GUI), possibly including writing some queries. These platforms include OneTick [19], Striim [29], Oracle Streams [20], Tibco StreamBase Complex Event Processing [30], DataTorrent [5], EsperTech [6], Informatica Vibe Data Stream [11] and Nastel AutoPilot [16]. Typical functionalities for the platforms are visualizing data streams, aggregating them, creating triggers based on some rules or patterns, detecting anomalous events and detecting patterns. Some of the platforms enable usage of sliding or tumbling windows (see section 2.1). In these platforms, anomaly detection or triggering is not based on historical data but some pre-defined thresholds, rules and patterns.

Oracle Streams and Informatica Vibe Data Stream process streaming data and transfer it to other systems. Oracle Streams can be used to monitor the streams and generate triggers. Informatica Vibe Data Stream can do some preprocessing and visualize the data before it is moved on. Combined with Informatica Rulepoint Complex Event Processing it can also trigger rule-based alerts.

OneTick is created for monitoring especially financial or stock data. It can do aggregation and users can create their own functions. It can be operated via a GUI or by writing and executing queries. It can combine historical and real-time data. Possible use cases are profit or loss monitoring, risk management, transaction cost analysis and regulatory surveillance. Striim processes streaming data by SQL queries and the results can be visualized via a GUI. Striim can perform aggregation, anomaly detection, pattern finding and fraud detection. Possible use cases are log monitoring, fraud detection, location monitoring and customer experience management.

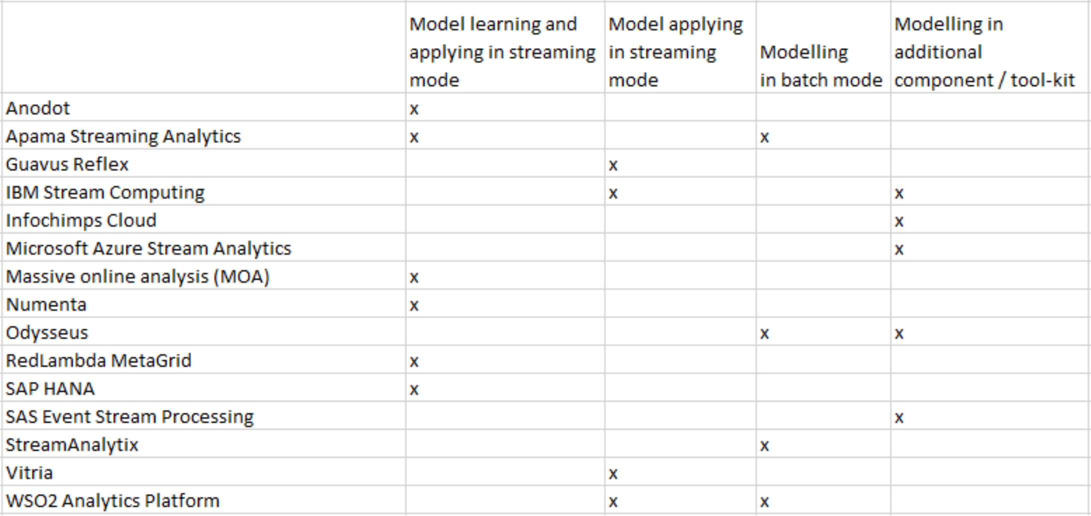
Tibco StreamBase Complex Event Processing is operated via a GUI and can perform data normalization, aggregation and correlation and generate rule-based triggers. It can combine historical and real-time data. Possible use cases are risk management, fraud detection, routing, market data management, real-time pricing, network management, clickstream analytics, analysing cross-sell and up-sell opportunities and patient data management.

DataTorrent is a platform that does stream processing but does not contain any analytics. EsperTech is a product for complex event processing and event series analysis. It has a GUI but can be combined with source code imported by the user. It enables sliding and tumbling windows and contains some aggregation functionalities as well as a couple of approximation algorithms for computing frequency or top elements (on some metric) or counting distinct elements. Nastel AutoPilot can monitor alerts and it can be used to manually trace the root cause for the alerts. It can do pattern matching, aggregation, sorting, merging, joining of events and metrics, computing standard deviation and moving averages and momentum indicators to determine which states are normal and which are abnormal. It can be used for application performance and end-user experience monitoring.

3.1.3 Platforms including modelling

Platforms including modelling may perform modelling in batch mode and only use the results in stream processing, they may learn the models in batch mode and apply them in streaming mode or they may both learn and apply the models on-line. Some of the solutions do not contain modelling in themselves but can be combined with a plug-in or additional package. It is often not explicitly stated whether modelling could be performed in streaming mode using these additional components. It is likely that many of them do not operate in streaming mode but only pass the results to the stream processor since these additional packages or plug-ins may be used with other than streaming products as well and it would often require different settings to operate in streaming mode. An overview of the types of modelling used in these platforms is shown in table 3.2. Platforms containing modelling include StreamAnalytix [28], Massive online analysis (MOA) [41], Numenta [17], Anodot [2], RedLambda MetaGrid [21], WSO2 Analytics Platform [32],

Figure 3.2: Platforms including modelling. This table shows whether the platforms perform model learning and applying in batch or on-line mode. Some of the platforms do not contain modelling but contain either tool-kits for modelling or can be combined with additional analytics packages. See the text for further details.



	Model learning and applying in streaming mode	Model applying in streaming mode	Modelling in batch mode	Modelling in additional component / tool-kit
Anodot	x			
Apama Streaming Analytics	x		x	
Guavus Reflex		x		
IBM Stream Computing		x		x
Infochimps Cloud				x
Microsoft Azure Stream Analytics				x
Massive online analysis (MOA)	x			
Numenta	x			
Odysseus			x	x
RedLambda MetaGrid	x			
SAP HANA	x			
SAS Event Stream Processing				x
StreamAnalytix			x	
Vitria		x		
WSO2 Analytics Platform		x	x	

SAP HANA [23], Apama Stream Analytics [4], Guavus Reflex [8], StreamAnalytix [28], Vitria [31], SAS Event Stream Processing [24], Odysseus [18], IBM Stream Computing [9] and Microsoft Azure Stream Analytics [15].

Numenta, Anodot, RedLambda MetaGrid and WSO2 Analytics Platform are anomaly detection platforms. Numenta and Anodot update the model continuously. Numenta builds a model for each metric to be monitored. Seasonality is included in the model which is based on hierarchical temporal memory methods. Anodot includes dozens of time series models of which it chooses the best for each stream. It updates the models in streaming mode and shows the users the streams which contain anomalies. It also correlates the streams and thus tries to identify causes for the anomalies. RedLambda MetaGrid is a solution created for threat detection that can be used to detect anomalies, meaningful events, sequences, rates, patterns and correlations. WSO2 Analytics Platform can be used for anomaly and fraud detection. The fraud detection functionality contains modelling in batch-mode. It can combine historical and streaming data.

MOA and SAP HANA update the models on-line. MOA is a platform operated by a GUI. It contains some algorithms for on-line clustering, classification, anomaly detection and recommendation engines. New methods can also be implemented. The results can be visualized. In SAP HANA, there are two machine learning methods integrated that operate in streaming mode: one for classification and one for clustering. In addition, developers can produce their own methods through query processing and a software development kit.

Apama Stream Analytics is a tool where models created by other tools can be imported, and it will generate actions and make predictions on streaming data based on the models. The platform is used with a GUI but the models need to be imported from elsewhere. Most of the analytics happens in batch mode but some models are also updated in streaming mode. Guavus Reflex is a platform that allows processing and analysing streaming data and visualization.

StreamAnalytix contains building blocks for analytics, which include certain algorithms. It also allows to visualize the data and generate triggers. It can be used via a GUI and new features can be programmed. It can combine historical and real-time data. Possible use cases are call center analytics, log analytics, predictive maintenance, and sentiment and topic analysis. Vitria can be used to predict threats and opportunities and trigger processes. It can look for pattern-based indicators and do aggregation and visualization. Analytics is performed in batch-mode.

SAS Event Stream Processing, Odysseus, IBM Stream Computing and Microsoft Azure Stream Analytics contain modelling as a tool-kit or as an additional package. SAS Event Stream Processing is used by creating workflows that allow monitoring data, following patterns such as threshold values or aggregates such as count or variance and generating triggers based on the patterns. The event stream processing tool can be combined with SAS Advanced Analytics to include modelling. Odysseus can be used either by defining queries or through a GUI, source code is also available and can be used for development. Dashboards can be created and triggers generated with it. It contains classification and clustering algorithms as plug-ins. Possible use cases include monitoring business activity and key performance indicators, monitoring and forecasting weather and monitoring manufacturing. IBM Stream Computing allows to create rule-based configurations for stream processing. Additional features and analytics are available as tool-kits allowing real-time prediction using models computed in batch mode. It is possible to develop new features by programming as well. Use cases include targeted marketing, e.g. sending offers based on customers' interests and location and monitoring acceptance or rejection of the offers, fraud detection and monitoring patients for emerging diseases. Microsoft Azure Stream Analytics can be used to detect anomalies, trigger alerts and visualize the data.

The platform can be combined with a machine learning package.

3.1.4 Summary

There is a multitude of solutions available for streaming and new ones are emerging constantly. Terminology used is not established, for instance the term ”streaming analytics” is referred to in the context of data processing or querying as well as applying modelling and machine learning in batch and on-line modes. Currently, there are few platforms where modelling, either learning or applying or both, are applied in streaming mode. Also, many of the products require the user to do some programming, querying or stream configuration and only a few solutions provide ready-made applications.

3.2 Anomaly detection for streaming data

In streaming time series data, anomalies are defined in the context of the data stream, sometimes also with respect to other data streams. Unusual changes, sequences and temporal patterns are interesting [64]. Anomalies in time series data can be detected in several ways. One is by making predictions and comparing the observed values to predictions. Another is constructing a profile of the time series and checking whether a new event fits in the profile. A third is by information theoretic approach where a point is an anomaly if removing it makes the fitting less erroneous. [64] In this section methods designed for on-line anomaly detection are presented.

There are methods which update the predictive models as new data points arrive. This way, the models are adaptive. However, some of the incremental algorithms still take a lot of time to perform a single update and are therefore not suitable for streaming data. Hence, also time complexity is important. Another aspect to consider is the number of fixed parameters in the model. Since parameters cannot be determined by for instance cross-validation, heuristic parameter choices are often needed.

The methods often provide anomaly scores and a threshold is needed to distinguish normal and anomalous scores. If the threshold is based on a confidence level, it gives a natural interpretation of the threshold. On the other hand, in for instance distance-based methods, choosing a threshold distance does not have such an interpretation and can be a difficult task.

For this work existing anomaly detection methods for streaming data have been divided into eight categories:

- **Density-based methods** estimate the probability distribution, or in non-parametric methods probability density for each point, and points lying in areas with low probability density are declared anomalous.

- **Distance-based methods** define anomalies based on how far they lie from their nearest neighbour(s).
- Methods based on prediction errors predict future values and define points as anomalies if the predicted values differ significantly from the observed values. These can be further divided to **regression-based methods** which make the predictions based on some attributes of the data and **methods based on time series** which use auto-regression, i.e. values of one or a few preceding data points affect the prediction of the next data point.
- **Methods using support vector machines** construct the normal class using support vectors and define as anomalous the points that lie outside the class boundaries.
- **Clustering-based methods** cluster the data and define as anomalous points that are either far away from any cluster or form atypical clusters.
- **Tree-based methods** construct classification or regression trees and define anomalousness based on how well the points map to the tree.
- **Projection-based methods** project data onto another, often lower-dimensional, space and points with high reconstruction error are defined anomalous.

The categories are somewhat overlapping and some methods could be assigned to multiple categories.

3.2.1 Density-based methods

Density-based methods form density estimates. Parametric methods fit one or more statistical distributions to the data and non-parametric methods estimate density by using e.g. histograms, kernels or grids. When a new data point arrives, its density is estimated and if the density estimate is below a given threshold, the point is defined anomalous.

In this section four density-based methods are presented. The method by Yamanishi et al. [99] is designed for mixed-attribute data sets and has a parametric and non-parametric variant. Subramaniam et al. [93] use kernel density estimation and present both a density- and a distance-based method using the density estimators. Faulkner et al. [52] present ways to define anomalies from density estimates. Pevny et al. [84] project data on one dimension and use density estimates of one-dimensional histograms.

Yamanishi et al. [99] present a method for mixed-attribute data sets, that is, data sets that contain both categorical and numerical variables. Their method is an on-line discounting learning algorithm that incrementally learns a probabilistic mixture model. The model uses a decaying factor to diminish effects of old data.

Histogram density is used for categorical variables by performing sequentially discounting Laplace estimation. For each categorical variable, the data values are partitioned into disjoint sets and a cell is created for each combination. At arrival of a new data point, the counts are adjusted with temporal discounting and appropriate Laplace smoothing is applied.

For continuous data, a Gaussian mixture model or a kernel mixture model is used. A model is built for each cell in the histogram density. A Gaussian mixture model is learned via sequentially discounting expectation maximization which is effectively incremental expectation maximization with discounting effects of past examples. A kernel mixture model is learned via sequentially discounting prototype updating algorithm where coefficients of mixture and variance matrix are fixed and means of kernels or prototypes are learned.

A joint distribution of the categorical and continuous attributes is formed and an anomaly score of a data point is computed based on how much the distribution changes when adding the point. The time complexity of the algorithm is cubic in the number of dimensions, hence the algorithm is not suitable for high-dimensional data.

Subramaniam et al. [93] present a method based on kernel density estimation. A point is classified as an anomaly if it lies in a space of low density. Kernel density estimators are easy to compute and maintain in a streaming environment. They can be easily combined and they scale well in multiple dimensions. The method by Subramaniam et al. assumes each attribute takes values in the range [0,1] or the values should be mapped to that range but they do not discuss how to do this. The anomalies are searched within a stream as well as anomaly streams are searched among all the streams. The anomalies are searched within a sliding window and the points are compared to the rest of the window. To compute the density estimators, a random sample of the sliding window and standard deviation of the values in the sliding window are maintained.

Subramaniam et al. present two different methods to use in this framework: distance-based and density-based anomaly detection. The density-based method takes into account local variations in the feature space by looking at the density in the neighbourhood of the point in addition to global density. It is thus more robust than simply looking at the density estimate. In the distance-based approach, the number of points within a given distance is estimated and the point is declared an anomaly if the estimate is below a given threshold.

Faulkner et al. [52] also use on-line density estimation but do not present density estimators themselves, instead they use existing ones. Anomaly thresholds are based on quantile summaries. They detect anomalies both within a stream and across multiple streams. Their approach can be extended to take seasonal variation into account.

Pevny et al. [84] use an ensemble of weak learners, one-dimensional histograms. The histograms are constructed by random projections. Each histogram approximates probability density of input data projected onto a single projection vector. The projections are constructed by randomly setting a certain amount of the weights in a weight vector to zero and the rest are drawn randomly from normal distribution. An anomaly score is given as an average log probability estimated on each projection vector. Histograms are updated as a new sample arrives.

Pevny et al. use equi-width histograms that are defined off-line. This requires that the features have approximately the same scale and the range of the values is approximately known. Pevny et al. state that there exists also a method to maintain histograms on-line without assumptions on the value range but they found the equi-width histogram more accurate. The method is able to handle missing values. The probability distributions of the histograms are assumed to be independent and even if this doesn't hold in practice, Pevny et al. claim they still get good results. Features are ranked according to their contribution to the sample's anomalousness and contribution of a certain feature can be assessed by a statistical test.

3.2.2 Distance-based methods

Distance-based methods compute or estimate distances to nearest neighbours or count how many data points are within a certain distance from each data point. Exact calculation is costly and either clever data structures or approximations are needed. Defining anomaly thresholds for distance-based methods is difficult at least without application-specific knowledge.

Angiulli et al. [37] propose a method where a point is considered an anomaly if it has less than k neighbours in a window. A neighbour is a point that is at most distance d away from the point. Angiulli et al. propose exact and approximate algorithms for the problem. In the exact algorithm, an indexed stream buffer (ISB) is kept. ISB is a data structure which supports range query search. That is, given a new object and a threshold distance, it returns all objects whose distance to the object is less than the threshold distance. ISB keeps track of time points of preceding neighbours and number of succeeding neighbours in a window. If number of succeeding neighbours is at least k, the point is a "safe inlier", that is, it cannot become an anomaly. When a data point arrives, its neighbours are queried and the preceding and succeeding neighbours are updated for the new point and its neighbours, respectively.

While the exact algorithm requires storing all points within the window in ISB, the approximate algorithm consumes less memory. There, only a certain fraction of randomly chosen safe inliers is stored. They cannot become anomalies but could of course affect the anomaly score of other points. In addition, instead of storing the list of preceding neighbours at the arrival of a sample, a ratio of preceding safe inlier neighbours and total number of safe inliers can be stored.

The algorithm is designed for defining anomalies not continuously but at given query points. To find anomalies at a given point, the ISB structure is scanned and the number of preceding and succeeding neighbours is computed. Since a point's anomalousness depends on future data points, its anomalousness cannot be determined immediately when the point arrives.

Pokrajac et al. [85] use an incremental local outlier factor (LOF) to detect anomalies. It is based on computing densities at local neighbourhoods. To compute the LOF of a point p, first distance to the kth nearest neighbour of p is computed. Then reachability distance with respect to another point is computed. Local reachability density is the inverse of average reachability distance based on k nearest neighbours of p. LOF is the ratio of average reachability density of the k nearest neighbours of q and that of q.

An incremental algorithm determines which LOF scores need to be updated in insertion and deletion of a point. The amount of updates needed is quite low. For efficient implementation, efficient nearest neighbour and reverse nearest neighbour queries are needed. These exist but are not efficient on high-dimensional data due to lack of efficient indexing structures on high-dimensional data.

Xie et al. [97] use a k nearest neighbours (KNN) -based method. The data are divided in hypercubes. Anomaly is based on the number of occurrences in the same hypercube.

3.2.3 Methods using time series prediction

The following methods fit models for time series and base their anomaly scores on deviations from the constructed models. Yamanishi et al. [98] use an autoregressive model, Lee et al. [77] use a Kalman filter multivariate autoregression model with Bayesian inference, Khreich et al. [72] use an ensemble of hidden Markov models and Richard et al. [88] use a kernel-based approach.

Yamanishi et al. [98] present a method with the same idea as they had for the density-based model in [99] (see section 3.2.1) but designed for time series data. The framework is the same but an autoregressive (AR) model is used and learning is done via sequentially discounting AR model estimation. The AR model of order k is a linear combination of k previous time points. Anomaly score can be calculated either by comparing distance of distributions before and after inserting the point as in [99] or by the probability density.

Lee et al. [77] use a Kalman filter multivariate autoregression model. In a Kalman filter an observed variable depends linearly on a hidden variable. A hidden variable at time t depends on the hidden variable at time $t-1$. The Bayes formula is applied to estimate the posterior distribution of the hidden variables. At each step, posterior estimate of the hidden variable at the previous time point is used as the prior for the new posterior. Observation is included to obtain the posterior. [81] The method by Lee et al. give probabilities for the observation given the point is normal. A simple thresholding could be used but they propose an approach using extreme value theory. The method gives scores for both anomalies and change points.

Khreich et al. [72] base their approach on an ensemble of hidden Markov models (HMM). They state that there are approaches to update HMMs incrementally but claim that doing so in one iteration as is required in streaming approaches does not result in good performance. Therefore, they do not update existing HMMs but instead build new ones as a new batch of data comes in. Incremental Boolean combination, which aims at maximizing the receiver operating characteristic (ROC) curve, is used to combine the HMMs. Less accurate HMMs, that have not been selected in the combination for a while, are discarded. The method requires validation data composed of normal and anomalous sub-sequences.

Richard et al. [88] present a method for on-line time series prediction using kernels. The model contains a set of kernel functions that are added and removed from the set on each update. Richard et al. do not present an anomaly detection scheme but it could be added, e.g. based on the prediction error.

3.2.4 Methods using support vector machines

Support vector machines (SVM) are an approach for classification where support vectors form the class boundaries. In anomaly detection, one-class

SVM can be used to define the normal class and points falling outside the boundaries can be defined as anomalies.

Davy et al. [49] construct SVMs from kernels since kernels are insensitive to data dimension and do not require fitting the data to a certain statistical distribution. A normal class is constructed and its boundaries are defined such that most points lie inside the boundaries. Davy et al. use data points classified as normal to train the class boundaries. Data without labels could also be used if it is assumed that number of anomalies is very small. Then, the anomalies would not have a big impact on the model. The method is incremental, new vectors can be added and old ones removed. However, adding and removing vectors are iterative processes and although they are expected to converge very quickly, there is a possibility that this would become a bottleneck in computation.

Zhang et al. [100] use a quarter-sphere one-class SVM [74] where origin and radius of a sphere are used to describe the normal class. It can be applied for multiple streams that each have a pre-defined subset of the other streams as neighbours. Each point is first checked for its anomalousness within its own stream to see if it lies outside the normal class. It is then compared with radii of the neighbouring streams. If it still seems anomalous it is declared an anomaly. The radii are updated after each observation of a new point, the new point is added and the oldest in the window is removed. Updates can be done either in sliding window mode, at fixed time intervals or when a point is identified as an anomaly or margin support vector. The data are normalized which is an additional challenge in streaming settings.

Also Laskov et al. [76] present a method for incremental SVM. The approach is general but can be extended to one-class incremental SVM to be used in anomaly detection. The approach of incremental learning can be reversed to obtain decremental forgetting and thus a sliding window mode is possible. Laskov et al. claim they have improved time complexity compared to previous approaches for incremental SVM but the memory consumption is large.

3.2.5 Regression-based methods

Regression-based methods construct a regression model on the data and define anomalies based on deviations from predictions. Hill et al. [66] use univariate regression models that are based on either naive predictors, nearest cluster predictors, single-layer linear network predictors or multilayer perceptron predictors. Ting et al. [96] use Bayesian linear regression. Chen et al. [48] present a method which takes contextual attributes into account.

Hill et al. [66] present a method that uses a univariate regression model and calculates a prediction interval from recent historical data to detect anomalies. A sliding window of size w is used. Training is done in batch mode using 10-fold cross-validation. There are two strategies: either the anomalies are included in the window that is used to calculate the predictions or they are discarded. This defines whether the model is adaptive to anomalies or not. For instance, in case concept drift would occur it would be useful to include the anomalies in the window. The other approach would be to use the predicted value instead of the real value for anomalous points. This can cause problems in two cases. First, if non-anomalous data are misclassified as anomalous, the false predicted value will be used in future predictions. Second, the variation in the predicted values is typically smaller than variation in the observed values, therefore a lot of anomalies would induce bias and a too centralized model.

Hill et al. use four different predictors in this framework, namely naive predictor, nearest cluster predictor, single-layer linear network predictor and multilayer perceptron predictor. The naive predictor only predicts the next value to be the previous value. The nearest cluster predictor assigns training data to clusters using k-means and predicts the cluster of the new data point and the mean of the cluster as the value of the data point. The single-layer linear network predictor is a linear combination of the values in the window and weights are updated on each iteration. The multilayer perceptron predictor on the other hand is a non-linear combination of input variables. A feed-forward network with sigmoid activation functions in the hidden layers and linear activation in output node is created. The network is trained using backpropagation with gradient descent and the model is retrained from time to time. From the predictions, the points are classified as anomalous based on Student's t-distribution.

The naive predictor is clearly very simple and not useful in many cases. The nearest cluster predictor and multilayer perceptron predictor on the other hand operate in batch mode and therefore are not very adaptive to changes in the data distribution. The single-layer linear network updates the weights on each iteration and therefore is the most suitable for an on-line learning problem.

Ting et al. [96] present a Bayesian linear regression approach for anomaly detection. It is assumed that observed outputs have unequal (heteroscedastic) variances. A weight is given for each data point. The model is learned via expectation maximization. The goal is to maximize incomplete log likelihood since hidden probabilistic variables are marginalized out. However, only lower bound of the incomplete log likelihood is analytically tractable and it is based on the expected value of the complete likelihood. Updates need to be run iteratively. For an incremental version of the algorithm a forgetting factor is introduced.

For large prediction error, weight of the data point is small. Hence, the algorithm automatically down-weights the anomalies. To explicitly detect anomalies, the algorithm should be slightly modified and a threshold set. Computational complexity for an update is cubic with respect to the data dimensionality but can be reduced to linear by using recursive least squares technique.

Chen et al. [48] discuss a method of doing regression analysis for time series where data is divided in hypercubes based on contextual attributes and aggregated within those. The method uses tilted time windows (see section 2.1). They present how only a subset of the data and cubes need to be stored and how they can be stored in a compressed way to save space. They don't describe an anomaly detection method but it could be implemented, e.g. based on prediction error.

3.2.6 Clustering-based methods

Clustering-based methods divide data into clusters. Incremental clustering requires clever data structures or approximations. How anomalies are defined, i.e. whether they are individual points far away from any cluster or cluster centre or atypical clusters, can vary but can often be tuned independent of the clustering method used.

Burbeck et al. [44] present an anomaly detection method with incremental clustering. It is assumed that training data do not contain anomalies. The model consists of a number of clusters and a tree index whose leaves contain the clusters. Clusters are stored as condensed information, cluster features, that contain the number of points in the cluster, sum of cluster points and sum of squares of cluster points. Centroid is the average of points and radius describes tightness of the cluster or how far the points are from the centroid. Centroid and radius can be computed from cluster features and cluster features of two clusters can be easily merged. A leaf node of the tree contains one or more cluster features. Cluster features also have a threshold indicating whether they can absorb more data points: either the radius must be smaller than a threshold or distance to a data point must be smaller than a threshold.

When a new data point arrives, its closest cluster is searched and it is checked whether it beats the threshold. If it does, it is added to the cluster. Otherwise, it forms a new cluster unless the maximum number of clusters is reached. In that case, the threshold is increased and the model is rebuilt. Clusters can be removed from the tree if there have not been occurrences in the cluster for some time. Also cluster size and frequency of occurrences can be taken into account in forgetting clusters, or a decaying factor for old clusters can be used.

Hsu et al. [69] and Gomes et al. [62] also present methods for incremental clustering that could be used for anomaly detection.

3.2.7 Tree-based methods

Tree-based methods construct a tree structure from the data. The trees are updated as new data points arrive but usually significant changes to the tree structure would require retraining which takes time. Tan et al. [94] define anomalies based on mass profile of the tree, whereas Guha et al. [63] define anomalies based on how much adding a point would change the tree structure.

Tan et al. [94] present a one-class anomaly detection method that is based on streaming half space trees (HS trees). HS trees are full binary trees. The model consists of a set of trees that are formed by a set of nodes where each node captures the number of data items, or mass, within a particular subspace of the data stream. The model assumes that data values are in the range [0,1] or they should be mapped to that range.

The method uses two windows: a reference window and a latest window. Latest window is the current window which is not full yet. The reference window is the window preceding the latest window. Mass profile of the reference window is learned and data points in latest window falling in low-mass subspaces of reference window are declared anomalies. The mass profile of latest window is recorded incrementally and once the latest window is full, it becomes the reference window. Building the tree is done by randomly sampling splitting points for a randomly sampled dimension until the predefined tree depth is reached. Updating a mass for a new instance is done by traversing down the tree and updating each node on the way. Thus, this requires time that is of the order of the depth of the tree. The anomaly scores are computed from the masses and depth of terminal nodes.

Guha et al. [63] present a similar method as Tan et al. which is based on a random cut forest. They define a robust random cut tree by making rules based on randomly splitting one attribute at a time at a randomly chosen point in the range of the attribute. A robust random cut forest is a collection of independent robust random cut trees. If a point is far from other points in the original space, it will also be far from others in a random cut tree on expectation. A random sample of the stream is maintained. Model complexity is defined as the sum of depths of the nodes in the forest. A point is identified as an anomaly when the joint distribution of including the point is significantly different from the distribution that excludes it. The method does not require data range to be restricted as in Tan et al. but knowledge about the data range should be available.

3.2.8 Projection-based methods

In projection-based methods the data is projected into a lower-dimensional space. The projection is constructed such that the projection vectors represent the normal data well. If a data point projects well and the residual is small, it is considered normal but if it does not project well and the residual is large, it is considered an anomaly.

Liu et al. [79] use principal component analysis (PCA) in sliding window mode. The PCA decomposition is divided to two parts. The first principal components constitute the normal part whereas the last principal components constitute the anomaly part. The norm of the anomaly part defines an anomaly distance and if the distance is above a threshold, the point is declared an anomaly. Liu et al. state that PCA has not been applied in window mode previously. They use variance histograms to perform updates on the PCA model.

Huang et al. [70] also use a projection to detect anomalies. They maintain a set of few orthogonal vectors that describe normality. They present algorithms for deterministic and randomized streaming updates for left singular vectors to maintain the vectors. If a new data point does not project well on the vectors or its residual is large, the point is identified as an anomaly. Huang et al. state that their method is close to real-time. The method can be extended to include decremental forgetting, i.e. operate in sliding window mode.

Ahmed et al. [35] present another method that uses projections. They build a dictionary to describe the normality using kernel recursive least squares. A new data point is then projected onto the current dictionary and projection error is computed. Based on the error, the point is then defined ”green”, ”orange” or ”red”. Green points are normal and red points are immediately defined as anomalies. Orange points are analysed later after some additional data is seen. This results in more trustworthy analysis but all anomalies cannot be spotted immediately as they arrive. The usefulness of orange points is tested and they are added to the dictionary if they are useful enough in classifying other points. The dictionary vectors are also checked regularly to see if they have become useless recently and should be removed from the dictionary.

3.2.9 Other

Otey et al. [82] present an algorithm for mixed-attribute datasets that uses "links" to take into account dependencies between the attributes in the data set. Anomalies are points that violate the dependencies. Links in categorical space are defined such that two points are linked if they share at least one attribute - value pair. Link strength equals the number of shared pairs. An anomaly is a point that has very few or very weak links to other points. Otey et al. present a way to compute approximations of the anomaly scores fast.

For continuous attributes, a covariance matrix is incrementally maintained to capture dependencies. Matrices are maintained for different attribute combinations. This captures dependencies between values in mixed attribute space. Two points are linked in the mixed attribute space if they are linked in categorical space and their continuous attributes are coming from the same distribution. Whether they come from the same distribution is determined based on how much the points violate the covariance matrix. To calculate the violation of a point, covariance score for each pair of continuous attributes is compared with the covariance matrix. Anomaly scores are then compared to average anomaly scores.

The time complexity grows exponentially with the number of categorical attributes and the number of their possible values and quadratically with the number of continuous attributes. Otey et al. present two techniques to reduce the exponential time complexity part but the time complexity still can be quite challenging. Otey et al. also state that the performance of their algorithm decreases when number of categorical variables decreases. Therefore, the method is not well suited for datasets with only continuous attributes or where the fraction of categorical attributes is small.

Bu et al. [43] present a method for detecting anomalies of trajectories, e.g. abnormal routes of a person. A base window as well as left and right windows (before and after the base window, respectively) are defined and the number of neighbours are computed. Neighbour is a trajectory in either left or right window that has a small enough distance to the base window trajectories. If number of neighbours is too small, the base window is an anomaly. Pruning and incremental clustering are used to obtain an efficient on-line method.

Hill et al. [67] present an approach based on dynamic Bayesian networks. Bayesian networks are directed acyclic graphs where each node contains probabilistic information regarding all possible values of a state variable. Dynamic Bayesian networks are Bayesian networks that evolve over time. The state variables can be hidden or observed and discrete- or continuous-valued. Inference is done via Kalman filtering or Rao-Blackwellized particle filtering and parameters are learned via expectation maximization algorithm. The anomalies are detected from Bayesian credible intervals or maximum a posteriori measurement statuses. The approach is constructed for two streams and Hill et al. state that adding more streams would be non-trivial. Therefore, the approach is not really suitable for handling a large amount of streams.

Salem et al. [90] first perform discrete Haar wavelet transform to divide a data point into approximation and detail signals. The approximation signal results from passing the data through a low pass filter and an inverse low pass filter and the detail signal from high pass and inverse high pass filters. Energy of a signal is defined as sum of squares of attribute values. Abnormal deviations are detected from the ratio of the energy of the detail signal with the total energy of both signals. Non-seasonal Holt-Winters model consisting of a level and a linear trend is used to predict the energy ratio. Difference of true and predicted energy ratios is assumed to be normally distributed and the parameters are estimated from a sliding window. Instead of mean and standard deviation, median and median absolute deviation are used as the parameters of the distribution. This avoids the effect of anomalies on the distribution but requires more memory to compute the parameters. After estimating the parameters, a statistical test is performed to see if the difference between true and predicted energy ratios is abnormal. Quantile ranges can be used to efficiently identify anomalous attributes from anomalous data points to try and identify causes of the anomaly.

3.2.10 Summary

In the survey it is presented that there exist many methods for anomaly detection that perform incremental updates. However, many of them are limited in practice. Some methods have high time complexity with respect to the data dimensionality, some are not even suitable for multi-dimensional data. Some require a lot of memory. Some perform incremental updates in an iterative manner, such that fast convergence is typical but cannot be guaranteed. Some require very complex data structures. Some require costly retraining if big model changes are required or concept drift occurs. Some require the data to be on a certain range. Some do not present a thresholding technique for defining anomalies.

Lightweight on-line detector of anomalies (LODA) is designed to do real-time model updating on even high-dimensional data. It has deterministic time complexity which can be even sub-linear w.r.t. data dimensionality. LODA is also able to handle missing values which is useful in real-world scenarios. It also allows to compute the effect of each dimension on the anomaly score, which is not always easy for high-dimensional data, but is useful in practice as there might be need to track down the cause of the anomaly. Therefore, LODA was chosen for further investigation in this work.

3.3 Lightweight on-line detector of anomalies

In this section the chosen method, Lightweight on-line detector of anomalies, is presented in more detail.

LODA consists of k one-dimensional histograms $h_i, i = [1...k]$. The histograms are formed by mapping data onto random projection vectors $w_i, i = [1...k]$. Let $x_j \in \mathbb{R}^d, j = 1...n$ denote the n d-dimensional data points. The anomaly score for point x_j is given by

$$f(x_j) = -\frac{1}{k}\sum_{i=1}^{k} \log \hat{p}_i(x_j^\top w_i) = -\log\left(\prod_{i=1}^{k} \hat{p}_i(x_j^\top w_i)\right)^{1/k}, \tag{3.1}$$

where $x_j^\top w_i$ is the projection of sample x_j on the weight vector of histogram i and $\hat{p}_i(x_j^\top w_i)$ is a density estimate at $x_j^\top w_i$ given by histogram i. For each projection vector, only $\sqrt{d}$ randomly chosen dimensions are non-zero. The non-zero elements are randomly drawn from $N(0, 1)$.

If the distributions p_i for the k projections were independent, $\prod_{i=1}^{k} \hat{p}_i(x_j^\top w_i)$ in the above formula would describe the joint probability distribution of the projections. Then the formula would become

$$f(x_j) = -\log p(x_j^\top w_1, x_j^\top w_2, ..., x_j^\top w_k).$$

However, since the independence assumption does not hold in practice, it is an approximation.

3.3.1 Histogram construction

When a sample x arrives, the anomaly score is computed and after that the histograms are updated. Three choices for constructing the histogram and estimating probabilities are considered, namely fixed-bin, equi-width and on-line histograms.

In fixed-bin histogram the bins are defined beforehand and for each new sample the bin count for the corresponding bin can be updated. The probability estimate of sample $z = x^\top w$ can be computed as the number of instances in the bin where z falls divided by the number of samples in total.

In equi-width histogram the width of the bin is defined beforehand. Denote the bin width by δ. Then bin for sample z is given by $b = \lfloor z/\delta \rfloor$, where

$\lfloor\rfloor$ denotes the floor operation. If b is not in the histogram, it is added there. If b is in the histogram, it's count is increased by one. Probability estimates are computed similarly as for the fixed-bin histogram.

In an on-line histogram the bins are updated constantly. Only the number of bins is defined beforehand. For a point z, it is checked if z already is in the histogram. If it is, the bin count is increased by one. If it is not, z is added to the histogram with bin count 1. Then two bins closest to each other are combined. Denote these two bins by z_i and z_{i+1} and the corresponding bin counts by m_i and m_{i+1}, respectively. These are replaced by a new bin $\frac{z_i m_i + z_{i+1} m_{i+1}}{m_i + m_{i+1}}$ with bin count $m_i + m_{i+1}$. The probability estimate of z is given by a weighted average of counts of the closest bins smaller and larger than z in the histogram.

For the fixed-bin histogram, for good choice of histogram bins it is required that the range of the data is at least approximately known beforehand. Then the bin boundaries could be set so that the bins cover the range of the data. Required space is determined by the number of bins. The equi-width histogram does not require knowledge of the range of the data but some value for bin width must be determined and a good choice requires some knowledge of the data. Time for updates and computing probability estimate is $O(1)$. Required space is not restricted but can in theory grow infinitely. The on-line histogram does not require any knowledge of the data range or distribution beforehand but it is not as fast as the equi-width histogram. Required space is determined by the number of bins.

3.3.2 Missing values and determining anomalous features

Missing values can be handled by calculating output from only the histograms whose projection vector's elements corresponding to missing variables are zero.

A feature's contribution to the sample's anomalousness is computed by a one-tailed two-sample t-test. The test statistic describes the difference in the anomaly score $f(x)$ (eq. 3.1) if only projections containing the feature were used vs. if only projections not containing the feature were used. If the difference is significant, the feature has a significant impact on the anomaly score. The test statistic is given by

$$t_j = (\mu_j - \bar{\mu}_j)/(\sqrt{s_j^2/|I_j| + \bar{s}_j^2/|\bar{I}_j|})$$

where I_j denotes the set of histograms that use the jth feature and $\bar{I}_j$ denotes the set of histograms that do not use the jth feature and μ_j and $\bar{\mu}_j$ and s_j^2

and $\bar{s}_j^2$ denote the mean and variance of $-\log \hat{p}_i$ calculated with $i \in I_j$ and $i \in \bar{I}_j$, respectively.

3.3.3 Drawbacks and improvement ideas to LODA

In this section I discuss some drawbacks of LODA and present some of my ideas to modify and improve LODA to overcome them.

LODA's output is an anomaly score for each data point. In practice, it would be useful to be able to define a threshold for the score to declare a point an anomaly. The anomaly score does not give an intuitive thresholding but as the score is based on probabilities, it could be mapped back to a number which would approximate the joint probability, namely by $\hat{p}(x) = e^{-f(x)}$, where $f(x)$ is the anomaly score given by equation (3.1). Probability is a more intuitive measure for thresholding and simple fixed thresholds could be used. Another way to perform theresholding would be to calculate the probability if the distribution was uniform and define a point an anomaly if the probability is at most some amount, e.g. 10% of the uniform probability. Also, different levels of anomalousness could be defined, e.g. mild anomaly if less than 50%, medium anomaly if less than 25%, strong anomaly if less than 10% and severe anomaly if less than 5% of the uniform probability. With multiple histograms, defining the probability under uniform distribution could be based on the mean or median of the number of bins.

The method assumes different variables to be approximately on the same scale. Imagine for instance variable 1 having range [0,1] for normal values and variable 2 having range [0,100] for normal values, i.e. previously observed values fill this range. Then a value 2 for variable 1 would be very anomalous, but the effect would disappear under relatively small fluctuations of variable 2, when the score is computed by a weighted sum of the variables. If the range of the variables is approximately known beforehand, some scaling techniques can be used or the weights can be adjusted accordingly. However, if nothing is known about the scales beforehand, this could be problematic. Then, some on-line normalization schemes would be required.

Squashing functions can be used to map the data to some interval, e.g. between 0 and 1. An example is the logistic function, $f(x) = \frac{1}{(1+e^{-x})}$, another is $f(x) = \frac{x}{x+1}$. However, squashing functions are a problematic approach for anomaly detection since while values in the middle of the range get mapped more or less linearly, extreme values get mapped close to each other and close to normal values. Another approach would be to map the values seen so far on part, e.g. 80%, of the range and the rest are squashed. This is a slightly better approach since at least most of the normal values get mapped to the linear range. However, it still does not make a big distinction between very extreme and almost normal points.

Unlike several other methods presented in section 3.2, LODA does not require the values to be mapped on certain range. Therefore, a possible approach would be to try to restrict most of the values to a given range but allow extreme values to flow outside the range. This could be obtained by taking a certain amount of samples from the beginning of the stream and using them to define the range, e.g. by $\hat{x} = \frac{x}{|max(X)-min(X)|}$, where X denotes the set of n points $x_1, x_2, ..., x_n$ used for scaling. If LODA was operated in sliding window mode, also the mapping could be done in sliding window mode, always using the range of the previous window for scaling the values in the next window.

If probability estimate of a point is zero, logarithm cannot be taken and thus LODA's anomaly score cannot be computed. This happens when there is no other point in the histogram bin where the current point is assigned. For fixed bin histogram and on-line histogram the number of bins is fixed (e.g. 20) but for fixed width histogram the number of bins is not restricted but grows based on the data. Scores for these points could be omitted but sometimes it would be beneficial to be able to compute them. A small positive constant value could be assigned as the estimated probability if it otherwise would be zero. Another approach would be to use bin count of e.g. 0.5 for those bins were the count is 0. This would also mimic the behaviour of the histograms that the point is less anomalous if there are fewer points in the histogram.

Pevny et al. present for on-line histogram the formula for computing the probability as $p(z) = \frac{z_i m_i + z_{i+1} m_{i+1}}{2M(z_{i+1} - z_i)}$ where $z_i < z \leq z_{i+1}$ and $M = \sum_i m_i$. However, this can be negative if z_i and z_{i+1} are negative. Therefore, there is probably an error in the formula. In the text it is said that the probability should depend on a weighted average of the bin counts. This could be given by $p(z) = \frac{(z_{i+1} - z)m_i + (z - z_i)m_{i+1})}{M(z_{i+1} - z_i)}$.

For on-line histogram, some small value δ could be defined such that if the distance of two points is at most δ, they are considered the same value. This would decrease computational cost, as bins would not have to be combined as often, without affecting the results much.

Pevny et al. claim that there are no hyperparameters that should be tuned manually. However, the number of histograms and number of histogram bins have to be set. It is mentioned that the optimal number of histograms should be based on how much variance is reduced after adding a new histogram. For selecting the number of bins, Pevny et al. present a likelihood function that should be maximized to choose an optimal number of bins. However, both of these require a training set which is assumed not to be available in the streaming case, and it would be practical to get some rules of thumb for selecting the parameters.

I suggest an approach for adjusting the bin width on the go after an initial guess has been made. This is relevant for the fixed width histogram as the number of bins is not restricted but can grow indefinitely. If a good number of histograms is found, denoted by k, at certain intervals the bin width, denoted by δ, can be adjusted such that the new bin width is $\hat{\delta} = \delta * (k/\hat{k})$ where $\hat{k}$ denotes the number of bins in a histogram at the moment.

Chapter 4

Data sets

4.1 Synthetic data sets

Synthetic data sets were created to assess the performance of LODA, to construct hypothesis on what kind of data LODA would or would not be suitable for and to test improvement ideas.

First, data sets were created to evaluate the parameters that should be used with LODA. 31 different data sets were created to get some variability to be able to better assess the most suitable parameters. All data sets contained $n = 100000$ samples and the amount of anomalies was 0.1% of the number of samples, i.e. there were 100 anomalies, which were inserted in random positions. The number of anomalies was set to be low to imitate real situations where anomalies are rare. Data was sampled either from i) sine function with noise from the normal distribution, ii) t-distribution or iii) two-modal uniform distribution. For the sine and t-distribution data sets anomalous values were peak values outside the range of normal data. For the two-modal uniform distribution anomalous values were values between the two modes. Dimension of the data was either 5, 10 or 100. For anomalous points, the number of dimensions containing exceptional values was varied. For some of the data sets generated from t-distribution, there was some correlation in the values between different dimensions. The 31 data sets were obtained from these parameter combinations. This created data sets that originated from different distributions and had different level of difficulty in terms of detecting anomalies to obtain more generalizable results. More detailed description of the data sets can be found in Appendix A.

Second, eight data sets were created to test the adaptation of LODA to concept drift. The data sets had either gradual or sudden concept drift. The data sets were 100-dimensional, they contained $n = 60000$ samples and there were three different concepts in each data set. In the gradual concept drift data sets there were additionally two shift phases. The data sets are described in more detail in Appendix A.

Third, 32 data sets were created to demonstrate how different dimensions having different scale affects the performance of LODA (see section 3.3.3). The data sets contained $n = 100000$ samples of which 100 were anomalies and the data had 10 dimensions. The data sets had either different range or scale in different dimensions. The data sets are described in more detail in Appendix A.

4.2 Publicly available data sets

In this section publicly available data sets, that are suitable for testing anomaly detection methods, are investigated. In addition to the synthetic data sets, these were used for testing LODA.

It would be preferable to have labels (normal / anomaly) for the data points to allow efficient and robust evaluation of accuracy. Data sets without labels have also been used for testing [80, 82, 91] but it requires manual identification of anomalous data points. The fraction of anomalies in the data set should be small to resemble real world scenarios. The data set used should also be large to avoid effect of random fluctuations. Especially when the fraction of anomalies is small, a large data set is needed so that evaluation is based on more than a handful of anomalies in the data set.

The UCI machine learning repository [78] contains many data sets that have been used in different machine learning tasks. One of them is the *KDD Cup 99* data set which has been widely used to test anomaly detection methods [40, 44, 76, 82, 91, 94, 99]. The data set contains network traffic including normal traffic and attacks. The intrusions have been simulated in a military network environment. There are around 5 000 000 data points and the data set contains 41 features out of which 34 are continuous and 7 are categorical. The amount of intrusions is high (80%) so to use it meaningfully in experiments it would be required to reduce the amount of anomalies significantly. Tavallaee et al. [95] demonstrate that there are some issues with the data set. Many of the records appear multiple times in the data set which biases learning and method evaluation. This should be taken into account and duplicate events could be removed.

The UCI machine learning repository also contains two data sets which monitor traffic amounts and are associated with events organized nearby. It is expected that before and after events traffic is higher than usual and these points could be identified as anomalies. The *Dodgers loop sensor* data set contains count of cars every five minutes and is associated with Dodgers game events nearby. The *Callt2 building people counts* data set contains counts of people walking in and out of a building every half an hour and is associated with events organized in the building. The *Dodgers* data set contains 50 000 instances and the amount of events is 81 whereas the *Callt2* data set contains 10 000 instances and the amount of events is 30.

Data sets used to test methods that identify concept drift are of course not sufficient alone to test anomaly detection but they could be used to test the adaptiveness of anomaly detection methods. When concept drift occurs, the first points should be identified as anomalous but then the model should adapt to the change and the following points should be considered normal. *SEA dataset* [92] is a generated dataset that has been used to test many concept drift methods [60]. It contains 60 000 samples divided into four concepts. There are 3 continuous and 1 categorical attributes.

Chapter 5

Experiments

In this chapter experiments on LODA, conducted for both synthetic and real data sets, are presented. Suitable parameter choices are addressed and some of the improvement ideas presented in section 3.3.3 are tested, focusing on normalizing the data and finding ways to set a threshold for the anomaly scores to define anomalies. Adaptation and its speed are also evaluated. These experiments are first conducted on synthetic data sets and then on real data sets.

5.1 Performance measures

Results were evaluated by comparing observed anomalies to true anomalies. As a measure mainly harmonic mean of precision and recall, F-score, was used. FP denotes the number of false positives, FN the number of false negatives, TP the number of true positives and TN the number of true negatives. Then,

- True positive rate or recall is defined as TP/(TP+FN).
- False positive rate is defined as FP/(FP + TN).
- Precision is defined as TP/(TP + FP).
- F-score is defined as 2TP/(2TP + FP + FN) or precision * recall / (precision + recall).

Receiver operating characteristic (ROC) curve, which shows false positive rate on x-axis and true positive rate on y-axis, or area under the ROC curve are often used measures. However, since false positive rate depends on the number of true negatives, whose proportion in anomaly detection is large, it is not a very good measure for assessing anomaly detection methods. It gives over optimistic results as the false positive rate grows very slowly. This was also noticed in the experiments, area under the curve was typically very close to one. Therefore, ROC curves are not shown in the results as they contain little information and make the results look better than they actually are.

The F-score can be presented for different thresholds on the anomaly scores. Here, mainly the best F-score for each data set is shown. This is an unrealistic theoretical best performance score but it allows easy, summarized comparison between the different experiments.

5.2 Experiments on synthetic data sets

The tests in sections 5.2.1-5.2.2 were performed using the first data set described in section 4.1. The data sets were generated from different distributions, had different number of dimensions and different level of anomalousness. The goal was to find optimal parameters for using LODA.

5.2.1 Zero probability replacement

Two different options for replacing zero probabilities as explained in section 3.3.3 were tried. First, assigning a very low probability (10^{-10}) that would mimic the probability being zero. Second, the probability was set to be $0.5/i$ where i is the number of samples seen so far. These experiments were run using the fixed width histogram with 10 histograms and bin width being either 1 or 2. Results can be seen in Figure 5.1. It can be observed that the results are better for the second option and hence that was used in all further experiments. The reason the results are better for the second option is probably due to that the first option gives a very low score and hence those points are identified as anomalies. This causes false positives especially in the beginning when there are not yet samples in the histograms.

All the F-scores are shown in a table in Appendix B.

5.2.2 Parameter settings

The three different histogram options (see section 3.3.1), fixed bin histogram, fixed width histogram and on-line histogram, were tested to see if there are differences in the results or in the speed of the algorithm. Ten histograms were used and the bin width for fixed width histograms and the bin boundaries for fixed bin histograms were varied. Twenty bins were used for fixed bin and on-line histograms. Results can be seen in Figure 5.2. On-line histogram

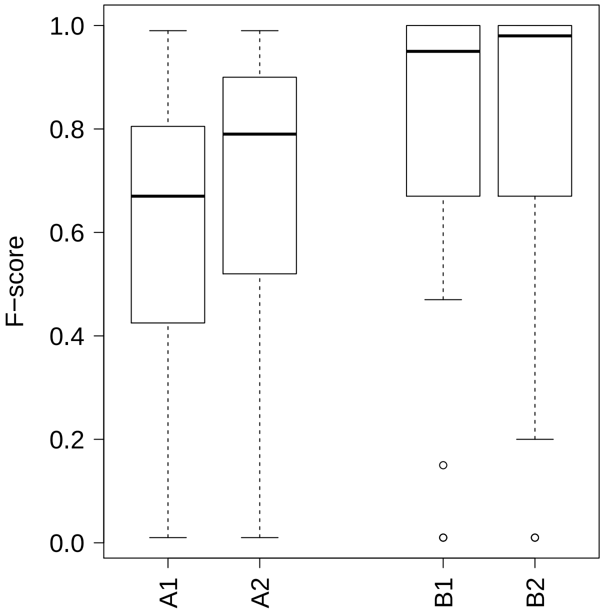


Figure 5.1: Boxplots of best F-scores showing results for two different zero replacement techniques. "A1" and "A2" denote zero probability replacement with 10^{-10}, using bin widths 1 and 2, respectively. "B1" and "B2" denote zero probability replacement with $0.5/i$ (where i is the number of samples seen so far), using bin widths 1 and 2, respectively.

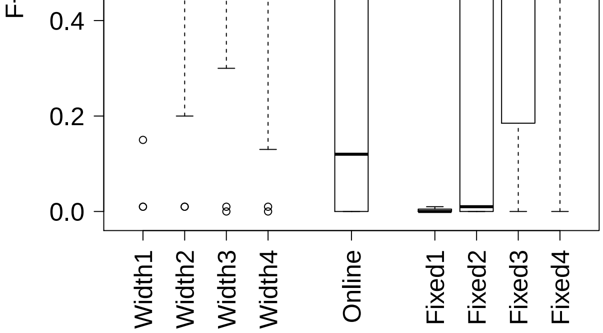


Figure 5.2: Boxplots of best F-scores using different histogram constructions. "Width1", "Width2", "Width3" and "Width4" denote fixed width histograms with bin widths 1, 2, 4 and 0.5, respectively. "Online" denotes the on-line histogram. "Fixed1", "Fixed2", "Fixed3" and "Fixed4" denote fixed bin histograms with bin boundaries uniformly divided in the ranges [-1,1],[-5,5],[-10,10] and [-15,15], respectively.

has worst performance and fixed width histograms have best performance. There is big variety in the performance of fixed bin histograms and the performance is highly dependent on the choice of the bin boundaries. On the other hand, for fixed width histogram the bin width does not have a great effect on the scores for most of the explored data sets. In addition, running times (on a single laptop, using one core, implementation in Scala, including reading the input from a CSV file one line at a time) with 10 histograms were for both fixed bin and fixed width histograms 0.002-0.03 ms/event, depending on the dimensionality of the data set, whereas for the on-line histogram running times were 0.2-4 ms/event. Since the fixed width histogram is both best performing and fast, it was used in all further experiments.

It was tested how much varying the number of histograms affects the results and what would be a good choice for the number of histograms. Given the dimensionality of the data is d, each histogram uses $\sqrt{d}$ dimensions. The hypothesis was that each dimension should be used at least once to get a representative estimate of the data. If the number of histograms is less than $\sqrt{d}$, each dimension is used on average less than once. If the histogram count is slightly higher than $\sqrt{d}$, each dimension is used on average more than once but it is very likely that some dimensions are not used at all. Therefore, it was expected that more than $\sqrt{d}$ should be used to get good results.

This was tested using 2, 5, 10, 20, 50 and 100 histograms. Results are shown in Figure 5.3. It can be seen that with higher number of histograms the results are often better. With higher-dimensional data higher number of histograms is required. This supports the hypothesis.

The time complexity is linear with respect to the number of histograms. Figure 5.4 shows for the 100-dimensional data sets how the time grows as the number of histograms grows. It can be observed that especially for the maximum time out of the data sets the relationship indeed is linear. For the minimum time, there is some more fluctuation.

Here it is assumed that in practice it might be impossible or impractical to use a very large number of histograms, especially if the data is very high-dimensional, even though it would give better results. Therefore, in further experiments as many histograms as there are dimensions were used to get realistic experimental settings. If data dimensionality is high, a lower number of histograms would also be possible. For instance, in Figure 5.3 it can be observed that for 100-dimensional data using 50 or 100 histograms gives equally good results. However, if time and space permit, it would be recommended to use histogram counts even higher than the dimensionality of the data.

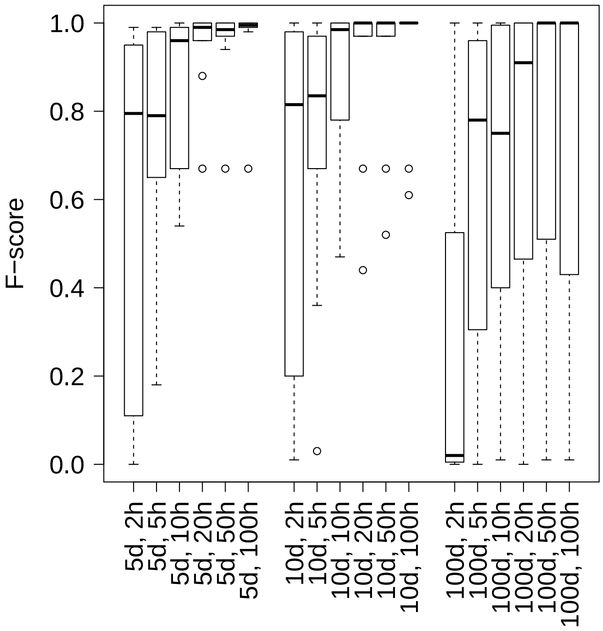


Figure 5.3: Boxplots of best F-scores using different number of histograms. Labels give the numbers of dimensions and histograms, e.g. "5d, 2h" denotes five-dimensional data sets where two histograms were used.

Here are some remaining general remarks.

- Results with both sine curve and t-distribution are fairly good in general (F-score above 0.8 in most cases), whereas results for the multimodal distribution are not (F-score 0.6-0.7). A possible reason for this is that for the sine curve and t-distribution data sets, anomalies deviated more from normal data, whereas for the multimodal distribution anomalies had values quite close to normal values.

- If there is an anomalous value in only one of the dimensions for anomalous points, the results are quite poor (F-score below 0.2 in most cases). Accuracy seems to increase as the number of dimensions containing exceptional value increases, as would be expected. For instance, when all dimensions contained exceptional values, F-scores were above 0.9 in most cases. However, further experiments should be done to get more reliable estimates on how many of the dimensions should have anomalous values to get good enough results.

- The number of bins in each histogram were looked at. It seemed that

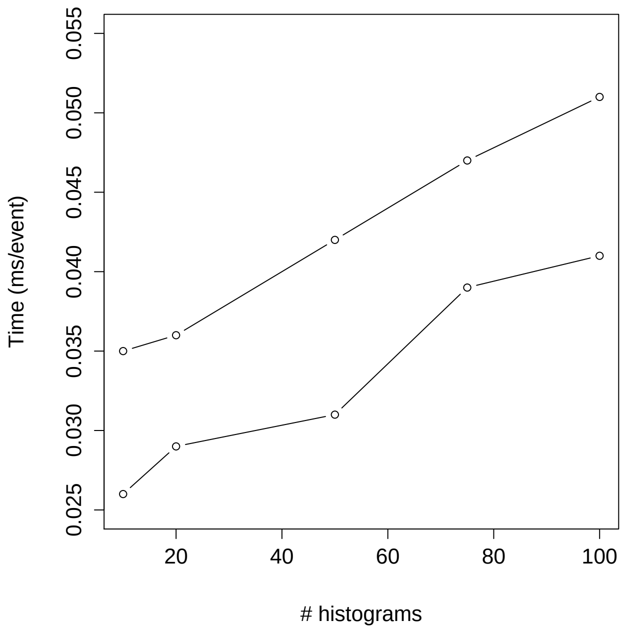


Figure 5.4: Time (ms/event) used by LODA for the 100-dimensional data sets using different number of histograms. The minimum and maximum out of 11 data sets is shown.

> results were best when bin counts did not vary a lot between the histograms and when bin counts were rather low (< 20). However, it was not possible to identify any rule of thumb for the bin width based on this. But, bin width did not seem to affect the results significantly. Therefore, in further experiments bin width 1 was used.

All the F-scores for the above experiments are shown in tables in Appendix B.

5.2.3 Concept drift

Since one of the advantages of LODA is on-line model updating and therefore adaptation, it should be able to adapt to concept drift. Here speed of adapting of LODA in theory is discussed and it is tested if adaptation follows the theory in practice as well.

Assume that an anomaly threshold of δ is used and a sudden concept drift occurs after n samples have been seen and inserted into the histograms. Assume for simplicity that there is only one histogram and all the samples from the new concept fall in the same bin. Then, the time to adapt, i.e. when a sample from the new concept is no longer considered an anomaly but the probability exceeds the threshold δ, is given by solving k from $k/(n+k) > \delta$. If, for instance, $\delta = 0.01$ and $n = 10000$, this gives that $k \geq 102$. In other words, 101 first samples from the new concept are considered anomalous. This can be quite a big number depending on the use case, at least if all anomalies are handled manually. In addition, all the samples from the new concept are unlikely to fall in the same bin. Therefore, even longer adaptation time is expected.

LODA was run on the synthetic concept drift datasets described in section 4.1. It was checked which samples had anomaly score mapped as probability (see section 3.3.3) less than 0.01. First concept drift was after 20000 samples and second after 40000 samples. For all the datasets, LODA performed as expected. New concepts were identified every time, i.e. anomalies were detected at the point of concept drift. Average number of data points after which the samples were no longer identified as anomalies, i.e. the method had adapted to the concept drift was for the first concept drift 183 (median 191) and for the second concept 381 (median 391). These results correspond to the theoretical calculations.

The results are highly dependent on the chosen threshold level and with tighter thresholding adapting would be faster. For instance, using a threshold of 0.004 all concepts are still identified but the time to adapt is much lower.

5.2.4 Normalization

It was tested how LODA performs on data having different scale in different dimensions and if the proposed normalization techniques improve the results for such data sets. Experiments were run on the third data set described in section 4.1, which contained data having different range or scale in different dimensions. LODA was tested without preprocessing the data, then using a few off-line normalization techniques to see the difference. Finally, it was experimented with some more or less on-line normalization techniques.

Denote by X the data set, by $x \in X$ a sample of the data set and by $\hat{x}$ the normalized sample. Off-line normalization techniques used were

1. $\hat{x} = x/max(X)$. Dividing by the maximum of the data set scales the data linearly in the range [0,1] (in case the values are non-negative).

2. $\hat{x} = x/(max(X) - min(X))$. Dividing by the difference of maximum and minimum values sets all data sets to have equal scale, although their range might be different.

3. $\hat{x} = (x - min(X))/(max(X) - min(X))$. In addition to the previous one, first subtracting the minimum of the data set sets all values to start from zero (in case the values are non-negative). Thus, the values are also approximately in the same range.

4. $\hat{x} = (x - mean(X))/sd(X)$. This is the commonly used z-normalization, subtracting the mean and dividing by the standard deviation of the data. This sets the mean of the data set to be zero and its standard deviation to be one.

These are referred to as off-line normalization techniques 1-4.

On-line normalization techniques were

1. Same as off-line technique 1, except using only first 1000 samples to define $max(X)$.

2. $\hat{x} = x/(x + 1)$. This sets the data values between [-1,1] and does not require any parameters.

3. Same as off-line technique 2, except using only first 1000 samples to define $max(X)$ and $min(X)$.

4. Same as off-line technique 3, except using only first 1000 samples to define $max(X)$ and $min(X)$.

These are referred to as on-line normalization techniques 1-4, respectively.

The results for data sets where dimensions had different scale and anomalies appeared in only the smaller-scale dimensions can be seen in Figure 5.5. Unnormalized data sets gave very poor results. Both off-line and on-line normalization techniques improved the results, z-normalization giving the best results. It can be observed that on-line normalization technique 2 worked poorly but other normalization techniques worked quite well. The second on-line technique working poorly was expected as it cuts big values and scales unevenly, which is unsuitable for anomaly detection. For these data sets, on-line techniques also perform well compared to off-line techniques.

A Wilcoxon signed-rank test was performed to see the statistical significance of the improvement. The Wilcoxon signed-rank test is equivalent to a paired t-test except that the variables are not assumed to be normally distributed. The test was conducted between the unnormalized data sets paired with the same data sets that had been using on-line normalization technique 1. A one-sided test was performed testing if the on-line method gives better results, i.e. if the scores are higher than for the unnormalized data sets. The

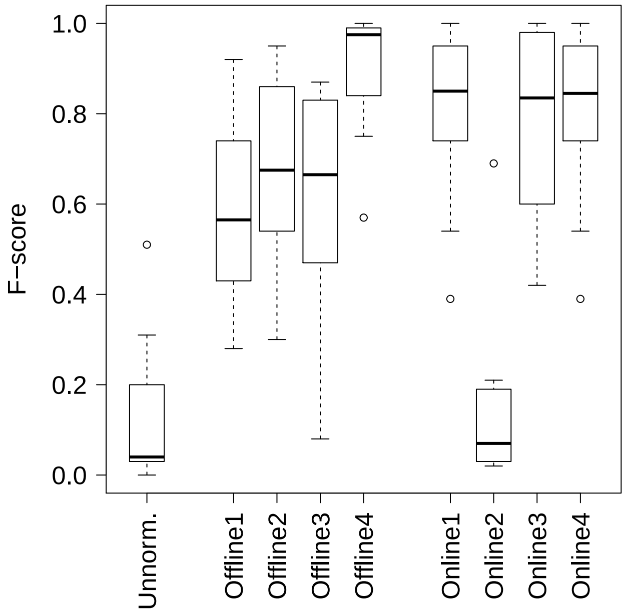


Figure 5.5: Boxplots of best F-scores using different normalization techniques. "Unnorm." refers to unnormalized data. "Offline1", "Offline2", "Offline3" and "Offline4" refer to off-line normalization methods 1-4, respectively. "Online1", "Online2", "Online3", and "Online4" refer to on-line normalization techniques 1-4, respectively. See text for further details.

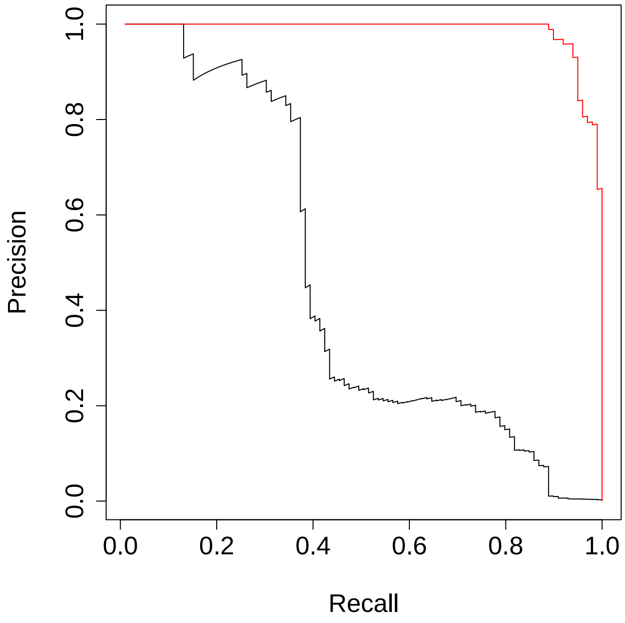


Figure 5.6: Precision-recall curve for one of the normalization data sets. The curve for unnormalized data is shown in black and the curve for data normalized using on-line technique 1 is shown in red.

p-value obtained was 0.0009766 indicating that indeed the on-line method performed better.

A precision-recall curve is shown for one of the data sets in Figure 5.6, both without normalization and using on-line normalization technique 1. The data set chosen was the one that performed best without any normalization. There it can be observed that the improvement is not only at a single point but there is quite a big difference in the precision-recall curve.

For the other normalization data sets, unnormalized results were good and using normalization techniques did not affect the results much, although on-line normalization technique 2 worked poorly for those data sets as well.

All the results for the normalization experiments can be seen in a table in Appendix B.

5.2.5 Scoring

So far the results have been shown using the best F-score as a measure. However, in practice, it is not trivial to obtain the best F-score since it requires correctly guessing the best cutoff value. The scores were mapped to probability as explained in section 3.3.3 and a couple of schemes to define a good threshold for defining anomalies were designed. The performance of these thresholding schemes is shown compared to the unrealistic theoretical best performance scores.

First approach would be to use fixed thresholds, e.g. 0.01, 0.02 or 0.05. However, it was examined which cutoff values give the best F-scores and it was discovered that the best cutoff values vary a lot. Therefore, another approach was designed which uses the amount of bins in determining the threshold. The average number of bins, m, in each histogram was computed after 1000 samples had been seen. Its multiplicative inverse, $1/m$ was computed – this would give the probability of a sample given the samples were uniformly distributed, i.e. all the bins received equal number of samples. The multiplicative inverse $1/m$ was divided with a fixed number, 3, 5 or 7. In other words, a point was defined an anomaly if its probability was one third, fifth or seventh of uniformly distributed points.

F-scores using these methods can be seen in Figure 5.7. It also shows the best F-scores as a reference. It can be seen that the scoring techniques do not reach the best F-scores, as expected. However, most of the thresholding techniques give fairly good results compared to the best scores.

For further comparison, Figure 5.8 shows the precisions for the same settings. It can be seen that the bin width dependent techniques as well as the fixed threshold at 0.01 give fairly good scores. With these data sets

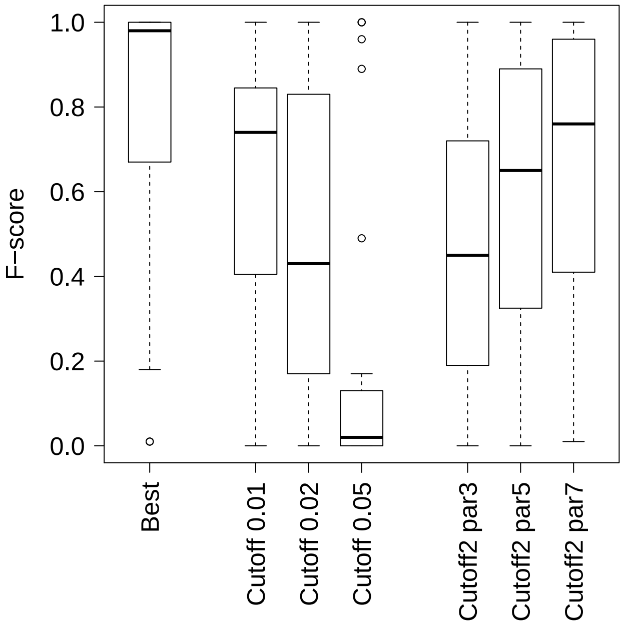


Figure 5.7: Boxplots of F-scores using different thresholding techniques. "Best" refers to the best F-scores. "Cutoff 0.01", "Cutoff 0.02" and "Cutoff 0.05" refer to fixed cutoffs at 0.01, 0.02 and 0.05, respectively. "Cutoff2 par3", "Cutoff2 par5" and "Cutoff2 par7" refer to cutoffs defined by bin width dependent technique with parameters 3, 5 and 7, respectively. See text for further details.

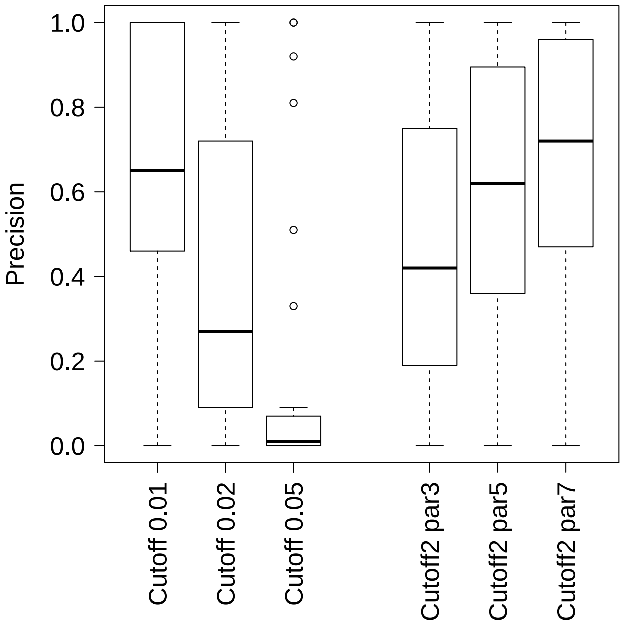


Figure 5.8: Boxplots of precisions using different thresholding techniques.

the results seem to improve as the parameter for the bin width dependent thresholding technique is increased.

All results for the scoring experiments are shown in a table in Appendix B.

5.3 Experiments on real data sets

In addition to synthetic data sets, experiments were run on four real data sets presented in section 4.2 to see the performance of LODA on real world data sets. Here it is shown how the improvements work for the KDD data set and how LODA is able to find anomalies for the other three data sets.

5.3.1 KDD cup 99 data set

Since the KDD cup data set contained an excess anomalies, it was preprocessed to set the proportion of anomalies to 0.1%. First, duplicate anomalies were removed after which the data set contained 20% anomalies. Then, fifty data sets were generated such that anomalies were randomly sampled from the set of all unique anomalies. Out of the 42 features, the four that contained non-numerical values were removed. These fifty data sets are dependent since they contain the same normal data and anomalies were sampled with replacement. Therefore, the differences in results when showing them on all the data sets seem higher than they actually are.

KDD cup data set contained features with very different scales. Off-line normalization techniques 1 and 4 and on-line normalization techniques 1-4 (see section 5.2.4) were used. Results can be seen in Figure 5.9. The results are poor without any normalization. There is significant improvement when off-line normalization is used, especially z-normalization. On-line normalization using first 1000 samples also improves the results compared to unnormalized data but the scores are still quite far from those of off-line normalized data.

Out of the on-line normalization techniques, the one which was clearly worst for the synthetic data sets, performed best. This may be due to that the data is less regular and therefore cannot be well normalized based on the first 1000 data points. Although on-line normalization technique 2 in general is not most suited for anomaly detection, it maps different dimensions on the same scale, whereas the other techniques may still leave different dimensions on very different scales.

To experiment if 1000 samples was too little for normalization, normalizing was also tried based on the first 5000 points. Results are clearly better and comparable to off-line normalized.

Precision-recall curves are shown for two of the KDD data sets in Figure 5.10, for both unnormalized data and data normalized with on-line technique 1 using 5000 samples for normalization. The two data sets chosen were those that performed best without any normalization. It can be seen that the improvement indeed is quite big, although very high precision is not obtained even with the improved method.

The thresholding schemes explained in section 5.2.5 were tested on the KDD data sets. With fixed cutoffs 0.01 or 0.02, the F-scores and precisions were below 0.1 or undefined. The situation was similar with the bin count dependent cutoff scheme. Changing the threshold did not affect much. When performing the bin count dependent cutoff scheme after 5000 samples had been seen, results were much better. However, comparing to the best obtained F-scores the results are significantly worse. These results with parameter 3 are shown in Figure 5.11. Although for the synthetic data set stricter thresholding seemed better, for KDD data set the situation is the opposite. This might be due to bin counts being higher for KDD data sets and therefore the threshold gets very low if the parameter is very high.

Even though in the last experiments quite a lot of the data (10 000 samples) was spent for tuning, it was still only 1.2% of the data. The overall results are still not very good (F-score around 0.5) but results improved quite a bit from the first unnormalized results to normalizing such that only a relatively small subset of the data is discarded to be used for normalizing and thresholding such that anomalies can be reported in real time.

Finally, some experiments were conducted on the KDD data sets with different settings that did not improve the results:

- Using constant weights or weights drawn from $U(0, 1)$ instead of weights drawn from the normal distribution.

- Adjusting the bin width on the go based on the bin counts (see section 3.3.3). The reason why this did not improve the results could be that there was no turning point after which the bin counts would have stabilized, rather they kept increasing slowly. Hence, there is no single point when it would be expected that adjusting the bin width at that point should yield better results. However, in real use it probably would be good to check the bin counts at times and adjust them if needed.

- Weighing the different histograms in scoring differently based on the bin counts such that those having higher bin counts had higher weights. This could have helped since in the histograms where the bin counts are higher, probability of a point falling in a certain bin is lower. Giving

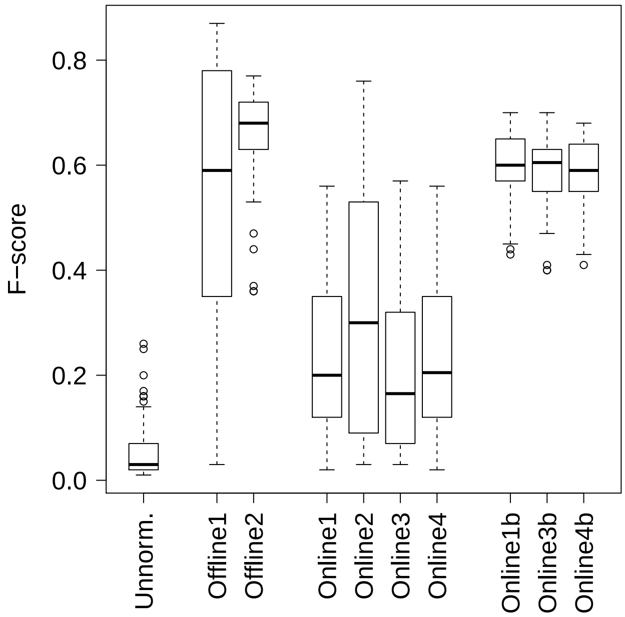


Figure 5.9: Boxplots of F-scores for KDD data sets. ”Unnorm.” refers to unnormalized data. ”Offline1” and ”Offline2” to off-line normalization techniques 1 and 4, respectively. ”Online1”, ”Online2”, ”Online3” and ”Online4” refer to on-line normalization techniques 1-4, respectively. ”Online1b”, ”Online3b” and ”Online4b” refer to on-line normalization techniques 1,3 and 4, respectively, using 5000 samples for defining normalization parameters. See text for further details.

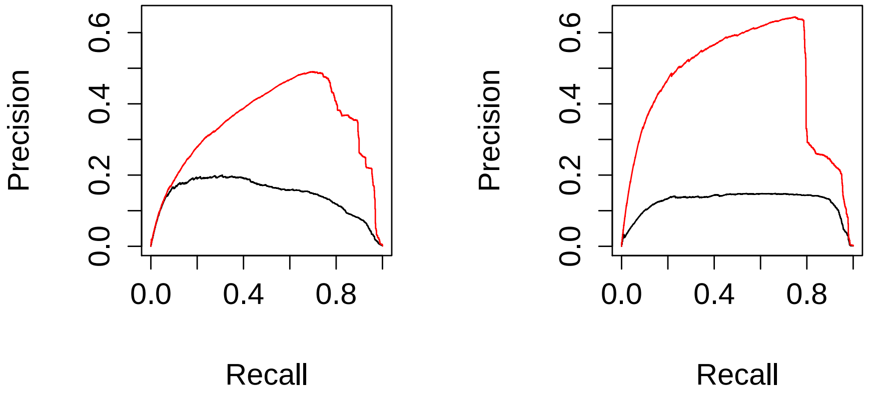


Figure 5.10: Precision-recall curves for two of the KDD data sets. The curves for unnormalized data are shown in black and the curves for data normalized using on-line technique 1 are shown in red.

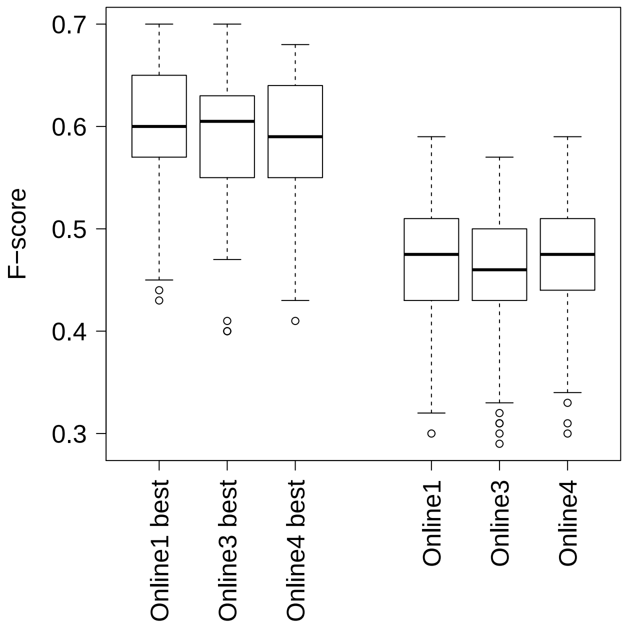


Figure 5.11: Boxplots of F-scores for KDD data sets using bin count dependent cutoff with parameter 3. On the left are shown the best F-scores for reference.

higher weight to histograms with higher number of bins would balance this.

The KDD cup data set has been used widely in the literature. Comparing to results obtained in other research is not trivial since different preprocessing techniques have been used. For instance, a much larger proportion of anomalies has typically been used than in this work. In those cases it is possible to use classification algorithms that are trained on part of the data and the amount of data used for training is typically a lot higher than the part used for normalization in this work. Also, since there are many replicas in the original data set, it is easy to classify them correctly if they have already been seen in the training set. Used measures also vary. Typically used measures are area under the curve (AUC), detection rate or true positive rate (TPR) and false positive rate (FPR). As discussed in the beginning of this chapter, FPR is not well suited for anomaly detection tasks. Using the best obtained settings for LODA (5000 samples for normalization with on-line normalization techniques 1, 3 and 4), the following results were obtained. AUC was above 0.98 in all cases and above 0.99 for many cases. Median of the detection rates, using bin width dependent thresholding with parameter 3, was 0.93-0.95, depending on the normalization technique used, while median of the false positive rates was 0.022-0.023. In light of these results LODA seems to be working well on this data set and is comparable to other methods presented in the literature (see e.g. [42],[44],[82],[94]).

The results varied quite a lot depending on which anomalies were chosen in the data set. Apparently some of the anomalies are harder to predict than others. This hypothesis is supported by Saphnani et al. [89] who state that some of the intrusions in the data set have been very difficult to classify for any algorithm.

5.3.2 Other real data sets

For the SEA data set, a similar approach was used as for the synthetic concept drift data sets: it was evaluated whether there are anomalies when concept drift occurs and if the method adapts, i.e. if the anomalies stop occurring after a while and how fast this happens. Concept drift was not identified by LODA. There were three concept drift points in the data set but none of them popped up as producing more extreme scores than other data points. Different normalization techniques were tried but those did not help either. Hence, LODA was not able to identify anomalies in this data set.

To better understand why LODA failed, differences in the SEA data set in different concepts were investigated. It was discovered that mean values for the three continuous dimensions varied between 4.95-5.04, 4.99-5.03, and 5.00-5.03 across the different concepts while the standard deviations within the concepts were approximately 2.9. The sum of all variables varied between 15.55-15.72 for different concepts while the standard deviations within the concepts were 5.2-5.3. Since LODA is based on weighted sums of the values in different dimensions, it is expected that it does not notice this small differences. Perhaps some other methods are able to identify some other kind of variation between the concepts.

For the Callt2 and Dodgers event data sets it was evaluated whether the anomalies occur at or near the times of the events. The points were sorted according to their anomalousness and 100 points with highest anomaly score were looked at.

For the Callt2 data set, among the top 100 scored data points, there were 50 points that matched with the event times or deviated by at most one hour from the event start and end times. Among the 100 top scored points, there were matches with the event times of 16 out of 30 events.

For the Dodgers data set, among the top 100 scored data points, there were 32 points that matched with the event end times or deviated by at most half an hour from one of them. Among the 100 top scored points, there were matches with the event times of 20 out of 81 events.

These results show that LODA is able to identify some of the events but many remain unidentified or are falsely identified. A challenge in evaluating the results is determining how much before and after the events would it be expected that the values should be higher. Choices for these experiments were done such that there would not be much different if the thresholds were varied slightly.

Using time of day as a contextual attribute could help in obtaining better results for these data sets, since especially for the Callt2 data set the inflow and outflow were higher during certain times of the day. Taking time of day into account could help in identifying values that are exceptional at certain times, not just in general.

Chapter 6

Discussion and conclusions

In this work I have given an overview of concepts related to streaming analytics, discussing challenges arising from real-time processing and analysis, and approaches to address these challenges. The discussion is beneficial for several areas and use cases around streaming analytics. Available software platforms and solutions for streaming analytics have been explored, and a survey on streaming anomaly detection methods has been conducted.

The survey shows that while there is plenty of research going on in the area, the field is still emerging and lacks well known and widely used methods. Normalization is one aspect that is not much covered in the research – many methods assume the data is normalized even though for streaming use cases this is not trivial. Many methods also are unable to operate on high-dimensional data or they have a high time complexity with respect to the data dimensionality.

Based on the literature survey I have chosen to investigate further Lighweight on-line detector of anomalies (LODA), which is one of the few methods that can achieve real-time model updating even for high-dimensional data. I have investigated that indeed it can process up to hundreds of thousands of events per second.

My experiments show how well LODA is able to identify anomalies. I have assessed parameter selection for LODA and proposed the most suitable parameters. For my synthetic data sets the performance is typically good and I have shown results on different kinds of data sets.

I have discussed a couple of defects of LODA and proposed ideas to overcome them. First, LODA is unable to provide the anomaly score for a small set of data points. I have provided a technique to compute the score for those points as well. Second, LODA can fail if proper normalization is not used. However, since off-line normalization is unsuitable for streaming data, some on-line techniques are required. I have designed these techniques and shown how they improve the performance of LODA. Third, to use LODA in practice, some kind of thresholding for anomalies is required. I have suggested mapping anomaly scores given by LODA to probabilities, which are easier to interpret, and I have suggested how to define the thresholding levels. The normalization and thresholding techniques presented are not restricted to be used with LODA but they could be used with other methods as well.

For real world data, the distribution typically is not known and does not follow a simple textbook form. Therefore, results on synthetic data sets cannot be generalized to real world situations. However, they give an understanding on the behaviour of the method and in what cases it might potentially fail. For the real data sets that LODA was tested on, there was still room for improvement in the performance of LODA. However, as I studied the performance of other algorithms on one of the data sets, which has been widely studied, I was unable to find methods that would show substantially better performance on this data set.

The thresholding and other settings that would be required if LODA were to be used in real world production systems have been assessed in this work. Future work would include investigating LODA on more real data sets to get a better understanding of its performance on different data sets. In real cases the required accuracy is also highly dependent on the use case and actions, as well as whether precision or recall should be emphasized more. For instance, if there is an action for each detected anomaly, it might be required that there are few false positives. But, it might be important to catch all exceptional events, e.g. malicious users, in which case there should be few false negatives. On the other hand, if anomaly detection is used for automating part of a process, a lower accuracy might be acceptable. In practice the thresholds probably would be adjusted slightly also on the go, depending on the data set and use case, but the approach presented gives a good starting point.

Instead of assigning a single threshold, anomalies could be given different severities based on how anomalous they are, i.e. setting thresholds for different anomaly levels. Anomalies could also be classified based on whether they appear as single anomalies or whether there is a set of consecutive anomalies.

This work shows that LODA is an adaptive method. However, adaptation is quite slow. For faster adapting, sliding windows could be used. Another possibility would be to ”silence” the anomalies after a certain number of consecutive anomalies.

LODA could also be used to monitor concept drift, e.g. a set of consecutive anomalies, and retrain the model when this is observed. This could be used for other use cases than anomaly detection as well.

For distributed computing windowing could be used such that local models would only be updated at certain intervals using a two-window model. When a new window is ready it would be sent to the center and combined with other local windows and sent back to the local device.

Some of the real data sets contained also categorical variables but their affect on the performance of LODA was not studied in this work. In theory LODA could also be used for categorical variables and there would be room for further investigations.

In this work using contextual variables with LODA was not addressed. LODA could take into account contextual variables by training a different model for each contextual variable. In practice, the user could divide the attributes into contextual attributes and attributes of interest. The contextual attributes could include device, time (e.g. time of day which could be divided into a couple of different sections: morning, day, evening, night) or location of the event. The attributes of interest would be those that are monitored and where deviations would indicate anomalies. Out of the contextual attributes the user could choose those that are interesting, e.g. if the time should affect on whether a point is considered anomalous. There could be some threshold to define how many data points from a certain contextual attribute are required such that it is possible to detect anomalies. A separate model could then be built for each of the contextual attribute combinations. This would require careful thinking such that there are not too many models (w.r.t. memory, speed, etc.) and that the possibilities stay clear to the user.

In addition, if most recent values are of most interest, the user could define a time frame that is of interest, e.g. each point would be compared to all events during the preceding week. This would define the sliding window of the model. Another approach would be a decaying factor such that effect of older events would decrease over time.

I see potential in LODA and recommend trying it out in real environments. The simplicity of the model also allows for easy modifications and further development of the method. However, further evaluations, especially with different kinds of data sets, are required to quantify the performance of the method.

Bibliography

[1] Amazon kinetics. https://aws.amazon.com/kinesis/streams/. Accessed: 10.05.2016.

[2] Anodot. http://anodot.com. Accessed: 23.05.2016.

[3] Apache flinks. http://flink.apache.org/. Accessed: 17.05.2016.

[4] Apama. http://www.softwareag.com/corporate/products/apama_webmethods/analytics/overview/. Accessed: 10.05.2016.

[5] Datatorrent. https://www.datatorrent.com/. Accessed: 17.05.2016.

[6] Espertech. http://www.espertech.com/. Accessed: 17.05.2016.

[7] Google data flow. https://cloud.google.com/dataflow/. Accessed: 17.05.2016.

[8] Guavus. https://www.guavus.com/platform/. Accessed: 12.08.2016.

[9] Ibm. http://www.ibm.com/analytics/us/en/technology/stream-computing/. Accessed: 17.05.2016.

[10] Infochimps. http://www.infochimps.com/infochimps-cloud/cloud-services/cloud-streams/. Accessed: 10.05.2016.

[11] Informatica. https://www.informatica.com/products/data-integration/real-time-integration/vibe-data-stream.html#fbid=MdIQDbu5906. Accessed: 23.05.2016.

[12] ipolicy networks. http://www.ipolicynetworks.com. Accessed: 02.06.2016.

[13] Kx systems. https://kx.com/. Accessed: 23.05.2016.

[14] Lambda architecture. http://lambda-architecture.net/. Accessed: 04.01.2017.

[15] Microsoft azure. http://azure.microsoft.com. Accessed: 10.05.2016.

[16] Nastel. http://www.nastel.com/products/autopilot-m6.html. Accessed: 23.05.2016.

[17] Numenta. http://numenta.com. Accessed: 20.07.2016.

[18] Odysseus. http://odysseus.informatik.uni-oldenburg.de/index.php?id=1&L=2. Accessed: 23.05.2016.

[19] One market data. https://www.onetick.com/products/onetick-event-stream-processing. Accessed: 23.05.2016.

[20] Oracle. http://www.oracle.com/technetwork/database/information-management/streams-fov-11g-134280.pdf. Accessed: 23.05.2016.

[21] Redlambda. http://www.redlambda.com/. Accessed: 23.05.2016.

[22] S4: Distributed stream computing platform. http://incubator.apache.org/s4/. Accessed: 11.08.2016.

[23] Sap hana. http://scn.sap.com/community/developer-center/streaming. Accessed: 23.05.2016.

[24] Sas event stream processing. http://www.sas.com/en_sg/software/data-management/event-stream-processing.html. Accessed: 17.05.2016.

[25] Spark streaming. http://spark.apache.org/streaming/. Accessed: 23.05.2016.

[26] Sqlstream. http://www.sqlstream.com/. Accessed: 17.05.2016.

[27] Storm. http://storm.incubator.apache.org/. Accessed: 11.08.2016.

[28] Streamanalytix. http://streamanalytix.com/. Accessed: 10.05.2016.

[29] Striim. http://www.striim.com/. Accessed: 17.05.2016.

[30] Tibco. http://www.tibco.com/products/event-processing/complex-event-processing/streambase-complex-event-processing. Accessed: 23.05.2016.

[31] Vitria. http://www.vitria.com/. Accessed: 23.05.2016.

[32] Wso2. `http://wso2.com/`. Accessed: 17.05.2016.

[33] Aggarwal, C. C., and Yu, P. S. Outlier detection for high dimensional data. In *ACM Sigmod Record* (2001), vol. 30, ACM, pp. 37–46.

[34] Agyemang, M., Barker, K., and Alhajj, R. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis 10*, 6 (2006), 521–538.

[35] Ahmed, T., Coates, M., and Lakhina, A. Multivariate online anomaly detection using kernel recursive least squares. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications* (2007), IEEE, pp. 625–633.

[36] Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996), ACM, pp. 20–29.

[37] Angiulli, F., and Fassetti, F. Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (2007), ACM, pp. 811–820.

[38] Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2002), ACM, pp. 1–16.

[39] Babcock, B., Datar, M., and Motwani, R. Sampling from a moving window over streaming data. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms* (2002), Society for Industrial and Applied Mathematics, pp. 633–634.

[40] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. Survey on incremental approaches for network anomaly detection. *arXiv preprint arXiv:1211.4493* (2012).

[41] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. Moa: Massive online analysis. *The Journal of Machine Learning Research 11* (2010), 1601–1604.

[42] Bolon-Canedo, V., Sanchez-Marono, N., and Alonso-Betanzos, A. Feature selection and classification in multiple class

datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications 38*, 5 (2011), 5947–5957.

[43] Bu, Y., Chen, L., Fu, A. W.-C., and Liu, D. Efficient anomaly monitoring over moving object trajectory streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), ACM, pp. 159–168.

[44] Burbeck, K., and Nadjm-Tehrani, S. Adaptive real-time anomaly detection with incremental clustering. *information security technical report 12*, 1 (2007), 56–67.

[45] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.

[46] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S. R., Reiss, F., and Shah, M. A. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (2003), ACM, pp. 668–668.

[47] Chen, M., Mao, S., and Liu, Y. Big data: A survey. *Mobile Networks and Applications 19*, 2 (2014), 171–209.

[48] Chen, Y., Dong, G., Han, J., Wah, B. W., and Wang, J. Multi-dimensional regression analysis of time-series data streams. In *Proceedings of the 28th international conference on Very Large Data Bases* (2002), VLDB Endowment, pp. 323–334.

[49] Davy, M., Desobry, F., Gretton, A., and Doncarli, C. An online support vector machine for abnormal events detection. *Signal processing 86*, 8 (2006), 2009–2025.

[50] Dong, G., Han, J., Lakshmanan, L. V., Pei, J., Wang, H., and Yu, P. S. Online mining of changes from data streams: Research problems and preliminary results. In *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams* (2003), pp. 739–747.

[51] Esling, P., and Agon, C. Time-series data mining. *ACM Computing Surveys (CSUR) 45*, 1 (2012), 12.

[52] Faulkner, M., Olson, M., Chandy, R., Krause, J., Chandy, K. M., and Krause, A. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on* (2011), IEEE, pp. 13–24.

[53] Flajolet, P., and Martin, G. N. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences 31*, 2 (1985), 182–209.

[54] Fu, T.-c. A review on time series data mining. *Engineering Applications of Artificial Intelligence 24*, 1 (2011), 164–181.

[55] Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. Mining data streams: a review. *ACM Sigmod Record 34*, 2 (2005), 18–26.

[56] Gama, J. *Knowledge discovery from data streams.* CRC Press, 2010.

[57] Gama, J. A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence 1*, 1 (2012), 45–55.

[58] Gama, J., Sebastião, R., and Rodrigues, P. P. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), ACM, pp. 329–338.

[59] Gama, J., Sebastião, R., and Rodrigues, P. P. On evaluating stream learning algorithms. *Machine Learning 90*, 3 (2013), 317–346.

[60] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR) 46*, 4 (2014), 44.

[61] Golab, L., and Özsu, M. T. Issues in data stream management. *ACM Sigmod Record 32*, 2 (2003), 5–14.

[62] Gomes, R., Welling, M., and Perona, P. Incremental learning of nonparametric bayesian mixture models. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8.

[63] Guha, S., Mishra, N., Roy, G., and Schrijvers, O. Robust random cut forest based anomaly detection on streams. In *Proceedings of The 33rd International Conference on Machine Learning* (2016), pp. 2712–2721.

[64] GUPTA, M., GAO, J., AGGARWAL, C., AND HAN, J. Outlier detection for temporal data. *Synthesis Lectures on Data Mining and Knowledge Discovery 5*, 1 (2014), 1–129.

[65] HEINTZ, B., CHANDRA, A., AND SITARAMAN, R. K. Towards optimizing wide-area streaming analytics. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (2015), IEEE, pp. 452–457.

[66] HILL, D. J., AND MINSKER, B. S. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software 25*, 9 (2010), 1014–1022.

[67] HILL, D. J., MINSKER, B. S., AND AMIR, E. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the Congress-International Association for Hydraulic Research* (2007), vol. 32, Citeseer, p. 503.

[68] HODGE, V. J., AND AUSTIN, J. A survey of outlier detection methodologies. *Artificial Intelligence Review 22*, 2 (2004), 85–126.

[69] HSU, C.-C., AND HUANG, Y.-P. Incremental clustering of mixed data based on distance hierarchy. *Expert Systems with Applications 35*, 3 (2008), 1177–1185.

[70] HUANG, H., AND KASIVISWANATHAN, S. P. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment 9*, 3 (2015), 192–203.

[71] KAMBATLA, K., KOLLIAS, G., KUMAR, V., AND GRAMA, A. Trends in big data analytics. *Journal of Parallel and Distributed Computing 74*, 7 (2014), 2561–2573.

[72] KHREICH, W., GRANGER, E., MIRI, A., AND SABOURIN, R. Adaptive roc-based ensembles of hmms applied to anomaly detection. *Pattern Recognition 45*, 1 (2012), 208–230.

[73] KLEINBERG, J. Temporal dynamics of on-line information streams, 2006.

[74] KOTENKO, I., LASKOV, P., AND SCHÄFER, C. Intrusion detection in unlabeled data with quarter-sphere support vector machines. *DIMVA 2004, July 6-7, Dortmund, Germany* (2004).

[75] Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al. Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter 16*, 1 (2014), 1–10.

[76] Laskov, P., Gehl, C., Krüger, S., and Müller, K.-R. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research 7*, Sep (2006), 1909–1936.

[77] Lee, H.-j., and Roberts, S. J. On-line novelty detection using the kalman filter and extreme value theory. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on* (2008), IEEE, pp. 1–4.

[78] Lichman, M. UCI machine learning repository, 2013.

[79] Liu, Y., Zhang, L., and Guan, Y. Sketch-based streaming pca algorithm for network-wide traffic anomaly detection. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on* (2010), IEEE, pp. 807–816.

[80] Ma, J., and Perkins, S. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 613–618.

[81] Meinhold, R. J., and Singpurwalla, N. D. Understanding the kalman filter. *The American Statistician 37*, 2 (1983), 123–127.

[82] Otey, M. E., Ghoting, A., and Parthasarathy, S. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery 12*, 2-3 (2006), 203–228.

[83] Page, E. Continuous inspection schemes. *Biometrika 41*, 1/2 (1954), 100–115.

[84] Pevnỳ, T. Loda: Lightweight on-line detector of anomalies. *Machine Learning 102*, 2 (2016), 275–304.

[85] Pokrajac, D., Lazarevic, A., and Latecki, L. J. Incremental local outlier detection for data streams. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on* (2007), IEEE, pp. 504–515.

[86] Qian, Z., He, Y., Su, C., Wu, Z., Zhu, H., Zhang, T., Zhou, L., Yu, Y., and Zhang, Z. Timestream: Reliable stream computation in the cloud. In *Proceedings of the 8th ACM European Conference on Computer Systems* (2013), ACM, pp. 1–14.

[87] Raghavan, M. R. H. P. Computing on data streams. In *External Memory Algorithms: DIMACS Workshop External Memory and Visualization, May 20-22, 1998* (1999), vol. 50, American Mathematical Soc., p. 107.

[88] Richard, C., Bermudez, J. C. M., and Honeine, P. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing 57*, 3 (2009), 1058–1067.

[89] Sabhnani, M., and Serpen, G. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intelligent Data Analysis 6* (2002), 1–13.

[90] Salem, O., Liu, Y., Mehaoua, A., and Boutaba, R. Online anomaly detection in wireless body area networks for reliable healthcare monitoring. *IEEE journal of biomedical and health informatics 18*, 5 (2014), 1541–1551.

[91] Song, X., Wu, M., Jermaine, C., and Ranka, S. Conditional anomaly detection. *Knowledge and Data Engineering, IEEE Transactions on 19*, 5 (2007), 631–645.

[92] Street, W. N., and Kim, Y. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (2001), ACM, pp. 377–382.

[93] Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., and Gunopulos, D. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB Endowment, pp. 187–198.

[94] Tan, S. C., Ting, K. M., and Liu, T. F. Fast anomaly detection for streaming data. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* (2011), vol. 22, p. 1511.

[95] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A.-A. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the*

Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009 (2009).

[96] Ting, J.-A., D'Souza, A., and Schaal, S. Automatic outlier detection: A bayesian approach. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (2007), IEEE, pp. 2489–2494.

[97] Xie, M., Hu, J., Han, S., and Chen, H.-H. Scalable hypergrid k-nn-based online anomaly detection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 24*, 8 (2013), 1661–1670.

[98] Yamanishi, K., and Takeuchi, J.-i. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), ACM, pp. 676–681.

[99] Yamanishi, K., Takeuchi, J.-I., Williams, G., and Milne, P. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery 8*, 3 (2004), 275–300.

[100] Zhang, Y., Meratnia, N., and Havinga, P. Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks. In *Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on* (2009), IEEE, pp. 990–995.

[101] Zimek, A., Schubert, E., and Kriegel, H.-P. Outlier detection in high dimensional data. In *Tutorial at the 12th International Conference on Data Mining (ICDM), Brussels, Belgium* (2012), vol. 10.

Appendix A: Data set descriptions

General synthetic data sets

Here the 31 data sets that were used in many of the synthetic data experiments are described. All data sets have 100 000 samples and contain 100 anomalous samples, i.e. 0.1% of the number of samples. Parameters for the t-distribution are standard deviation (1 corresponds to normal distribution), skew (1 corresponds to normal distribution), and shape (20 corresponds to normal distribution, smaller values have lower and bigger values higher kurtosis).

1. 5-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 5 * amplitude = 25, anomalous values in one dimension for anomalous points.

2. 5-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in three dimensions for anomalous points.

3. 5-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in three dimensions for anomalous points.

4. 5-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in all dimensions for anomalous points.

5. 5-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 *

amplitude = 12.5, anomalous values in all dimensions for anomalous points.

6. 10-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in one dimension for anomalous points.

7. 10-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in five dimensions for anomalous points.

8. 10-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in five dimensions for anomalous points.

9. 10-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in all dimensions for anomalous points.

10. 10-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in all dimensions for anomalous points.

11. 100-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in ten dimensions for anomalous points.

12. 100-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in ten dimensions for anomalous points.

13. 100-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in twenty dimensions for anomalous points.

14. 100-d data set. Normal data from sine curve with amplitude 5 and noise from the normal distribution. Anomalous values are peaks of 2.5 * amplitude = 12.5, anomalous values in fifty dimensions for anomalous points.

15. 5-d data set. Normal data from t-distribution with standard deviation 0.5, shape 20 and skew 1. Anomalous values are peaks of 5 * standard deviation = 2.5, anomalous values in all dimensions for anomalous points.

16. 5-d data set. Normal data from t-distribution with standard deviation 1, shape 10 and skew 1. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in all dimensions for anomalous points.

17. 5-d data set. Normal data from t-distribution with standard deviation 1, shape 50 and skew 1. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in three dimensions for anomalous points.

18. 5-d data set. Normal data from t-distribution with standard deviation 2, shape 20 and skew 1. Values for two randomly chosen dimensions were randomly drawn from the t-distribution. Values for the other three dimensions were taken from the preceding dimension and random noise from the normal distribution $\mathcal{N}(0, 0.5)$ was added such that these dimensions correlated with some other dimensions. Anomalous values are peaks of 5 * standard deviation = 10, anomalous values in three dimensions for anomalous points.

19. 10-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 1. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in five dimensions for anomalous points.

20. 10-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 1. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in all dimensions for anomalous points.

21. 10-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 2. Values for five randomly chosen dimensions were randomly drawn from the t-distribution. Values for the other five dimensions were taken from the preceding dimension and random noise from $\mathcal{N}(0, 0.5)$ was added such that these dimensions correlated

with some other dimensions. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in five dimensions for anomalous points.

22. 10-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 2. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in all dimensions for anomalous points.

23. 100-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 1. Values for the first dimension were randomly drawn from the t-distribution. Values for the other 99 dimensions were taken from the preceding dimension and random noise from $\mathcal{N}(0, 0.5)$ was added such that the dimensions correlated with each other. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in one dimension for anomalous points.

24. 100-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 1. Values for the first dimension were randomly drawn from the t-distribution. Values for the other 99 dimensions were taken from the preceding dimension and random noise from $\mathcal{N}(0, 0.5)$ was added such that the dimensions correlated with each other. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in fifty dimensions for anomalous points.

25. 100-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 1. Values for fifty randomly chosen dimensions were randomly drawn from the t-distribution. Values for the other fifty dimensions were taken from the preceding dimension and random noise from $\mathcal{N}(0, 0.5)$ was added such that these dimensions correlated with some other dimension. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in ten dimensions for anomalous points.

26. 100-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 1. Values for fifty randomly chosen dimensions were randomly drawn from the t-distribution. Values for the other fifty dimensions were taken from the preceding dimension and random noise from $\mathcal{N}(0, 0.5)$ was added such that these dimensions correlated with some other dimension. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in fifty dimensions for anomalous points.

27. 100-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 2. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in one dimension for anomalous points.

28. 100-d data set. Normal data from t-distribution with standard deviation 1, shape 20 and skew 2. Anomalous values are peaks of 5 * standard deviation = 5, anomalous values in twenty dimensions for anomalous points.

29. 5-d data. Normal data from two uniform distributions, either $U(0,1)$, denoting the uniform distribution in the range [-1,-0.1], or from $U(0.1,1)$. Anomalies were sampled from $U(-0.1,0.1)$. Anomalous data points had anomalous values in all dimensions.

30. 10-d data. Normal data from two uniform distributions, either $U(0,1)$, denoting the uniform distribution in the range [-1,-0.1], or from $U(0.1,1)$. Anomalies were sampled from $U(-0.1,0.1)$. Anomalous data points had anomalous values in all dimensions.

31. 100-d data. Normal data from two uniform distributions, either $U(0,1)$, denoting the uniform distribution in the range [-1,-0.1], or from $U(0.1,1)$. Anomalies were sampled from $U(-0.1,0.1)$. Anomalous data points had anomalous values in all dimensions.

Concept drift data sets

Here the 8 data sets created to assess concept drift are described. All data sets contain 60 000 samples and have three different concepts, one containing samples 1 - 20 000, second containing samples 21 000 - 40 000 and third containing samples 41 000 - 60 000. All data sets have 100 dimensions.

1. Sudden concept drift data set. Values for all dimensions randomly drawn for the first concept from U(0,1), for the second concept from U(2,3) and for the third concept from U(4,5).

2. Sudden concept drift data set. Values for half the dimensions randomly drawn for the first concept from U(0,1), for the second concept from U(2,3) and for the third concept from U(4,5). For the other half of the dimensions, values for all three concepts were drawn from U(0,1).

3. Sudden concept drift data set. Values for 20% of the dimensions randomly drawn for the first concept from $U(0,1)$, for the second concept from $U(2,3)$ and for the third concept from $U(4,5)$. For the remaining 80% of the dimensions, values for all three concepts were drawn from $U(0,1)$.

4. Sudden concept drift data set. Values for half the dimensions randomly drawn for the first concept from $U(0,1)$, for the second concept from $U(2,3)$ and for the third concept from $U(4,5)$. The other half of the dimensions had values randomly drawn from $U(2,3)$ for the first concept, from $U(4,5)$ for the second concept and from $U(6,7)$ for the third concept.

5. Sudden concept drift data set. Values for half the dimensions randomly drawn for the first concept from $U(0,1)$, for the second concept from $U(2,3)$ and for the third concept from $U(2,3)$. The other half of the dimensions had values randomly drawn from $U(2,3)$ for the first concept, $U(0,1)$ for the second concept and $U(2,3)$ for the third concept.

6. Gradual concept drift data set. Values for all dimensions randomly drawn for the first concept from $U(0,1)$, for the second concept from $U(2,3)$ and for the third concept from $U(4,5)$. Between concepts values shifted from first to second concept in 100 steps having a step size of 0.01 with noise from $U(0,0.2)$.

7. Gradual concept drift data set. Values for half the dimensions randomly drawn for the first concept from $U(0,1)$, for the second concept from $U(2,3)$ and for the third concept from $U(4,5)$. Between concepts values shifted from first to second concept in 100 steps having a step size of 0.01 with noise from $U(0,0.2)$. For the second half of dimensions values were randomly drawn from $U(0,1)$ for all three concepts.

8. Gradual concept drift data set. Values for half the dimensions randomly drawn for the first concept from $U(0,1)$, for the second concept from $U(2,3)$ and for the third concept from $U(4,5)$. Between concepts values shifted from first to second concept in 100 steps having a step size of 0.01 with noise from $U(0,0.2)$. For the second half of dimensions values were randomly drawn from $U(2,3)$ for all three concepts.

Normalization data sets

Here the 32 data sets created to test the effect of data normalization are described. All data sets have 100 000 samples and 10 dimensions.

1. Values for normal data points were drawn from $U(0{,}1)$. Anomalies were drawn from $U(1{,}2)$. Anomalous points had anomalous values in all dimensions.

2. Values for normal data were drawn from $U(0{,}1)$ for half the dimensions and from $U(2{,}3)$ for the other half. Anomalies were drawn from $U(1{,}2)$. Anomalous points had anomalous values in all dimensions.

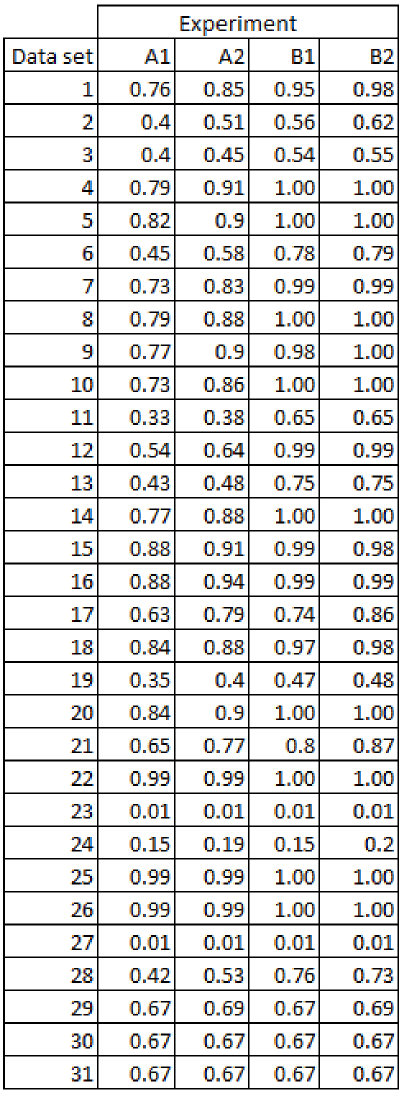
3-12. Values for normal data were drawn from $U(0{,}1)$ for half the dimensions and from $U(2{,}3)$ for the other half. Anomalies were drawn from $U(1{,}2)$ for half the dimensions and from $U(3{,}4)$ for the other half. Anomalous points had anomalous values in all dimensions.

13-22. Values for normal data were drawn from $U(0{,}1)$ for half the dimensions and from $U(0{,}100)$ for the other half. Anomalies were drawn from $U(1{,}2)$ for the first half of dimensions and from $U(100{,}101)$ for the second half. Anomalous points had anomalous values in all dimensions.

23-32. Values for normal data were drawn from $U(0{,}1)$ for half the dimensions and from $U(0{,}100)$ for the other half. Anomalies were drawn from $U(0{,}100)$. Anomalous points had anomalous values in the first half of dimensions.

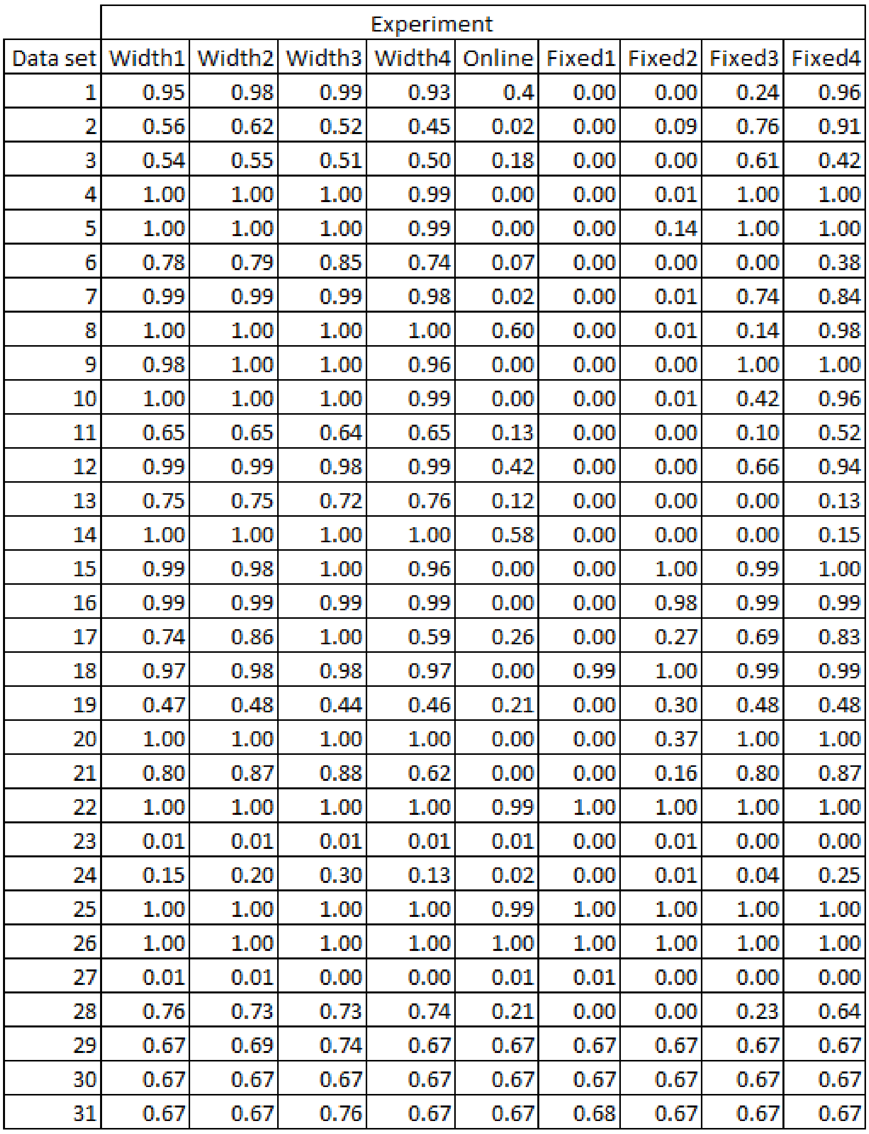
Appendix B: Supplementary tables

Table 1: F-scores for the zero replacement experiments (section 5.2.1).



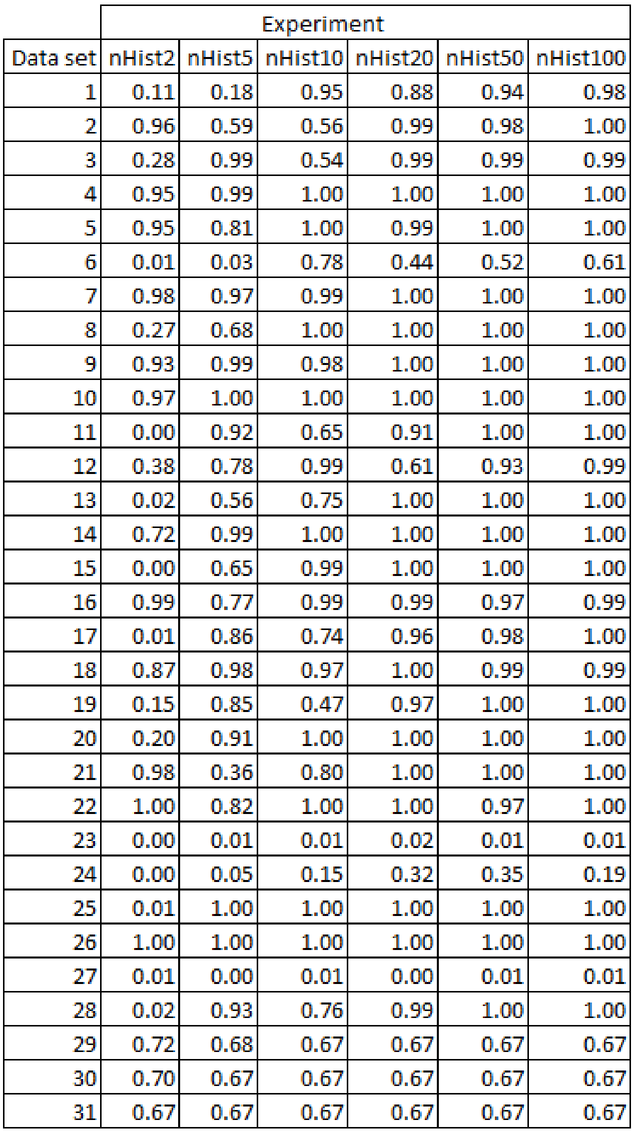
	Experiment			
Data set	A1	A2	B1	B2
1	0.76	0.85	0.95	0.98
2	0.4	0.51	0.56	0.62
3	0.4	0.45	0.54	0.55
4	0.79	0.91	1.00	1.00
5	0.82	0.9	1.00	1.00
6	0.45	0.58	0.78	0.79
7	0.73	0.83	0.99	0.99
8	0.79	0.88	1.00	1.00
9	0.77	0.9	0.98	1.00
10	0.73	0.86	1.00	1.00
11	0.33	0.38	0.65	0.65
12	0.54	0.64	0.99	0.99
13	0.43	0.48	0.75	0.75
14	0.77	0.88	1.00	1.00
15	0.88	0.91	0.99	0.98
16	0.88	0.94	0.99	0.99
17	0.63	0.79	0.74	0.86
18	0.84	0.88	0.97	0.98
19	0.35	0.4	0.47	0.48
20	0.84	0.9	1.00	1.00
21	0.65	0.77	0.8	0.87
22	0.99	0.99	1.00	1.00
23	0.01	0.01	0.01	0.01
24	0.15	0.19	0.15	0.2
25	0.99	0.99	1.00	1.00
26	0.99	0.99	1.00	1.00
27	0.01	0.01	0.01	0.01
28	0.42	0.53	0.76	0.73
29	0.67	0.69	0.67	0.69
30	0.67	0.67	0.67	0.67
31	0.67	0.67	0.67	0.67

Table 2: F-scores for the histogram choice experiments (section 5.2.2).



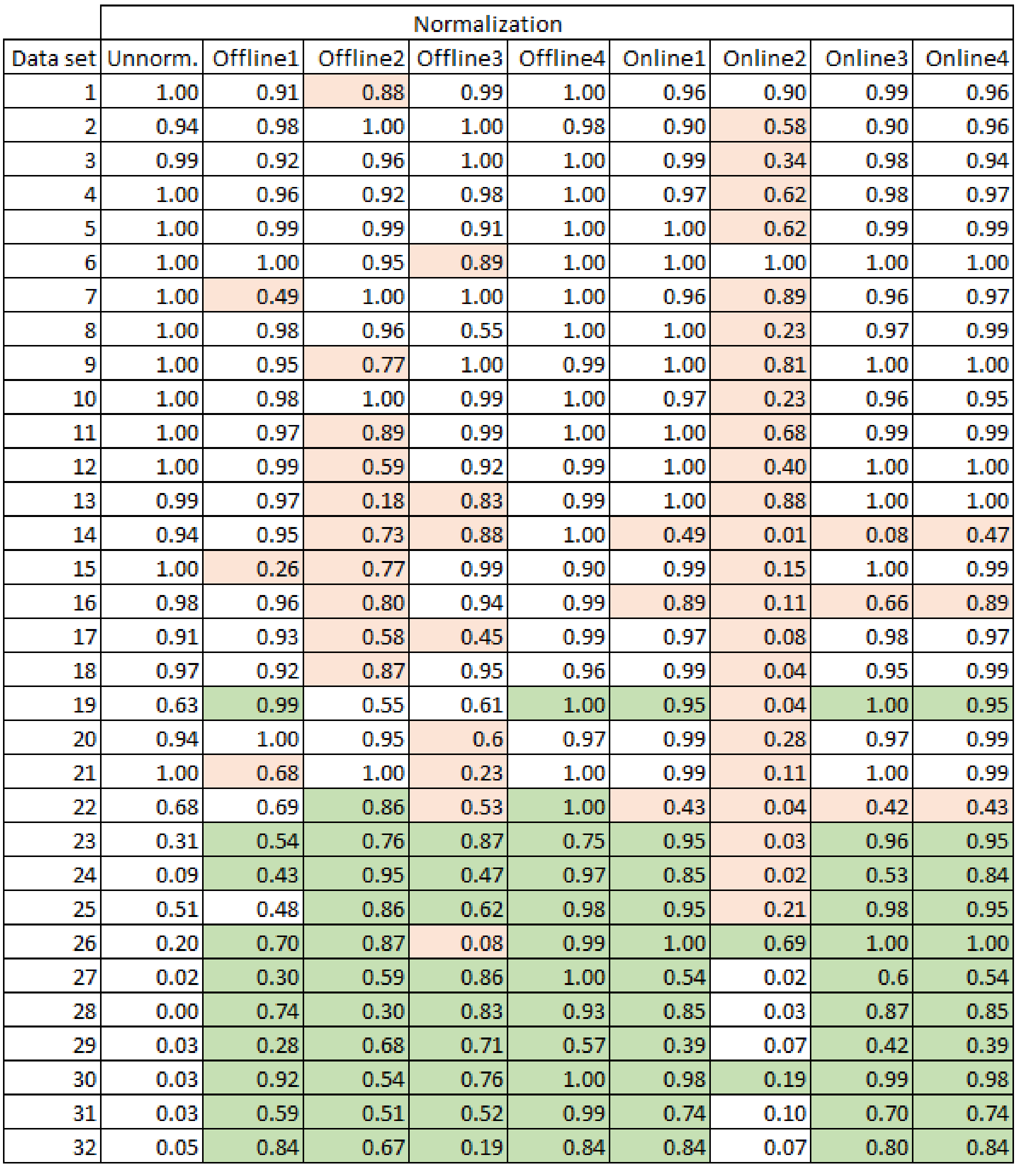
	Experiment								
Data set	Width1	Width2	Width3	Width4	Online	Fixed1	Fixed2	Fixed3	Fixed4
1	0.95	0.98	0.99	0.93	0.4	0.00	0.00	0.24	0.96
2	0.56	0.62	0.52	0.45	0.02	0.00	0.09	0.76	0.91
3	0.54	0.55	0.51	0.50	0.18	0.00	0.00	0.61	0.42
4	1.00	1.00	1.00	0.99	0.00	0.00	0.01	1.00	1.00
5	1.00	1.00	1.00	0.99	0.00	0.00	0.14	1.00	1.00
6	0.78	0.79	0.85	0.74	0.07	0.00	0.00	0.00	0.38
7	0.99	0.99	0.99	0.98	0.02	0.00	0.01	0.74	0.84
8	1.00	1.00	1.00	1.00	0.60	0.00	0.01	0.14	0.98
9	0.98	1.00	1.00	0.96	0.00	0.00	0.00	1.00	1.00
10	1.00	1.00	1.00	0.99	0.00	0.00	0.01	0.42	0.96
11	0.65	0.65	0.64	0.65	0.13	0.00	0.00	0.10	0.52
12	0.99	0.99	0.98	0.99	0.42	0.00	0.00	0.66	0.94
13	0.75	0.75	0.72	0.76	0.12	0.00	0.00	0.00	0.13
14	1.00	1.00	1.00	1.00	0.58	0.00	0.00	0.00	0.15
15	0.99	0.98	1.00	0.96	0.00	0.00	1.00	0.99	1.00
16	0.99	0.99	0.99	0.99	0.00	0.00	0.98	0.99	0.99
17	0.74	0.86	1.00	0.59	0.26	0.00	0.27	0.69	0.83
18	0.97	0.98	0.98	0.97	0.00	0.99	1.00	0.99	0.99
19	0.47	0.48	0.44	0.46	0.21	0.00	0.30	0.48	0.48
20	1.00	1.00	1.00	1.00	0.00	0.00	0.37	1.00	1.00
21	0.80	0.87	0.88	0.62	0.00	0.00	0.16	0.80	0.87
22	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
23	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.00
24	0.15	0.20	0.30	0.13	0.02	0.00	0.01	0.04	0.25
25	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
26	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
27	0.01	0.01	0.00	0.00	0.01	0.01	0.00	0.00	0.00
28	0.76	0.73	0.73	0.74	0.21	0.00	0.00	0.23	0.64
29	0.67	0.69	0.74	0.67	0.67	0.67	0.67	0.67	0.67
30	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.67
31	0.67	0.67	0.76	0.67	0.67	0.68	0.67	0.67	0.67

Table 3: F-scores for the number of histograms experiments (section 5.2.2).



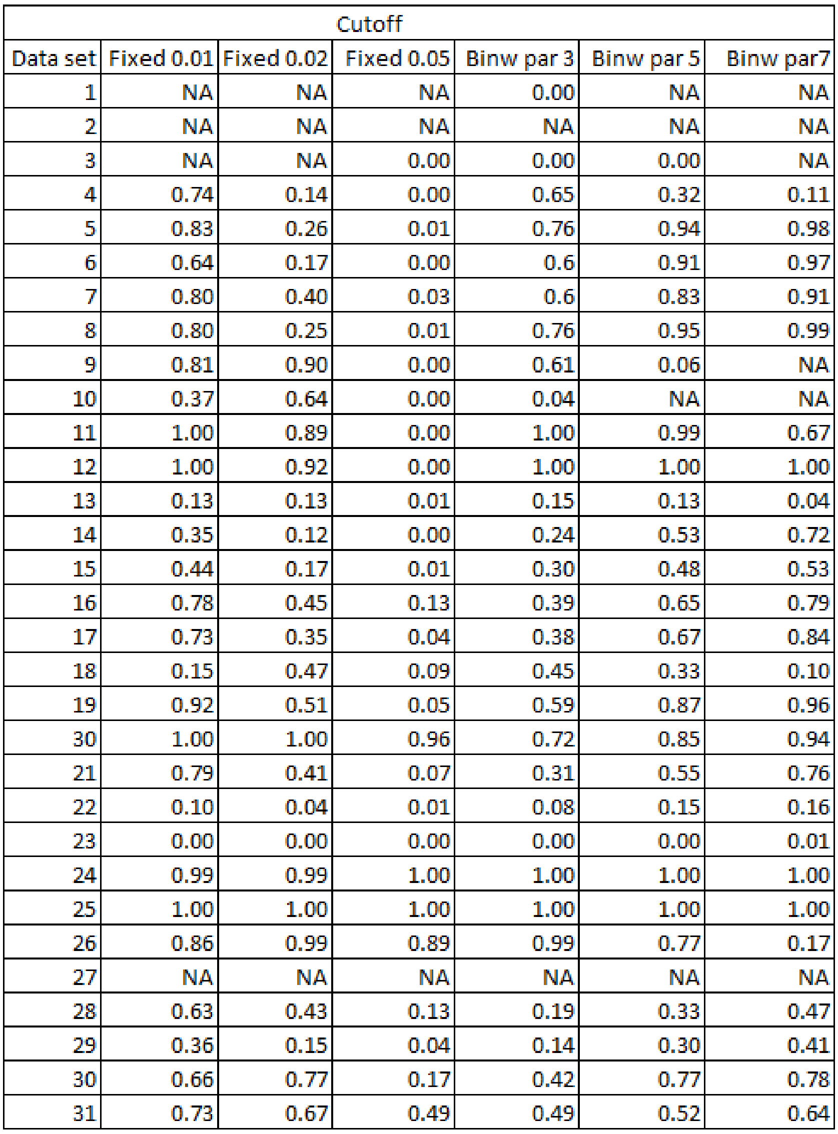
	Experiment					
Data set	nHist2	nHist5	nHist10	nHist20	nHist50	nHist100
1	0.11	0.18	0.95	0.88	0.94	0.98
2	0.96	0.59	0.56	0.99	0.98	1.00
3	0.28	0.99	0.54	0.99	0.99	0.99
4	0.95	0.99	1.00	1.00	1.00	1.00
5	0.95	0.81	1.00	0.99	1.00	1.00
6	0.01	0.03	0.78	0.44	0.52	0.61
7	0.98	0.97	0.99	1.00	1.00	1.00
8	0.27	0.68	1.00	1.00	1.00	1.00
9	0.93	0.99	0.98	1.00	1.00	1.00
10	0.97	1.00	1.00	1.00	1.00	1.00
11	0.00	0.92	0.65	0.91	1.00	1.00
12	0.38	0.78	0.99	0.61	0.93	0.99
13	0.02	0.56	0.75	1.00	1.00	1.00
14	0.72	0.99	1.00	1.00	1.00	1.00
15	0.00	0.65	0.99	1.00	1.00	1.00
16	0.99	0.77	0.99	0.99	0.97	0.99
17	0.01	0.86	0.74	0.96	0.98	1.00
18	0.87	0.98	0.97	1.00	0.99	0.99
19	0.15	0.85	0.47	0.97	1.00	1.00
20	0.20	0.91	1.00	1.00	1.00	1.00
21	0.98	0.36	0.80	1.00	1.00	1.00
22	1.00	0.82	1.00	1.00	0.97	1.00
23	0.00	0.01	0.01	0.02	0.01	0.01
24	0.00	0.05	0.15	0.32	0.35	0.19
25	0.01	1.00	1.00	1.00	1.00	1.00
26	1.00	1.00	1.00	1.00	1.00	1.00
27	0.01	0.00	0.01	0.00	0.01	0.01
28	0.02	0.93	0.76	0.99	1.00	1.00
29	0.72	0.68	0.67	0.67	0.67	0.67
30	0.70	0.67	0.67	0.67	0.67	0.67
31	0.67	0.67	0.67	0.67	0.67	0.67

Table 4: F-scores using different normalization techniques (section 5.2.4). The first column indicates the result with unnormalized data. The result for a normalized data set is marked with yellow if it is worse than the unnormalized result and the difference is more than 0.1 and it is marked with green if the result is better than the unnormalized and the difference is more than 0.1.



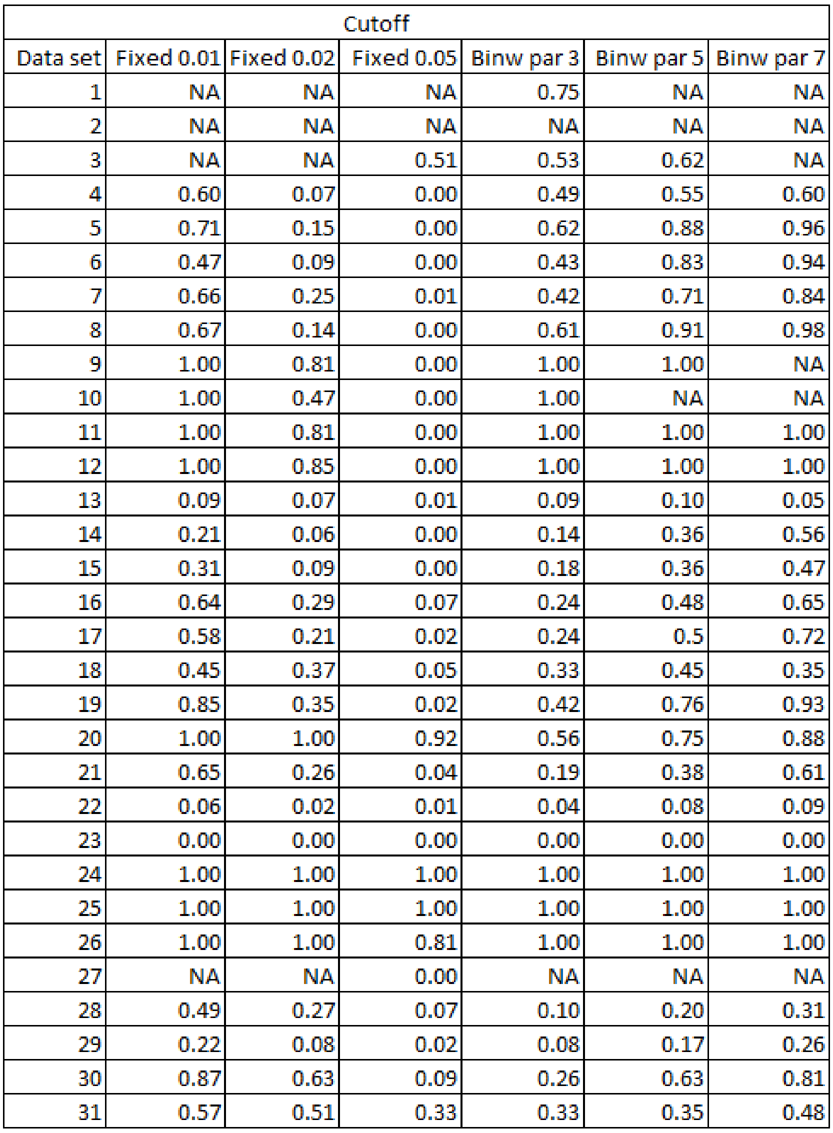
	Normalization								
Data set	Unnorm.	Offline1	Offline2	Offline3	Offline4	Online1	Online2	Online3	Online4
1	1.00	0.91	0.88	0.99	1.00	0.96	0.90	0.99	0.96
2	0.94	0.98	1.00	1.00	0.98	0.90	0.58	0.90	0.96
3	0.99	0.92	0.96	1.00	1.00	0.99	0.34	0.98	0.94
4	1.00	0.96	0.92	0.98	1.00	0.97	0.62	0.98	0.97
5	1.00	0.99	0.99	0.91	1.00	1.00	0.62	0.99	0.99
6	1.00	1.00	0.95	0.89	1.00	1.00	1.00	1.00	1.00
7	1.00	0.49	1.00	1.00	1.00	0.96	0.89	0.96	0.97
8	1.00	0.98	0.96	0.55	1.00	1.00	0.23	0.97	0.99
9	1.00	0.95	0.77	1.00	0.99	1.00	0.81	1.00	1.00
10	1.00	0.98	1.00	0.99	1.00	0.97	0.23	0.96	0.95
11	1.00	0.97	0.89	0.99	1.00	1.00	0.68	0.99	0.99
12	1.00	0.99	0.59	0.92	0.99	1.00	0.40	1.00	1.00
13	0.99	0.97	0.18	0.83	0.99	1.00	0.88	1.00	1.00
14	0.94	0.95	0.73	0.88	1.00	0.49	0.01	0.08	0.47
15	1.00	0.26	0.77	0.99	0.90	0.99	0.15	1.00	0.99
16	0.98	0.96	0.80	0.94	0.99	0.89	0.11	0.66	0.89
17	0.91	0.93	0.58	0.45	0.99	0.97	0.08	0.98	0.97
18	0.97	0.92	0.87	0.95	0.96	0.99	0.04	0.95	0.99
19	0.63	0.99	0.55	0.61	1.00	0.95	0.04	1.00	0.95
20	0.94	1.00	0.95	0.6	0.97	0.99	0.28	0.97	0.99
21	1.00	0.68	1.00	0.23	1.00	0.99	0.11	1.00	0.99
22	0.68	0.69	0.86	0.53	1.00	0.43	0.04	0.42	0.43
23	0.31	0.54	0.76	0.87	0.75	0.95	0.03	0.96	0.95
24	0.09	0.43	0.95	0.47	0.97	0.85	0.02	0.53	0.84
25	0.51	0.48	0.86	0.62	0.98	0.95	0.21	0.98	0.95
26	0.20	0.70	0.87	0.08	0.99	1.00	0.69	1.00	1.00
27	0.02	0.30	0.59	0.86	1.00	0.54	0.02	0.6	0.54
28	0.00	0.74	0.30	0.83	0.93	0.85	0.03	0.87	0.85
29	0.03	0.28	0.68	0.71	0.57	0.39	0.07	0.42	0.39
30	0.03	0.92	0.54	0.76	1.00	0.98	0.19	0.99	0.98
31	0.03	0.59	0.51	0.52	0.99	0.74	0.10	0.70	0.74
32	0.05	0.84	0.67	0.19	0.84	0.84	0.07	0.80	0.84

Table 5: F-scores for different cutoffs (section 5.2.5). "Binw par 3", "Binw par 5" and "Binw par 7" denote the bin width dependent cutoff technique with parameters 3, 5 and 7, respectively. "NA" indicates that it was not possible to compute the score, i.e. number of true anomalies or observed anomalies was zero at the cutoff.



Cutoff						
Data set	Fixed 0.01	Fixed 0.02	Fixed 0.05	Binw par 3	Binw par 5	Binw par7
1	NA	NA	NA	0.00	NA	NA
2	NA	NA	NA	NA	NA	NA
3	NA	NA	0.00	0.00	0.00	NA
4	0.74	0.14	0.00	0.65	0.32	0.11
5	0.83	0.26	0.01	0.76	0.94	0.98
6	0.64	0.17	0.00	0.6	0.91	0.97
7	0.80	0.40	0.03	0.6	0.83	0.91
8	0.80	0.25	0.01	0.76	0.95	0.99
9	0.81	0.90	0.00	0.61	0.06	NA
10	0.37	0.64	0.00	0.04	NA	NA
11	1.00	0.89	0.00	1.00	0.99	0.67
12	1.00	0.92	0.00	1.00	1.00	1.00
13	0.13	0.13	0.01	0.15	0.13	0.04
14	0.35	0.12	0.00	0.24	0.53	0.72
15	0.44	0.17	0.01	0.30	0.48	0.53
16	0.78	0.45	0.13	0.39	0.65	0.79
17	0.73	0.35	0.04	0.38	0.67	0.84
18	0.15	0.47	0.09	0.45	0.33	0.10
19	0.92	0.51	0.05	0.59	0.87	0.96
30	1.00	1.00	0.96	0.72	0.85	0.94
21	0.79	0.41	0.07	0.31	0.55	0.76
22	0.10	0.04	0.01	0.08	0.15	0.16
23	0.00	0.00	0.00	0.00	0.00	0.01
24	0.99	0.99	1.00	1.00	1.00	1.00
25	1.00	1.00	1.00	1.00	1.00	1.00
26	0.86	0.99	0.89	0.99	0.77	0.17
27	NA	NA	NA	NA	NA	NA
28	0.63	0.43	0.13	0.19	0.33	0.47
29	0.36	0.15	0.04	0.14	0.30	0.41
30	0.66	0.77	0.17	0.42	0.77	0.78
31	0.73	0.67	0.49	0.49	0.52	0.64

Table 6: Precisions for different cutoffs (section 5.2.5). "Binw par 3", "Binw par 5" and "Binw par 7" denote the bin width dependent cutoff technique with parameters 3, 5 and 7, respectively. "NA" indicates that it was not possible to compute the score, i.e. number of observed anomalies was zero at the cutoff.



	Cutoff					
Data set	Fixed 0.01	Fixed 0.02	Fixed 0.05	Binw par 3	Binw par 5	Binw par 7
1	NA	NA	NA	0.75	NA	NA
2	NA	NA	NA	NA	NA	NA
3	NA	NA	0.51	0.53	0.62	NA
4	0.60	0.07	0.00	0.49	0.55	0.60
5	0.71	0.15	0.00	0.62	0.88	0.96
6	0.47	0.09	0.00	0.43	0.83	0.94
7	0.66	0.25	0.01	0.42	0.71	0.84
8	0.67	0.14	0.00	0.61	0.91	0.98
9	1.00	0.81	0.00	1.00	1.00	NA
10	1.00	0.47	0.00	1.00	NA	NA
11	1.00	0.81	0.00	1.00	1.00	1.00
12	1.00	0.85	0.00	1.00	1.00	1.00
13	0.09	0.07	0.01	0.09	0.10	0.05
14	0.21	0.06	0.00	0.14	0.36	0.56
15	0.31	0.09	0.00	0.18	0.36	0.47
16	0.64	0.29	0.07	0.24	0.48	0.65
17	0.58	0.21	0.02	0.24	0.5	0.72
18	0.45	0.37	0.05	0.33	0.45	0.35
19	0.85	0.35	0.02	0.42	0.76	0.93
20	1.00	1.00	0.92	0.56	0.75	0.88
21	0.65	0.26	0.04	0.19	0.38	0.61
22	0.06	0.02	0.01	0.04	0.08	0.09
23	0.00	0.00	0.00	0.00	0.00	0.00
24	1.00	1.00	1.00	1.00	1.00	1.00
25	1.00	1.00	1.00	1.00	1.00	1.00
26	1.00	1.00	0.81	1.00	1.00	1.00
27	NA	NA	0.00	NA	NA	NA
28	0.49	0.27	0.07	0.10	0.20	0.31
29	0.22	0.08	0.02	0.08	0.17	0.26
30	0.87	0.63	0.09	0.26	0.63	0.81
31	0.57	0.51	0.33	0.33	0.35	0.48