University of Zurich^UZH

Indep. Study Feb. 13, 2020

Anomaly Detection on Data Streams

Roland Schläfli
of Bern BE, Switzerland

Student-ID: 12-932-398
rolandschlaefli@gmail.com

Advisor: **Dr. Daniele Dell'Aglio**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Abstract

Comparing approaches in the area of streaming anomaly detection can be challenging, as the descriptive research output is often not equally complete, not similarly structured, or not evaluated coherently and on similar data. This work introduces a set of foundational features that can be used to categorize, evaluate, and compare anomaly detection approaches in the streaming context, and should aid in doing so more fairly. Some of the proposed features describe capabilities that are essential in real-world application, while others help judge the quality and completeness of the performed evaluation. We introduce some existing research for each of the different categories and align that research according to our proposed features. Finally, we provide some intuition on open challenges that can hinder the real-world applicability of approaches.

Table of Contents

Abbreviations

AUC Area Under Curve

HTM Hierarchical Temporal Memory

IDS Intrusion Detection System

kNN k-Nearest-Neighbor

PCA Principal Component Analysis

ROC Receiver Operating Characteristic

SVM Support Vector Machine

1

Introduction

In a straightforward definition, anomaly detection describes a method that compares expected with observed behavior and, in doing that, tries to find the observations that are most dissimilar from the norm. When the dissimilarity between expected and observed behavior is quantified, it yields a class label or suspicion score that can be used to mark observations for further investigation or even to make a decision based on a threshold automatically (Bolton and Hand, 2002). Training an anomaly detection system most often yields a categorical classifier that can then process a new observation and assign it such a suspicion score (Hayes and Capretz, 2015).

Anomaly detection methods are often used for fraud detection and fraud prevention. Fraud detection describes cases in which fraud (e.g., finding misuse of someone's credit card) is to be detected after the fact. In contrast, fraud prevention tries to prevent fraud from happening at all (e.g., blocking a fraudulent transaction before it goes through). Further use cases for anomaly detection include parts of factory machines and vehicles like airplanes that could be observed to ensure their reliability, or computer behavior and networks that could be observed to detect intrusions (Chandola et al., 2012). Additionally, malicious actions (e.g., vandalism) in social networks and public knowledge bases could potentially be addressed with automated anomaly detection systems (Heindorf et al., 2016).

Modern systems are often highly complex and composed of many components, making it hard to understand the relationships and connections across components. A large number of components leads to a high potential for failure, and the missing understanding makes it challenging to find the root of any such failure. Traditional multi-class classification (i.e., training a system to detect specific failures based on domain knowledge) is therefore not suitable for anomaly detection on such systems, as the potential failures are not enumerable or not even known (Pimentel et al., 2014). Anomaly detection on complex modern systems can, therefore, be interpreted as a one-class classification problem: the "normal" class is very well-sampled, and its distribution can be estimated through training, but the anomalous classes are too undersampled to learn explicitly. In one-class classification, once a model of "normality" has been built, new events and patterns are evaluated against that model and scored based on the likelihood of them belonging to the "normal" class. If that anomaly likelihood is sufficiently large, a pattern can be deemed anomalous and handled accordingly (Pimentel et al., 2014).

While anomaly detection is a standard term, outlier detection and novelty detection are commonly-used synonyms. All of these terms apply to the one-class classification problem as described here, but they all stem from different application domains and are not used universally. Outliers often carry the connotation of being undesired data points in a dataset that can have a substantial effect on statistical analyses and should, therefore, be filtered out before any further processing. Similarly, anomalies refer to data points that are irregular and lie far from the mean or other means of normality in the dataset, but that are potentially important to know about (Pimentel et al., 2014).

Past research on anomaly detection has established several different categories of anomaly detection systems: probabilistic approaches learn the distribution of the training data and estimate the probability of new data being generated by that distribution using statistical measures; distance-based approaches use a (metric) distance measure to compute clusters or neighborhoods and classify based on membership or neighbors; reconstruction-based approaches learn to predict expected observations and classify based on the error between prediction and observation; domain-based approaches learn a classification boundary and classify based thereon; information-theoretic approaches evaluate whether the addition of new observations alters the entropy (or another measure) of the dataset (Pimentel et al., 2014).

The main challenges in anomaly detection include the significant imbalance of classes (i.e., there are many more normal observations than anomalies), potential uncertainties in class membership (i.e., an observation could belong to a class only with a certain probability), the high importance of time to detection (i.e., a fraud should be found as quickly as possible), as well as the commercial compromises regarding the cost of misclassification errors (i.e., a false negative might be much more costly than a false positive) and finding vs. not finding a fraud (Bolton and Hand, 2002).

Many existing approaches for anomaly detection do not scale well to large datasets as the training time grows overproportionately to the number of observations, or the testing time (i.e., when making a prediction) grows as more observations arrive (Schneider et al., 2016). Also, many existing approaches assume that the entire dataset fits into memory and are not capable of processing observations as they arrive (i.e., online), which means that dedicated algorithms are needed for streaming anomaly detection (dos Santos Teixeira and Milidiú, 2010).

Furthermore, when we process an evolving data stream instead of working on batch data, the potential evolution of normal behavior becomes one of the most crucial issues. For example, when a consumer changes his spending behavior, the new normal monthly spending on his credit card might lie at a higher level, which an anomaly detection system would need to recognize and adapt to (Bolton and Hand, 2002). Additionally, the computational complexity (e.g., an algorithm's usage of computing time and system resources) of algorithms that are applied in a streaming context is often critical, as streaming systems in practice are overloaded with a high-volume and high-velocity inflow of new observations. It is generally impractical to store all of the data arriving in a stream, so algorithms should only process each observation once to improve efficiency (Hayes and Capretz, 2015; Schneider et al., 2016).

Comparing different anomaly detection methods is challenging in terms of capabilities as well as performance, as even the definition of what is an anomaly differs significantly across application domains (Hayes and Capretz, 2015). To improve on that situation and allow for more coherent evaluation, Chapter 2 introduces a foundational set of features using which one could investigate an anomaly detection approach. Subsequently, Chapter 3 introduces a collection of research on streaming anomaly detection, focusing on an analysis of the features as introduced in Chapter 2.

2

Features in Streaming Anomaly Detection

Before we delve into specific approaches developed for anomaly detection on data streams, we introduce a framework to compare such approaches. This section presents a foundational set of features that apply across all the categories of algorithms later described. We make use of these features to categorize and compare algorithms in Chapter 3.

The anomaly detection approaches we focus on work on sequences (or streams), which are ordered series of events of varying data type and dimensionality. The events in a sequence can be discrete (binary or part of a predefined alphabet), continuous (e.g., floating numbers), or multivariate. For batch datasets without an intrinsic ordering, we could apply various of the same approaches as well as many others (Chandola et al., 2009; Hodge and Austin, 2004). This work, however, focuses on anomaly detection for sequences/streams, which is also reflected in the features presented in this chapter.

2.1 ANOM: What types of anomalies does an approach recognize?

Anomalous observations in a sequence are often also referred to as spatial or *point anomalies*, while anomalous (sub-)sequences that are anomalous in entirety are called *collective anomalies*. Another definition of collective anomaly would be that an observation or (sub-)sequence is only anomalous in the context of multiple datasets or data streams (Chandola et al., 2012). When a dataset is very sparse, it can be hard to detect anomalies reliably, as anomaly detectors will have difficulties deriving useful patterns or distributions. However, when multiple sparse datasets are evaluated in conjunction, a collective anomaly detector could find more anomalies (Zheng et al., 2015). An additional definition of anomaly, so-called *contextual anomalies*, are events that are anomalous in one particular context but not necessarily in another one (Chandola et al., 2012). According to Hayes and Capretz (2015), the context consists of descriptive attributes in the dataset that could potentially explain anomalous behavior or reinforce the conclusion (e.g., location, time, or relationships).

Anomaly detection approaches generally focus on detecting a single one of three types of anomalies: a single anomalous event in a sequence (further referred to as "Point"), an anomalous subsequence when compared to the remainder of the sequence (i.e., a "discord" as defined by Keogh et al. (2005), further referred to as "Collective"), or a sequence that is anomalous in its entirety when compared to other sequences (further referred to as "Sequence"). The length of an anomaly (from a single event to a subsequence of arbitrary length) is often not fixed, not known beforehand, and significantly varies across application domains, which poses a significant challenge for anomaly detection algorithms of all kinds (Chandola et al., 2012).

2.2 TRAIN: How is an approach trained?

There are three different ways in which anomaly detection algorithms, as well as machine learning algorithms in general, can be trained: supervised learning, unsupervised learning, and semi-supervised learning. In supervised learning, an algorithm learns patterns in a dataset using a set of labeled observations. Once a supervised algorithm has been trained, it can assign labels to previously unseen data. While there are approaches to label data automatically, labeling is often performed in manual labor, making supervised algorithms expensive to develop and maintain. Supervised algorithms are often not directly applicable to real-time anomaly detection, as the streaming context requires continuous learning without manual interventions (Ahmad et al., 2017).

In unsupervised learning, algorithms are trained on an unlabeled dataset, based on which they learn patterns on their own. Unsupervised algorithms work under the assumption that data contains mostly normal data with only a few anomalies, which is reasonable as it is typically possible to collect a lot of normal data (Schneider et al., 2016).

Semi-supervised algorithms are a mixture of supervised and unsupervised algorithms, where the training dataset contains labels, but only for a single class or a selection of all classes. For example, an algorithm for spam detection could be trained on a set of valid email conversations. The algorithm would learn to classify based on the information it has been given as well as its view of patterns in the dataset. Finally, the algorithm could classify future emails as normal or not-normal, the latter of which would (Wang et al., 2013).

A learning capability that can be especially important in an evolving streaming context is incremental learning, which indicates that the model (i.e., the prediction function) is continuously updated based on incoming observations (Haibo He et al., 2011; Kontaki et al., 2011). The capability for incremental learning is often a requirement in practical situations, as many streams change over time and as it can be impractical to gather and store representative training data from a potentially infinite high-velocity data stream, e.g., in traffic monitoring (Haibo He et al., 2011; Schneider et al., 2016). Furthermore, incremental learning algorithms often incorporate sophisticated sampling, randomization, and approximation techniques (Gama, 2012).

Depending on the definition, an algorithm using incremental learning can either access past and current data to evolve its model, or it can only access the past model and the current data. As past data is often not persisted in stream processing systems, the latter is frequently a more realistic assumption (Haibo He et al., 2011). While the semantics of incremental learning and forgetting are well-researched for predictive systems, that work is still largely missing in areas like data stream clustering (Gama, 2012).

2.3 PROC: How does an approach process observations?

The most intuitive approach to process newly arriving observations in streaming systems is processing them as soon as they arrive (i.e., “online”), and either persisting or forgetting them after processing. Many application areas like credit card fraud prevention require online processing, as the time a transaction takes to go through is critical to the business goal. Most online processing systems are built to quickly provide an approximate answer that is correct with a high probability instead of requiring an exact answer at all times (Gama, 2012). Some algorithms, such as many clustering-based ones, work in a partially online fashion: an initial batch processing phase constructs a baseline model, after which an online processing phase updates that model (Ahmad et al., 2017).

Contrary to processing events online, traditional batch processing is based on the idea that the entire dataset is available in memory before training/processing starts, which makes it inappropriate for handling high-velocity high-volume systems. Batch processing cannot keep up with the continuous changes that occur in data streams (Gama, 2012).

The most practical approach combines batch and online processing in that it processes events in small (rolling) batches. These batches consist of events that have been collected in time- or count-based windows applied to the data stream and are continuously updated. Such windows can, for example, collect all events that arrive within five subsequent minutes, after which all events are processed, and the window slides further in time. This approach enables algorithms to forget objects that are no longer active instead of having to persist them in memory (Kontaki et al., 2011).

2.4 DIM: What input dimensionality can an approach work with?

As previously motivated, data streams can be either simple univariate time-series or complex multivariate sequences with many content-related and contextual variables. In general, all anomaly detection approaches differ in the dimensionality they expect and are capable of handling: while most approaches work on multivariate data streams, some cannot process data with more than one variable.

Data in high-dimensional datasets are generally sparsely distributed, which diminishes the meaning of distance and statistical measures and is hard to handle for many algorithms, a problem that is also called the "curse of high-dimensionality" (Yoon et al., 2019). While there are many statistical and machine learning methods for anomaly detection, most are not practically applicable to high-dimensional, large-scale problems (Schneider et al., 2016). Some anomaly detection approaches apply sophisticated feature selection or dimensionality reduction to reduce the issues presented by very high-dimensional input data.

2.5 DRIFT: How does an approach account for concept drift?

A key challenge in anomaly detection on temporal data and data streams is the idea of concept drift. *Concept drift* describes the fact that the target concept and the distribution that generates incoming events can both change over time. For example, the spending behavior of customers changes over time, which means that a model trained on past data becomes inconsistent, requiring algorithms to update their model of what is "normal" behavior, and to forget outdated information. A sequence that is subject to concept drift is also often described as being generated by a non-stationary distribution (Tsymbal, 2004).

There have been several categorizations of concept drift in previous research, the most basic one being the distinction between sudden and gradual concept drift. While a *sudden drift* occurs at a single point in time, *gradual drift* spans a more extended period. For example, a production machine could gradually deteriorate, resulting in a decrease in the quality of its outputs. There have also been distinctions in the speed of gradual drifts, categorizing them into moderate and slow concept drift (Tsymbal, 2004).

Additionally, the notion of *virtual concept drift* has been introduced to describe a concept drift that only changes the generative distribution but leaves the target concept untouched (Tsymbal, 2004; Widmer and Kubat, 1993). In a more formal definition, virtual concept drift (or loose concept drift) is described as being a drift in which only the prior class probability changes, while in rigorous concept drift, the prior and conditional probabilities change (Bifet et al., 2009). In some cases, concepts might also drift in recurring patterns (e.g., if seasonality is involved). In such cases, algorithms could persist their model for later reuse once they detect and adapt to concept drift. However, the majority of learners does not yet account for recurring drift (Tsymbal, 2004).

There are various ways to account for concept drift in an algorithm: by using only a window of the most recent instances to train the "normal" model, by weighting instances with their age to enable gradual forgetting of old events, or by using ensemble learning and applying aging to members of the ensemble, amongst others. Some of these approaches require specific machine learning techniques (e.g., algorithms that support weighting), while others are more generally applicable (e.g., windowing) (Tsymbal, 2004).

We can further distinguish between blind adaptation, which means using a static decay rate or window size (i.e., updating the model continuously independent of the occurrence of drift), or informed adaptation, which dynamically sets these parameters only when concept drift has been detected and thereby saves on computational efforts (Schneider et al., 2016; Talagala et al., 2019). Most algorithms regularly adapt without considering if there has even been a change. Contrarily, methods that apply explicit change detection only adapt if they detect a significant change (e.g., a so-called changepoint). Changepoints can be detected by monitoring performance metrics of the algorithm, as well as by monitoring how the distribution evolves across windows (Gama, 2012).

2.6 NOISE: How does the approach account for noise?

While anomalies are defined as real events that do not conform to the expected notion of a "normal" event, noise describes events that stem from an entirely different distribution or a random process. Noise can be introduced by failing sensors, data transmission errors, user inputs, and many other failures. Noise is especially challenging in a streaming context, as noise removal algorithms often depend on having a view of the entire dataset (i.e., work on batches), which does not hold in the streaming context (Agrawal et al., 2017).

Some algorithms integrate the handling of noise, which makes them more resilient and less prone to random errors. The main challenge in handling noise is how anomalous events can be distinguished from events that are noise, as well as how concept drift and noise can be separated. If an algorithm is too sensitive, it might classify noise as concept drift and update its model, while it might adapt to changes very slowly if it is highly resilient regarding noise (Tsymbal, 2004). According to Tsymbal (2004); Widmer, Gerhard and Kubat, Miroslav (1996), "an ideal learner should combine robustness to noise and sensitivity to concept drift."

2.7 EV1-4: How was the approach evaluated?

The methods chosen to evaluate anomaly detection algorithms are a vital part of any reliable and reproducible analysis. Therefore, we propose that a common feature for comparison should be dedicated to how an approach was evaluated, as well as how the evaluation procedure and artifacts are documented.

This section introduces four components that we deem critical for any evaluation for anomaly detection approaches. While our further analysis focuses on these four components, it is important to note that there can be many other factors depending on the domain.

EV1: Choice of Evaluation Measures. As the percentage of anomalies in a dataset is, by definition, extremely low, the commonly used false positive and true negative rates are not very meaningful in separation. Instead, evaluations should be based on more significant measures like precision and recall, or their combination, the F1-measure (dos Santos Teixeira and Milidiú, 2010). Another combination of measures results in the Receiver Operating Characteristic (ROC) curve that is often used to evaluate the tradeoff between the detection rate and the false alarm rate (Pimentel et al., 2014).

A well-balanced anomaly detection algorithm should strive for high accuracy with a low number of false alarms. Furthermore, using commonly used measures for evaluation like precision and recall might not suffice in all cases, as they do not give weight to early detection of anomalies (i.e., it should be rewarded if algorithms detect an anomaly early rather than late) (Lavin and Ahmad, 2015). In addition to measures evaluating accuracy, an approach can also be evaluated for its efficiency in processing based on time and memory requirements.

EV2: Choice of Evaluation Datasets. A critical issue in comparing anomaly detection algorithms for data streams is the data and framework using which they are compared. Most datasets that were specifically designed for performance evaluations have been designed for static evaluations with fixed training and test datasets (Ahmad et al., 2017; Lavin and Ahmad, 2015). There are only a few datasets freely available for streaming algorithm benchmarking, and many of them do not include concept drift and other challenges that occur in real-world applications. To evaluate specific issues like concept drift, it can often be useful to generate datasets artificially, as real-world datasets can contain concept drift, but do so in places that we do not know of (Bifet et al., 2009).

Previous work has also introduced more general guidelines for the evaluation of streaming algorithms: one should either evaluate algorithms periodically using a holdout set of the data (that needs to evolve similarly to the stream) or by applying a performance measure as soon as new observations arrive (prequential). The holdout method is advantageous as it allows for usage of a balanced test set (with the same number of anomalies as normal observations), which is not the case in a real stream with prequential benchmarking (Schneider et al., 2016).

EV3: Availability of Data and EV4: Availability of Algorithm. Having the ability to reproduce an approach based on a concise but complete algorithmic description and being able to evaluate an approach on the same data as used in the original research is a critical part of the scientific review process. These features are used to distinguish research that provides access to all the data used in the evaluation, as well as an algorithmic description or pseudocode of the algorithms used by the approach.

3

Approaches to Streaming Anomaly Detection

Ahmad et al. (2017) describe an ideal real-life anomaly detection algorithm as being able to: work unsupervised and automated, predict in an online fashion, continuously learn and evolve, adapt to concept drift, detect anomalies as early as possible, and avoid misclassifications. Additionally, an ideal algorithm would have a low computational cost in space and time, be supported with strong performance guarantees, and have a minimal number of parameters (Bifet et al., 2009).

Based on the features introduced in Chapter 2, this chapter will introduce and summarize several anomaly detection approaches from existing research. Table 3.1 provides an overview of how the approaches align with each of the previously defined features, while the descriptive text later in this chapter is oriented toward a concise summarization of the approach instead of covering every single feature.

3.1 Probabilistic Approaches

Probabilistic approaches learn a model of normality based on the distribution of the training data and compare all new test data against that distribution. If it is sufficiently improbable that the distribution has generated the new data (i.e., if it lies in a low-density area of the distribution), that test data is classified as anomalous. Probabilistic approaches have a strong mathematical foundation and are very effective if they manage to estimate the generative distribution. Furthermore, probabilistic methods work in a very space-efficient way in that only the learned distribution needs to be persisted (Pimentel et al., 2014).

A critical factor in the complexity of probabilistic approaches is the distinction between *parametric* and *nonparametric models*. The former are based on the assumption that a finite set of parameters derived from the training data can capture every detail of the data. While this results in simpler models, there is also a higher chance of the model not being able to fit the dataset due to high bias (e.g., by trying to fit a gaussian normal distribution on a much more complex phenomenon). Contrarily, the latter approaches learn a distribution without constraints, with fewer assumptions, and with the capability of growing the model with the amount of data, resulting in the ability to learn very intricate patterns (Pimentel et al., 2014). Gaussian mixture models combine multiple Gaussian distributions to approximate the true distribution but require a large number of parameters that rise even faster in higher dimensions (Schneider et al., 2016).

Approach	TYPE	ANOM	TRAIN	PROC	DIM	DRIFT	NOISE	EV1	EV2	EV3	EV4
Wang and Stolfo (2004)	Prob.	Point	Unsupervised Incremental	Online	n	Y	Y	Accuracy, Confusion, ROC, Time, Memory	Real	N	N
Bifet et al. (2009)	Prob.	Point	Unsupervised Incremental	Online	n	Y	N	Accuracy, Time, Memory	Both	Y	N
Schneider et al. (2016)	Prob.	Point	Unsupervised Incremental	Batch/Online	n	Y	N	Accuracy, AUC, Time, Memory, Stat. Tests	Both	Y	N
Talagala et al. (2019)	Prob.	Sequence	Unsupervised	Online	1	Y	Y	Accuracy, Confusion, OP, PPV	Both	N	Y
Cao et al. (2006)	Dist.	Point	Unsupervised Incremental	Online	n	Y	Y	Cluster Purity, Time, Memory	Both	Y	Y
Kontaki et al. (2011)	Dist.	Point	Unsupervised Incremental	Online	n	Y	N	Time, Memory, Event Count	Both	Y	Y
Miller et al. (2014)	Dist.	Point	Unsupervised Incremental	Online	n	Y	N	Accuracy, ROC, Precision, Recall	Real	N	Y
Yoon et al. (2019)	Dist.	Point	Unsupervised Incremental	Online	n	Y	N	Time, Memory, Concentration Ratio	Both	Y	Y
Hayes and Capretz (2015)	Recon.	Contextual	Unsupervised	Online	1	N	N	Time, Accuracy	Real	N	N
Shipmon et al. (2017)	Recon.	Point/Collective Only Periodic	Supervised	Batch/Online	n	N	Y	MSE, Confusion	Both	N	N
Ahmad et al. (2017)	Recon.	Collective	Unsupervised Incremental	Online	n	Y	Y	NAB Score, Time, Capabilities	Both	Y	Y
Kanarachos et al. (2017)	Recon.	Contextual Temporal	Unsupervised Incremental	Batch/Online	1	N	Y	ROC, AUC, Significance	Real	N	N
Liu et al. (2008)	Domain	Point	Unsupervised	Online	n	N	Y	AUC, Time	Both	Y	Y
dos Santos Teixeira and Milidiú (2010)	Domain	Collective	Unsupervised Incremental	Online	n	Y	N	Confusion, Precision, Recall, F1	Both	Y	Y
Muter and Asaj (2011)	Info.	Point	Unsupervised	Online	1	N	N	Analysis of Entropy	Real	N	N
Wang et al. (2011)	Info.	Contextual Temporal	Unsupervised	Online	1	Y	N	Confusion, Recall	Both	N	Y
Rettig et al. (2015)	Info.	Contextual Temporal	Unsupervised	Batch/Online	n	Y	N	Domain Knowledge, Time	Real	N	N

Table 3.1: Overview of the references surveyed in this work. The **TYPE** of each reference describes which category the presented approach belongs to. **ANOM** describes what types of anomalies are detected by the approach. **TRAIN** describes, whether the approach is trained in a supervised or unsupervised manner. **PROC** describes how the approach processes new observations (online also includes windowed quasi-online approaches). **DIM** describes, whether an approach works on univariate (1) or multivariate (n) data. **DRIFT** is marked as fulfilled (Y), if the approach explicitly accounts for concept drift. **NOISE** is marked as fulfilled (Y), if the approach considers noise and has also been evaluated on noisy data. **EV1** summarizes, what measures were used to evaluate the approach (both complexity and accuracy-wise). **EV2** signifies, whether only real data, or both artificial and real data was used in the evaluation. **EV3** is marked as fulfilled, if all the data used for the evaluation is available for replication. **EV4** is marked as fulfilled, if the algorithms used by the approach are explicitly stated and not only described textually.

Due to their potential for complexity, nonparametric approaches are also more prone to overfitting, meaning that they too tightly fit a complex pattern and, therefore, do not generalize to unseen data. Additionally, probabilistic approaches, especially nonparametric ones, tend to require more training data to get to a reasonable level of accuracy. When probabilistic approaches are applied to datasets with high dimensionality, the density estimation of the distribution can often become inaccurate due to the inherent sparsity of the dataset (Kontaki et al., 2011; Pimentel et al., 2014).

3.1.1 Anomalous Payload-based Network Intrusion Detection (Wang and Stolfo, 2004)

Traditional Intrusion Detection Systems (IDS) often work with signatures, which are attack-patterns that have been observed before, and that can be generalized and shared to detect further occurrences of similar attacks. However, zero-day attacks, which are the first occurrences of new threats, cannot be prevented by such signature-based IDS. The PAYL approach presented by Wang and Stolfo (2004) seeks to close the gap in early detection by applying anomaly detection to recognize new threats as soon as they appear.

PAYL builds a model of the typical network payloads based on their byte frequency (i.e., an abstraction of the content of requests) and compares the distance of new observations to that typical distribution using the Mahalanobis distance. If the Mahalanobis distance of an observation crosses a preset threshold (e.g., a number of standard deviations), that observation is marked as anomalous. In an online processing context, the model incrementally evolves based on new observations and a decay parameter that gradually phases out older observations.

Wang and Stolfo (2004) show that their approach accurately signals anomalies on labeled data while keeping a false positive rate of less than 1%. The approach managed to pick up anomalies from an unlabeled dataset but has not been tested on real-life data streams. As PAYL is a simple approach, Wang and Stolfo (2004) recommend that it be used in ensemble with other IDS for increased coverage and a reduced number of false alarms.

3.1.2 New Ensemble Methods For Evolving Data Streams (Bifet et al., 2009)

Bifet et al. (2009) approach the anomaly detection problem using an ensemble of Hoeffding decision trees that approximates the generative distribution of a multivariate data stream and evolves that distribution incrementally alongside the stream. The individual predictions of each tree in the ensemble are aggregated with bagging and/or boosting, which means that individual decisions are averaged (either weighted or unweighted) to find an ensemble decision. Using a weighted average (i.e., boosting) based on the prediction error of individual trees causes well-performing trees to be weighted higher and partially accounts for concept drift, as outdated predictors that model a past distribution tend to perform worse.

In their best-performing configuration, Bifet et al. (2009) also include the ADWIN change detector in their system. ADWIN is an algorithm that buffers a window of past events, which allows computing the statistics necessary to detect a change in concepts. The ADWIN bagging that Bifet et al. (2009) apply makes use of ADWIN to estimate the weights of the predictors in the ensemble. When ADWIN detects a change in concepts, the algorithm removes the predictor with the highest error and adds a new one to the ensemble.

The ensembles presented in Bifet et al. (2009) outperform the baseline Naive Bayes model and the other decision tree algorithms they were benchmarked against. However, the ensemble approach has a relatively high overall complexity in the time and memory dimensions, as it combines several systems to make a decision. While ADWIN bagging tends to be the most accurate method in a drifting environment, it is also one of the slowest in terms of processing time.

3.1.3 Expected Similarity Estimation for Large-Scale Batch and Streaming Anomaly Detection (Schneider et al., 2016)

The EXPoSE approach as proposed by Schneider et al. (2016) is a kernel-based algorithm that constructs an embedding from the underlying data and, by working in that embedding space, can efficiently calculate the similarity of new observations and normal data. EXPoSE does not make assumptions regarding the size or shape of the underlying generative distribution (except that there is a distribution), which is advantageous when compared to parametric statistical models. Furthermore, the algorithm is able to process complex multivariate data like images and videos.

EXPoSE can learn incrementally in constant time per observation and, when trained as such, produces the same model as when trained offline. Additionally, it can also make predictions online and in constant time, using constant memory. The design of the algorithm allows for parallelization, which makes it much more applicable to big data problems. EXPoSE can be extended to account for concept drift by working on sliding windows over the data stream, or by applying a gradual forgetting mechanism to outdated observations. The main advantage of EXPoSE lies in its higher efficiency and lower complexity when compared to state-of-the-art algorithms that, overall, perform similarly well.

3.1.4 Anomaly Detection in Streaming Nonstationary Temporal Data (Talagala et al., 2019)

Talagala et al. (2019) propose an approach based on a bivariate two-sample nonparametric test. The approach projects several univariate time-series into a two-dimensional problem space and applies the statistical test to find anomalous sequences. Before the approach can process new observations from multiple data streams in an online fashion, it first needs to be warmed-up using a batch of typical data. This warm-up period serves the initialization of model parameters to non-arbitrary values.

To account for drift, Talagala et al. (2019) propose an informed approach that evaluates the statistical distance between the current typical distribution and the distribution of the current test window. The model only needs to be updated when that distance grows significantly large, which allows for quicker decisions and reduced computational complexity. In their evaluation, Talagala et al. (2019) find that their framework performs well when applied to nonstationary streams with noise and multi-modal distributions (i.e., more than one kind of typical behavior).

3.2 Distance-based Approaches

Distance-based approaches are based on the concepts of neighbors and cluster analysis, which are generally both easy to implement and interpret (Schneider et al., 2016). While normal data is in close proximity to other data (i.e., tightly clustered or close to its neighbors), anomalies are scattered far off from the majority of other data. The first subcategory of distance-based approaches consists of nearest-neighbor approaches like k-nearest-neighbor (kNN), which are among the most popular for anomaly detection in general. However, these methods are not easily applicable in an incremental learning streaming context (Pimentel et al., 2014). The second subcategory is comprised of clustering approaches like k-means clustering, which are approaches that have been applied to the streaming context in a considerable amount of research.

Distance-based approaches are highly dependent on a suitable distance measure. Approaches often work under the assumption that the data space is metric, as this allows for efficient indexing (Kontaki et al., 2011). As distance-based methods rely on the computation of distances between data points, they run into scalability and efficiency issues when applied to high-dimensional datasets. The complexity of the individual distance calculations can be high, and, as there is a high number of operations, the overall computation can grow to be impractical. Additionally, distance measures tend to fail in distinguishing anomalies and normal data in high-dimensional contexts due to the "curse of high dimensionality" (Pimentel et al., 2014).

In a streaming context, distance-based approaches present several challenges like maintaining a correct clustering over time, using small amounts of time and memory, a requirement for incremental clustering, and the appearance and disappearance of clusters due to concept drift (Gama, 2012). As the distance computations cause a relatively high computational complexity in the test phase, distance-based approaches need to be carefully tuned for usage in an online context, as they could otherwise cause bottlenecks (Pimentel et al., 2014). A foundational element that is often used to improve the efficiency in data stream clustering is the idea of cluster features. As Gama (2012) state: "A cluster feature, or microcluster, is a compact representation of a set of points." Algorithms that work with cluster features first reduce the overall data points to local microclusters. Based on these microclusters, it takes much less work to generate a correct overall clustering, and the algorithm requires less of the very constrained memory (Cao et al., 2006; Gama, 2012).

Additional challenges include the fact that clustering algorithms for stream environments cannot assume an initial or fixed number of clusters or other parameters, as the stream continuously evolves with new clusters appearing and old ones disappearing. The clusters can have an arbitrary shape, and new observations can initialize a new cluster if they arrive far from any existing cluster. Algorithms also need to be able to handle noise, as data streams are susceptible to sensor failures and other random events. An ideal algorithm would even be able to distinguish noise and anomalies from newly developing normal clusters (Cao et al., 2006).

3.2.1 Density-Based Clustering over an Evolving Data Stream with Noise (Cao et al., 2006)

The DenStream approach introduced by Cao et al. (2006) builds on the idea of cluster features and builds its internal model of normality using core-micro-clusters, which are dense approximations with attributes for weight, center, and radius (Miller et al., 2014). Core-micro-clusters can approximate clusters of arbitrary shape and size and are used to represent clusters of both normal observations and anomalies. New observations can then be evaluated against the existing clusters, and if warranted, marked as anomalous.

DenStream applies a damped window to the underlying data stream, allowing it to weigh observations with a decay function (i.e., old observations are weighted exponentially less than new ones) so that only a part of the infinite stream needs to be kept in memory. The observations in the window are summarized into compact micro-clusters (which are more than natural clusters, but less than observations). To account for the appearance of new clusters, clusters of outliers (o-micro-clusters) that reach a certain weight are promoted to potential core-micro-clusters (p-micro-clusters), and vice-versa for p-micro-clusters that shrink comparably. After outlier and potential micro-clusters have been formed, DenStream applies a variant of the DBSCAN clustering algorithm to all p-micro-clusters, which yields a final clustering. Each p-micro-cluster is represented as a point with a weight derived of its size.

In general, DenStream achieves a very good clustering quality with a purity that is consistently above 95% when applied to the evaluation datasets. The evaluation shows that DenStream can reliably separate noise from relevant observations and that it works much more memory-efficiently than comparable algorithms. The computational complexity of DenStream is linear in the number of observations processed, which means that it is well capable of handling high-velocity data streams.

3.2.2 Continuous Monitoring of Distance-Based Outliers over Data Streams (Kontaki et al., 2011)

Kontaki et al. (2011) introduce a distance-based approach that reduces the number of computations with a novel event-based framework that only schedules distance-computations if potentially relevant events occur. For example, an observation that is considered to be normal based on the current window can only become anomalous if its neighbors leave the current window, which is an event that can be planned for.

By tracking the expiration times of objects in the current window, related computations only need to be executed when such an object leaves the window. As the same kind of scheduling cannot be done for the inverse event (i.e., new arrivals cannot be planned for), the approach only schedules the expiration of existing normal observations. An additional optimization applied in COD is based on the notion that certain normal observations can never become anomalous and thus do not need to be watched in the event queue. This notion of "safe inliers" captures the idea that if there are enough neighbors following an observation in the current window, even the expiration of all its preceding neighbors cannot make it anomalous.

As an extension of their COD algorithm, Kontaki et al. (2011) also propose the MCOD algorithm that extends COD with the notion of micro-clusters (i.e., dense regions of normal observations). MCOD applies micro-clusters to outlier detection in that it classifies any observation within a micro-cluster as certainly normal, and any observation outside as an anomaly. Using micro-clusters as such significantly reduces the number of distance computations that need to be performed. Overall, the focus of the evaluation has been on the overall efficiency, which greatly improved in comparison to an existing benchmark algorithm (i.e., Abstract-C).

3.2.3 Twitter spammer detection using data stream clustering (Miller et al., 2014)

The amount of spam on the Twitter platform has dramatically increased in recent years, which can be attributed to the growing pervasiveness of social media platforms in general. To address that spam problem, Miller et al. (2014) applies a density-based clustering approach that can learn clusters of normal tweets and subsequently judges unknown tweets based on their distance to these clusters. Each tweet is represented by a vector containing 107 numerical features that summarize both content-related and contextual properties.

The first layer in the system, the StreamKM++ algorithm, is based on an extended k-means++ algorithm (Ackermann et al., 2012). The extensions that make it applicable to data streams is the windowing of the data stream in conjunction with the data reduction performed using coresets, which are approximative aggregations of observations. To improve the performance of the system, a second layer based on the DenStream algorithm makes another pass on the observations processed by the first layer. While the first layer is optimized such that it produces a minimal amount of false-negatives and, conversely, produces more false-positives, the second layer processes the stream more efficiently by only reconsidering flagged anomalies (normal observations are assumed to be flagged correctly).

The overall system processes observations online and only requires a single pass at each observation. The application of a two-layer approach significantly reduces both false-positive and false-negative rates and improves all other metrics when compared with each layer in separation. After tuning parameters on a training set of tweets, the two-layer system manages to achieve very high accuracy (>95%), 100% recall, and a low rate of false positives on the test set.

3.2.4 NETS: Extremely Fast Outlier Detection from a Data Stream via Set-Based Processing (Yoon et al., 2019)

Yoon et al. (2019) propose a distance-based approach that exploits a newly discovered characteristic when windowing a data stream: namely, that "the change in the locations of data points in the data space is typically very insignificant." As new observations enter the window and old ones expire, it is very probable that new observations replace expired ones. Yoon et al. (2019) suggest that this leads to many redundant distance-computations that can be avoided if observations at similar locations are handled in groups, and the so-called "net effect" is exploited (i.e., the effect that new observations can cancel out the effects of expired ones if they arrive in similar places).

To exploit the "net effect", Yoon et al. (2019) propose a set-based approach that processes both the expired window and the new window concurrently and groups spatially close observations, allowing the approach to skip calculations that would be redundant (e.g.,+1 and -1 observation in the same place). In comparison, a traditional approach would first remove expired observations from the window, then add new observations, and finally compute anomalies on that window. Due to the "curse of high dimensionality", the efficiency of NETS drops as the dimensionality of the data grows to a large scale. To ameliorate the issue, NETS applies two-level dimensional filtering, which means that it detects anomalies and normal data in a subspace with fewer dimensions, after which it processes the remainder of the data in the full-dimensional space. This approach allows NETS to process even a sparse high-dimensional data stream efficiently.

As Yoon et al. (2019) show in their evaluation, these optimizations allow NETS to outperform other state-of-the-art algorithms like MCOD and LEAP by several orders of magnitude (10x and 24x, respectively). The time-complexity of NETS is lower than comparable algorithms, but the space-complexity is about the same.

3.3 Reconstruction-based Approaches

Approaches with a foundation on reconstruction perform a regression on the training set to predict a new event and compute the distance between the predicted value and the actual observation. The anomaly score of an event can be computed based on that distance (i.e., the reconstruction error), allowing for the classification of an incoming event as anomalous or normal. Reconstruction-based approaches work without any assumptions about the data and work well even in high-dimensionality contexts. They tend to have high complexity in the training phase but are fast when it comes to evaluating a new observation (Pimentel et al., 2014).

A special set of reconstruction-based approaches are called spectral methods, which are methods that transform the high-dimensional data space into a lower-dimensional one before attempting to distinguish anomalies from normal observations. These methods work under the assumption that anomalies are easier to distinguish in fewer dimensions. The most common technique for performing such a reduction in dimensionality is Principal Component Analysis (PCA) (Pimentel et al., 2014).

Another subgroup of reconstruction-based approaches, the neural-network methods, is based on a hill-climbing optimization of a fixed set of parameters. Neural-network models can be highly sensitive to the parameters they are trained with, which can make it difficult to use them with high-dimensional data. Additionally, as many reconstruction-based approaches, including neural networks, are based on iterative optimization, the choice of optimization method is often critical for the success of the model (Kanarachos et al., 2017).

3.3.1 Contextual anomaly detection framework for big sensor data (Hayes and Capretz, 2015)

Hayes and Capretz (2015) apply a two-layer approach consisting of a content anomaly detector that detects spatial anomalies with a fast probabilistic predictor, as well as a contextual anomaly detector that subsequently processes anomalies with a clustering-based approach. The contextual detection layer includes contextual information by building clusters for known sensor profiles (i.e., the average over a set of sensors in the same context) and subsequently training a classifier for each cluster. Once a point anomaly is detected based purely on its content, it can be further evaluated if the anomaly persists as a contextual anomaly in the context of its sensor profile (i.e., by checking whether it belongs to the specific cluster). This two-layer approach has the potential of significantly reducing the rate of false-positives. However, as both layers of the system are trained offline, a periodical retraining would be necessary to account for concept drift.

When compared against the outliers package as included in the R distribution, the contextual detection framework in Hayes and Capretz (2015) performed equally well but introduced the capability of working in an online environment. The described performance benefit, however, only holds under the assumption that the algorithm processes many low-dimensional records instead of fewer high-dimensional ones, as it focuses on reducing the number of records running through the entire process. Furthermore, the approach includes several optimizations specific to the big data context. The algorithm has been designed to be parallelizable such that only the contextual clustering layer requires centralized execution. The content-based detection layer can be executed in parallel on a cluster of machines. Additionally, the computational requirements are reduced by the layering approach, as the computationally expensive contextual detector only needs to process events that have been marked as anomalies by the fast content-based detection layer.

3.3.2 Time Series Anomaly Detection (Shipmon et al., 2017)

Shipmon et al. (2017) perform a reconstruction of observations in univariate time-series using neural-network-based regression. The reconstruction-error of their model is analyzed with an ensemble of a probabilistic gaussian tail-probability thresholding and an accumulator that increases for collective anomalies (i.e., on sequences of point anomalies), thereby filtering out anomalies caused by pure noise. As their model profits from a higher number of features, Shipmon et al. (2017) extend the dimensionality of their time-series by engineering additional features from the timestamp and past labels (e.g., by using derivatives).

When evaluating their approach on various static datasets, both artificial and real ones, and comparing the results with simple baseline models, Shipmon et al. (2017) show that the approach shows robust results on periodic time-series, but does not perform well on non-periodic ones. While the used neural-network can, in principle, be trained incrementally, the need for labeling would make it hard to apply the approach to online processing scenarios.

3.3.3 Unsupervised real-time anomaly detection for streaming data (Ahmad et al., 2017)

Ahmad et al. (2017) propose a "theoretical framework for sequence learning in the cortex" that applies the Hierarchical Temporal Memory (HTM) sequence learning methodology to the domain of anomaly detection. HTM is based on principles of neuroscience that have been transferred to the area of machine intelligence, resulting in a system that emulates synapses and neurons in a human brain.

The anomaly detector introduced by Ahmad et al. (2017) builds an internal model from spatiotemporal data and continuously evolves during processing. When processing a new observation, the HTM system predicts expected future behavior and compares it to the real observation to get a measure of the reconstruction error. Based on the reconstruction error, the approach probabilistically computes the likelihood of the system being in an anomalous state, and thresholds that likelihood to reach its final decision. As the threshold is not directly applied to the reconstruction error, the number of false alarms because of noise can be significantly reduced.

In their evaluation based on the open-source NAB scoring system and datasets, Ahmad et al. (2017) show that their approach outperforms almost any other of the evaluated state-of-the-art approaches, especially when a low number of false-negatives is rewarded. However, the HTM approach incurs a significantly higher latency (i.e., time for processing a new observation in real-time) when compared to these same other approaches.

3.3.4 Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform (Kanarachos et al., 2017)

The framework proposed by Kanarachos et al. (2017) transforms univariate time series into multiple signal subcomponents (each on a different time-scale). This so-called "wavelet transformation" reduces the influence of noise and enables the framework to evaluate the input at different time-scales. Every subcomponent ("wavelet") is processed by a dedicated neural network that learns the normal behavior of its respective time-scale. Combining the component networks and integrating them with additional layers then results in a deep neural network model that can reconstruct the expected signal. The final decision of the anomaly detector is based on features extracted by applying a Hilbert transformation to the error signal (i.e., the time series of reconstruction errors), and by thresholding the resulting values with a probabilistic method that optimizes statistical significance and ROC.

Using a neural network enables incremental learning and online processing, as weights can be updated with each new observation (or mini-batch thereof). However, the anomaly detector detects temporal anomalies (i.e., anomalies in a time context) based on signal frequencies and transformations and is, therefore, dependent on having a window of observations available. Furthermore, while incremental learning aids in the evolution of the model and in recognizing emerging patterns, concept drift has not been explicitly accounted for in the approach (e.g., by means of expiration).

Kanarachos et al. (2017) evaluated their work on two real-life use cases from signal processing (e.g., earthquake detection), and showed that the approach could be applied to different datasets without requiring significant manual tuning or the possession of domain knowledge, which was the main goal of the overall work.

3.4 Domain-based Approaches

Domain-based methods compute a decision boundary around the "normal area" of a dataset without deriving an explicit generative distribution. Observations that lie outside of this boundary can then be classified as anomalies. While working only on the domain is advantageous and requires relatively less training data, this also means that a domain-based model is easily negatively influenced by outliers in the dataset (Pimentel et al., 2014).

Support Vector Machines (SVM) are a popular method for domain-based anomaly detection as they build a decision boundary (a hyperplane) using only data points close to the border (i.e., support vectors), ignoring the data points within the boundary. Working only with support vectors means that SVMs do not depend on knowing the distribution of the training data and can work only with the domain. As SVMs are only applicable to a batch processing context, there has been some research on other domain-based methods that can work with online processing and incremental learning (Pimentel et al., 2014).

3.4.1 Isolation Forest (Liu et al., 2008)

Liu et al. (2008) propose an anomaly detection approach that isolates observations on a multivariate data stream by recursively partitioning the data space with decision-tree rules. As an anomaly is easier to isolate than normal observations, anomalies tend to require only a few levels of partitioning and should thus be located higher-up in the resulting decision-tree. The approach exploits this pattern in computing an anomaly score based on the normalized path-length of a new observation. To account for the randomness in tree partitioning, the approach combines a parametrized number of trees in an ensemble and uses the more robust averaged statistics. Additionally, Liu et al. (2008) apply a subsampling strategy that enables learning on a small part of the overall data (e.g., a subset of 256 observations).

In-depth evaluation shows that the approach achieves the highest AUC on most of the considered evaluation datasets. The subsampling strategy does not significantly impact accuracy but reduces the processing time of the approach by a large amount, allowing the approach to work faster than benchmarked approaches (on large datasets). As the approach separates training and testing stages, it has to be retrained periodically to account for concept drift in evolving data streams.

3.4.2 Data Stream Anomaly Detection through Principal Subspace Tracking (dos Santos Teixeira and Milidiú, 2010)

dos Santos Teixeira and Milidiú (2010) introduce a new approach for domain-based anomaly detection that works in a projected subspace of univariate numerical input streams. The proposed approach performs a dimensionality reduction in an incremental way (by estimation) and tracks the number of latent variables needed to explain the observations in the resulting subspace. An increase in the number of required variables indicates the beginning of a new collective anomaly.

Existing approaches most often apply PCA and are not practical in the streaming context, as they require expensive calculations or need to persist all data in memory. Additionally, such methods tend to be parameterized for the number of dimensions in the projection space, which makes it hard to adapt to changes in the stream. The method proposed by dos Santos Teixeira and Milidiú (2010), which is called FRAHST, automatically estimates the number of dimensions and incrementally estimates the statistical properties of the data using a decay factor, allowing the approach to efficiently perform dimensionality reduction in a dynamic and continuously evolving context.

The advantages of FRAHST are its robustness and its practicable complexity, as well as the low number of parameters (decay rate and desired error). When evaluated against four other algorithms on several real and synthetic datasets, FRAHST is the only technique that consistently identifies anomalies with an F1 score above 80 percent. Additionally, the algorithm has been implemented and tested in the data centers of an internet provider, which motivates its applicability to a real context.

3.5 Information-theoretic Approaches

Approaches based on information theory calculate the information content of data with measures like entropy. If a new observation significantly changes the information content of the overall dataset, that observation can, therefore, be considered anomalous. Information-theoretic methods are highly dependent on the choice of the measure for information, as the measure needs to be able to detect the effects of anomalies, which is only possible if that effect is sufficiently large. Additionally, it is often challenging to derive a suitable score from the measured effects. The exact computation of information content is also relatively expensive (i.e., the test phase), which is a critical impediment in the streaming context (Kanarachos et al., 2017; Pimentel et al., 2014).

3.5.1 Entropy-Based Anomaly Detection for In-Vehicle Networks (Muter and Asaj, 2011)

As automotive systems are increasingly connected and integrated with external systems, their attack surface grows likewise. To address this issue, Muter and Asaj (2011) propose a system that analyzes vehicle network communications at runtime. As vehicles have a very long lifetime but are hard to update, Muter and Asaj (2011) suggest that an anomaly detection system could be a better solution than traditional signature-based intrusion detection that needs to be updated regularly.

Muter and Asaj (2011) define their notion of normality using the concept of entropy, as it does not require knowing anything about the data. As Muter and Asaj (2011) state, "entropy allows an abstract representation of the randomness of this data." While entropy can be high in regular computer networks with high randomness in network communications, the communication in a vehicular network is highly standardized (e.g., with allowed values, value ranges, and frequencies), which results in lower entropy. Because of this, changes in entropy (e.g., a sudden drop due to flooding with many identical messages) can serve as a suitable anomaly detection measure in such a network.

In their evaluation on real vehicles, Muter and Asaj (2011) find that their approach can successfully identify manually executed attacks in vehicular networks and is more practically applicable than related work. Firstly, it only requires data that represents normal behavior, not explicit specifications about message formats or content restrictions, which makes it easy to transfer and deploy to a variety of vehicles. Additionally, the approach is usable in the low-resource embedded computing context, as it is based on relatively cheap computations like tracking changes in entropy and applying a threshold.

3.5.2 Statistical Techniques for Online Anomaly Detection in Data Centers (Wang et al., 2011)

Current approaches for data center monitoring are often based on assumptions about the distribution of the data and thresholds that are trained offline and kept at a constant level during processing. Therefore, the methods are not well-fit for the evolving patterns and bursty network behavior prevalent in data centers. To improve on this, Wang et al. (2011) introduce a new online processing approach for univariate time series that processes streaming data in windows and examines the window contents using a statistical test. More specifically, the distribution of observations in each window is compared to the distribution of observations in different periods/contexts using a test based on relative entropy (i.e., multinomial goodness-of-fit). When the two distributions do not match well enough, the null hypothesis can be rejected, and the window is marked as anomalous.

To account for changing usage patterns of network traffic based on the time-of-day, Wang et al. (2011) additionally compute the distribution of different temporal contexts (e.g., traffic between 11:00 and 12:00) and compare windows to the matching context. In their evaluation, Wang et al. (2011) show that the approach performs better than comparable Gaussian approaches, especially if different contexts are evaluated to form an ensemble decision (e.g., the corresponding time bucket, the recent past, and all past). Furthermore, the computations performed by the approach are lightweight in both time and space, as all that is required is the tracking and computation of statistics regarding frequencies of occurrence.

3.5.3 Online Anomaly Detection over Big Data Streams (Rettig et al., 2015)

Rettig et al. (2015) introduce an online anomaly detection approach that detects contextual (temporal) anomalies on multiple data streams. The approach continuously maintains aggregate statistics across windows on a stream (relative entropy) and across streams (correlation coefficient). These separate measures enable the detection of anomalous changes in the behavior of one stream (i.e., if the relative entropy changes significantly), as well as the detection of behavior that is anomalous in the context of a related stream (e.g., abrupt changes causing the correlation to drop). Rettig et al. (2015) implemented their approach in a real stream processing system (Kafka and Spark) to showcase their focus on generalization and real-world applicability.

When evaluated on a real telecommunications infrastructure, the approach detects both anomalies caused by users and anomalies through infrastructure failure. Additional experiments with different numbers of processing nodes showed that the approach scales well and can be used to process large amounts of data simultaneously. The overall evaluation is predominantly based on domain knowledge and in-depth manual analysis, which clarifies the reasons for the observed results, but could make the evaluation harder to replicate.

4 Conclusion

In the previous chapters, we have introduced a foundational set of features and properties that can be used to compare anomaly detection approaches for streaming systems, including the types of anomalies that can be identified, whether an approach explicitly accounts for concept drift and noise in the underlying data stream, and how an approach has been evaluated and documented. Furthermore, we have introduced and categorized several approaches from past and recent research, including popular methods like DenStream and MCOD. We finally used our features to align the approaches on 11 dimensions (see Table 3.1), providing a big-picture overview and allowing for easier comparison.

Besides the foundational features we have described, many often more challenging problems cannot be quantified as easily and have been left for further analysis. We conclude our work with a short motivation for some of these problems.

Applicability in Practical Environments. In today's high-velocity, high-volume streaming context, it is often practically infeasible to process a stream in a centralized fashion (i.e., on a single processing unit). However, a large part of algorithms for data mining and anomaly detection (e.g., kNN and SVM) have been developed for and evaluated using single machine processing (Hayes and Capretz, 2015). What is needed to satisfy the requirements of today's context are systems whose computations are efficiently parallelizable so that they can run in distributed processing environments like Apache Flink (Toliopoulos et al., 2019). It is necessary to incorporate parallelism (e.g., by applying data parallelism paradigms like MapReduce), as according to Pimentel et al. (2014), "Efficient novelty detection techniques should be scalable to large and high-dimensional datasets." Furthermore, emerging patterns like distributing data collection and analysis to low-power devices or moving computation and persistence closer to where it is used (i.e., *edge computing*) should be accounted for as well.

Computational Complexity and Performance. Another factor that is important in stream processing systems is the speed of processing (i.e., the computational complexity) (Ahmad et al., 2017). Modern computing environments require a paradigm shift towards less expensive approximative algorithms that can potentially provide similar prediction performance (even though they perform worse during training) (Hayes and Capretz, 2015). Online stream processing systems are highly dependent on low-latency algorithms, meaning that the testing phase of such algorithms must be rapid. Computations during testing that have a high complexity can quickly lead to bottlenecks and slow down processing to unusable levels (Ahmad et al., 2017).

Unclear or Unrealistic Assumptions. Many anomaly detection algorithms base on a set of statistical assumptions about the data and the generative process. If these assumptions do not hold (they often do not) or are not documented, the algorithms might not work as well as would be expected. One assumption that is often made is that the distribution that generates a data stream must have a particular shape and be subject to specific constraints (e.g., "normal" distribution). Other common assumptions concern the size or dimensionality of the dataset to be processed. Some algorithms work better on large datasets with high dimensionality, while others suffer from the "curse of high dimensionality". In general, the fewer assumptions an algorithm is based on, the more generalizable it is to real-world problems, and the better it tends to work (Ahmad et al., 2017).

Transferability. Transferability describes how well an algorithm can be transferred to other use cases and domains and is important when working in highly complex environments, but is rarely explicitly considered in research (Kanarachos et al., 2017). Many anomaly detection approaches are tightly coupled to a specific context, have been tuned with significant domain knowledge, or do not provide the parameters necessary to apply them in a different context or to different data. This makes it harder to exploit research that has been performed in different application areas, even if it could otherwise be very useful.

Commercial Compromise. While we have focused on approaches that detect or classify anomalies as autonomously as possible, the anomaly detection problem is often also a problem of finding a suitable commercial compromise between cost and accuracy. Due to the impossibility of catching every single anomaly even with large investments, it often makes sense to build systems with several layers, including manual processes. For example, once a fraud-detection system detects a highly-probable anomaly, it might assign this anomaly to a human for immediate investigation (Bolton and Hand, 2002). Anomaly detection approaches in research should account for this by appropriate means of parametrization (i.e., allowing for the system to be tuned in terms of the tradeoff between accuracy and cost).

References

Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., and Sohler, C. (2012). StreamKM++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics*, 17(1):2.1.

Agrawal, B., Wiktorski, T., and Rong, C. (2017). Adaptive real-time anomaly detection in cloud infrastructures. *Concurrency and Computation: Practice and Experience*, 29(24):e4193.

Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009). New Ensemble Methods For Evolving Data Streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, page 139, Paris, France. ACM Press.

Bolton, R. J. and Hand, D. J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235–255.

Cao, F., Estert, M., Qian, W., and Zhou, A. (2006). Density-Based Clustering over an Evolving Data Stream with Noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 328–339. Society for Industrial and Applied Mathematics.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58.

Chandola, V., Banerjee, A., and Kumar, V. (2012). Anomaly Detection for Discrete Sequences: A Survey. *IEEE Trans. Knowl. Data Eng.*, 24(5):823–839.

dos Santos Teixeira, P. H. and Milidiú, R. L. (2010). Data Stream Anomaly Detection through Principal Subspace Tracking. In *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, page 1609, Sierre, Switzerland. ACM Press.

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

Haibo He, Sheng Chen, Kang Li, and Xin Xu (2011). Incremental Learning From Stream Data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914.

Hayes, M. A. and Capretz, M. A. (2015). Contextual anomaly detection framework for big sensor data. *Journal of Big Data*, 2(1).

Heindorf, S., Potthast, M., Stein, B., and Engels, G. (2016). Vandalism Detection in Wikidata. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16*, pages 327–336, Indianapolis, Indiana, USA. ACM Press.

Hodge, V. J. and Austin, J. (2004). A Survey of Outlier Detection Methodologies. page 42.

Kanarachos, S., Christopoulos, S.-R. G., Chroneos, A., and Fitzpatrick, M. E. (2017). Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform. *Expert Systems with Applications*, 85:292–304.

Keogh, E., Lin, J., and Fu, A. (2005). HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages

226–233, Houston, TX, USA. IEEE.

Kontaki, M., Gounaris, A., Papadopoulos, A. N., Tsichlas, K., and Manolopoulos, Y. (2011). Continuous Monitoring of Distance-Based Outliers over Data Streams. In *2011 IEEE 27th International Conference on Data Engineering*, pages 135–146, Hannover, Germany. IEEE.

Lavin, A. and Ahmad, S. (2015). Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44, Miami, FL, USA. IEEE.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, Pisa, Italy. IEEE.

Miller, Z., Dickinson, B., Deitrick, W., Hu, W., and Wang, A. H. (2014). Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73.

Muter, M. and Asaj, N. (2011). Entropy-Based Anomaly Detection for In-Vehicle Networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115, Baden-Baden, Germany. IEEE.

Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99:215–249.

Rettig, L., Khayati, M., Cudre-Mauroux, P., and Piorkowski, M. (2015). Online Anomaly Detection over Big Data Streams. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1113–1122, Santa Clara, CA, USA. IEEE.

Schneider, M., Ertel, W., and Ramos, F. (2016). Expected Similarity Estimation for Large-Scale Batch and Streaming Anomaly Detection. *Machine Learning*, 105(3):305–333. arXiv: 1601.06602.

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., and Edwards, S. (2017). Time Series Anomaly Detection. *arXiv:1708.03665*, page 9.

Talagala, P. D., Hyndman, R. J., Smith-Miles, K., Kandanaarachchi, S., and Muñoz, M. A. (2019). Anomaly Detection in Streaming Nonstationary Temporal Data. *Journal of Computational and Graphical Statistics*, pages 1–21.

Toliopoulos, T., Gounaris, A., Tsichlas, K., Papadopoulos, A., and Sampaio, S. (2019). Continuous Outlier Mining of Streaming Data in Flink. *arXiv:1902.07901 [cs]*. arXiv: 1902.07901.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Technical Report 106.2, Computer Science Department, Trinity College Dublin, Dublin.

Wang, C., Viswanathan, K., Choudur, L., Talwar, V., Satterfield, W., and Schwan, K. (2011). Statistical Techniques for Online Anomaly Detection in Data Centers. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 385–392, Dublin, Ireland. IEEE.

Wang, K. and Stolfo, S. J. (2004). Anomalous Payload-Based Network Intrusion Detection. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Jonsson, E., Valdes, A., and Almgren, M., editors, *Recent Advances in Intrusion Detection*, volume 3224, pages 203–222. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wang, W., Lu, D., Zhou, X., Zhang, B., and Mu, J. (2013). Statistical wavelet-based anomaly detection in big data with compressive sensing. *EURASIP Journal on Wireless Communications and Networking*, 2013(1).

Widmer, G. and Kubat, M. (1993). Effective learning in dynamic environments by explicit context tracking. In Siekmann, J., Goos, G., Hartmanis, J., and Brazdil, P. B., editors, *Machine Learning: ECML-93*, volume 667, pages 227–243. Springer Berlin Heidelberg, Berlin, Heidelberg.

Widmer, Gerhard and Kubat, Miroslav (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101.

Yoon, S., Lee, J.-G., and Lee, B. S. (2019). NETS: Extremely Fast Outlier Detection from a Data Stream via Set-Based Processing. *Proceedings of the VLDB Endowment*, 12(11):1303–1315.

Zheng, Y., Zhang, H., and Yu, Y. (2015). Detecting collective anomalies from multiple spatio-temporal datasets across different domains. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '15*, pages 1–10, Bellevue, Washington. ACM Press.