University of Zurich[UZH]

Online Anomaly Detection on Multivariate Data Streams

Thesis August 18, 2019

Christoph Weber
of Rorschach SG, Switzerland

Student-ID: 10-924-231
christoph.weber3@uzh.ch

Advisor: **D. Dell'Aglio, PhD**

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

Acknowledgements

I would like to express my sincere gratitude to my advisor Daniele Dell'Aglio, PhD for his guidance, advice, and valuable suggestions during the whole thesis. I want to thank Professor Abraham Bernstein, PhD, for giving me the opportunity to work on such an exciting research project. Further, I want to thank our project partners to letting me participate in their project and supporting me whenever needed. Moreover, I want to thank Angela Estermann and David Ackermann for motivating me during the whole thesis and proof-reading my work.

Zusammenfassung

Immer mehr Datenquellen erzeugen kontinuerlich und sich schnell ändernde Datenströme und benötigen massgeschneiderte Lösungen, um unerwartete Ereignisse zu erkennen. Die Erkennung von Ausreissern in univariaten Datenströmen findet bereits erhebliche Beachtung, vor allem im Finanzbereich, während die Erkennung multivariater Anomalien, insbesondere ohne Grundwahrheit, weniger erforscht ist. Wir präsentieren den Stand der Forschung in der Anomalieerkennung im Allgemeinen, ihre Anwendung für Datenströme und Techniken zur Auswertung ohne Grundwahrheit. Wir implementieren einen dichtebasierten Clustering-Algorithmus, der multivariate Datenströme mit Mikro-Clustern zusammenfasst, und bewerten ihn anhand von synthetischen und realen Datensätzen. Wir schlagen eine Erweiterung des Algorithmus vor, um Trends zu berücksichtigen und zwischen Pionieren und Ausreissern richtig zu unterscheiden. Die durchgeführten Experimente zeigten eine Leistungsverbesserung, die durch die vorgeschlagenen Trendeinflussparameter wursacht wurde, und zeigen eine Korrelation zwischen einer intrinsischen Dateneigenschaft und der Performanc der Anomalieerkennung, was die Abstimmung von Hyperparametern ohne Grundwahrheit ermöglicht.

Abstract

The number of data sources continuously producing fast-changing data streams and needing tailor-made solutions to detect unexpected events increases rapidly. Outlier detection in univariate data streams already receives considerable attention, mainly in financial data, while multivariate anomaly detection, especially without ground truth, is less explored. We present state of the art in anomaly detection in general, its adoption for data streams and techniques for evaluation without ground truth. We implement a density-based clustering algorithm that summarizes multivariate data streams with micro clusters, and we evaluate it on synthetic and real-world data sets. We propose an extension of the algorithm to incorporate data drift to distinguish between pioneers and outliers correctly. The performed experiments show a performance improvement caused by the proposed drift-influence hyperparameters and revealed a correlation between an intrinsic data property and the anomaly detection performance, which allows hyperparameter tuning without ground truth.

Table of Contents

Abbreviations

ARIMA AutoRegressive Integrated Moving Average

AUC Area Under Curve

c-micro cluster Core micro cluster

o-micro cluster Outlier micro cluster

p-micro cluster Potential core micro cluster

PCA Principal Component Analysis

PR Precision-Recall

ROC Receiver Operating Characteristic

$\boldsymbol{\alpha}$ Outlier score

$\boldsymbol{\beta}$ Outlier tolerance factor

$\boldsymbol{\delta}$ Drift-distance-influence hyperparameter

$\boldsymbol{\epsilon}$ Maximum micro cluster radius

$\boldsymbol{\zeta}$ Minimum micro cluster size

$\boldsymbol{\eta}$ Random angle for synthetic data set creation

$\overrightarrow{\boldsymbol{\theta_t}}$ Overall drift vector

$\boldsymbol{\lambda}$ Decay factor

$\boldsymbol{\mu}$ Core weight threshold

$\boldsymbol{\sigma}$ Standard deviation

$\boldsymbol{\Psi}$ Cosine similarity

$\boldsymbol{\omega}$ Drift-weight-influence hyperparameter

$\boldsymbol{t_{1/2}}$ Half-life time

1

Introduction

The number of data sources continuously producing fast-changing data streams and needing tailor-made solutions to detect unexpected events increases rapidly. Data streams are fast-changing and potentially infinite and produce massive amounts of temporally ordered samples [Han and Ding, 2009]. Evolving data streams are ubiquitous, common fields of use are applications of the Internet of Things (IoT) which continuously produce large amount of sensor data, social media feeds, or real-time stock exchange data. In all these fields, we can find the motivation to detect anomalous events as fast as possible, e.g. a malfunctioning sensor or fraudulent behavior can inflict more damage the longer they go undetected. Traditional data analysis methods can not extract the full potential or are not applicable. E.g. conventional anomaly detection methods require the whole data set to define normality, which conflicts with the continuity and potential infinity of data streams.

While the data sources are often already massive data streams, the techniques used to analyze them are not tailored for the use case. Large amounts of data are dropped early in the process or never analyzed. We apply approaches that can overcome the challenges of data streams and profit of their full potential.

Univariate time series receive already a decent amount of attention, mainly in financial data. Multivariate sequential data poses greater challenges and is thus less explored. We implement DenStream proposed by [Cao et al., 2006] to find anomalies in an evolving multivariate data stream with arbitrary shape. The algorithm summarizes samples in a data stream with micro clusters and adopts them to changes over time.

If a data stream contains drift, pioneer samples are marked as anomalous as long as not enough samples lie in the pioneer region. The goal of this thesis is to improve the anomaly detection in multivariate data streams by incorporating drift to label pioneers as inliers and apply a state of the art solution to real data.

We introduce two additional hyperparameters and evaluate their contribution to anticipate data drift and to improve the outlier detection performance. We perform multiple experiments on synthetic data with ground truth to assess our contribution and apply the DenStream outlier detection algorithm to real-world data to determine the value when used in a productive environment. We perform a thorough hyperparameter analysis and show their sensitivity and influence on the used anomaly detection algorithm. The two additional hyperparameters can improve the outlier detection performance, but their usage comes with an increase in uncertainty. Our results support one of the two

proposed cardinality reduction methods proposed by [Putina et al., 2018] while they discourage the other. We found a correlation between an intrinsic algorithm property and the performance score, which helps to tune hyperparameters when no ground truth is available.

Chapter 2 presents the related work of stream mining and anomaly detection. It highlights the challenges and potential solutions when performing outlier detection in a streaming environment. We cover the methodology in Chapter 3. It presents the knowledge discovery process and describes the fundamental algorithm. We evaluate the algorithm properties and our contribution on synthetic data in Chapter 4. We apply the anomaly detection algorithm to real-world scenarios in Chapter 5. We state the limitations and deduce appropriate future work in Chapter 6. Chapter 7 concludes our work.

2

Related Work

This chapter covers the related work of this master's thesis. It highlights the theoretical foundations of knowledge discovery in data streams, anomaly detection in general and the adaptions needed in a streaming environment. Furthermore, it introduces DenStream, an algorithm for clustering in data streams, and its implementation to detect anomalies.

2.1 Stream Mining

Stream mining encompasses techniques centering around knowledge discovery in continuous data streams. These data streams are temporally ordered, fast-changing, massive, and potentially infinite [Han and Ding, 2009]. There are three main limiting factors in data mining: Time, memory, and sample size [Domingos and Hulten, 2000]. More and more applications continuously produce so much data that traditional data analysis is unfeasible or impossible because of the aforementioned limiting factors.

Stream mining algorithms typically work with only one pass of each sample, i.e. raw data is accessible only once. Additionally, the temporal component of the data stream must be taken into consideration, which may induce data evolution and decaying importance of a point over time. Generally, an algorithm can only use data in a limited time window to extract relevant information [Aggarwal, 2007]. Furthermore, algorithms typically need to deliver results (near) online, required by the temporal component and the need of data analysts to take action as soon as possible [Khan and Fan, 2012].

Traditional data analysis techniques such as classification or clustering compare data samples of the complete data set to reach a conclusion. In a streaming environment, aggregations of past samples can represent values outside the time window to some degree. Rolling mean and standard deviation instead of single points are examples for primitive aggregations. Micro clusters which represent multiple points within a particular area through their weight and influence radius is a more advanced method. Partitioning the data space or pruning mechanisms ensures finitely many micro clusters, required because of limited memory space [Aggarwal, 2007].

2.2 Anomaly Detection

Anomaly detection in machine learning is the goal to find "not-normal" samples in data. These abnormal samples are commonly referred to as anomalies or outliers. This thesis utilizes the terms interchangeably. This section shows the characteristics of anomalies and explains how to detect and evaluate them.

2.2.1 Characteristics of Anomalies

Anomalies have two important characteristics [Chandola et al., 2009]:

Different The features of anomalous samples are different from the features of normal samples.

Rare The occurrence of anomalies is rare compared to normal samples.

Anomalies and noise are not the same. The distinction may vary, but anomalies generally contain abnormalities that contain *useful* and *true* information that may arise from the natural variation, while noise is not useful and should be removed from a data set [Salgado et al., 2016]. [Chandola et al., 2009] describe three different types of anomalies. They differ in the relevant scope and form:

Point Anomalies A sample is a *point anomaly* if it is considered anomalous on the global scale, i.e. to all other data points. This is the simplest case, as no connections to a neighborhood or context must be considered. E.g. the highest building of the world, Burj Khalifa[1], with a height of 828m, is much taller than the average building and also than the second highest in the world (Shanghai Tower[2] with 632m).

Contextual Anomalies A sample is a contextual anomaly if it is anomalous for a given context but not otherwise. The context must be defined as part of the anomaly detection process. Features can be split into contextual and behavioral. Contextual features define the context, e.g. time or coordinates, while behavioral features are non-contextual, e.g. number of people watching TV or temperature. In time-series data, the focus mostly lies on contextual anomalies in consequence of the time axis. E.g. 1 million viewers watching the TV channel SRF1 during prime time at 19:30 is normal, but 1 million viewers during the night at 04:00 is anomalous.

Collective Anomalies Multiple samples together form an anomalous collection, while the single samples may be normal. The data points must be related to each other to be able to form a collective anomaly. Techniques to find this type differ significantly from the other two and are not relevant in this thesis. E.g. doors in a building may be unlocked electronically, but all of them unlocking at the same time is anomalous and may indicate fraudulent actions.

[1] https://www.burjkhalifa.ae/en/the-tower/facts-figures/
[2] http://www.shanghaitower.com/shanghaizhongxinEnglish/15.php

2.2.2 Detection Techniques

The approaches to find anomalies can be categorized into three different anomaly detection types [Goldstein and Uchida, 2016]:

Supervised Anomaly Detection The training and test data contains fully labeled samples. The label classifies each sample into normal and abnormal. This is very similar to a binary classification task with the difference that the classes are highly unbalanced. Because of this difference, some classification algorithms are better suited than others.

Semi-Supervised Anomaly Detection The training data is fully labeled as in the supervised case, but it contains only normal samples and no anomalies. Thus an algorithm can only learn the normal behavior. It detects anomalies by comparing it to the model describing normality.

Unsupervised Anomaly Detection No labels exist on the data set. A model finds anomalies only based on intrinsic properties such as density or distance.

[Chandola et al., 2009] highlight multiple challenges in the field of which two are especially relevant for this thesis. First, labeled data is hard to find. Even if there exists some ground truth, it represents only known behavior but does not prepare for possible new approaches, e.g. the user behavior during a security attack can be very different from case to case. This makes unsupervised anomaly detection the most common type. Second, normality can change over time, demanding the model to adopt. Anomalies at one point in time may become normality at another. This thesis focuses on tackling the second challenge. From here on, we focus on unsupervised anomaly detection.

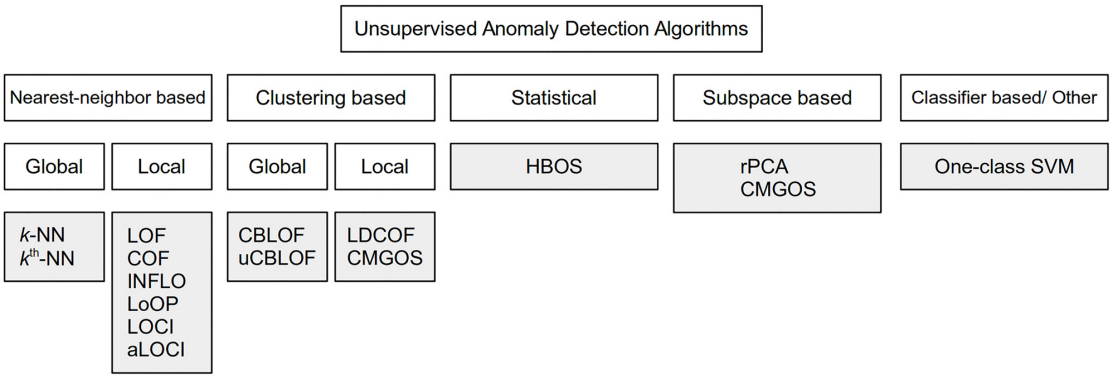

Figure 2.1: Categorization of unsupervised anomaly detection algorithms [Goldstein and Uchida, 2016]

[Goldstein and Uchida, 2016] describe multiple types of algorithms for unsupervised anomaly detection. The two main types are density- and cluster-based. In the first type, samples in low-density areas are anomalous. An algorithm can compare the density globally by calculating the anomaly score via the distance from the sample to the k^{th}

or all k nearest neighbors. Local outliers can be found with the Local Outlier Factor (LOF) which compares the density locally with other samples in the same area. Cluster-based algorithms determine multiple clusters and calculate the outlier score based on the distance to these clusters. Figure 2.1 depicts a categorization of unsupervised anomaly detection algorithms.

2.3 Anomaly Detection in Streams

A streaming environment introduces new challenges, as explained in Section 2.1. This section explores the techniques presented in related work and shows appropriate performance evaluation methods for anomaly detection in streams.

2.3.1 Techniques

Computationally lightweight statistical methods such as sliding threshold, extreme studentized deviate, ARIMA, and PCA are the most common methods in practice [Ahmad et al., 2017]. Advanced model-based techniques are also possible, but require more background knowledge and adoption to the specific domain. [Mousavi et al., 2015] present an overview of clustering methods in data streams:

Hierarchical Methods Create a taxonomy which summarizes the samples into more and more general categories.

Partitioning Methods Construct k partitions of the data space and evaluate each by some criterion, e.g. sum of square errors.

Grid-based Methods Separate the data space into cells through a grid, then merge the cells to build clusters.

Density-based Methods Evaluate how close the samples lie in the areas of the data space.

Model-based Methods Run a hypothesized model for every cluster and provides the data that fits the model perfectly.

The main limitations - data processed only once and limited storage - reduce the options for unsupervised anomaly detection methods. Most approaches to find anomalies presented in literature focus on distance and density-based algorithms. [Angiulli and Fassetti, 2007] and [Kontaki et al., 2011] find anomalies based on the neighborhood in a sliding window - if a sample has less than k neighbors within distance R, it is an anomaly. The size of the time-based sliding window limits the number of samples the algorithm has to classify as inlier/outlier. Instead of distance-based, [Amini and Wah, 2010] apply density-based clustering. [Assent et al., 2012] propose a hierarchical clustering approach with a tuple based sliding window. The algorithm traverses the hierarchical clustering tree until interrupted by an incoming sample. Thus, the anomaly detection gets better

the more time passes until the next sample arrives. This ensures constant memory size independent from data frequency.

2.3.2 Evaluation Techniques

Anomaly detection in data streams incorporates multiple difficulties concerning performance evaluation. Anomalies are per definition rare occurrences, improvements in predicting the rare class thus change the overall classification result only marginally. The area under curve (AUC) of the receiver operating characteristic (ROC) curve which is often used in classification is thus unfeasible, as it does not balance the true-positives and true-negatives. [Davis and Goadrich, 2006] propose the use of the precision-recall (PR) curve to evaluate the performance of highly skewed data sets.

The PR-AUC is not sensitive to biased data sets but still requires a ground truth to evaluate the performance. However, anomaly data sets are rarely labeled, as explained in Section 2.2.2. Evaluation of unsupervised learning is an ongoing research topic, primarily focused on clustering algorithms. [Rand, 1971] describes four characteristics that define the quality of a clustering algorithm:

Natural Clusters How well does the method retrieve the clusters inherent in the data?

Sensitivity How sensitive is the clustering result to small changes in the data?

Missing data How sensitive is a method to missing data?

Method comparison How different is the result when applying two methods to the same data?

Generally, data analysts evaluate anomaly detection tasks with background knowledge and use some known anomalies as a baseline to tune the hyperparameters and data preparation. [Goix, 2016] proposes Excess-Mass and Mass-Volume curves to discriminate accurately between algorithms in an unsupervised anomaly detection setting w.r.t. PR and ROC based criteria. This solves both explained problems in anomaly detection evaluation but still needs to be adopted by research and industry.

2.4 DenStream

This section first describes the DenStream clustering algorithm by [Cao et al., 2006]. Second, it evaluates the implementation presented by [Putina et al., 2018], which utilizes DenStream but optimizes the algorithm to find anomalies instead of clusters. Third, it presents the hyperparameters used in the algorithm by [Cao et al., 2006] and the proposed improvements by [Putina et al., 2018]. The DenStream clustering algorithm and its adoption for anomaly detection is the main foundation of this thesis.

2.4.1 DenStream Clustering

DenStream clusters evolving data streams with arbitrary shapes, requires no assumption about the numbers of clusters and can handle anomalies [Cao et al., 2006]. The algorithm summarizes individual samples of the incoming data stream into micro clusters during the online phase. It clusters these micro clusters into macro clusters during the offline phase, which is initiated on user request. Micro clusters represent the summarized samples through their weight w and radius r.

During the online phase, DenStream adds incoming samples to a micro cluster or creates a new one if none exist in the neighborhood of the sample. Micro clusters with weight below a certain threshold are outlier micro clusters (o-micro clusters) of Type I. Otherwise, they are potential core micro clusters (p-micro clusters) [Aggarwal et al., 2004]. The only way to change (i.e. promote) an o-micro cluster to a p-micro cluster is by increasing the weight of an o-micro cluster over the threshold. E.g. some pioneers (i.e. *anomalies*) establish a new settlement on an island which over time grows large enough to be considered *normal*.

During the offline phase, DenStream clusters all p-micro clusters into macro clusters with the DBSCAN algorithm [Ester et al., 1996]. All p-micro clusters that are now part of a macro cluster are core micro clusters (c-micro cluster), hence the micro cluster prefix "potential", the rest are o-micro clusters of Type II. Figure 2.2 depicts the different micro clusters in DenStream.

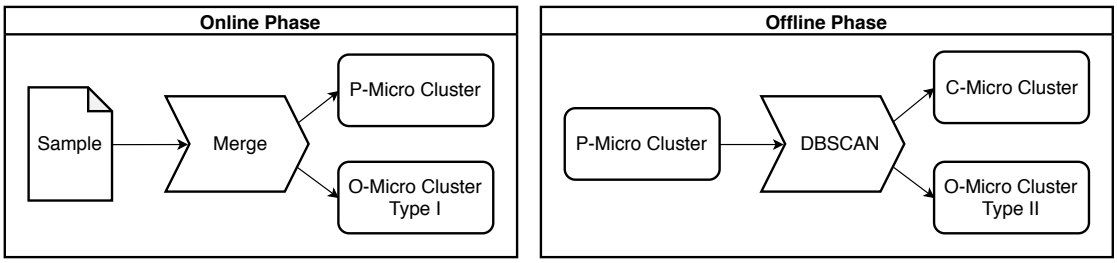

Figure 2.2: Phases in DenStream Clustering [Cao et al., 2006]

Algorithm 2.1 shows the DenStream micro cluster maintenance algorithm by [Cao et al., 2006]. It first tries to add a sample to a p-micro cluster. If it fails, it tries to add the sample to an o-micro cluster of Type I or creates a new one. This may promote the o-micro cluster because it gained enough weight. The algorithm summarizes the samples during the merge process. It updates weight, mean, and variance by adding the new value and reduces the old values with the decay function $2^{-\lambda t}$, further explained in Section 2.4.3.

2.4.2 DenStream Anomaly Detection

In DenStream clustering, anomalies are a by-product, while [Putina et al., 2018] adapt the algorithm to focus on finding anomalies. They treat every sample that is added to an

```
def denstream(ϵ, β, μ, λ):
  for s in data_stream:
    # try merge sample into nearest p−micro cluster c_p
    c_p = get_nearest_p_micro_cluster(s)
    c_p^+ = merge(c_p, s, λ, s_w)
    if radius(c_p^+) ≤ ϵ:
      c_p = c_p^+
    else:
      # try merge sample into nearest o−micro cluster c_o
      c_o = get_nearest_or_new_o_micro_cluster(s)
      c_o^+ = merge(c_o, s, λ, s_w)
      if radius(c_o^+) ≤ ϵ:
        # check if big enough for promotion
        if weight(c_o^+) > β·μ:
          p_micro_clusters.append(c_o^+)
          o_micro_clusters.remove(c_o)
        else:
          c_o = c_o^+
      else:
        # new o−micro cluster
        o_micro_clusters.append(c_o^+)
```

Algorithm 2.1: DenStream micro cluster maintenance [Cao et al., 2006]

o-micro cluster Type I during micro cluster maintenance as contextual anomaly. Algorithm 2.2 shows the extension to capture outliers in real-time. Additionally, they present a way to reduce the cardinality of DenStream by calculating the two hyperparameters core weight threshold μ and maximum radius ϵ automatically, shown in Section 2.4.3.

```
      ...
          c_o = c_o^+
          write_real_time_outlier(s)
      else:
        # new o−micro cluster
        o_micro_clusters.append(c_o^+)
        write_real_time_outlier(s)
```

Algorithm 2.2: DenStream outlier detection by [Putina et al., 2018], extract from Algorithm 2.1

2.4.3 Hyperparameters

The DenStream algorithm *outlier_detection(*$\epsilon, \beta, \mu, \lambda$*)* by [Cao et al., 2006] uses four different hyperparameters. These section explains their characteristics and the proposed simplifications by [Putina et al., 2018].

Decay Factor λ

A micro cluster does not store each sample individually but summarizes them through weight w, mean $\bar{x}$ and variance σ^2. The decay factor controls the influence of past samples. The higher λ, the lower the importance of the past. The Equations 2.1 to 2.3 show the decay applied to the summarizing properties of a micro cluster at timestamp t, with the last sample s added at timestamp t_{last} with weight s_w and value s_x.

$$w_t = w_{t_{last}} \cdot 2^{-\lambda \cdot (t - t_{last})} + s_w \tag{2.1}$$

$$\bar{x}_t = \bar{x}_{t_{last}} + \frac{s_w \cdot (s_x - \bar{x}_{t_{last}})}{w_t} \tag{2.2}$$

$$\sigma_t^2 = \sigma_{t_{last}}^2 \cdot \frac{w_t - s_w}{w_{t_{last}}} + s_w \cdot (s_x - \bar{x}_t) \cdot (s_x - \bar{x}_{t_{last}}) \tag{2.3}$$

Maximum Micro Cluster Radius ϵ

The radius of a core micro cluster must be below or equal to the maximum radius ϵ with $\epsilon > 0$. [Putina et al., 2018] propose to calculate ϵ based on a set of samples taken from the stream before the outlier detection starts and add all samples to the same micro cluster. The largest micro cluster radius during this preprocessing then results in ϵ. They recommend the dynamic threshold selection because of better recall and precision in their experiments. This reduces the DenStream hyperparameter cardinality by one from *outlier_detection(*$\epsilon, \beta, \mu, \lambda$*)* to *outlier_detection(*β, μ, λ*)*.

Core Weight Threshold μ

The weight of a core micro cluster must be at least $w \geq \mu$ with $\mu > 0$ [Cao et al., 2006]. [Putina et al., 2018] proposes to set μ to the maximum possible weight of a micro cluster (i.e. adding all points to the same micro cluster) and only set the outlier tolerance factor β by hand. The maximum possible weight in a fixed-rate sampling stream can be calculated by adding all samples to the same micro clusters and discount it with the decay function $f(t) = 2^{-\lambda t}$. This leads to the geometric series shown in Equation 2.4 with limit μ^+. With $\mu = \mu^+ = \frac{1}{1-2^{-\lambda}}$, the DenStream hyperparameter cardinality reduces by one from *outlier_detection(*$\epsilon, \beta, \mu, \lambda$*)* to *outlier_detection(*ϵ, β, λ*)*.

$$\mu^+ = 2^{-\lambda(t-t)} + 2^{-\lambda(t-(t-1))} + 2^{-\lambda(t-(t-2))} + \cdots \tag{2.4a}$$

$$= 1 + \frac{1}{2^\lambda} + \frac{1}{2^{2\lambda}} + \cdots \tag{2.4b}$$

$$= \sum_{t=0}^{\infty} \left(\frac{1}{2^\lambda}\right)^t \tag{2.4c}$$

$$= \frac{1}{1-2^{-\lambda}} \quad since \quad \frac{1}{2^\lambda} < 1 \quad for \quad \lambda > 0 \tag{2.4d}$$

$$\tag{2.4e}$$

Outlier Tolerance Factor β

The outlier tolerance factor β is the threshold between o-micro clusters Type I relative to c-micro clusters. This reduces the core weight threshold μ and leads to the concept of potential core micro clusters, i.e. $w_{potential} \geq \beta \cdot \mu$ and $w_{outlier} < \beta \cdot \mu$ [Cao et al., 2006]. P-micro cluster may then become core micro clusters or o-micro clusters Type II during DBSCAN clustering.

3

Methodology

This thesis heavily bases on the DenStream algorithm by [Cao et al., 2006] and its modification for outlier detection by [Putina et al., 2018]. We first established a knowledge discovery process to follow a structured approach. Second, we prepared the data in a way such that the algorithm can consume it correctly. Third, we implemented the DenStream outlier detection algorithm. Fourth, we added the drift-influence hyperparameters with the hypothesis of an anomaly detection performance increase.

3.1 Knowledge Discovery Process

[Fayyad et al., 1996] describes the Knowledge Discovery in Databases (KDD) as the act to detect new, useful, and understandable patterns in data. It is a structured approach that guides through the different phases of knowledge discovery.

We organized the knowledge discovery process based on CRISP-DM (CRoss-Industry Standard Process for Data Mining). It builds on the KDD process and implements it according to common industry requirements [Shearer, 2000]. It comprises six phases, shown in Figure 3.1: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment.

Business Understanding defines the business objectives and success criteria used to evaluate the project; Data Understanding delivers the data mining objectives and the data mining success criteria. These two phases differ in a way that the first sets the goals to satisfy the reason why the project was started, while the latter defines ways to reach these goals through data mining. Data Preparation preprocesses data to best fit the used models. In the Modeling phase, machine learning models are built and evaluated by the data mining goals set previously. The last two phases consist of assessing the whole project against the business goals and deploying the solution.

3.2 Data Preparation

The data stream needs preparation for an algorithm to work [Pyle, 1999]. DenStream demands all features to be numerical and have the same scale to work correctly. E.g. calculation of the micro cluster radius is not reasonable if features have differently scaled

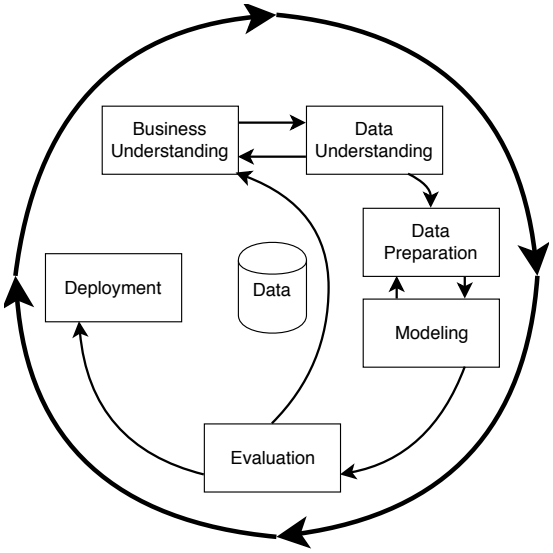

Figure 3.1: CRISP-DM process diagram [Shearer, 2000]

distances. The measurement unit plays a significant role in many machine learning models [Han et al., 2011]. We implemented two different ways to rescale our data, min-max normalization and z-score normalization.

Min-Max Normalization Performs a linear transformation such that all values of a feature lie between the range [0, 1], based on the smallest value x_{min} and biggest value x_{max}:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

Z-Score Normalization Normalizes the values of a feature based on the mean $\bar{x}$ and standard deviation σ_x. The further away from the mean, the bigger the value. This works best with normally distributed data:

$$x' = \frac{x - \bar{x}}{\sigma_x} \tag{3.2}$$

These calculations must be adapted in the streaming environment as we lack knowledge about the global maxima or standard deviation. We propose two different ways to overcome this. First, the use of a representative sample at the beginning of the stream, based on which we calculate the normalization factors. Second, setting the factors with domain knowledge, based on experience or future expectations.

The standard deviation of a data stream may deviate over time (i.e. heteroscedasticity) and may do so differently for each feature. This introduces the problem that the hyperparameter values like the maximum micro cluster radius ϵ must adapt if the underlying data samples become closer together or further apart. We propose to tackle this issue as future work.

3.3 DenStream Implementation

We implemented the DenStream algorithm by [Cao et al., 2006] shown in Algorithm 2.1 with the additions of [Putina et al., 2018] in Algorithm 2.2 and the proposed preprocessing to calculate μ and ϵ automatically to reduce the hyperparameter cardinality. We did not implement DBSCAN clustering on p-micro clusters, and thus all o-micro clusters are of Type I.

We calculate the decay factor λ based on the time needed for a point to lose half of its weight influence, i.e. the half-life time $t_{1/2}$. Equation 3.3 shows the calculation of lambda in the case of 50% influence loss after one minute.

$$2^{-\lambda \cdot t_{1/2}} = f(t) \tag{3.3a}$$
$$2^{-\lambda \cdot 60} = 0.5 \tag{3.3b}$$
$$\lambda = \frac{-log_2(0.5)}{60} \tag{3.3c}$$
$$\lambda \approx 0.0167 \tag{3.3d}$$

Similar to λ, we can rearrange the definition of the outlier tolerance factor β, as shown in Equation 3.4. We calculate it as the minimum size ζ a cluster needs for a promotion if we add all samples consecutively to the same micro cluster.

$$\beta = \zeta * (1 - 2^{-\lambda}) \tag{3.4}$$

3.3.1 Assumptions

We elicited certain assumption on which the DenStream anomaly detection relies on to work properly. They encompass the distribution of the data points in the domain space and the nature of the data stream. These assumptions also apply to the extended algorithm Denstream* presented in Section 3.4.

One macro cluster Only one macro cluster exists in the data stream which may or may not move.

Normal distribution The data is normally distributed around the macro cluster center.

Homoscedasticity The variance of the data stream does not change.

One sample at a time No two samples of the data stream have the same timestamp.

3.4 DenStream*

We added certain modifications and additions to the DenStream anomaly detection explained in Section 2.4.1, which we will discuss in this section.

We record the timestamp when the algorithm promotes an o-micro cluster to a p-micro cluster, shown in Algorithm 3.1. This allows us to evaluate the time needed to form p-micro clusters and further analyze the characteristics of correctly and falsely promoted o-micro clusters.

```
15          ...
16          o_micro_clusters.remove(c_o)
17          # promotion time for all samples
18          for x in get_micro_cluster_samples(c_o):
19            write_promotion_time(x)
20        else:
21          ...
```

Algorithm 3.1: Recording promotion time, extract from Algorithm 2.1

A data set can encompass a drift direction where the mean moves into a certain direction over time. This leads to a different outlier/inlier labeling for two samples with the same value but different creation times. E.g. a man in Switzerland with height 179cm was an outlier in the year 1896[1] but is not in 1996[2] . The value changed from being an outlier to an inlier over time.

We calculate the overall drift $\overrightarrow{\theta_t}$ based on the movement of the center of all p-micro clusters. As DenStream regards recent samples as more important than older ones, we apply the decay factor to the previous overall drift vector and add the vector from the overall p-micro cluster center P_t to the sample S_t, shown in Equation 3.5.

$$\overrightarrow{\theta_t} = 2^{-\lambda \cdot 1} \cdot \overrightarrow{\theta_{t-1}} + \overrightarrow{P_t S_t} \tag{3.5}$$

We introduce two adjustment hyperparameters that change the promotion procedure according to the drift in the data stream, positively or negatively discriminating samples according the cosine similarity Ψ between $\overrightarrow{\theta_t}$ and $\overrightarrow{P_t S_t}$ defined in Equation 3.6. We named the adjusted algorithm *outlier_detection*$(\epsilon, \beta, \mu, \lambda, \delta, \omega)$ DenStream*.

$$\Psi(\overrightarrow{\theta_t}, \overrightarrow{P_t S_t}) = \frac{\overrightarrow{\theta_t} \cdot \overrightarrow{P_t S_t}}{\|\overrightarrow{\theta_t}\| \|\overrightarrow{P_t S_t}\|} \tag{3.6}$$

3.4.1 Drift-Distance-Influence Hyperparameter δ

The drift-distance-influence hyperparameter δ adjusts the area in which samples of a micro cluster can lie. Figure 3.2 depicts the micro cluster areas with and without drift distance influence in a two-dimensional space. Samples that lie in the drift direction can be further away from the micro cluster center than samples that lie in the opposite direction. Sample S1 is a pioneer and can be merged into the micro cluster because we

[1] 1896: median 167.5cm, 95% confidence interval [165.1cm, 169.9cm] [NCD-RisC et al., 2016]
[2] 1996: median 178.4cm, 95% confidence interval [177.5cm, 179.3cm] [NCD-RisC et al., 2016]

adjusted the area. S2 lies in the opposite direction in regards to the drift and can thus not be merged in DenStream*. Nothing changes for S3 (outside) and S4 (inside) in the depicted example. With this approach, we expect to adopt to drift faster and improve the outlier detection.

We implemented this approach in DenStream* by comparing the micro cluster radius to the maximum micro cluster radius ϵ, adjusted by the drift distance influence δ and the cosine similarity Ψ as shown in Algorithm 3.2.

```
10        ...
11        c_o^+ = merge(c_o, s, λ, s_w)
12        if radius(c_o^+) ≤ ϵ * (1 + Ψ · δ):
13           # check if big enough for promotion
14           ...
```

Algorithm 3.2: Drift distance adjustment in DenStream*, extract from Algorithm 2.1

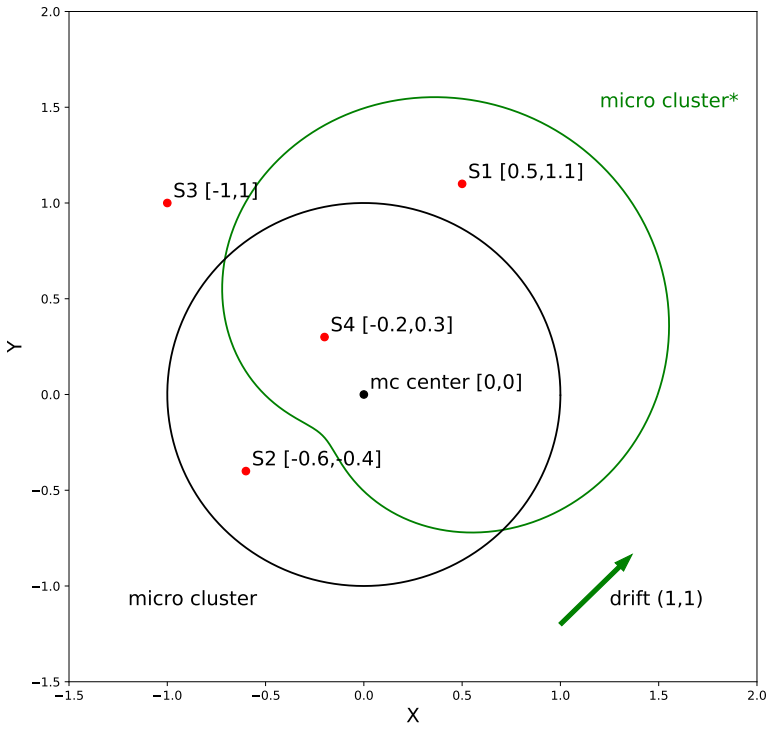

Figure 3.2: Valid micro cluster areas adjusted by δ.

3.4.2 Drift-Weight-Influence Hyperparameter ω

The drift-weight-influence hyperparameter adjusts the weight a sample adds to a micro cluster. If the sample lies in the drift direction, it adds more weight and vice-versa. Figure 3.3 depicts the sample weight by marker size. The weight of S1 increases because the vector from the overall center to the sample is similar to the drift direction. S2 loses weight because it lies in the opposite direction. The weight of S3 does not change as its vector is orthogonal to the drift. With this approach, pioneers add more weight to a micro cluster and should thus promote an o-micro cluster faster than samples that do not lie in the drift direction.

We implemented this approach in DenStream* by increasing or decreasing the sample weight s_w with the drift weight influence ω and the cosine similarity Ψ as shown in Algorithm 3.3.

```
9        ...
10       c_o = get_nearest_or_new_o_micro_cluster(s)
11       c_o^+ = merge(c_o, s, λ, s_w · (1 + Ψ · ω))
12       if radius(c_o^+) ≤ ε:
13          ...
```

Algorithm 3.3: Drift weight adjustment in DenStream*, extract from Algorithm 2.1

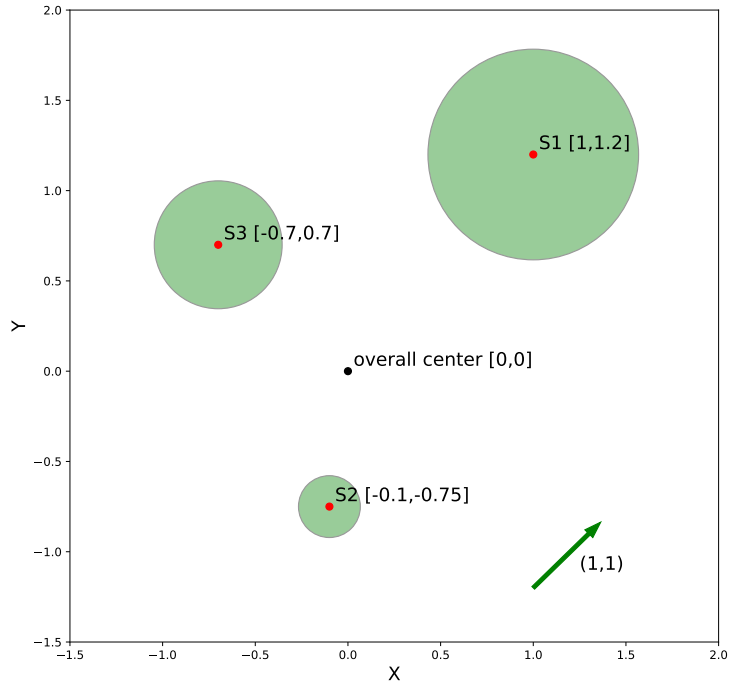

Figure 3.3: Weight of samples S1, S2, S3 adjusted by ω, depicted through blob size

3.4.3 Outlier Score α

[Putina et al., 2018] classify samples as inliers or outliers depending on whether they are part of a p-micro cluster or o-micro cluster. We extend this binary classification with the outlier score α. The euclidean distance from the outlier-sample S_o to the center of the closest p-micro cluster MC_{p_close} is a metric to understand how far away an outlier lies from all the inliers. We normalize this distance with the maximum radius ϵ to make the outlier scores comparable over multiple data sets with different density:

$$\alpha = \frac{\|distance(center(MC_{p\_close}), S_o)\|}{\epsilon} \tag{3.7}$$

We implemented the outlier score by calculating the distance to the next p-micro cluster whenever a sample is labeled anomalous, as shown in Algorithm 3.4.

```
16          ...
17        else:
18          c_o = c_o^+
19          # calculate outlier score
20          α = outlier_score(s, p_micro_clusters)
21          write_real_time_outlier(s, α)
22      else:
23        # new o-micro cluster
24        o_micro_clusters.append(c_o^+)
25        # calculate outlier score
26        α = outlier_score(s, p_micro_clusters)
27        write_real_time_outlier(s, α)
```

Algorithm 3.4: Outlier score α instead of binary labeling, extract from Algorithm 2.1

4

Evaluation

We conducted all the experiments on Ubuntu 18.02 on a Notebook with Intel Core(TM) i5-5200U CPU running at 2.20GHz and 8GB of RAM. The computation time does not affect the results as we treated all samples as evenly spaced by a constant time interval, independent from the real-time passed. We used three different data sets to perform our experiments. We optimized the DenStream hyperparameters and then evaluated the influence of the drift-influence hyperparameters (i.e. drift-distance-influence δ and drift-weight-influence ω). We used the PR-AUC described in Section 2.3.2 as a performance measure. We also analyzed the time needed to promote an o-micro cluster to a p-micro cluster.

4.1 Data

To our knowledge, there exists no benchmark data set to test multivariate time series outlier detection methods. We regard this as a result of the general lack of labeled outlier data sets, described in Section 2.2.2. Thus, we created data sets where we injected outliers. We built three synthetic data sets with the same underlying random distribution, but the mean of each data set follows a different drift-path. The data sets have two features x and y.

Static $x, y = 0$ The data contains no drift, i.e. the mean stays at the same point.

Line $y = x$ The mean follows a straight line, i.e. the drift vector is static.

Sinus $y = sin(x)$ The mean follows a sinus curve.

We created $n = 10\,000$ synthetic samples in each data set and increased x by 0.004 with $-4 \leq x \leq 4$ every $q = 5$ steps in the *line* and *sinus* data set. We calculated the output y based on the drift function and a given x. We added noise to every sample such that the noisy inliers can lie everywhere in a circle within a maximum radius r around the noiseless sample. We inject outliers as values that lie uniformly distributed as a ring outside r and inside $2 \cdot r$.

Algorithm 4.1 shows the synthetic data set generation. We draw the distance d and angle η from the uniform random distributions $\mathcal{U}$, i.e. $d \sim \mathcal{U}(0, r)$ and $\eta \sim \mathcal{U}(0, 2\pi)$.

```
for x in range(n/q):
  for _ in range(q):
    η = random(0, 2π)
    d = random(0, 1)
    if random(0, 1) < outlier_propability:
      # inject outlier
      r* = r · (1 + d)
    else:
      r* = r · d
    x* = x + cos(η) · r*
    y* = y(x) + sin(η) · r*
```

Algorithm 4.1: Synthetic data set generation

We then inject an outlier with a defined probability. At the end we calculate the noisy sample coordinates x* and y*.

We ran the experiments with a random seed to ensure consistent data over multiple runs. Figure 4.1 depicts the three synthetic data sets (with only $n = 600$ for better visualization). For the line and sinus data set, we see some points labeled as anomalies that lie in between inliers. This is correct as these points were anomalous at creation time and the inliers were created at a later point in time in the same area because of drift.

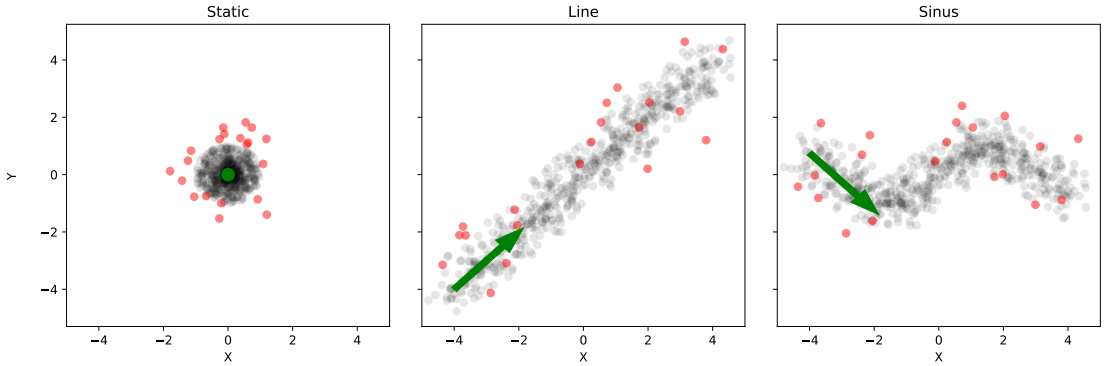

Figure 4.1: Synthetic data sets used in experiments - anomalies in red, drift at $x = -4$ in green

4.2 Experiments

For each experiment, we collected the used hyperparameters, rescaling factors, and metadata for repeatability and reproducibility. These values comprise the following: Time

of execution, name of used data set, decay factor λ, tolerance factor β, maximum micro cluster radius ϵ, drift-distance-influence δ, drift-weight-influence ω, the feature scaling method (none, min-max normalization, z-score normalization) and the feature scaling parameters (minimum, maximum, mean, standard deviation).

The algorithm writes every sample that is added to an o-micro cluster to the database. It adds a promotion time to a sample if its o-micro cluster is promoted to a p-micro cluster. An entry in the database comprises the rescaled features, the cosine-similarity to the drift-center, the insertion time, the promotion time (if applicable), and the outlier score.

We used the precision-recall AUC (PR-AUC) described in Section 2.3.2 as the performance measure to compare the outcome of different run configurations. We first optimized the DenStream hyperparameters decay factor λ, tolerance factor β, and maximum micro cluster radius ϵ via grid search. We further classify the anomalies into real-time and persistent anomalies, as the interpretation of an anomaly is dependent on the point of view.

Real time anomalies Every sample that the algorithm adds to an o-micro cluster is an anomaly at this point of time.

Persistent anomalies Persistent anomalies are real time anomalies that are never part of a p-micro cluster, i.e. their corresponding o-micro cluster never gets promoted.

The real-time anomalies are a superset of persistent ones, as shown in Equation 4.1. Thus we end up with two sets of optimal hyperparameters for each data set, one optimized where we compare the ground truth with every element that the algorithm adds to an o-micro cluster (i.e. real-time outliers) and one optimized where we compare the ground truth with every element that the algorithm did not promote to an o-micro cluster (i.e. persistent outliers).

$$AllSamples \supseteq RealtimeAnomalies \supseteq PersistentAnomalies \tag{4.1}$$

Thus we compare the ground truth with both interpretations of an anomaly to take both points of view into consideration.

4.2.1 DenStream Hyperparameter Optimization

Figure 4.2 depicts the results for the grid search on the sinus data set with real-time outliers. The hyperparameters show a correlation between each other in regards to the PR-AUC. The best hyperparameter value for ζ with some specific ϵ and $t_{1/2}$ is not necessarily the best for other ϵ and $t_{1/2}$ in regards to PR-AUC. This complicates the process of finding optimal hyperparameters as we have to optimize multiple axes simultaneously.

The proposed calculation of the maximum cluster radius by [Putina et al., 2018] as described in Section 2.4.3 returned $\epsilon \approx 0.67$ for all three data sets. This value for ϵ is

very far away from all well-performing hyperparameter-triples we found through grid-search. Therefore we did not pursue this approach any further. Instead, we included ϵ in the grid-search and set it explicitly.

We also analyzed the sensitivity of each hyperparameter to understand the influence of each better. Therefore we set two hyperparameters to their top overall optimal values and vary the third. Figure 4.3 shows this sensitivity analysis for the top three PR-AUC scores on the sinus data set with real-time outliers, e.g. the sensitivity of ζ if we set $[t_{1/2}, \epsilon]$ to [28, 0.40], [34, 0.40] and [36, 0.42] in the leftmost graph.

For reasons of clarity and comprehensibility, the complete list of visualizations for all data sets, and both real-time and persistent outliers can be found in Section A.1.

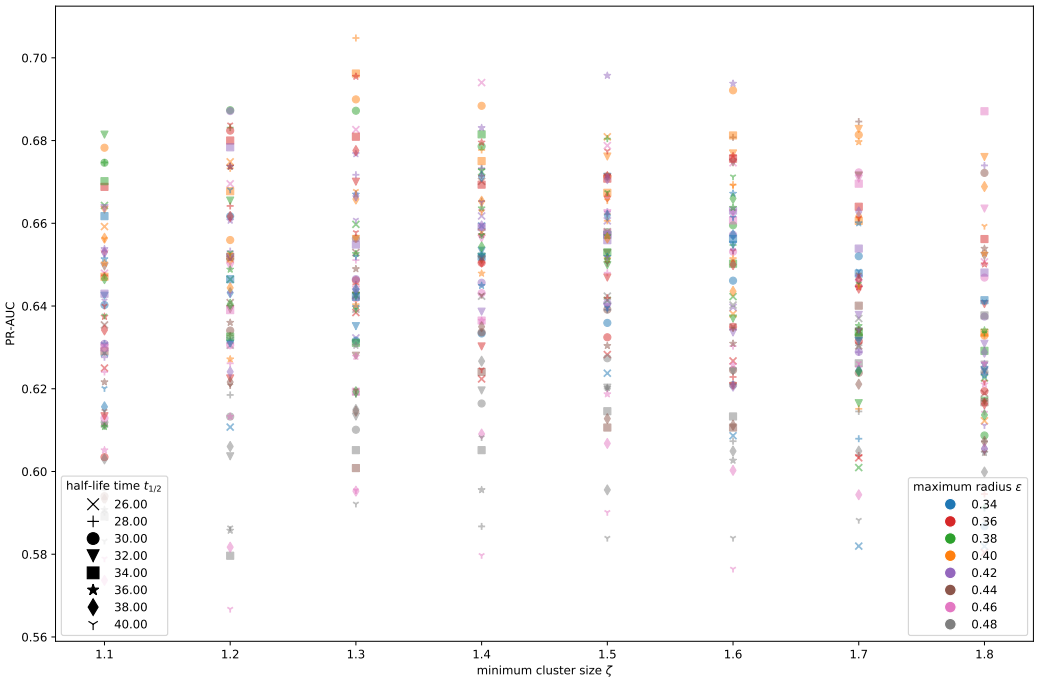

Figure 4.2: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on sinus data set with real-time outliers

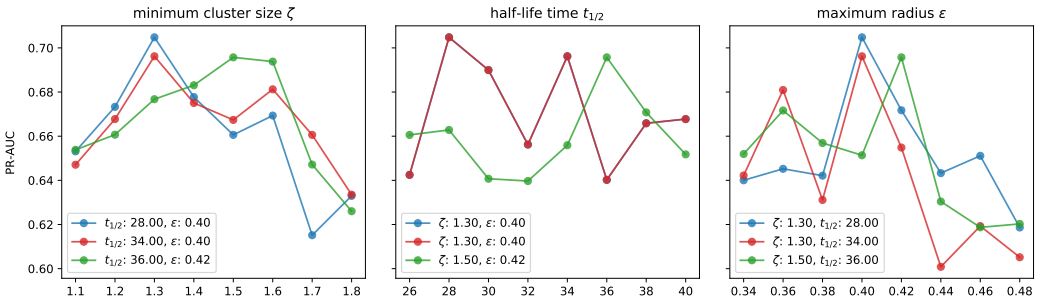

Figure 4.3: Sensitivity of ϵ, ζ and $t_{1/2}$ on sinus data set with real-time outliers

4.2.2 Experiment 1 - DenStream* Performance Influence

Hypothesis 1 *The drift-influence hyperparameters δ and ω improve the PR-AUC of DenStream*.*

We analyzed the influence of distance-influence δ and weight-influence ω on the PR-AUC of DenStream* during this experiment. We used the optimal values for the hyperparameters λ, β and ϵ, i.e. the configurations which resulted in the highest PR-AUC in the hyperparameter analysis described in Section 4.2.1. Figure 4.4 shows the grid search for δ and ω on the sinus data set with real-time outliers.

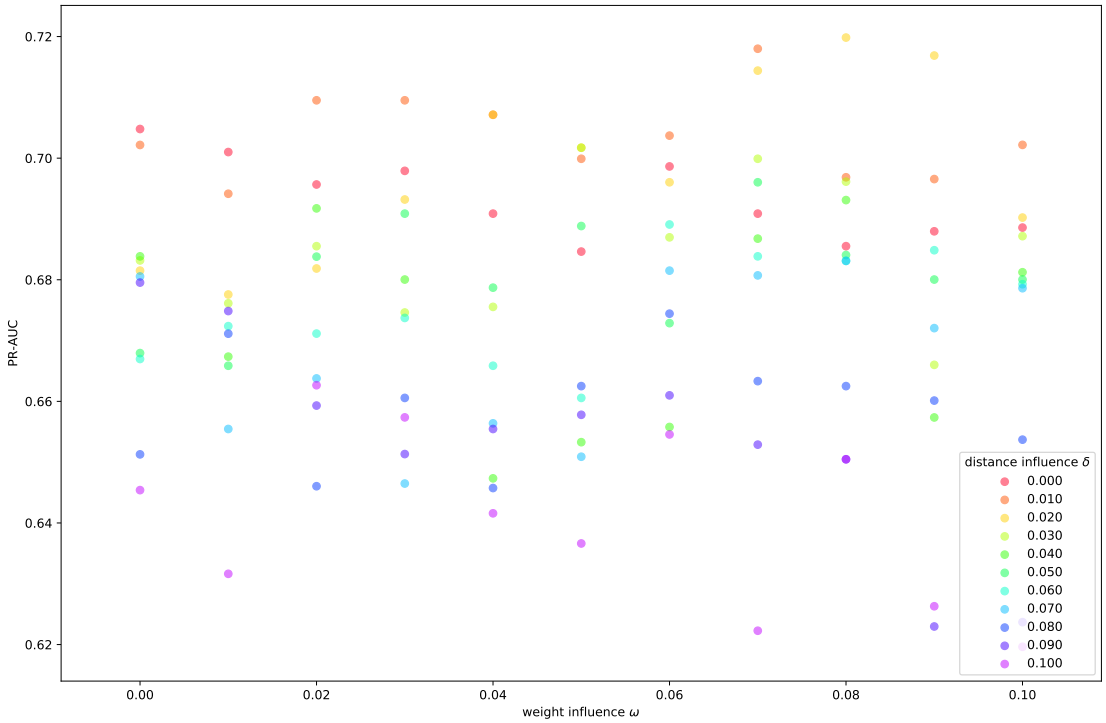

Figure 4.4: Grid search on δ and ω on sinus data set with real-time outliers

Figures 4.5 to 4.7 show the performance influence of δ and ω respectively, keeping all DenStream hyperparameters and one drift-influence hyperparameter unchanged. We see that δ tends to reduce the PR-AUC faster than ω. The complete set of grid search figures on the drift-influence hyperparameters can be found in Section A.2.

In the static data set, both influence factors only reduce performance. This can be expected, as the underlying assumption of drift is not fulfilled, i.e. the data is static. We can see an improvement for the linear drift and the sinus curve path, the most significant for the sinus data set with real-time outliers.

We broke down the test results with the optimized drift-influence hyperparameters to highlight cases where the two algorithms decided differently or equally about classifying a sample as an anomaly. Tables 4.1 to 4.3 show the breakdown for the three data sets.

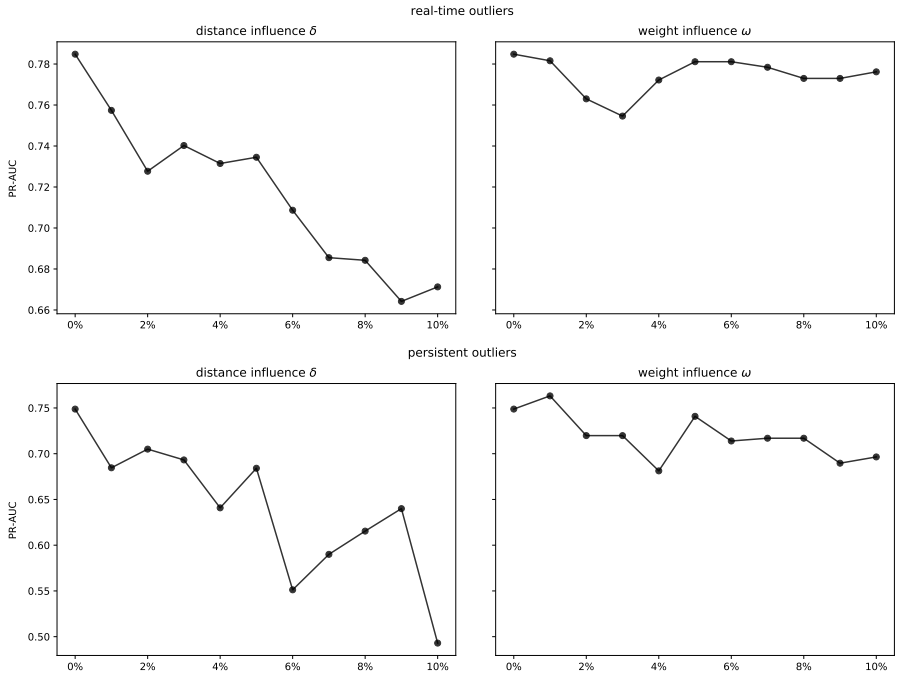

Figure 4.5: Separate influence of δ and ω on static data set

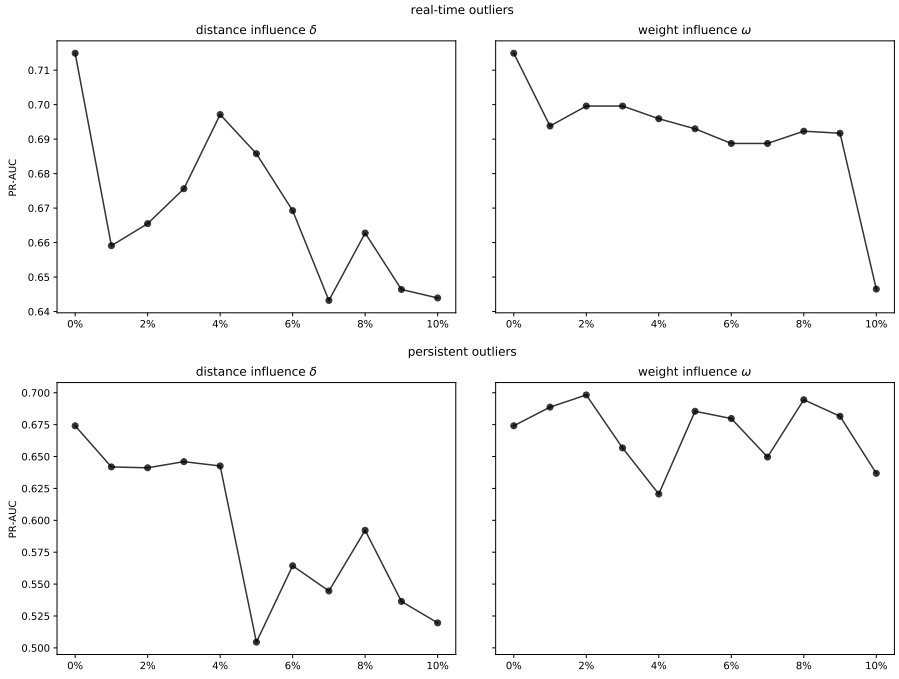

Figure 4.6: Separate influence of δ and ω on line data set

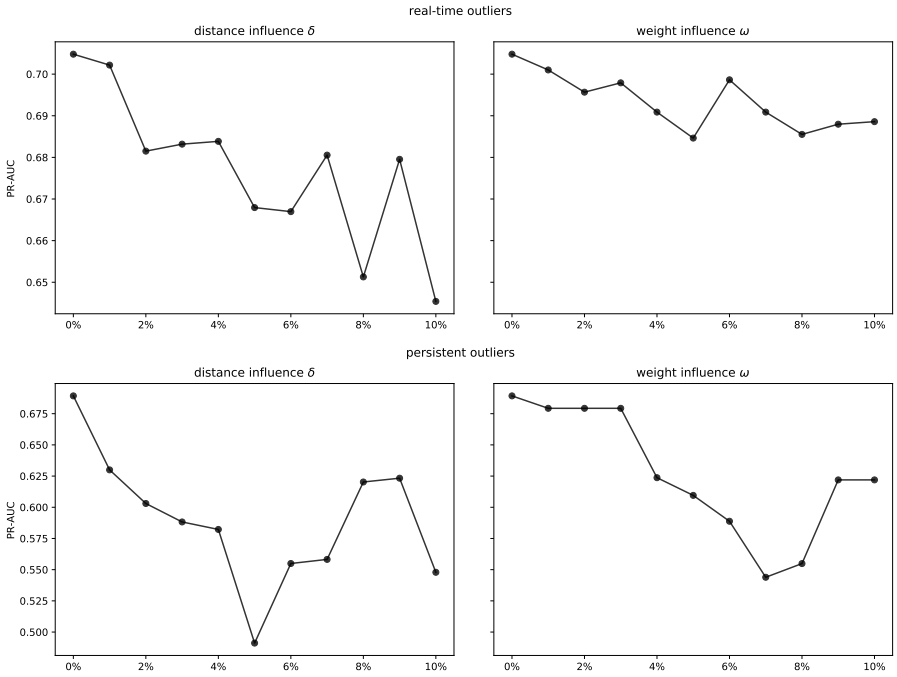

Figure 4.7: Separate influence of δ and ω on sinus data set

Overall, the confusion matrix shows a highly unbalanced data set. For that reason, the PR-AUC is more expressive than the ROC-AUC performance measurement, as explained in Section 2.3.2. DenStream* improves the PR-AUC for persistent outliers in the line data set primarily through a lower rate of wrongly labeled outliers (i.e. false negatives), while the score for real-time outliers in the sinus data set improves because of fewer wrongly labeled inliers (i.e.false positives).

The hypothesis holds for specific data sets. If there is drift in the data set, δ and ω can lead to a higher PR-AUC for the right hyperparameter values. If there is no drift, the additional hyperparameters do not improve the PR-AUC.

DenStream					DenStream*				
	Real-Time		Persistent			Real-Time		Persistent	
	Inlier	Outlier	Inlier	Outlier		Inlier	Outlier	Inlier	Outlier
Inlier	9790	11	9778	23	Inlier	9790	11	9778	23
Outlier	65	134	60	139	Outlier	65	134	60	139

Table 4.1: Static dataset: *Denstream* and *Denstream** confusion matrices

DenStream					DenStream*				
	Real-Time		Persistent			Real-Time		Persistent	
	Inlier	Outlier	Inlier	Outlier		Inlier	Outlier	Inlier	Outlier
Inlier	9791	10	9792	9	Inlier	9791	10	9793	8
Outlier	92	107	108	91	Outlier	92	107	101	98

Table 4.2: Line dataset: *Denstream* and *Denstream** confusion matrices

DenStream					DenStream*				
	Real-Time		Persistent			Real-Time		Persistent	
	Inlier	Outlier	Inlier	Outlier		Inlier	Outlier	Inlier	Outlier
Inlier	9763	38	9800	1	Inlier	9777	24	9800	1
Outlier	67	132	118	81	Outlier	74	125	118	81

Table 4.3: Sinus dataset: *Denstream* and *Denstream** confusion matrices

4.2.3 Experiment 2 - DenStream* Promotion Time Influence

Hypothesis 2 *The drift-influence hyperparameters δ and ω reduce the time passed between the creation of an o-micro cluster and its promotion to a p-micro cluster.*

We analyzed the time needed to promote an o-micro cluster to a p-micro cluster. We expected that the drift-aware DenStream* would promote wrongly labeled real-time outliers (i.e. false positive) faster. We compared DenStream and DenStream* in regard to the time needed for samples to be promoted, i.e. the promotion time. We used the optimal values for δ and ω found through grid-search, as discussed in Section 4.2.2. Tables 4.4 to 4.6 show the samples median of the promotion time, split into correctly (✓) and wrongly (✗) promoted anomalies.

	Real-time			Persistent		
	✓	✗	$\sum$	✓	✗	$\sum$
DS	781	11.5	735	682.5	65.5	257.5
*DS**	781	11.5	735	118.5	40	112.5

Table 4.4: Median promotion time comparison for static data set

The hypothesis holds for the cases where δ and ω improve the performance, i.e. if there is drift in the data set, as explained in Section 4.2.2. This is in accord with the general observation that lower overall promotion time is an indicator for better performance. We can observe this effect on all data sets, as shown in Figure 4.8.

	Real-time			Persistent		
	✓	✗	$\sum$	✓	✗	$\sum$
DS	1047	7.5	840	1335	148	479
*DS**	1047	7.5	840	521.5	14	287.5

Table 4.5: Median promotion speed comparison for line data set

	Real-time			Persistent		
	✓	✗	$\sum$	✓	✗	$\sum$
DS	627	279	580.5	510.5	24	187
*DS**	858	90	717.5	510.5	24	187

Table 4.6: Median promotion speed comparison for sinus data set

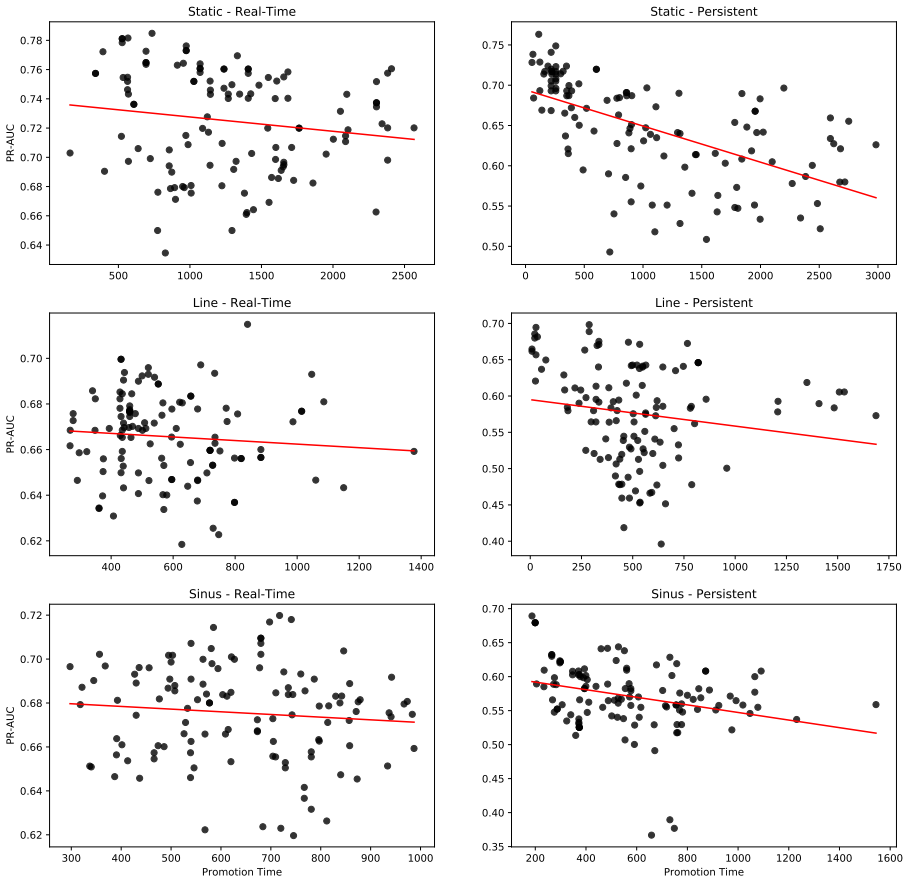

Figure 4.8: Median promotion time vs. PR-AUC

4.3 Discussion

The use of PR-AUC showed the performance difference in the different algorithms settings adequately, even in the highly biased data sets with an inlier-to-outlier ratio of around 2%. Still, the inherent anomaly-property *rareness* leads to the conclusion that incline and decline in performance often lie in the area of statistical uncertainty because the absolute number of outliers remains low even for bigger data sets.

The DenStream hyperparameter optimization showed that the decay factor λ, tolerance factor β and maximum micro cluster radius ϵ can not be optimized separately as their PR-AUC influence is not independent of the other hyperparameters. The proposed calculation of ϵ by [Putina et al., 2018] through a test set resulted in much lower performance scores, and therefore, we did not pursue it further. The reduction of the hyperparameter space because of the automatic calculation of μ, as explained in Section 2.4.3 was invaluable. It reduced grid-search time significantly because we had to explore only three dimensions instead of four.

We introduced the distinction between real-time and persistent outliers. A data analyst must decide to optimize for one or the other. It has a direct influence on the optimal hyperparameters for the algorithm. It results in a trade-off between knowing what is anomalous at the moment vs. what is anomalous in the longer run.

The drift-influence hyperparameters δ and ω in DenStream* showed the potential to increase the performance of DenStream. They indicate similar behavior like the conventional DenStream hyperparameters λ, β and ϵ: They should not be analyzed separately as their PR-AUC influence is not independent and the performance is sensitive to small changes of the hyperparameters.

We achieved the best performance increase on the sinus-path drift data set through the modification of both δ and ω. In general, the drift-distance-influence hyperparameter δ reduces the performance score significantly in most of the possible hyperparameter space. The drift-weight-influence ω shows more stable behavior than drift-distance-influence δ. We thus propose to use δ only with a reliable evaluation base as the risk for lowering the performance score is otherwise higher than the potential gain.

We found a correlation between promotion time and performance score, especially if we optimize the hyperparameters to find persistent outliers. Better hyperparameters thus not only improve the probability of a correct inlier/outlier labeling. If the algorithm promotes a sample where the underlying ground truth labels it as an inlier, an analyst can come to the correct conclusion faster (i.e. sample is an inlier). The algorithm needs to keep all samples of o-micro clusters in memory; thus, an improvement in promotion time also allows for better scaling.

Although the hyperparameters show satisfying performance in a reasonably big parameter space, we see the need for thorough empirical evaluation before using the algorithm on a new data set. A data analyst must apply background knowledge to decide what kind of anomalies are interesting and what influence the time component (i.e. decay factor λ) should have. The described correlation between promotion time to PR-AUC can further act as a guideline.

5

DenStream in Real Scenarios

We applied the DenStream algorithm in two scenarios. The different settings tested the applicability of the algorithm with real-world data. Both data sets did not provide ground truth. Thus, the findings act as a guideline for further research on the data and can highlight areas where we encourage investigation. We were able to apply the learned heuristics and provide input for the two ongoing projects. We executed the unmodified DenStream algorithm to establish a more stable baseline for the project partners and because the impact of the drift-influence hyperparameters could not be evaluated without ground truth.

5.1 PigData

PigData is a collaborative research project, funded by the Swiss National Science Foundation National Research Program 75 "Big Data"[1]. It includes several partners: the Dynamic and Distributed Information Systems Group of the University of Zurich, the Vetsuisse Faculty of the University of Berne and the Swiss Federal Institute for Forest, Snow and Landscape Research (WSL). It applies Big Data methods to research the Swiss swine and pork production industry [Berezowski et al., 2016]. Multiple industry partners deliver data sets regularly with the plan of continuous data delivery. The data contains data about veterinary visits, reproduction, slaughtering process, and meat quality. This thesis does not include visualizations or exempts of this project for reasons of confidentiality.

The provided data is a collection of multiple data sets that contain data over a period of four to seven years. Many timestamps did not contain data values. They originated from the case when no event happened, so they can be imputed with zero. The treatment of missing values can have a significant impact on the DenStream performance as micro clusters lose importance over time. For sequential data where the time between measurements is irrelevant, the algorithm would need adjustment to treat all intervals as equal. We executed multiple anomaly detection runs with different data grouping. We grouped based on the data sources (e.g. all data from the slaughterhouse), point in the life of a pig (e.g. birth and death) and all data as one data set. We continuously fed the time sample into the DenStream algorithms to simulate the data stream.

[1] http://www.nfp75.ch/

We tuned the hyperparameters based on background knowledge about some known anomalies during the measuring time. We found multiple possible anomalous samples and readjusted the analysis after we presented the results to the project partners. We faced issues concerning the hyperparameter tuning because of only limited to no ground truth. The evaluation and discussion of the found anomalies with the industry partners happened after we finished this thesis and can thus not be included.

As future work, it would be essential to study how to make the algorithm less sensitive to wrongly chosen hyperparameter values. We see the possibility of a hybrid anomaly detection approach for the presented use case: First, execute a non-streaming outlier detection method on the available static data set as they do not have the drawbacks of a streaming environment, and then use the gained knowledge to tune the DenStream algorithm and run it continuously.

5.2 IPTV

The data originates from one of the biggest TV-streaming providers in Europe. The data set comprises TV sessions viewed by users over multiple days. Each session contains a timestamp and metadata about the user type, duration, device, and location [Philipp, 2015]. We needed to rescale all values to establish comparability between the features, as explained in Section 3.2. We reshaped the sessions into intervals of five minutes and aggregated them according to the used device. Table 5.1 shows an exempt from the aggregated data set. We then fed the aggregated interval results into the DenStream algorithm to simulate a continuous data stream.

timestamp	bigscreen	desktop	mobile	tablet
t_0	∼5700	∼4700	∼1800	∼1300
t_1	∼5800	∼4500	∼1800	∼1300
t_2	∼5800	∼4400	∼1900	∼1400
t_3	∼5900	∼4400	∼1800	∼1400

Table 5.1: Exempt from IPTV data set aggregated on devices with $\Delta t = 5min$

We tuned the hyperparameters heuristically and via grid search. We searched for hyperparameter values in which small changes did result in only small changes in the found outliers. Figure 5.1 shows only real-time outliers during the three analyzed days. They appear to be connected with the day-change at midnight which terminates open sessions. This indicates that the anomalies may be connected to the way the IPTV provider collects the data. The outlier at 04:00 of the first day originates from the cold-start effect as the very first samples are always real-time outliers because they must start a new o-micro cluster.

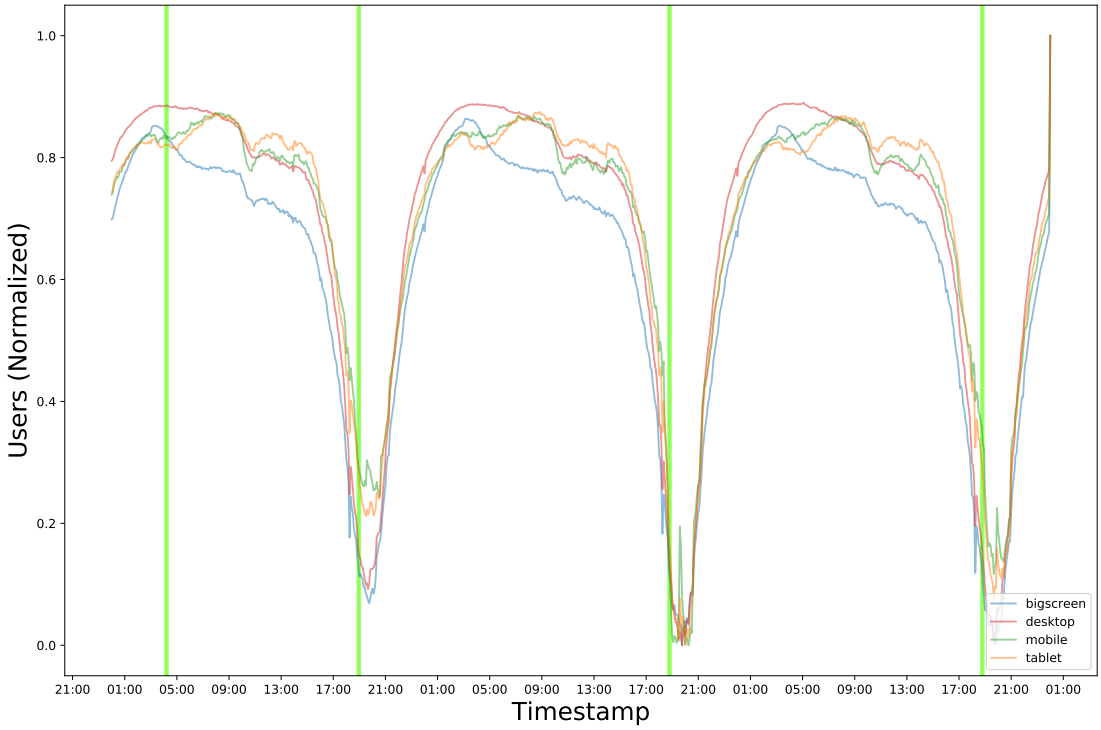

Figure 5.1: DenStream outlier detection on IPTV data set

6

Limitations and Future Work

The drift-influence calculation assumes only one (macro) cluster in the data stream at a time. Multiple clusters moving in arbitrary directions can not be represented and may lead to wrong application of the drift-influence hyperparameters. The discussed rescaling methods require homoscedasticity in the data stream, i.e. the variance of the features stays the same. Otherwise, some features gain more influence than others.

We set the sample size to $n = 10\,000$ during the evaluation to keep the duration of grid search manageable. The *rareness* characteristic of anomalies results in only a small number of samples that can be labeled *outlier* correctly (i.e. true-positive) so that part of the changes in performance score can be attributed to natural variance.

We executed all experiments on synthetic data sets that we generated ourselves. Thus the performance increase of DenStream* over DenStream anomaly detection may not generally be applicable on other synthetic data or real-world data.

During the master's thesis, we discovered multiple areas for future work that may overcome some of these limitations or extend on the topic:

Improve algorithm robustness [Putina et al., 2018] proposed ways to automatically calculate and adjust DenStream hyperparameters to make the algorithm more robust. We still see potential research in this area, especially about the automatic calculation of ϵ through clustering of a training set instead of adding all samples to the same micro cluster. We propose to automatically detect drift and adjust the hyperparameters δ and ω. Dynamically adapting hyperparameters could also tackle the issue of heteroscedasticity, e.g. continuously adjust ϵ based on the mean and variance of cluster radii.

DBSCAN The drift-influence hyperparameters δ and ω currently assume only one cluster that drifts through the variable space. We propose to apply DBSCAN in DenStream* and apply the drift-influence hyperparameters on incoming samples based on the closest macro cluster. This also allows to evaluate samples of o-micro cluster Type II and analyze the influence of δ and ω on this type of anomaly.

Finding root cause of an anomaly The features of a multivariate data set contribute to the outlier score in varying degree. The interest of a data analyst often lies in finding the justification of an anomaly. We propose further research in finding

or improving models for the streaming environment. We see sensitivity analysis, point-wise variance for every feature or combining every feature with each other to determine which combinations create an outlier (e.g. $M = \{a, b, c\} \rightarrow [a, b, c, ab, ac, bc, abc]$) as possible directions.

Missing Values The evaluated scenarios in Chapter 5 highlighted the difficulty of missing values. While our use case allowed to impute missing values with zero because of background knowledge, we propose research on other methods, e.g. rolling mean, interpolation or regression, and to apply them to the DenStream anomaly detection algorithm.

7

Conclusions

The goal of this thesis was to improve the anomaly detection in multivariate data streams by incorporating drift to label pioneers as inliers and apply a state of the art solution to real data.

The related work in Chapter 2 explores the challenges and possible solutions for anomaly detection in a data stream. We managed to collect previous work on the less explored variate of stream processing, mainly about processing multivariate data streams and how to evaluate them.

We implemented DenStream proposed by [Cao et al., 2006] to find anomalies in an evolving multivariate data stream with arbitrary shape in Chapter 4. The algorithm summarizes samples in a data stream with micro clusters and adopts them to changes over time. We introduced two additional hyperparameters δ and ω that influence micro cluster radius and weight to anticipate drift in the data stream. We introduced an outlier score α to compare the degree of abnormality of an outlier. Additionally, we presented ways to rearrange the decay factor λ, and the outlier tolerance factor β to make them easier to understand heuristically.

We performed multiple experiments on synthetic data sets. First, we presented a hyperparameter analysis of the DenStream algorithm and evaluated their sensitivity and influence. Second, we showed that the introduced drift-influence hyperparameters δ and ω of DenStream* can improve the anomaly detection performance but add additional statistical uncertainty to the result. We analyzed the two hyperparameter cardinality reductions of DenStream proposed by [Putina et al., 2018]: Our evaluation supports the first proposed reduction of setting the core weight threshold μ equal to the maximum possible micro cluster weight μ^+. We discarded the second proposed reduction of calculating the maximum cluster radius ϵ through test sampling because of the considerable performance reduction. We found a slight correlation between the promotion time of outliers and the performance score, which helps to tune hyperparameters when no ground truth is available.

The two scenarios described in Chapter 5 showed the application of the DenStream algorithm on real-world use cases. The results give insight into the data structure and guide further research. We present a foundation to base the future analysis on, even though the difficulty of missing ground truth persists.

With the current trend of more and more devices continuously producing data, data streaming research will further increase in importance. The ability to automatically

detect anomalies and to act accordingly is of high importance. We propose further research in the area in general, especially in improving the robustness of the process to achieve satisfying performance continually, even without ground truth.

References

[Aggarwal, 2007] Aggarwal, C. C. (2007). *Data streams: models and algorithms*, volume 31, pages 1–7. Springer Science & Business Media.

[Aggarwal et al., 2004] Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863. VLDB Endowment.

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

[Amini and Wah, 2010] Amini, A. and Wah, T. Y. (2010). Density micro-clustering algorithms on data streams: A review. In *World Congress on Engineering 2012. July 4-6, 2012. London, UK.*, volume 2188, pages 410–414. International Association of Engineers.

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

[Assent et al., 2012] Assent, I., Kranen, P., Baldauf, C., and Seidl, T. (2012). Anyout: Anytime outlier detection on streaming data. In *International Conference on Database Systems for Advanced Applications*, pages 228–242. Springer.

[Berezowski et al., 2016] Berezowski, J., Bernstein, A., Grütter, R., Nathues, C., and Nathues, H. (2016). Full Proposal: Research Plan NRP 75 "Big Data".

[Cao et al., 2006] 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. SIAM.

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

[Davis and Goadrich, 2006] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

[Domingos and Hulten, 2000] Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Kdd*, volume 2, page 4.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

[Fayyad et al., 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

[Goix, 2016] Goix, N. (2016). How to evaluate the quality of unsupervised anomaly detection algorithms? *arXiv preprint arXiv:1607.01152*.

[Goldstein and Uchida, 2016] Goldstein, M. and Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173.

[Han and Ding, 2009] Han, J. and Ding, B. (2009). *Stream Mining*, pages 2831–2834. Springer US.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[Khan and Fan, 2012] Khan, L. and Fan, W. (2012). Tutorial: data stream mining and its applications. In *International Conference on Database Systems for Advanced Applications*, pages 328–329. Springer.

[Kontaki et al., 2011] 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. IEEE.

[Mousavi et al., 2015] Mousavi, M., Bakar, A. A., and Vakilian, M. (2015). Data stream clustering algorithms: A review. *Int J Adv Soft Comput Appl*, 7(3):13.

[NCD-RisC et al., 2016] NCD-RisC et al. (2016). A century of trends in adult human height. *Elife*, 5:e13410.

[Philipp, 2015] Philipp, B. (2015). A flexible viewership analytics system for online tv. Master's thesis.

[Putina et al., 2018] Putina, A., Rossi, D., Bifet, A., Barth, S., Pletcher, D., Precup, C., and Nivaggioli, P. (2018). Telemetry-based stream-learning of bgp anomalies. In *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pages 15–20. ACM.

[Pyle, 1999] Pyle, D. (1999). *Data preparation for data mining*. morgan kaufmann.

[Rand, 1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

[Salgado et al., 2016] Salgado, C. M., Azevedo, C., Proença, H., and Vieira, S. M. (2016). Noise versus outliers. In *Secondary Analysis of Electronic Health Records*, pages 163–183. Springer.

[Shearer, 2000] Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22.

A

Evaluation Results

This appendix provides the complete set of plots for the conducted experiments. We first present the hyperparameter analysis on ϵ, ζ and $t_{1/2}$. The second section contains the evaluation of the DenStream* drift-influence hyperparameters δ and ω.

A.1 DenStream Hyperparameter Analysis ($\epsilon, \zeta, t_{1/2}$)

For each data set *static, line, sinus* we performed a grid search on the three hyperparameters ϵ, ζ and $t_{1/2}$. We then compared the ground truth once with the real-time anomalies and once with the persistent anomalies and calculated the PR-AUC. Figures A.1 to A.6 show the results in one plot, Figures A.7 to A.12 use the set of best hyperparameters and only vary one hyperparameter.

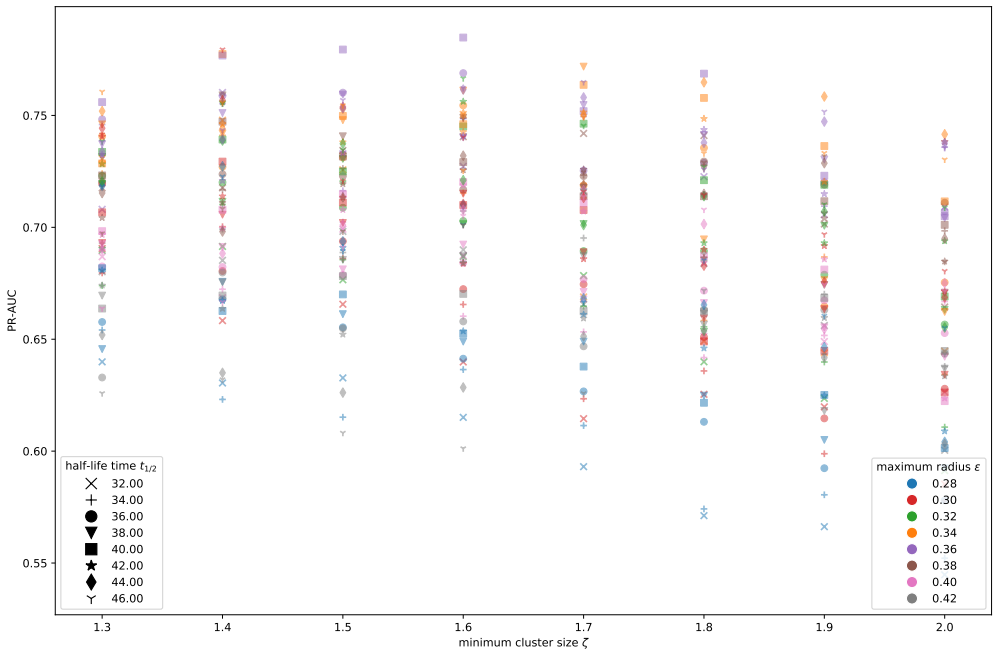

Figure A.1: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on static data set with real-time outliers

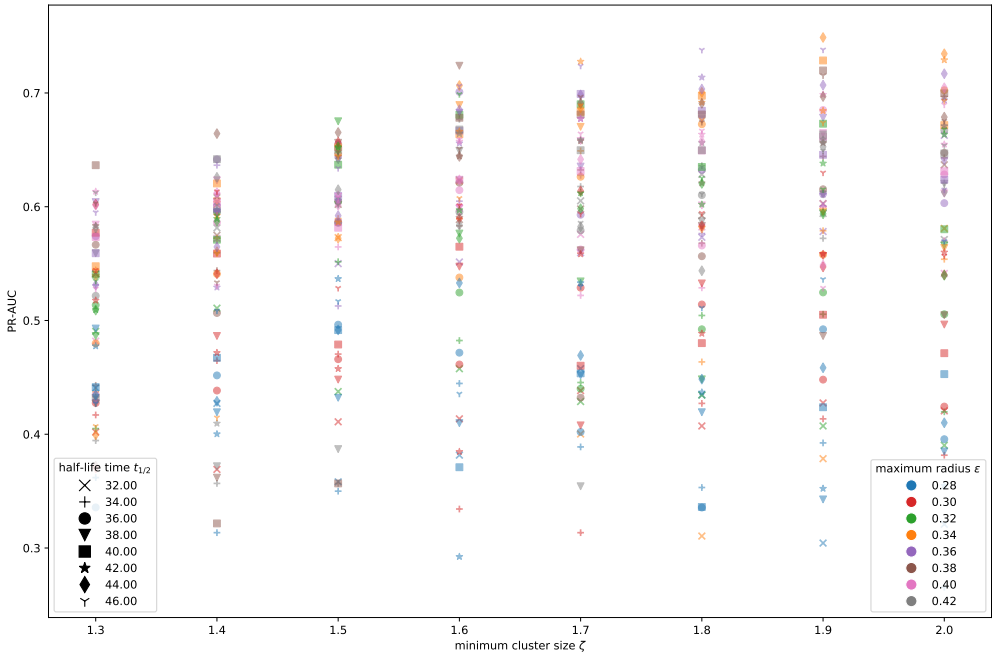

Figure A.2: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on static data set with persistent outliers

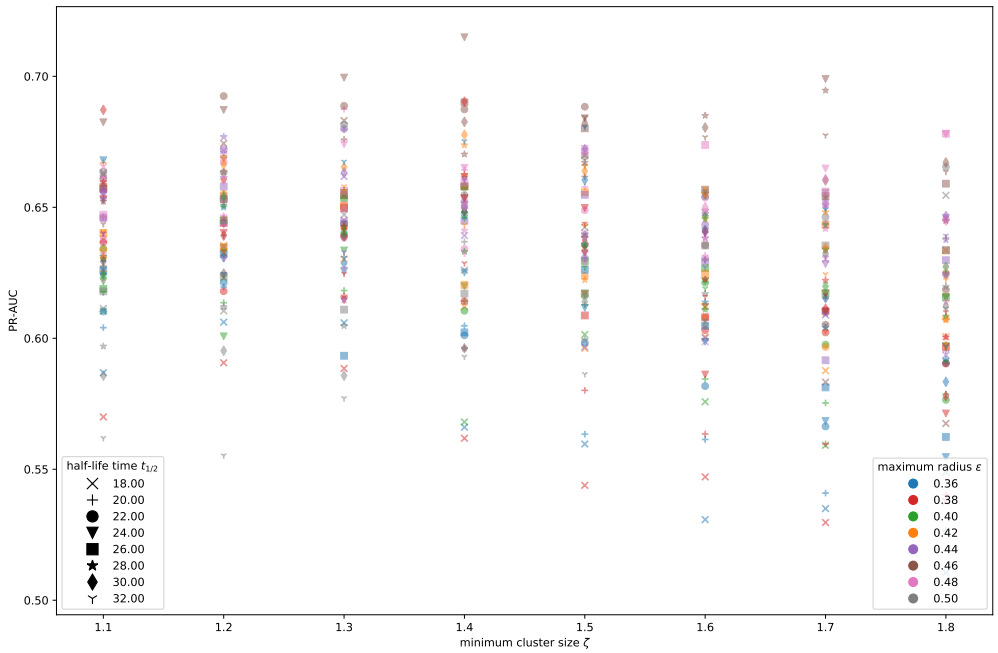

Figure A.3: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on line data set with real-time outliers

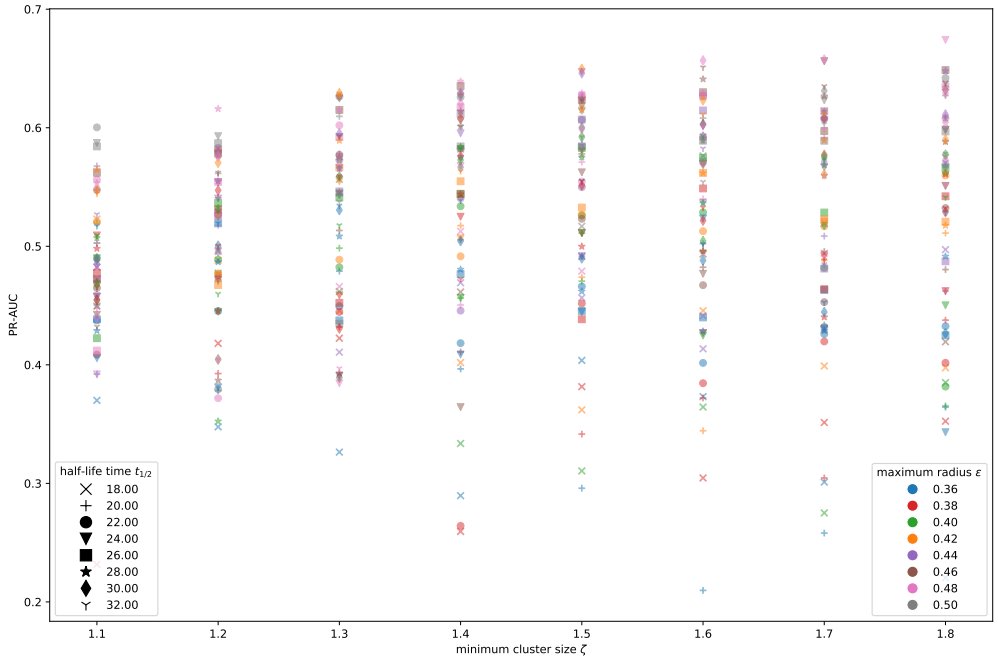

Figure A.4: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on line data set with persistent outliers

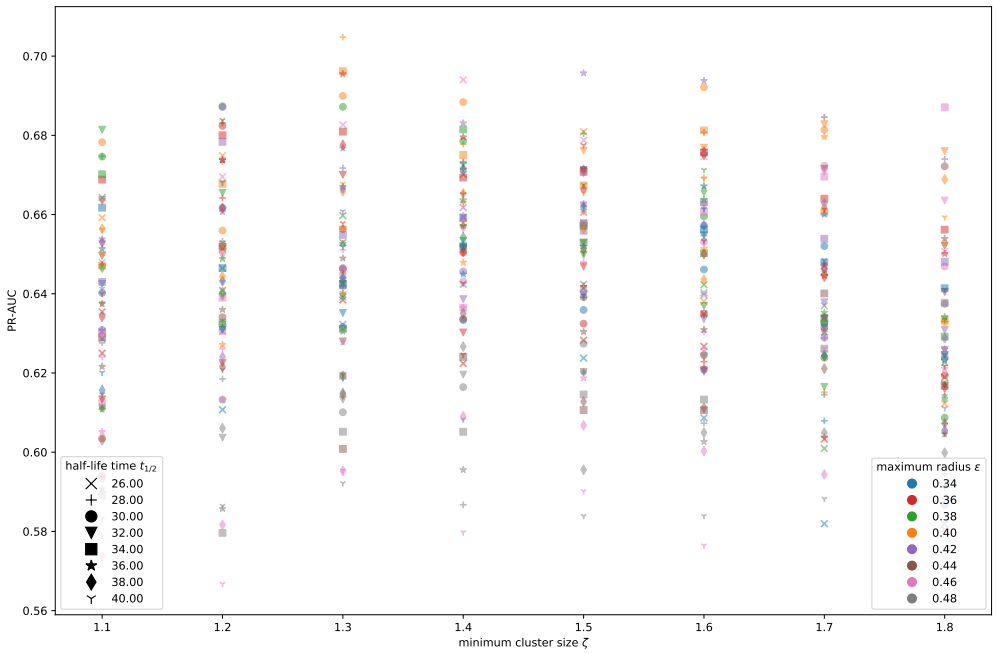

Figure A.5: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on sinus data set with real-time outliers

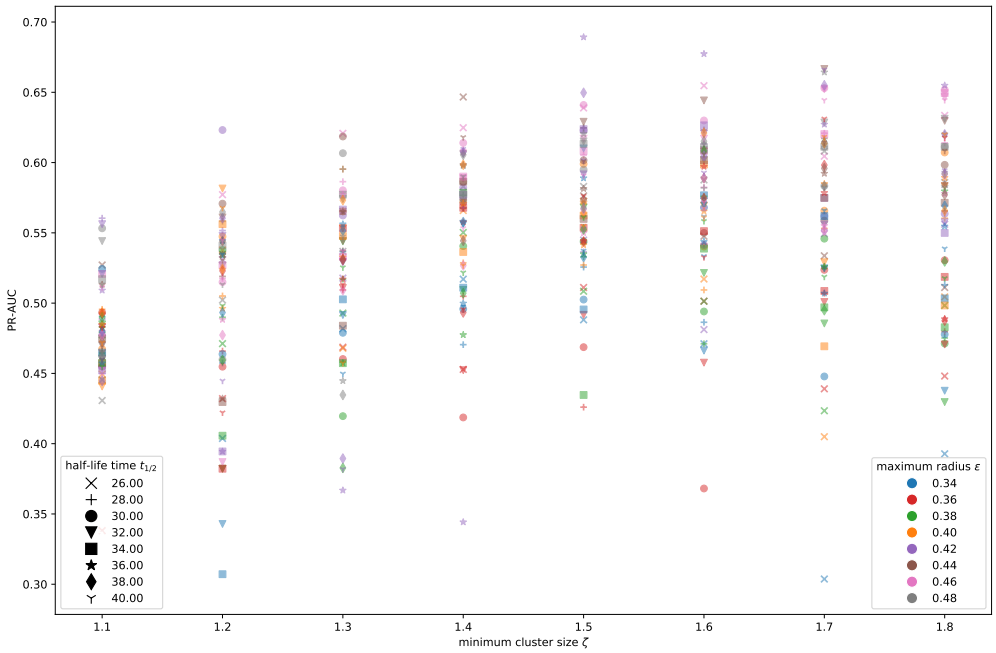

Figure A.6: ζ, $t_{1/2}$ and ϵ grid search for PR-AUC on sinus data set with persistent outliers

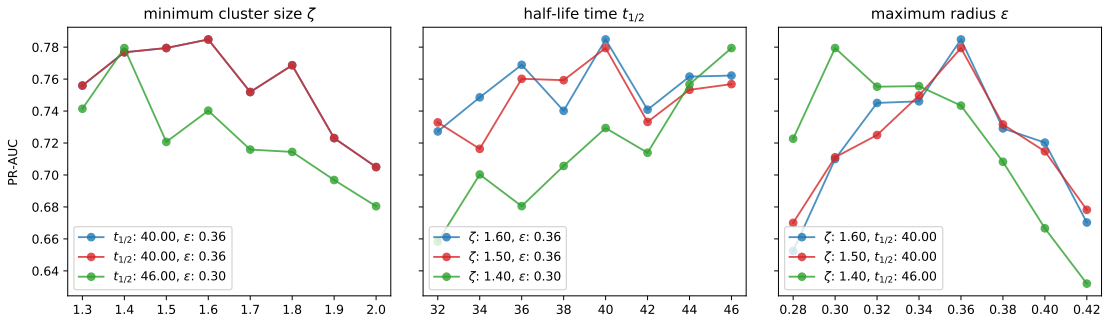

Figure A.7: Sensitivity of ϵ, ζ and $t_{1/2}$ on static data set with real-time outliers

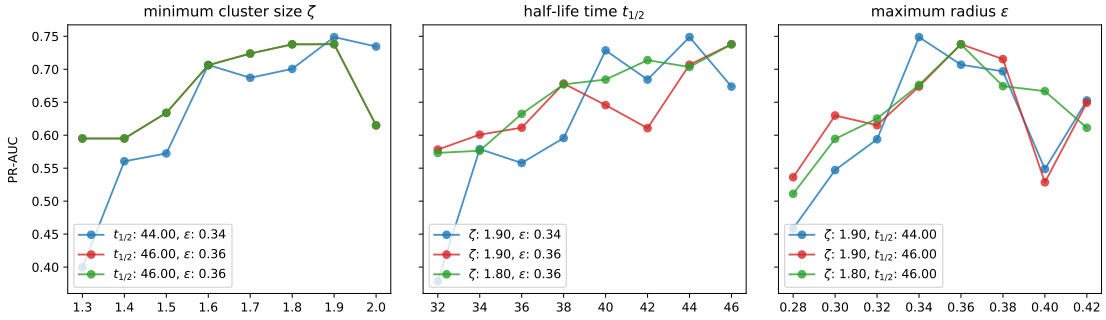

Figure A.8: Sensitivity of ϵ, ζ and $t_{1/2}$ on static data set with persistent outliers

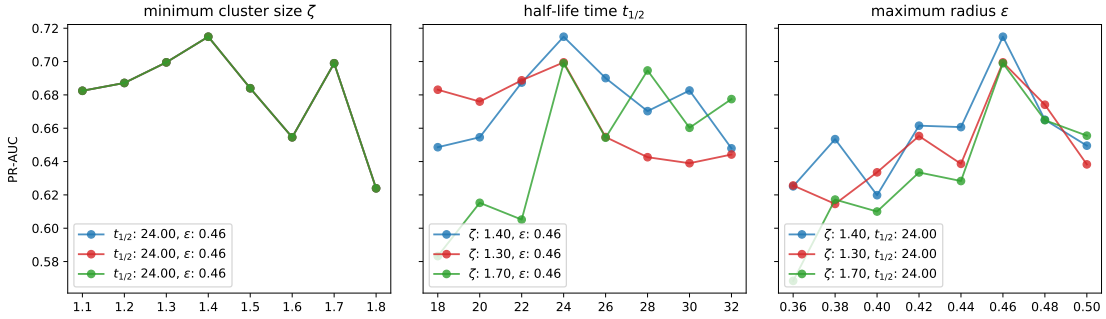

Figure A.9: Sensitivity of ϵ, ζ and $t_{1/2}$ on line data set with real-time outliers

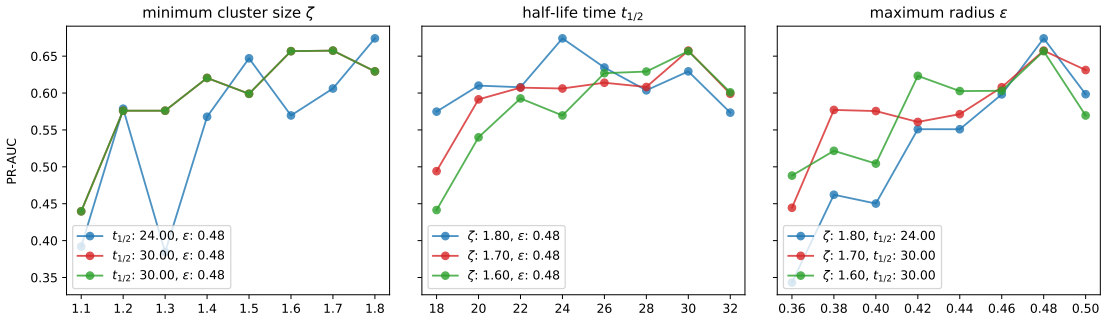

Figure A.10: Sensitivity of ϵ, ζ and $t_{1/2}$ on line data set with persistent outliers

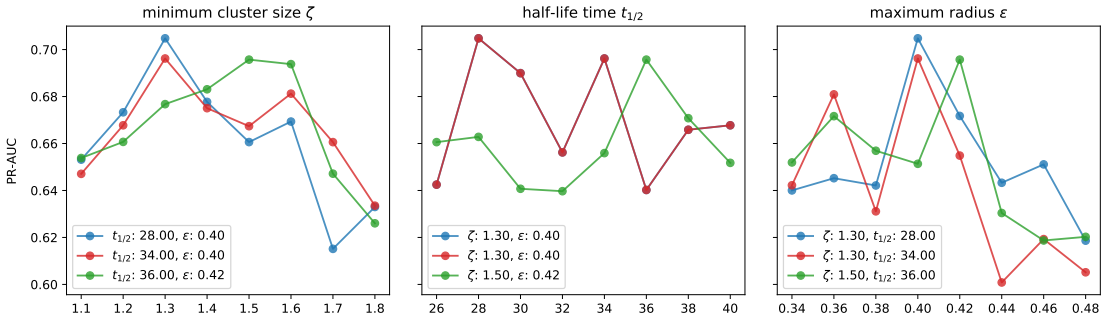

Figure A.11: Sensitivity of ϵ, ζ and $t_{1/2}$ on sinus data set with real-time outliers

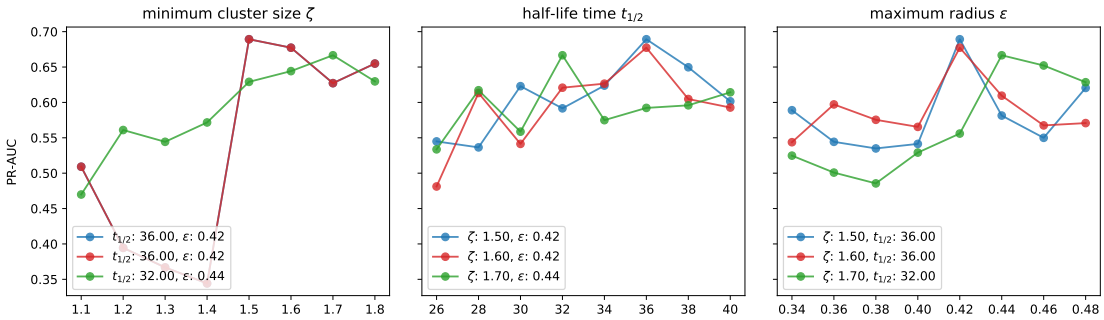

Figure A.12: Sensitivity of ϵ, ζ and $t_{1/2}$ on sinus data set with persistent outliers

A.2 Drift-Influence Hyperparameter Analysis (δ, ω)

For each data set *static, line, sinus* we performed a grid search on the two drift-influence hyperparameters δ and ω and set the other hyperparameters $(\epsilon, \zeta, t_{1/2})$ to the value of the best previous performance. We then compared the ground truth once with the real-time anomalies and once with the persistent anomalies and calculated the PR-AUC.

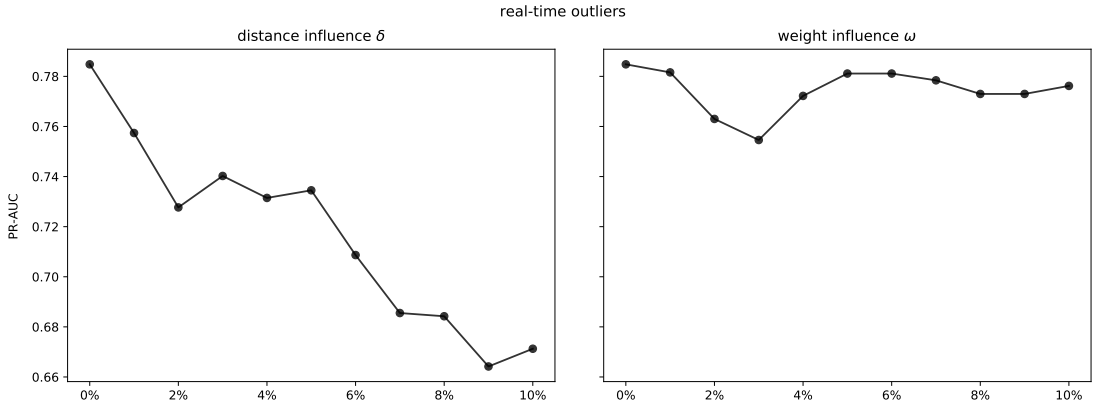

Figure A.13: Separate influence of δ and ω on static data set with real-time outliers

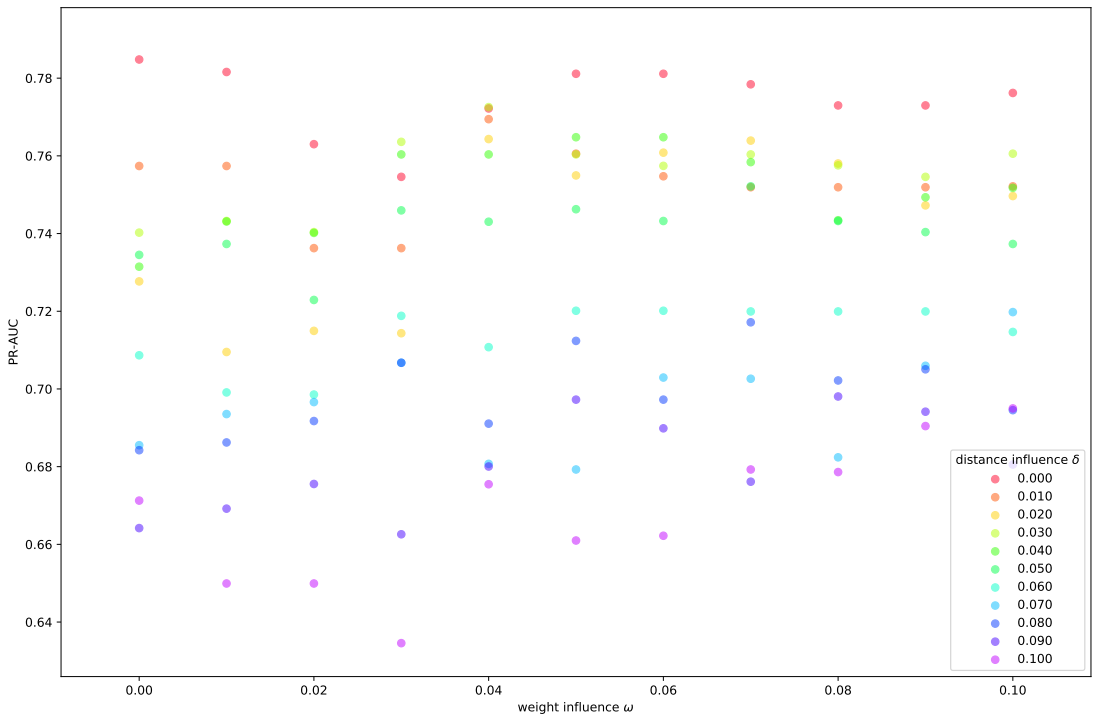

Figure A.14: Joined influence of δ and ω on static data set with real-time outliers

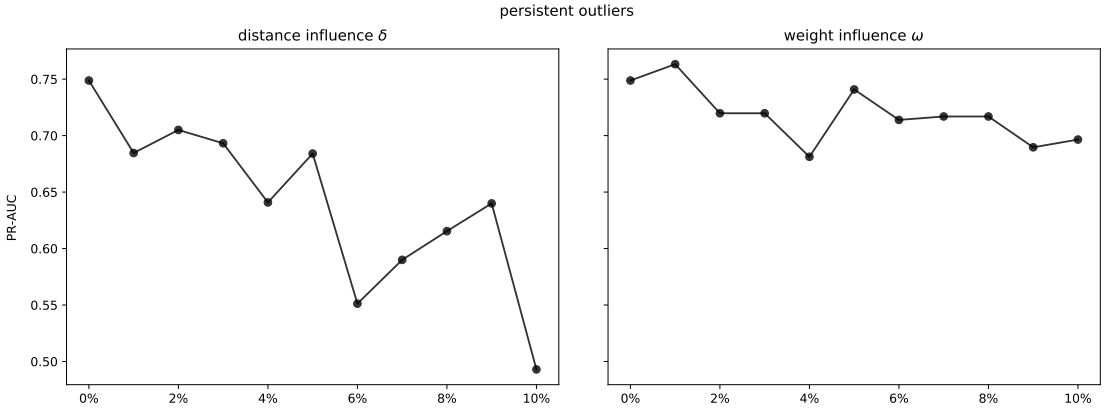

Figure A.15: Separate influence of δ and ω on static data set with persistent outliers

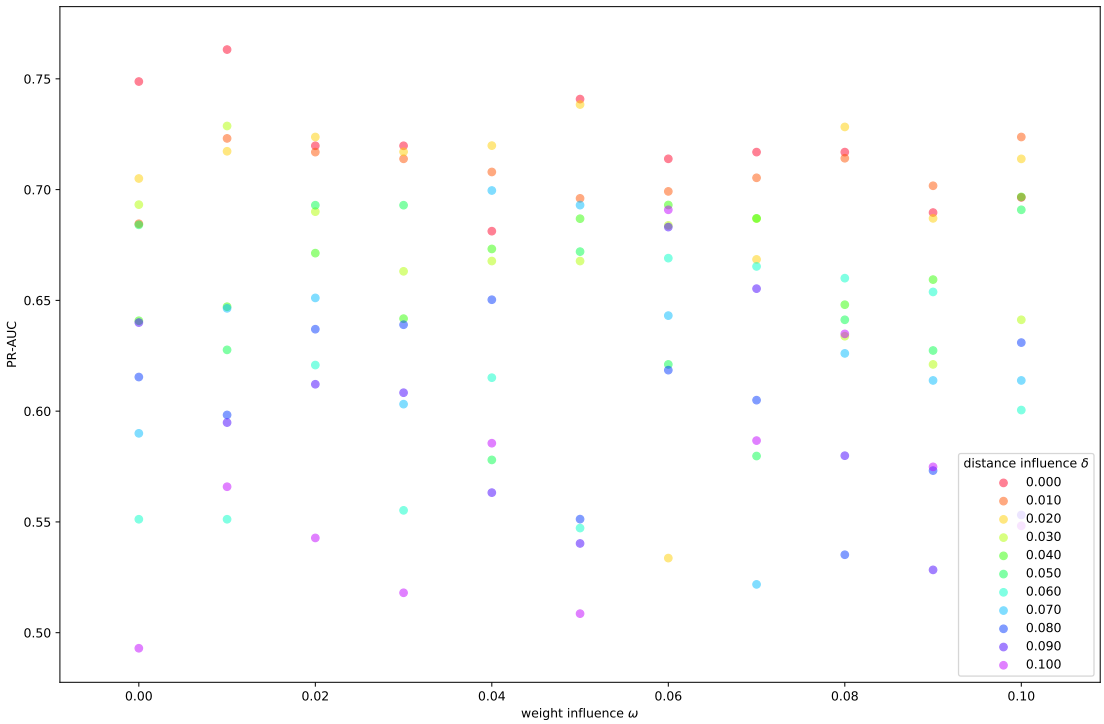

Figure A.16: Joined influence of δ and ω on static data set with persistent outliers

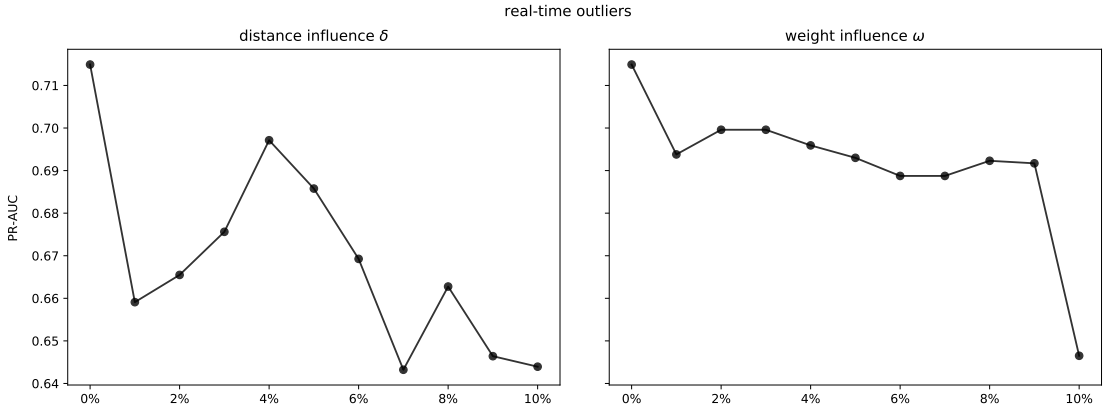

Figure A.17: Separate influence of δ and ω on line data set with real-time outliers

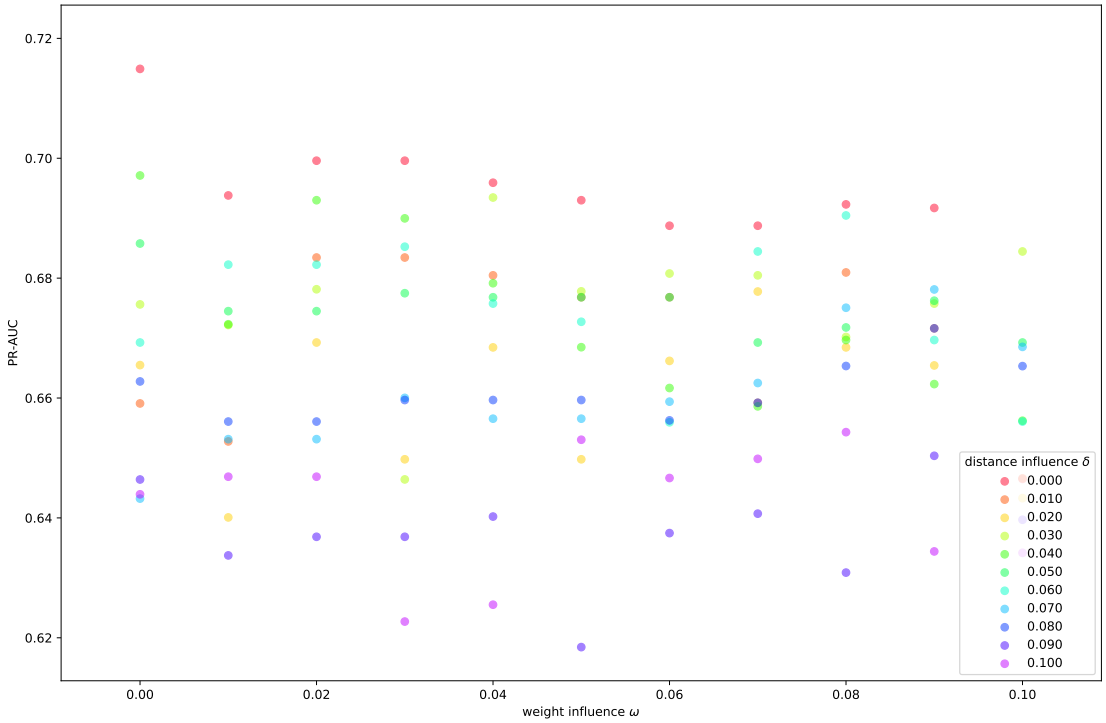

Figure A.18: Joined influence of δ and ω on line data set with real-time outliers

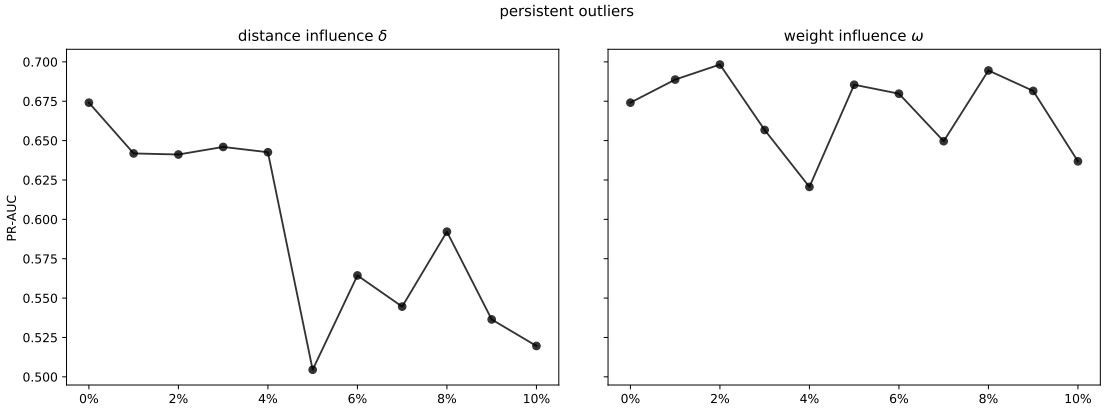

Figure A.19: Separate influence of δ and ω on line data set with persistent outliers

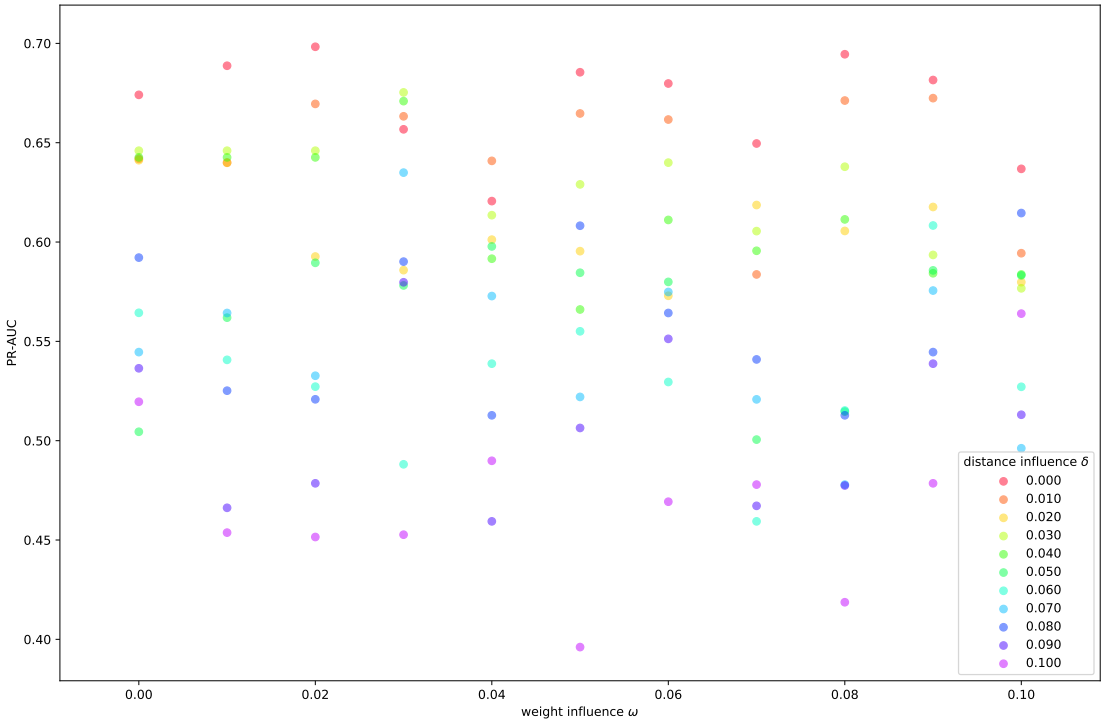

Figure A.20: Joined influence of δ and ω on line data set with persistent outliers

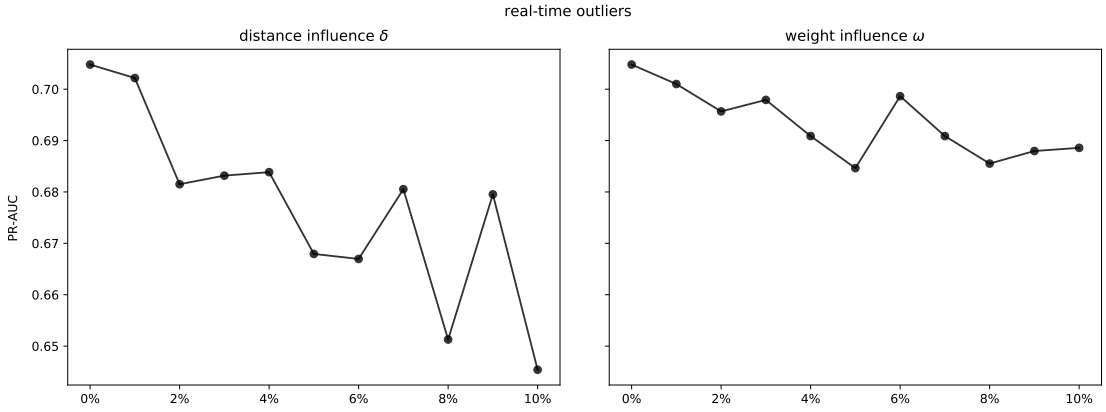

Figure A.21: Separate influence of δ and ω on sinus data set with real-time outliers

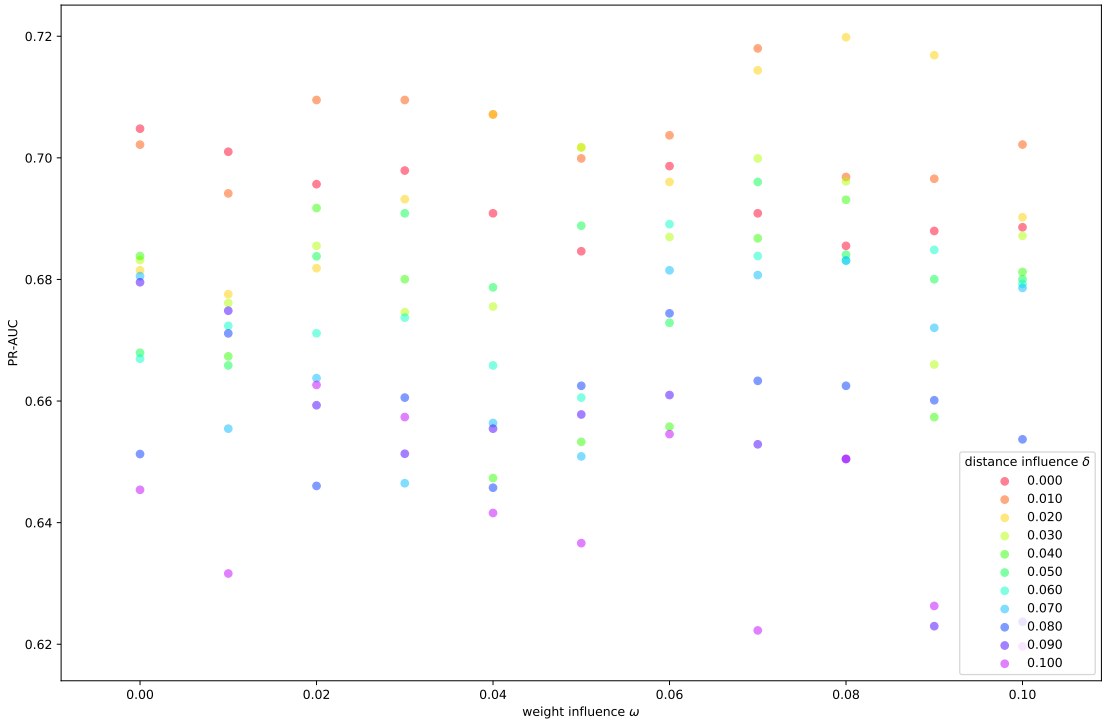

Figure A.22: Joined influence of δ and ω on sinus data set with real-time outliers

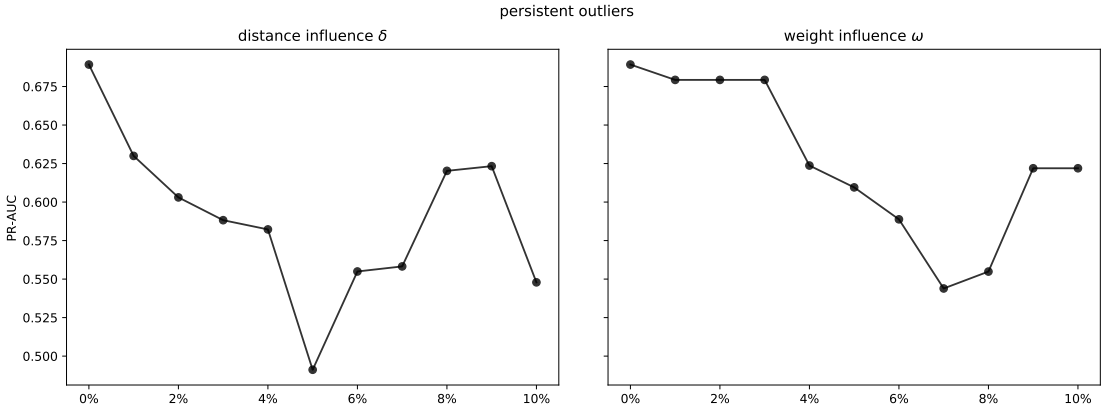

Figure A.23: Separate influence of δ and ω on sinus data set with persistent outliers

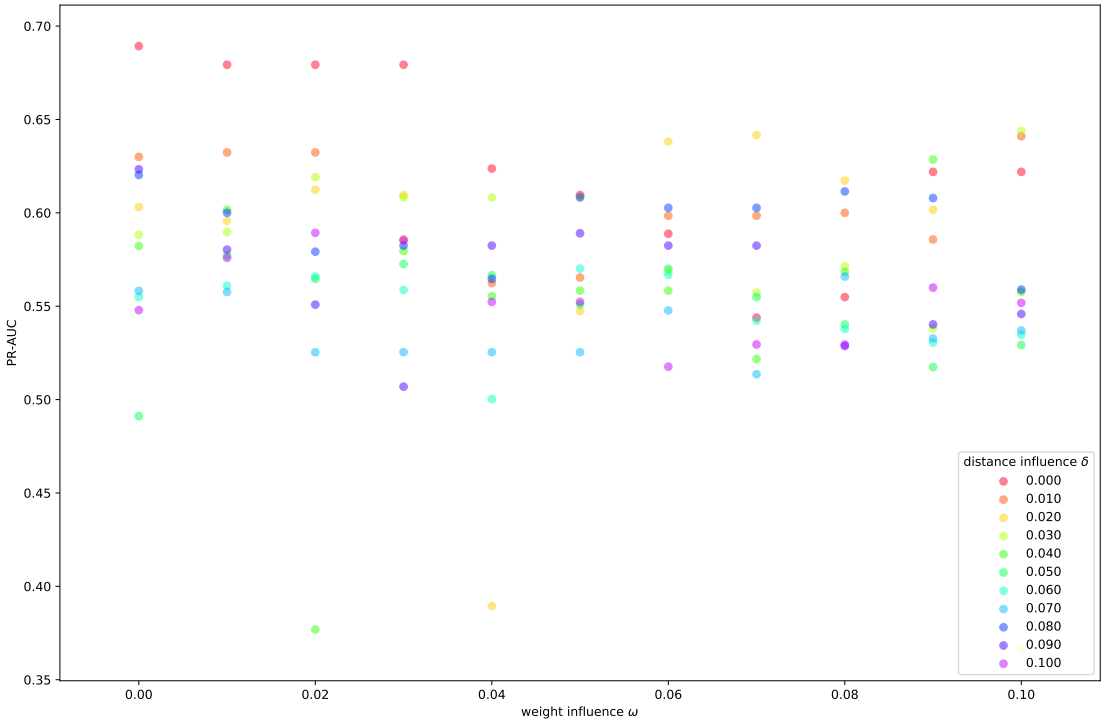

Figure A.24: Joined influence of δ and ω on sinus data set with persistent outliers

B

Content of CD

This appendix lists the content of the CD attached to this master's thesis.

- Abstract.txt (English abstract)
- Masterarbeit.pdf (Master's thesis in PDF-format)
- online-outlier-detection.zip
  - Archive (Files not in final version)
  - Data (Data sets)
  - Notebooks (Jupyter notebooks for evaluation)
  - PurePython (Python source code)
  - README.md (Description file)
- Zusfsg.txt (German abstract)

C

Installation & Run Guidelines

This appendix lists the steps to run the DenStream algorithm on the synthetic and PigData data sets and how to perform the evaluation. Further information can also be found in the *README.md* file inside the folder *online-outlier-detection* on the attached CD.

All source code and evaluation for the IPTV scenario lie on a remote server owned by the Dynamic and Distributed Information Systems Group of the University of Zurich in the folder *cweber/** for reasons of confidentiality. We also did not add the PigData data files to the attached CD.

General Installation:

1. Install Python 3.6 or newer
2. Install InfluxDB[1] and create a new database called *osd*
3. Recommended: Install PyCharm IDE[2] to run Python Code
4. Copy folder *source_code* to a working directory or clone the git repository[3]
5. Install the following Python Packages with `pip install`: *pandas, matplotlib, influxdb, jupyter, sklearn*
6. Set InfluxDB configuration in *PurePython/pig_data.py* and *PurePython/path_transition*
7. Run DenStream or run Jupyter notebooks

Run DenStream* on Synthetic Data Sets:

- `PurePython/main.py path_transition [path_type] [config_parameter]`
- Grid-Search on DenStream hyperparameters $(\lambda, \beta, \epsilon)$:
  - e.g. `PurePython/main.py path_transition sinus grid_hyperparameters`

[1] https://www.influxdata.com/
[2] https://www.jetbrains.com/pycharm/
[3] https://gitlab.ifi.uzh.ch/dellaglio/online-outlier-detection

- Grid-Search on DenStream drift-influence hyperparameters (δ, ω):
  - e.g. `PurePython/main.py path_transition sinus grid_drift_realtime`
  - e.g. `PurePython/main.py path_transition sinus grid_drift_persistent`
- Single run with custom hyperparameters and plot/noplot:
  - e.g. `PurePython/main.py path_transition sinus plot`
  - e.g. `PurePython/main.py path_transition sinus`

Run DenStream* on PigData data

- Copy PigData data sets to folder *Data/PigData* (not on CD)
- `PurePython/main.py pig [data_source]`
- Run specific data set or group:
  - e.g. `PurePython/main.py pig organ`

Create new synthetic data set:

- `PurePython/main.py path_creation [path_type]`
- Synthetic *line* data set:
  - e.g. `PurePython/main.py path_creation line`

Jupyter notebooks:

- Run `jupyter notebook`
- Hyperparameter evaluation
  - *Notebooks/DenstreamEvaluation.ipynb*
- PigData evaluation
  - *Notebooks/PigData.ipynb*

List of Figures

List of Tables

List of Algorithms