Karlsruhe Institute of Technology

Cost-Quality Trade-Offs in One-Class Active Learning

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Adrian Englhardt

aus Lauf an der Pegnitz

Tag der mündlichen Prüfung:	14. April 2021
Erster Gutachter:	Prof. Dr.-Ing. Klemens Böhm
Zweiter Gutachter:	Prof. Dr. Ira Assent

Acknowledgments

My PhD would not have been possible without the support of many people. I would like to express my appreciation to everyone who has helped me throughout this journey.

First of all, I would like to express my sincere gratitude to my supervisor Prof. Klemens Böhm. Throughout my PhD, I have found your dedicated support and guidance at all times. You gave me the freedom to set and pursue my own research agenda while always seeing the bigger picture of my PhD project. Thank you for tirelessly editing my manuscripts.

I would like to thank Prof. Ira Assent for being my second reviewer. Thank you for hosting me at Aarhus University even in the difficult pandemic times of summer 2020. The discussions with you have helped me tremendously.

I would like to thank all colleagues of my group and the members of the research training group "Energy Status Data". The discussions with you were always inspiring. You challenged me to always bring my best and encouraged me when I needed support. You helped me to reflect on the need and the implications of my work for other research fields. A special thank you goes to Holger Trittenbach who has devoted so much time to countless discussions. Working on joint research projects with you has been a blast.

Finally, I would like to thank my family and friends for continuously supporting me. You helped me to see the light at the end of the tunnel in difficult times and you made celebrating success even more enjoyable.

Abstract

Active learning is a paradigm to involve users in a machine learning process. The core idea of active learning is to ask a user to annotate a specific observation to improve the classification performance. One important application of active learning is detecting outliers, i.e., unusual observations that deviate from the regular ones in a data set. Applying active learning for outlier detection in practice requires to design a system that consists of several components: the data, the classifier that discerns between inliers and outliers, the query strategy that selects the observations for feedback collection, and an oracle, e.g., the human expert that annotates the queries. Each of these components and their interplay influences the classification quality. Naturally, there are cost budgets limiting certain parts of the system, e.g., the number of queries one can ask a human. Thus, to configure efficient active learning systems, one must decide on several trade-offs between costs and quality. The existing literature on active learning systems does not provide an overview nor a formal description of the cost-quality trade-offs of active learning. All this makes the configuration of efficient active learning systems in practice difficult.

In this thesis, we study different cost-quality trade-offs that are pivotal for configuring an active learning system for outlier detection. We first provide an overview of the costs of an active learning system. Then, we analyze three important trade-offs and propose ways to model and quantify them. In our first contribution, we study how one can reduce classification training costs by training only on a sample of the data set. We formalize the sampling trade-off between classifier training costs and resulting quality as an optimization problem and propose an efficient algorithm to solve it. Compared to the existing sampling methods in literature, our approach guarantees that a classifier trained on our sample makes the same predictions as if trained on the complete data set. We can therefore reduce the classification training costs without a loss of classification quality. In our second contribution, we investigate how selecting multiple queries allows trading off costs against quality. So-called batch queries reduce classifier training costs because the system only updates the classifier once for each batch. But the annotation of a batch may give redundant information, which reduces the achievable quality with a fixed query budget. We are the first to consider batch queries for outlier detection, a generalization of the more common case to query sequentially. We formalize batch active learning and propose several strategies to construct batches by modeling the expected utility of a batch. In our third contribution, we propose query synthesis for outlier detection. Query synthesis allows to artificially generate queries at any point in the data space without being restricted by a pool of query candidates. We propose a framework to efficiently synthesize queries and develop a novel query strategy to improve the generalization of a classifier beyond a biased data set with active learning. For all contributions, we derive recommendations for the cost-quality trade-offs from formal investigations and empirical studies to facilitate the configuration of robust and efficient active learning systems for outlier detection.

Zusammenfassung

Verfahren zur Ausreißererkennung suchen seltene und ungewöhnliche Beobachtungen in großen Datenbeständen. Die Suche nach Ausreißern spielt in einer Vielzahl von Anwendungen eine Rolle. Beispiele sind die Fehlererkennung in der maschinellen Fertigung, die Aufdeckung von finanziellem Betrug und die Überwachung von Rechnernetzen.

Ein fundamentales Problem der Ausreißererkennung ist, dass die Definition eines Ausreißers stark von der Anwendung abhängt. Eine Möglichkeit dieses Problem zu adressieren ist es, den Nutzer interaktiv in den Lernprozess einbinden. Sogenannte „Aktive Lernverfahren“ fragen den Nutzer nach Feedback zu einzelnen Beobachtungen, die für den Klassifikator einen hohen Informationsgehalt haben. Dabei sind die Fragen meist in der Form „Handelt es sich bei der Beobachtung um einen Ausreißer (ja/nein)?“. Das so gewonnene Nutzerfeedback wird verwendet, um den Klassifikator zu aktualisieren.

Ein aktives Lernverfahren besteht aus mehreren Komponenten: den Daten, dem Klassifikator, dem Nutzer und der Anfragestrategie, die Beobachtungen für Feedback auswählt. Zum einen hat jede dieser Komponenten einen Einfluss darauf, wie effektiv das System die Erkennung von Ausreißern verbessert. Zum anderen sind diese Komponenten auf vielfältige Art und Weise voneinander abhängig. Beispielsweise ist der Rechenaufwand zur Aktualisierung eines Klassifikators und zur Anfrageauswahl abhängig von der Datensatzgröße. Ein anderes Beispiel ist, dass der Nutzer womöglich mehrere ähnliche Anfragen schneller beantworten kann, als wenn sich die Anfragen stark unterscheiden. In der Praxis muss der Anwender stets zwischen dem Aufwand, also der benötigten Rechenzeit für den Klassifikator und die Anfragestrategie, der Zeit für die Annotation der Beobachtungen, und dem Nutzen, also der resultierenden Klassifikationsgenauigkeit, abwägen. Die Literatur zur Ausreißererkennung benennt zwar unterschiedliche Einflüsse auf die Effektivität des aktiven Lernens, gibt aber weder eine strukturierte Übersicht noch eine formale Beschreibung der Kosten-Nutzen-Kompromisse. Zudem ist unklar wie sich diese Kompromisse empirisch quantifizieren lassen. All dies erschwert es aktive Lernverfahren effizient einzusetzen.

In dieser Thesis beschäftigen wir uns mit der Frage, wie sich verschiedenen Kosten-Nutzen-Kompromisse modellieren und quantifizieren lassen, sodass sich geeignete Designentscheidungen bei der Realisierung aktiver Lernsysteme für die Ausreißererkennung treffen lassen. Wir geben dafür eine Übersicht über die verschiedenen Kompromisse, die beim aktiven Lernen auftreten. Anschließend betrachten wir drei wichtige Kompromisse genauer, geben jeweils theoretische Grundlagen zu deren Bewertung und schlagen algorithmische Lösungen vor, damit der Nutzer diese Kompromisse kontrollieren kann.

Klassifikatortrainingslaufzeit reduzieren durch Auswahl einer Teilmenge der Daten

In einem ersten Schritt betrachten wir die Trainingslaufzeiten des Klassifikator und wie diese reduziert werden kann, indem nur eine Teilmenge der Daten für das Training ausgewählt wird. Wir stellen uns dabei die Frage, wie die Anzahl der Beobachtungen reduziert werden kann, ohne die Klassifikationsgenauigkeit zu verschlechtern. Hierzu formalisieren wir die Datenreduktion als Optimierungsproblem. Aufbauend auf dem Optimierungsproblem schlagen wir einen neuen Ansatz zur Datenreduktion vor. Wir garantieren für unsere Reduktion, dass ein auf den reduzierten Daten gelernter Klassifikator die Daten genauso klassifiziert, wie wenn er auf dem kompletten Datensatz gelernt wird. Mit unserem neuen Ansatz lassen sich komplexe, hochdimensionale Datensätze effizient reduzieren und lange Trainingszeiten vermeiden.

Klassifikatoraktualisierung verzögern durch Batch-Anfragen

Als nächstes beschäftigen wir uns mit der Anfragenauswahl. Bisherige Ansätze im Bereich der Ausreißererkennung arbeiten sequentiell. Sie selektieren eine einzelne Anfrage und aktualisieren dann den Klassifikator basierend auf dem Feedback. Um die Anzahl der Aktualisierungen des Klassifikators zu reduzieren, kann man mehrere Anfragen auf einmal auszuwählen, sogenannte Batch-Anfragen, und den Klassifikator dann nur einmal für jedes Batch aktualisieren. Die Auswahl von Batch-Anfragen ist schwieriger als die Selektion sequentieller Anfragen. Bei der Auswahl von Batches muss nicht nur der Informationsgehalt einzelner Annotationen betrachten werden, sondern auch wie divers die Beobachtungen eines Batches über den Datenraum verteilt sind. Wir stellen verschiedene Möglichkeiten vor Batch-Anfragen für die Ausreißererkennung zu selektieren und evaluieren diese gegeneinander. Eine Einsicht unserer Evaluation ist, dass man mit der geeigneten Auswahl der Batch- Anfragestrategie die Laufzeit für die das Training des Klassifikators um eine Größenordnung verringern kann, ohne an Klassifikationsgenauigkeit zu verlieren.

Exploration des Datenraums zur Erhöhung der Klassifikationsgenauigkeit

Zum Abschluss der Thesis beschäftigen wir uns mit der Frage, wie ein Klassifikator zur Ausreißererkennung über einen gegebenen Datensatz hinaus verbessert werden kann. Damit der Klassifikator besser auf neue und während des Trainings nicht verfügbare Daten generalisiert, muss das aktive Lernverfahren Feedback in unbekannten Regionen des Datenraums sammeln, in denen bisher noch keine Beobachtungen vorkommen. In einem hochdimensionalen Datenraum sind jedoch zufälligen Anfrage ineffizient. Daher stellen wir uns die Frage, wie man ausgehend von einem gegebenen Datensatz und einem darauf gelernten Klassifikator den Datenraum explorieren kann. Wir entwickeln dafür ein neues Verfahren, das Anfragen effizient synthetisiert. Das gewonnene Nutzerfeedback erweitert Bereiche, die als „normal“ klassifiziert werden, also keine Ausreißer enthalten, und verbessert damit die Klassifikationsgenauigkeit über den Trainingsdatensatz hinaus.

Über die Thesis hinweg geben wir Empfehlungen, wie mit unseren Ansätzen die Kosten des aktiven Lernens gesenkt werden können, ohne an Klassifikationsqualität zu verlieren. Dabei ziehen wir unsere Schlüsse aus empirischen Untersuchungen und formal Garantien. In der Praxis unterstützen unsere Formalisierungen den Anwender dabei Kosten und Nutzen in einem Anwendungsfall abzuwägen und die Komponenten des aktiven Lernsystems entsprechend zu konfigurieren.

Contents

Part I.

Introduction

1. Motivation

Machine learning systems are omnipresent in our everyday life. They help us solve many ordinary tasks more quickly, such as transcribing conversations via speech recognition or categorizing images. They give us personal recommendations that affect our consumption behavior [MS10] and can now surpass the human level of play in many games, like chess, Shogi (Japanese chess), and Go by only learning from self-play [Sil+17].

The goal of a machine learning system is to learn a mathematical model for a given task from experience without requiring explicit programming [Alp20]. Here, the experience is data collected upfront and during the learning process. The learning process is a data-driven. Thus, the performance of the learned machine learning system hinges on the quantity and quality of the data. Since data collection incurs costs, researchers investigate how to collect data for machine learning efficiently [FLS06; Set12].

One common application of machine learning is outlier detection. Outlier detection searches for observations that are significantly different from the rest of the data [Agg15]. Outlier detection is well-motivated in real-world applications such as spam filtering [BB08; Cor08], fraud detection [AFR97; FP99], and monitoring of machines [FYM05; KLC02; YWF18], of health sensors [Lin+05; Won+03], or of network traffic [Gör+09; Laz+03; Sto+08]. Detecting outliers is a difficult task because they are rare and are thus difficult to model. One approach to outlier detection is to use one-class classifiers. These classifiers learn the concept of one target class. They then classify all observations that are different from this concept as outliers.

A fundamental problem with outlier detection is that the definition of an outlier depends on the use case. For example, during network traffic monitoring, a sudden high amount of traffic identified as unusual in a residential area may be completely average in a data center. One way to address this problem is to interactively include an expert in the learning process [AZL06]. So-called active learning systems ask a user for feedback on individual observations to improve the classification. They select queries that have high informativeness for the classifier, e.g., observations for which the classifier is unsure about their classification. The questions for the user are usually in the simple form "Is the observation an outlier (yes/no)?". After answering the query, the system updates the classifier with the user feedback. This way, a user influences the classifier and improves its performance for the given use case.

Labeling queries with a human annotator is the most prevalent scenario for active learning. However, there also are other scenarios where the oracle is not a human expert [BSM19]. One such scenario is design space exploration [CF17a; CF17b; LM12]. There, one tries to learn feasible regions in a potentially unbound data space. An example application is a search for designs that improve the aerodynamics or the robustness in crash tests [Gor+09]. In design space exploration, annotations come from simulations or real-world experiments. In some cases, the active learning system can automatically run the simulation. In other cases, the system requires additional external help from a human to set up and run the experiment. Although the intellectual effort for annotating queries in these cases no longer lies with a human, there still are annotation costs. For instance, computing the answer for a query requires resources such as computational power, actual materials and takes time.

An active learning system consists of several components: the data, the classifier, the oracle that answers the questions, and the query strategy that selects observations for feedback. On the one hand, each of these components impacts how effective the system is in improving the detection of outliers. On the other hand, these components are interdependent in many ways. For example, the computational costs for updating a classifier and for query selection depend on the size of the data set. Another example is that the oracle may answer several similar queries faster than if they are very different to each other. For instance, this is the case if there are change-over costs between experiments, which are negligible when experiments are similar to each other but high for dissimilar experiments [GK11].

Cost budgets limit certain aspects of an active learning system. For example, a project deadline limits the time for active learning, and there may be a fixed monetary budget. There may also be physical limits, such as the laboratory space for running experiments. An operator that configures an active learning system must adhere to these budgets. So on the one side, the interplay of all components influences the classification quality. On the other side, cost budgets limit certain design choices when configuring an active learning system. In practice, the operator must consider the trade-off between costs and quality to configure an efficient active learning system. Otherwise, he may configure an inefficient system that wastes resources or an ineffective system that does not achieve a required classification quality.

Although the literature on active learning for outlier detection is aware of a few of the cost-quality trade-offs that influence active learning, it gives neither a structured overview nor a formal description of these trade-offs. Existing works focus on individual components, e.g., speed up the classifier training [Ala+20; HZH14; Kra+18; Li+18; Li+19; Li11; Qu+19; Sun+16; Xia+14; Zhu+14] without considering all other active learning system components. Understanding these trade-offs is crucial to efficiently apply active learning for outlier detection. In this thesis, we address the question of how one can model and quantify different cost-quality trade-offs to facilitate the configuration of outlier-detection active learning systems.

1.1. Challenges

Configuring an active learning system for outlier detection is difficult. It requires expertise in the research area of outlier detection as well as active learning. We identify the following three challenges that make the application of active learning for outlier detection challenging.

(C1) Large Configuration Space Recall that an active learning system consists of multiple components: the data, the classifier, the query strategy, and the oracle. The use case typically determines the data and the oracle, i.e., these two components are fixed. However, the classifier and query strategy are configurable. The operator has to select one out of the many available algorithms for both of the components. Additionally, there are dependencies between these two components that restrict the choices. Some query strategies require a specific classifier, while others are independent of the classifier choice. Next, when the data set is large, classifier training and query selection have long runtimes. Here, the operator must rely on *approximations*, e.g., suboptimal query selection from only a fraction of the data. Approximations require to trade off costs against quality. When considering approximations, there are various options that the operator can choose from to reduce the computational runtime during active learning. Literature lacks an overview of the various components and approximations available. This makes it difficult to configure an active learning system in practice.

(C2) Incoherent Evaluation Standards Another challenge is the difficulty of accessing the performance of an active learning system. The lack of coherent evaluation standards makes it difficult to cope with the large configuration space. This is challenging on two levels: On the one hand, there are many algorithms for specific components and approximations, e.g., reducing the classifier training time [Ala+20; HZH14; Kra+18; Li+18; Li+19; Li11; Qu+19; Sun+16; Xia+14; Zhu+14]. Since literature lacks an exhaustive experimental comparison, it is difficult to choose a particular method over another one. On the other hand, the performance of the active learning system depends on the interplay of all these components and approximations. While there is a study comparing existing one-class active learning classifiers and query strategies [TEB21], it does not cover approximations, i.e., scenarios where the operator has to take additional measures to trade off costs against quality. Literature does not provide ways to quantify the influence of approximations on the resulting quality of the active learning system.

(C3) Outlier Properties Outliers are rare, unusual, and may not follow a common distribution. These properties challenge all components of an active learning system. Generally, solutions and insights from multi-class active learning may not apply to active learning for outlier detection. For example, classifiers and query strategies that use density estimation require modifications before one can use them. Outliers may also affect the annotation because they are more difficult or expensive to annotate. The operator must therefore factor in the outlier properties when configuring an active learning system.

1.2. Contributions

In this thesis, we address the question of how to model and quantify different cost-quality trade-offs so that an operator can make appropriate design decisions when configuring active learning systems for outlier detection. To this end, we give an overview of the different cost-benefit trade-offs that occur in active learning. We discuss the various components of an active learning system as well as existing approximations that allow trading between quality and costs. This overview addresses challenge *(C1) Large Configuration Space.* Afterwards, we take a closer look at three important trade-offs, give theoretical ground for their evaluation, and propose new state-of-the-art algorithms so that the operator can control these trade-offs.

Reduce classifier training time by selecting a data subset In a first step, we look at the training runtimes of the classifier and how one can reduce them by selecting a sample of the data for training. We ask how one can reduce the number of observations without reducing the classification accuracy. For this purpose, we formalize sample selection as an optimization problem. Based on the optimization problem, we propose a new approach for data reduction named Reducing sAmples by Pruning of Inlier Densities (RAPID). RAPID guarantees that a classifier learned on the reduced data yields the same predictions as if it was learned on the complete data set. With our new approach, we can efficiently reduce complex, high-dimensional data sets and avoid long training times.

Delay classifier updates through batch queries Next, we deal with the selection of queries. Previous approaches in outlier detection work sequentially. They select a single query and then update the classifier based on the feedback. To reduce the number of updates of the classifier, one can select multiple queries at once, so-called *batch queries*. One must then update the classifier only once for each batch. But the annotation of a batch may give redundant information if the queries are similar to each other, which reduces the quality. Selecting batch queries is, therefore, more complex than selecting sequential queries. Batch query strategies must consider both the expected informativeness for annotating a single observation and the annotation dependency between them. Thereby, they must decide how to distribute the queries of a batch over the data space. We propose different ways to select batch queries for outlier detection suitable for the *(C3) Outlier Properties*. We then evaluate them against each other. One takeaway from of our evaluation is that with the appropriate selection of the batch query strategy, one can reduce the runtime to train the classifier by one order of magnitude without losing classification accuracy.

Exploration of the data space to increase classification accuracy Finally, we deal with the question of how to improve the generalization of classifier for outlier detection to new data. One way is to collect feedback in unknown regions of the data space where no observations were available during the training. However, in a high-dimensional data space, randomly querying or even structured querying, for example, with a grid, is inefficient because such queries do not take acquired annotations into account. Therefore, we ask ourselves how to actively explore the data space starting from a given data set and a trained classifier. We propose a framework for one-class query synthesis to efficiently find queries based on an expected informativeness function of a query. Based on our framework, we develop a novel query synthesis strategy named Domain Expansion Stragegy (DES). DES asks for feedback to extend the inlier regions. DES improves the classification accuracy beyond the training data set. Evaluating query synthesis is challenging because one requires an oracle that can annotate arbitrary queries. Existing works on outlier detection only consider queries from a fixed pool of observations. We propose three different ways to simulate oracles to evaluate query synthesis for outlier detection with respect to *(C2) Evaluation Standards* and *(C3) Outlier Properties*.

1.3. Thesis Outline

The thesis consists of three parts. In the first part, we explain the fundamentals of outlier detection and active learning (Chapter 2). We go over the various components of an active learning system for outlier detection and discuss their interplay with each other. In Chapter 3, we review related work. The second part is the core of this thesis and describes our three contributions. In Chapter 4 we propose RAPID, our novel approach to sample data set to reduce training runtimes. Chapter 5 discusses the selection of *batch queries* to further reduce classifier and query selection runtimes during active learning. Finally, we present our novel approach DES to improve a classifier beyond a given data set in Chapter 6. In the last part of this thesis, we conclude with a summary (Chapter 7) and discuss open questions for future research (Chapter 8).

2. Fundamentals

This chapter introduces the notation and fundamentals for outlier detection and active learning. In Section 2.3.3 we discuss the cost-quality trade-offs of active learning. This chapter reuses Section 2 of [Eng+20a].

2.1. Preliminaries

Data Let $\mathbf{X} = \langle x_1, x_2, \dots, x_N \rangle$ be a data set of N *observations* from the domain $\mathcal{X} = \mathbb{R}^M$ where M is the number of dimensions. Each observation $x \in \mathbf{X}$ is an M-dimensional numerical vector. Let a *sample* be a subset $\mathrm{S} \subseteq \mathbf{X}$ of the data set with a sampling ratio $^{|\mathrm{S}|}/_{N}$. Further, we denote $x \in \mathrm{S}$ as a *selected*, and $x \notin \mathrm{S}$ as a *not-selected* observation.

Class Labels Let $\mathrm{Y} = \langle y_1, y_2, \dots, y_N \rangle$ be the class labels of a data set $\mathbf{X}$. Each entry $y \in \mathrm{Y}$ is the realization of a dichotomous variable $\mathcal{Y} = \{\text{in}, \text{out}\}$. The class labels of a data set are also called the *ground truth.*

Label Pools When working with a new data set, one may not have a class label for all the observations. We differentiate between the set of unlabeled observations $\mathcal{U} \subseteq \mathbf{X}$ and labeled observations $\mathcal{L} \subseteq \mathbf{X}$, which are complementary, i.e., $\mathcal{U} \cup \mathcal{L} = \mathbf{X}$ and $\mathcal{U} \cap \mathcal{L} = \emptyset$. Furthermore, we split the labeled observations $\mathcal{L}$ into a set of inliers $\mathcal{L}_{in}$ and outliers $\mathcal{L}_{out}$ so that $\mathcal{L}_{in} \cup \mathcal{L}_{out} = \mathcal{L}$ and $\mathcal{L}_{in} \cap \mathcal{L}_{out} = \emptyset$.

Data Density Let the probability density of $\mathbf{X}$ be $p(x)$. One can estimate the empirical density of $\mathbf{X}$ by kernel density estimation by calculating

$$d_{\mathbf{X}}(x) = \sum_{x' \in \mathbf{X}} k(x, x') \tag{2.1}$$

where k is a kernel function with $k(x, x) = 1$. We use the shorthand $d_x = d_{\mathbf{X}}(x)$ when the reference to $\mathbf{X}$ is unambiguous. Note that $d_{\mathbf{X}}$ requires normalization to represent a probability density.

One can use densities to characterize observations in different ways.

Definition 1 (Level Set) *A level set is a set of observations with equal density* $L_\theta := \{x \in \mathbf{X} \colon d_x = \theta\}$*. A super-level set is a set of observations with* $L_\theta^+ := \{x \in \mathbf{X} \colon d_x \geq \theta\}$*.*

One way to use level sets to categorize observations is to define a *level-set classifier* as a function of type $g \colon \mathcal{X} \rightarrow \mathcal{Y}$ with

$$g_\theta^{\mathbf{X}}(x) = \begin{cases} \text{in} & \textit{if}\, x \in L_\theta^+ \\ \text{out} & \textit{else.} \end{cases} \tag{2.2}$$

Another useful categorization is to separate observations into boundary points and inner points. There are various ways to define a boundary of $\mathbf{X}$ [Ala+20; HZH14; Li+18; Li+19; Li11; Qu+19; Xia+14; Zhu+14]. In this thesis, we define boundary points as observations with density values close to the minimum empirical density.

Definition 2 (Boundary Point) *Let $d_{min} = \min_{x \in \mathbf{X}} d_x$, and let δ be a small positive value. An observation $x \in \mathbf{X}$ is a boundary point of $\mathbf{X}$ if $x \in \mathbf{B}^{\mathbf{X}}$ with $\mathbf{B}^{\mathbf{X}} = L^+_{d_{min}} \setminus L^+_{(d_{min}+\delta)}$.*

2.2. Outlier Detection

The goal of outlier detection, often also called anomaly detection, is to find unusual, rare observations in a data set. Hawkins gives the following definition of an outlier:

> *The intuitive definition of an outlier would be "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism".* *Douglas Hawkins, 1980, [Haw80]*

Outlier detection appears in a variety of real-world applications, see [Agg15] for an overview. Additionally, one can apply outlier detection to clean a data set, i.e., remove the outliers, to facilitate further analysis [SM11].

There are a plethora of different approaches for outlier detection. We categorize them by two criteria: the availability of class labels and the detection output they provide.

2.2.1. Class Label Availability

There are three scenarios for which class labels are available.

Unsupervised In the *unsupervised* scenario, no class labels are available, i.e., all observations are unlabeled. Formally the label pools are $\mathcal{U} = \mathbf{X}$ with $\mathcal{L}_{in} = \mathcal{L}_{out} = \emptyset$. To perform outlier detection in this scenario, one assumes that the majority of the observations are inliers. The observations that deviate "considerably" from these inliers are outliers. Here, the outlier detection approach defines what "considerably" is. When working with a new data set, the unsupervised scenario is most common because, in a new application, no labels are available initially.

Supervised In the *supervised* scenario, each observation has a label. Formally the label pools are $\mathcal{U} = \emptyset$ with $\mathcal{L}_{in} \cup \mathcal{L}_{out} = \mathbf{X}$. Since all labels are available, one can directly learn a classifier. The key difference from other machine learning tasks is the unbalanced nature of the labels. There are much fewer outliers than inliers, i.e., $|\mathcal{L}_{out}| \ll |\mathcal{L}_{in}|$. The supervised scenario for outlier detection is uncommon in practice because acquiring a fully annotated data set is expensive.

Semi-supervised In the *semi-supervised* scenario, there are both labeled and unlabeled observations. Formally the label pools are $\mathcal{L}_{in} \cup \mathcal{L}_{out} \cup \mathcal{U} = \mathbf{X}$ with $\mathcal{U} \neq \emptyset$. Typically, there are far fewer labeled observations than unlabeled ones, i.e., $|\mathcal{L}| \ll |\mathcal{U}|$. Semi-supervised outlier-detection approaches are often extensions of unsupervised methods. These extensions bias the detection with the available label information.

2.2.2. Detection Output

We follow [Agg15] and distinguish between two types of outlier detection outputs: outlier scores and binary classification. We define both types in the following.

2.2.2.1. Outlier Scores

Outlier detection approaches of this type output a score for each observation that quantifies the level of "outlierness" [Agg15].

Definition 3 (Outlier Score Function) *An* outlier score function *is a function of type* $score : \mathcal{X} \longrightarrow \mathbb{R}$.

The *score* function assigns a real-valued number to each observation in the data set $\mathbf{X}$. There are various ways to quantify the "outlierness" of an observation, e.g., based on statistical analysis [AAR96; KSZ08] or with proximity-based models [Bre+00; Pap+03].

The *score* function induces a ranking of the observations. Depending on the *score* function, observations with higher (or lower) values have higher "outlierness". One can inspect this ranked list of observations with their scores to find outliers. However, analyzing a list of thousands of values is difficult. In many cases, one is only interested in a binary decision whether an observation is outlying or not. Then, one can set a *threshold* value. One classifies observations with a score higher (or lower) than the threshold as outliers. Finding this *threshold* is difficult in practice and may require domain knowledge.

2.2.2.2. Binary Classification

Outlier detection approaches of this type directly output a binary classification for each observation. Their idea is to learn a boundary around the inliers and classify everything as outliers outside of this boundary. Since they learn one concept, they are often also called *one-class classifiers*. Formally the boundary is a decision function:

Definition 4 (Decision Function) *A* decision function f *is a function of type* $f : \mathcal{X} \mapsto \mathbb{R}$. *An observation belongs to the outlier class if* $f(x) > 0$ *and to the inlier class otherwise.*

The decision function of a one-class classifier is more flexible than a score-based output. The decision function can assign labels to any data point in the data space after training the classifier. Score-based approaches calculate the scores once for a fixed data set and fix the threshold. Classifying new observations manipulates these scores and thus requires a recalculation of the scores and the threshold. For active learning, one-class classifiers are also preferred over score-based approaches because they allow us to inject feedback, as we see later in Section 2.3.

2.2.3. One-Class Classifiers

There are two categories of one-class classifiers: support-vector classifiers and non-support-vector classifiers [KM14]. Support-vector classifiers are the prevalent choice for one-class active learning [TEB21]. We now introduce two support-vector classifiers that we use in this thesis.

SVDD Support Vector Data Description (SVDD) [TD04] is a one-class classifier that learns a decision function. SVDD is an unsupervised classifier, i.e., it cannot use label information. SVDD is a quadratic optimization problem that searches for a minimum enclosing hypersphere with center a and radius R around the data. The goal of SVDD is to include all observations in hypersphere. SVDD may take a cost C to exclude an observation from the hypersphere if this reduces the radius significantly. The SVDD optimization problem is

$$\begin{aligned} \text{SVDD}: \underset{a,\, R,\, \xi}{\text{minimize}} \quad & R^2 + C \cdot \sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & \|\phi(x_i) - a\|^2 \leq R^2 + \xi_i,\ i = 1, \ldots, N \\ & \xi_i \geq 0,\ i = 1, \ldots, N \end{aligned} \tag{2.3}$$

with a cost parameter $C \in (0, 1]$, slack variables ξ, and a function $\phi : \mathcal{X} \mapsto \mathcal{H}$ that maps observations $x \in \mathcal{X}$ to a reproducing kernel Hilbert space $\mathcal{H}$. Solving SVDD gives a fixed a, R, and a decision function learned on a data set $\mathbf{X}$

$$f^{\mathbf{X}}(x) = \begin{cases} \text{in} & \textit{if}\ \|\phi(x) - a\|^2 \leq R^2 \\ \text{out} & \textit{else.} \end{cases} \tag{2.4}$$

Evaluating the decision function of the SVDD gives the distance to the decision boundary; positive for outliers, negative or zero for inliers.

One can obtain a solution of SVDD by solving the dual optimization problem with Lagrangian dual variables α_i for each observation $x_i \in \mathbf{X}$

$$\begin{aligned} \text{SVDD}_{\text{dual}}: \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{N} \alpha_i \langle \phi(x_i), \phi(x_i) \rangle - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle \\ \text{subject to} \quad & \sum_{i=1}^{N} \alpha_i = 1 \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, N. \end{aligned} \tag{2.5}$$

Since the SVDD$_{\text{dual}}$ only contains inner products of the form $\langle \phi(x_i), \phi(x_j) \rangle$ one can apply the *kernel trick*. This is, all the inner products are replaced with a kernel function $k(x_i, x_j) \rightarrow \mathbb{R}$ where $x_i, x_j \in \mathcal{X}$. Solving SVDD$_{\text{dual}}$ gives fixed values α_i. One can then calculate R^2 with any support vector (SV), i.e., an observation $\widehat{x} \in \mathbf{X}$ with an $\widehat{\alpha}$ where $0 < \widehat{\alpha} < C$ holds:

$$R^2 = k(\widehat{x}, \widehat{x}) - 2 \sum_{i=1}^{N} \alpha_i k(\widehat{x}, x_i) + \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j). \tag{2.6}$$

The decision function based on the dual solution is

$$f(x) = k(x,x) - 2\sum_j \alpha_j k(x,x_j) + \underbrace{\sum_{j=1}^{N}\sum_{k=1}^{N} \alpha_j \alpha_k k(x_j,x_k)}_{\text{const}} - R^2. \tag{2.7}$$

The decision function $f(x)$ based on the SVDD$_{\text{dual}}$ solution only relies on inner product calculations between x and some training observations, the support vectors. So classification with SVDD is efficient if the number of support vectors is low.

SVDD has two hyperparameters: C and a kernel function k. $C \in \mathbb{R}_{(0,1]}$ is a trade-off parameter to allow observations to fall outside of the hypersphere. Formally, observations outside the hypersphere with positive slack $\xi > 0$ are weighted by C, see Equation 2.3. High values for C make excluding observations expensive; based on the dual of SVDD, one can see that if $C = 1$, SVDD degenerates to a hard-margin classifier [TD04].

The kernel function k allows a decision boundaries of arbitrary shape. The – by far – most popular kernel with SVDD is the Gaussian kernel, often also called the Radial Basis Function (RBF) kernel,

$$k_{\text{RBF}}(x,x') = e^{-\gamma\|x-x'\|^2}. \tag{2.8}$$

There is an equivalent formulation with a parameter σ with $k_{\text{RBF}}(x,x') = e^{-\|x-x'\|^2/2\sigma^2}$ where $\gamma = 1/2\sigma^2$. The bandwidth parameter γ controls the flexibility of the decision boundary. Large values make the kernel pointy and thus increase the flexibility of the decision boundary. Small values, in comparison, increase the smoothness. For $\gamma \longrightarrow 0$, the decision boundary approximates a hypersphere. Choosing good values for the two hyperparameters γ and C is difficult [TBA20]. There is no established way of setting the parameter values, and one must choose one of the many heuristics to tune SVDD in an unsupervised setting [Lia+18; Sco15; TBA20; TD04; Wan+18]. Note, that there is an alternative formulation of SVDD, called ν-SVM, that learns a separating hyperplane instead of a hypersphere but both classifiers are equivalent under mild assumptions [Sch+01].

SVDDneg Support Vector Data Description with negative Examples (SVDDneg) [TD04] is a semi-supervised extension of SVDD to incorporate negative labels. SVDDneg assigns costs C_1 for observations in $\mathcal{U} \cup \mathcal{L}_{in}$ and costs C_2 for $\mathcal{L}_{out}$ where $C_1, C_2 \in (0,1]$. By adding a constraint to SVDD, SVDDneg enforces that labeled observations in $\mathcal{L}_{out}$ fall outside of the decision boundary.

$$\begin{aligned} \text{SVDDneg} : \underset{a,R,\xi}{\text{minimize}} \quad & R^2 + C_1 \cdot \sum_{i:\, x_i \in \mathcal{U}\cup\mathcal{L}_{in}} \xi_i + C_2 \cdot \sum_{i:\, x_i \in \mathcal{L}_{out}} \xi_i \\ \text{subject to} \quad & \|\phi(x_i) - a\|^2 \le R^2 + \xi_i, \; i: x_i \in \mathcal{U} \cup \mathcal{L}_{in} \\ & \|\phi(x_i) - a\|^2 \ge R^2 - \xi_i, \; i: x_i \in \mathcal{L}_{out} \\ & \xi_i \ge 0, \; i = 1, \dots, N. \end{aligned} \tag{2.9}$$

Analogous to SVDD, the dual of SVDDneg only contains inner products of the form $\langle \phi(x_i), \phi(x_j) \rangle$, and one can apply the *kernel trick*.

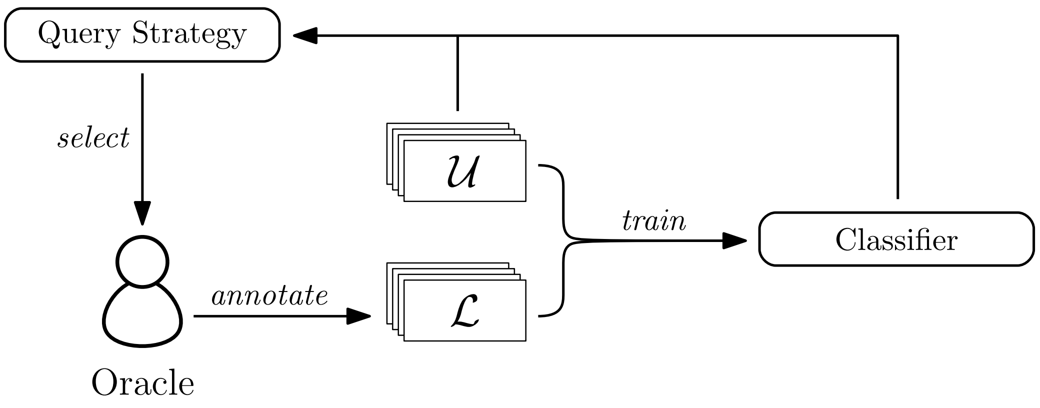


Figure 2.1.: Overview of an active learning system.

2.3. Active Learning

The basic idea of active learning is that a learning algorithm performs better and requires less training if it can *select* the data from which it learns compared to the case where one provides a static training data set [Set12]. The active learning system *selects* so-called *queries* that an external entity, called the *oracle*, *annotates*. Active learning is well-motivated in practice. There often is an abundance of unlabeled data but acquiring the labels is time-consuming and expensive. Example applications are fraud detection, speech recognition, information extraction, or computational biology, see [Set12] and [FZL12] for an overview.

In this section, we first introduce the concept of active learning and explain the individual components. Then, we give an overview of the cost-quality trade-offs an operator faces when configuring a one-class active learning system. Finally, we discuss the specifics of one-class active learning.

2.3.1. Overview and Components

An active learning system consists of four components: the data and label pools, the classifier, the query strategy, and the oracle. Active learning is an iterative process where each iteration consists of the following three steps: First, the system *trains* the classifier with the available data and label pools. Second, the query strategy *selects* the queries. Third, the oracle *annotates* these queries. The system iterates until it reaches some termination criteria. Examples of termination criteria are the convergence of the classifier or the exhaustion of a query budget. Figure 2.1 is an overview of an active learning system.

Data and Label Pools The data and label pools follow our description at the beginning of this chapter in Section 2.1. Note that the data and label pools change throughout the active learning process. The active learning system adds queries with their annotations, i.e., the query labels, to the existing data and updates the label pools. Updating in this case refers to adding the queries to the label pool $\mathcal{L}$. One adds a query x_{in} labeled as inlier to $\mathcal{L}'_{in} = \mathcal{L}_{in} \cup \{x_{\text{in}}\}$ and a query x_{out} labeled as outlier to $\mathcal{L}'_{out} = \mathcal{L}_{out} \cup \{x_{\text{out}}\}$. If a query x is from the unlabeled pool, i.e., $x \in \mathcal{U}$, one updates the pool $\mathcal{U}' = \mathcal{U} \setminus \{x\}$.

Classifier The classifier is a model trained by the active learning system on the available data and label pools. One can describe a classifier C as a mathematical function that maps from the data space to the label space: $\mathcal{X} \mapsto \mathcal{Y}$. In the case of one-class classification, we have formalized this as a *decision function* in Definition 4. The classifier choice for active learning depends on the class label availability. We discuss the classifier choice for one-class classification in Section 2.4.

Query Strategy A *query strategy selects queries* for annotation. Here, in each active learning iteration *queries* are one or multiple observations $x \in \mathcal{X}$. These queries may be existing observations from the pool of unlabeled observations $\mathcal{U}$ or artificially generated ones. We discuss the different query scenarios in Section 2.3.2.

In an active learning system, the query strategy takes the data, the pools and the trained classifier to *select* the *queries*.

Definition 5 (Query Strategy) *Given a data set* $\mathbf{X}$ *with label pools* $\mathcal{U}$, $\mathcal{L}_{in}$, $\mathcal{L}_{out}$, *and a classifier* C, *a query strategy* QS *is a function of type* $QS\colon C, \mathcal{U}, \mathcal{L}_{in}, \mathcal{L}_{out} \rightarrow \mathcal{Q}$. $\mathcal{Q}$ *is a query that contains a set of observations with* $\forall q \in \mathcal{Q}\colon q \in \mathcal{X}$.

Depending on the size of the query chosen by a query strategy, we differentiate between a *sequential* and a *batch* query strategy.

Definition 6 (Sequential Query Strategy) *A* sequential query strategy *is a* query strategy *according to Definition 5 that outputs a single element query* $\mathcal{Q}$ *with* $|\mathcal{Q}| = 1$.

Definition 7 (Batch Query Strategy) *Given a batch size* $k \in \mathbb{N}^+$, *a* batch query strategy *is a* query strategy *according to Definition 5 that outputs a query* $\mathcal{Q}$ *with* $|\mathcal{Q}| = k$.

For the case $k = 1$ a batch query strategy equals a sequential query strategy.

Generally, the goal of a query strategy is to improve the classification quality of the classifier. The annotation of a query is unknown during the query selection. A query strategy has to estimate the "benefit" it expects from obtaining annotations for a query. Literature formalizes the "benefit" with the *informativeness* criterion [TEB21]. Sometimes the *informativeness* is also called the *expected information gain* of a query [Set12].

Definition 8 (Informativeness) *Given a classifier* C *and label sets* $\mathcal{U}$, $\mathcal{L}_{in}$, $\mathcal{L}_{out}$, *the informativeness is a function* $\tau\colon x, C, \mathcal{U}, \mathcal{L}_{in}, \mathcal{L}_{out} \mapsto \mathbb{R}$ *that maps an arbitrary* $x \in \mathcal{X}$ *to* $\mathbb{R}$.

τ quantifies the informativeness, i.e., how valuable an arbitrary observation $x \in \mathcal{X}$ is, and generally higher values are better. As $C, \mathcal{U}, \mathcal{L}_{in}$ and $\mathcal{L}_{out}$ are fixed in a given active learning iteration, we only write $\tau(x)$ for brevity.

There are various ways to quantify the informativeness of a query. The most common idea is *uncertainty sampling* [LC94]. Here, queries have high informativeness if the classifier is unsure about their classification. Other concepts to quantify informativeness are *querying-by-committee*, observing the *expected model change* annotations would yield, or calculating *expected error reduction* queries would give. We refer to [Set12] for an overview. We present query strategies for one-class classification in Section 2.4.

Oracle The *oracle* is the entity that *annotates* a *query* by providing labels. Active learning abstracts from the entity that provides the labels:

Definition 9 (Oracle) *An* oracle *is a function of type $O\colon \mathcal{X} \to \mathcal{Y}$ that annotates an observation $x \in \mathcal{X}$ with a label $y \in \mathcal{Y}$.*

The abstraction of the annotation entity as an oracle is convenient when developing active learning algorithms. However, in some applications it is an oversimplification. The oracle may return wrong labels or cannot annotate a query at all. Oracles returning errors are called *noisy* oracles in literature [Set12]. To cope with a noisy oracle, one requires robust query strategies and classifiers that handle oracle uncertainty [Set11]. For now, we assume that the oracle always returns the ground-truth label. This simplifies the evaluation of active learning systems since we can simulate the annotations. We discuss noisy oracles in future work in Chapter 8.

We further distinguish between a *human* oracle and an *experiment* oracle. With a *human* oracle, an individual receives, inspects, and tries to annotate the queries. With an *experimental* oracle, an external process produces the annotations, e.g., by running an experiment or a computer simulation. Next, annotations may have variable costs depending on the query [Set11]. Some queries may be cheaper to annotate than others. All these aspects of an oracle challenge the application of active learning. We come back to this when discussing the active learning trade-offs later in this chapter.

2.3.2. Query Scenarios

A fundamental categorization of active learning methods is the *query scenario* that defines from which data the query strategy selects the queries. There are three query scenarios [Set12]: pool-based sampling, steam-based selective sampling, and query synthesis.

Pool-based Sampling *Pool-based sampling* assumes that there is a large pool of unlabeled observations $\mathcal{U}$. This pool is usually "closed", i.e., no observations are added to $\mathcal{U}$ during the active learning. A pool-based query strategy selects the queries from this pool. By using the *informativeness* criterion, one can rank the unlabeled observations in $\mathcal{U}$ and query them greedily. Formally a *sequential pool-based* query strategy takes an informativeness function τ and computes the query

$$Q = \operatorname*{arg\,max}_{x \in \mathcal{U}} \tau(x). \tag{2.10}$$

For a *batch* pool-based query strategy, the naive approach is to choose the *TopK* observations with the highest informativeness as a query with a $k \in \mathbb{R}^+$

$$Q = \operatorname*{arg\,max}_{Z \subseteq \mathcal{U},\ |Z|=k} \sum_{x \in Z} \tau(x). \tag{2.11}$$

More sophisticated solutions for pool-based batch query strategy include the diversity of a batch during query selection. We discuss batch queries in Chapter 5.

Stream-based Selective Sampling The second scenario is defined for a continuous stream of unlabeled observations arriving from a data source. For each arriving observation, a *stream-based* query strategy has to decide whether to query or discard it. One cannot query any discarded observation later, it is gone forever.

Query Synthesis The third scenario is *query synthesis*. In this scenario, the query strategy is not restricted by the available observation. Instead, a *query synthesis* strategy can select any query in the data space $\mathcal{X}$. Literature also uses the terms "artificially generated queries" or "queries generated de novo" when describing query synthesis [Set12]. On the one hand, query synthesis gives more freedom to the query selection than *pool-based* query selection because one is not restricted by the pool $\mathcal{U}$. On the other hand, generating these artificial queries is also more difficult. One has to find the best queries in a potentially unbound data space. When working with human oracles, artificially generated queries may be unrealistic, making an annotation hard or even impossible. We formalize the *query synthesis* framework for one-class active learning as one of our contributions in Section 6.1.

2.3.3. Cost-Quality Trade-offs

The overarching goal of active learning is to maximize classification quality while minimizing costs. In practice, there are external constraints for the costs. For example, the monetary costs of active learning must not exceed a monetary budget of a project. The operator must adhere to these budgets when configuring an active learning system. So, in some cases, he may have to accept a reduced quality to cut costs. Thus, there is a trade-off between costs and quality. In this section, we discuss the cost-quality trade-offs of active learning. We first individually explain the various costs and quality dimensions of active learning. We then describe approximation methods to balance both aspects.

2.3.3.1. Costs

We differentiate between two types of costs during active learning: *annotation* costs and *computational* costs. These costs can be measured as temporal and monetary costs.

Annotation Costs The *annotation* cost is the price for querying an oracle for annotations. The annotation costs highly depend on the use case, i.e., the real-world entity that produces the labels. There are two aspects of the annotation costs that one must consider during query selection. For the first aspect, we differentiate whether the annotation costs are *static* or *variable* for each observation. Annotations costs are variable when they depend on the complexity of the query, e.g., the costs may linearly depend on the query length. The second aspect is that annotation costs can be *known* or *unknown* [Set11]. An active learning system can consider known annotation costs during query selection. For example, annotation costs are unknown when the oracle is a simulation where the runtime is unknown in advance. When annotation costs are unknown, one can try to predict them, e.g., by learning a regression cost-model alongside the active learning system [Set11].

Computational Costs We discern between four *computational* costs for *classifier training*, *query selection*, *query presentation*, and *data flow management*.

First, there is the *classifier training cost* for learning the classifier. The cost depends on the type of classifier, i.e., its training complexity. For SVDD and SVDDneg, the classifier training complexity to solve the QP problem is in $O(N^3)$ [CTK04], i.e., it only depends on the size of the training data set.

Second, there is the *query selection cost*, i.e., the cost for choosing a query. While the exact cost varies between the query strategies, the cost generally depends on the number of considered *query candidates*. For example, the cost of a pool-based query selection depends on the number of unlabeled observations $|\mathcal{U}|$. All observations in $\mathcal{U}$ are *query candidates*. Then, for each query candidate, the query strategy evaluates the informativeness function. Some query strategies additionally require the evaluation of the decision function of the classifier for each query candidate.

Third, there is the *query presentation cost* to present a query to the oracle. This cost only appears when working with a *human* oracle. The system presents the selected query to the human in the *annotation interface*. Usually, it is not enough to present the query in raw form, e.g., the numeric vector, because it is too difficult for a human to annotate. Instead, the active learning system computes additional contextual and explanatory information to support the human during the annotation [TEB19]. An example is the visualization of the classifier decision function and data set in a plot. Computing these additional elements results in a query presentation cost.

Fourth, there are the costs to manage the *data flow* [TEB19]. These costs depend on the software architecture and implementation of the active learning system. The system must transfer the data set to the classifier and query strategy. Some query strategies require the decision function of the classifier. So there is direct communication between both components. When working with human oracles, the system has to provide data and visualizations to the annotation interface. Finally, the system has to store all annotations acquired during active learning. Such data flows incur costs for transferring and storing data. While literature mostly neglects the question of how to implement and optimize the data flow, we have identified that they are crucial when developing an active learning system in practice, see our previous work in [TEB19].

Budgets In practice, there are external constraints for the costs. We differentiate between two types of budgets. On the one hand, there is a *global* budget, such as a fixed project deadline or a monetary limit. The *global* budget limits the overall active learning process, i.e., it gives a maximum that the sum of all costs over all the active learning iterations cannot exceed. On the other hand, there are more fine-granular *local* budgets. They are externally given as well and apply to individual steps during one active learning iteration. An example is the downtime between two annotations, which may not exceed a threshold that guarantees that an annotator has low idle times. Budgets are restrictions that the operator has to adhere to when configuring an active learning system.

2.3.3.2. Quality

The quality of active learning depends on the learning objective. The prevalent objective is to improve the prediction quality of the classifier. In this case, one can quantify the *classification quality* of active learning by evaluating the classification on independent test data. To do so, one computes an evaluation metric suitable for the given data distribution. The evaluation metric quantifies the quality of the active learning progress at a given iteration. We discuss evaluation metrics for active learning systems in Section 2.5.2 and continue with the abstract discussion here.

A second aspect to consider is the *annotation quality* that is the correctness of the answers of the oracle. Annotation quality is high if the annotations correspond to the ground truth. Typically, there is a strong dependency between *classification quality* and *annotation quality*. The capabilities of the classifier determine the correlation between classification and annotation quality. On the one hand, the classifier may have limited capability and thus cannot consider all labels. Then a high annotation quality may still result in a low classification quality. On the other hand, a robust classifier internally corrects incorrect annotations. Thus, classification quality may still be high if there are many low-quality annotations. As we have discussed earlier, we focus on oracles that return correct annotations, for now, we refer to the classification quality when we speak of the quality of an active learning system in the following. We come back to the annotation quality in our outlook in Chapter 8.

2.3.3.3. Approximation methods

So far, we have described the two aspects, costs, and quality of active learning, independently. However, in an active learning system, one must jointly consider them. There are cases where one may have to trade off costs against quality. A reason is that the system exceeds a budget and there is no alternative algorithm with the same efficiency, and the budget cannot be changed. One way to deal with this is to accept a decrease in quality to cut costs. We call methods that trade costs against quality *approximations*. We use the term “approximation” because they are lossy, i.e., they reduce the costs and may also reduce the quality. During the configuration of a system, the operator may use multiple approximation methods to trade off costs against quality.

Note that an alternative method to reduce costs without changing the quality is not an approximation, and we call it a *lossless alternative*. The operator can add them to the active learning system without any effect on the quality, i.e., they are free. For instance, there may be a more efficient solver for the QP problem of an SVDD that produces a solution of the same quality as the original solver. The operator should always pick the best lossless alternative when configuring a system initially.

The literature proposes a plethora of approximation methods to reduce the costs of active learning. These methods target all components of the system. However, there is no overview over all the different methods one may use when configuring a system. In the following, we explain the basics of approximation methods by categorizing them by their underlying concept of how they work. We review literature on approximation methods for outlier detection with active learning later in Chapter 3.

Sampling The first concept is *sampling*. Sampling selects a subset of the data as the input for a subsequent algorithm. For algorithms that depend on the input data size, sampling reduces computational runtime. Thus, sampling reduces the classifier training costs. One can also use sampling during query selection. Selecting a subset of query candidates reduces the query selection costs. However, sampling may reduce active learning quality since one does not use not all available data.

Projections The second concept are *projections*. While sampling targets the data set size, projections reduce the dimensionality of the data. Projections, for instance, reduce the search space when generating artificial queries. In an annotation interface, projections can reduce the complexity of the visualizations for a human. A downside of using projections is that complex dependencies may not be present anymore, which in turn reduces the annotation quality.

Delaying/Skipping The third concept is *delaying* or *skipping*. Methods of this category delay or skip the execution of specific steps of an active learning system. An example is training the classifier after acquiring annotations for multiple queries instead of retraining it after each annotation. Active learning components that depend on the classifier may then use the last available trained classifier, e.g., for query selection or visualization. However, using outdated information may reduce the active learning quality. For instance, a query strategy would have selected a different query if the classifier had been updated with all annotations.

Updating The fourth concept is *updating*. Instead of rerunning an algorithm every iteration from scratch, updating reuses the output of the previous iteration. One can use the concept updating to incrementally train a classifier with new annotations. But update mechanisms may yield suboptimal solutions. For instance, retraining a classifier on all available data may yield a higher classification quality than incrementally updating.

Overall, the concrete trade-off the operator takes when configuring an active learning system depends on the actual data set and the specific algorithms. Approximation methods that come with recommendations or guarantees for the cost-quality trade-off facilitate the configuration for the operator. For instance, a guarantee for a sampling algorithm could be that it halves the runtime while the classification quality does not deteriorate by more than 5%. A central topic of this thesis is how one can derive such recommendations and guarantees from theoretical investigations or empirical studies.

2.4. Active Learning for Outlier Detection

In the previous two sections, we have introduced the two distinct research areas, outlier detection, and active learning. One-Class Active Learning (OCAL) for outlier detection brings both areas together. We have given an overview and evaluated state-of-the-art methods for OCAL in a comprehensive benchmark in a previous work of ours [TEB21].

Here, we only summarize the most important parts for this thesis, namely the learning scenario, and the choice of classifiers and query strategies for OCAL.

2.4.1. Learning Scenario

The learning scenario consists of the different assumptions researchers make regarding the application and the interaction with the oracle. Literature on OCAL often does not explicitly state these assumptions, which makes it difficult to configure an active learning system for a new application. We have identified the following assumptions that restrict the applicability of classifiers and query strategies in [TEB21].

Class Distribution One-class classification allows learning one concept in an imbalanced data sets. We distinguish between two cases. In the first case, the target concept is the "minority class", one groups all other classes to form the outlier class. Literature calls this the one-vs-all multi-class classification task [Gha+11a; JD03]. In the second case, the target concept is the "majority class", and all other observations are unusual. This second case is typical for outlier detection and what we assume in this thesis.

Learning Objective The most common learning objective of an active learning system is to improve the prediction quality of the classifier. Literature proposes to deviate from this objective when there is a specific interest in one of the two classes. In that case, one may prefer queries that the system expects to be inliers [Gha+11a; Gha+11b] or outliers [Das+16]. In this thesis, we do not assume any class preference during the query selection.

Initial Pools The label information available at the beginning of the active learning defines the initial pools. We can distinguish between the case that (1) all observations are unlabeled and that (2) there are some labeled observations available. In the second case, the amount of available labels depends on the use case. Additionally, some query strategies technically require a certain amount of labels, e.g., to avoid a singular correlation matrix. Acquiring these initial labels results in annotation costs. Since active learning is specifically designed to acquire labels cost-efficiently, case (1) where all observations are unlabeled is more common. So in this thesis, we assume initially unlabeled pools.

2.4.2. Classifier

For OCAL, we use a one-class classifier that discerns between the inliers and outliers by learning a decision function. We have reviewed one-class classifiers for outlier detection in Section 2.2.3. There is an essential difference between *unsupervised* and *semi-supervised* classifiers and how they can use the annotations acquired during active learning. *Unsupervised* classifiers work without labels and require an additional mechanism to use the label information. One way is to only expose them to a subset of the data set during training, e.g., only to the labeled inliers. Another way is to tune the parameters of the classifier by evaluating the classifier predictions on the available labels. *Semi-supervised* classifiers, in turn, do not require an additional mechanism. They can explicitly use both unlabeled and labeled data for training.

Existing works on OCAL for outlier detection use the unsupervised SVDD [TD04], semi-supervised SVDDneg [TD04] that uses labeled outliers, and semi-supervised Semi-Supervised Anomaly Detection (SSAD) [Gör+13] that uses labels from both classes. We have introduced SVDD and SVDDneg in Section 2.2.3. SSAD extends SVDDneg with additional constraints to use labeled inliers. SSAD adds a hyperparameter to balance the weight between the two classes. Our benchmark shows that tuning this hyperparameter is difficult and SSAD does perform better than SVDDneg [TEB21]. Therefore, we only use SVDD and SVDDneg in this thesis.

2.4.3. Query Strategies

All query strategies that literature proposes for OCAL are pool-based query strategies. We have given an overview of them and have evaluated them against each other in our benchmark [TEB21] and only provide a summary here. We propose a framework for query synthesis for OCAL in Section 6.1.

Following [TEB21], we categorize the query strategies into *data-based*, *model-based*, and *hybrid* query strategies. *Data-based* query strategies select queries independent of the classifier, i.e., solely based on the available data and label pools $\mathcal{U}$ and $\mathcal{L}$. *Model-based* query strategies directly use the decision function of the one-class classifier to select queries. Finally, *hybrid* query strategies combine both approaches by using the classifier information and data statistics.

Our overview and evaluation in [TEB21] shows that *model-based* query strategies are most flexible, i.e., they work with any initial pool. Additionally, they outperform the other strategies in the experimental comparison. In this thesis, we use the two *model-based* query strategies proposed in literature: *decision-boundary* and *high-confidence* querying. Recall that f is the decision function of a one-class classifier, which returns $f(x_{\text{out}}) > 0$ for outliers and $f(x_{\text{in}}) \leq 0$ for inliers.

Decision Boundary Querying [Gör+09] This query strategy selects observations closest to the decision boundary. The informativeness function is

$$\tau_{\text{DB}} = -|f(x)|. \tag{2.12}$$

High-Confidence Querying [BBJ15] This query strategy selects observations that match the inlier class the least. Given the decision function f, the informativeness is

$$\tau_{\text{HC}} = f(x). \tag{2.13}$$

2.5. Evaluating One-Class Active Learning

In this section, we discuss the evaluation of one-class active learning. We present the data sets and metrics we use in this thesis.

Table 2.1.: Overview over the benchmark data sets published in [Cam+16].

Data set	Observations (N)	Dimensions (M)	p_{out}
ALOI	49534	27	0.0304
Annthyroid	7129	21	0.0749
Arrhythmia	450	259	0.4578
Cardiotocography	2114	21	0.2204
Glass	214	7	0.0421
HeartDisease	270	13	0.4444
Hepatitis	80	19	0.1625
InternetAds	1966	1555	0.1872
Ionosphere	351	32	0.3590
KDDCup99	48113	40	0.0042
Lymphography	148	3	0.0405
PageBlocks	5393	10	0.0946
Parkinson	195	22	0.7538
PenDigits	9868	16	0.0020
Pima	768	8	0.3490
Shuttle	1013	9	0.0128
SpamBase	4207	57	0.3991
Stamps	340	9	0.0912
WBC	223	9	0.0448
WDBC	367	30	0.0272
WPBC	198	33	0.2374
Waveform	3443	21	0.0290
Wilt	4819	5	0.0533

2.5.1. Benchmark Data

To evaluate the classification quality of OCAL, we require labeled outlier detection data sets. In this thesis, we run experiments on real-world and synthetic data.

Real-world data There only exist few labeled real-world data sets for outlier detection. One common solution in literature is to resample binary or multi-class classification data sets [Cam+16; Dom+18; Emm+13]. One class is selected as the inlier class and all other classes are downsampled and grouped to form the outlier class. In this thesis, we work with the well-established benchmark suit published in [Cam+16]. Table 2.1 shows the data sets of the benchmark. The benchmark contains data set from various applications, and the data sets have different sizes (80 to 49 534 observations), dimensionality (3 to 1555 dimensions), and outlier ratios p_{out} (0.2 % to 75.38 %, median 9.12 %). The size and diversity of the benchmark data sets enable us to study the performance of our contributions in comprehensive experiments.

Synthetic data We also generate synthetic data sets in this thesis. Synthetic data sets allow us to study specific desired properties of the outlier detection method. We can control all the characteristics of a data set during the generation, e.g., the size, the number of dimensions, or the complexity of a data set. In this thesis, we generate small 2-dimensional data sets to visualize and explain how our algorithms work. Additionally, we generate synthetic data sets to evaluate how well algorithms scale with increasing data set size, dimensionality, and complexity. We introduce the generation process in the corresponding subsections in Section 4.2 and Section 6.4.

2.5.2. Metrics

To evaluate our experiments and quantify the quality of an OCAL system, we require evaluation metrics. The metrics must cope with the imbalance of outlier detection data sets. We first present the metric we use to evaluate a one-class classifier and then what we use to evaluate an active learning system.

One-Class Classification One metric robust for evaluating a binary classification is the Matthews Correlation Coefficient (MCC). MCC is robust against the imbalance of a data set. Given a confusion matrix with true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), one calculates

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}. \tag{2.14}$$

The value range of MCC is $[-1, 1]$. A perfect classification results in a MCC value of 1 since $FP = FN = 0$. A completely wrong prediction, i.e., predicting the inverse label for each observation, results in a MCC value of -1. With a MCC value of 0, the classification is not better than random.

Active Learning Evaluating active learning is more involved than a static evaluation of a classifier. During active learning, we can evaluate the classifier after each iteration. Instead of a single value like in a static setting, we then have a sequence of values, which we call an *active learning progress curve* [TEB21].

Visually comparing the performance of different active learning systems does not scale with the number of experiments. Manually comparing hundreds of progress curves is prohibitive. To facilitate extensive evaluations, one can summarize a progress curve. We have studied summary statistics of progress curves in a previous work of ours in [TEB21]. There, we discuss summary statistics proposed in literature and develop new ones to capture some characteristic of the learning progress. For example, we have distinct metrics to capture the quality increase at the start of the training or the convergence by calculating how stable the classification quality is over multiple iterations.

In this thesis, we focus on the *End Quality (EQ)* of an active learning system. We calculate *EQ* by evaluating the classification quality after the system has exhausted the budget and updated the classifier with the last annotation from the oracle. As an evaluation metric for the classification quality, one may use any quality metric. Since we are evaluating one-class classifiers, we use the previously presented MCC.

3. Related Work

In this chapter, we review the related work for our contributions. In the first part, we discuss approximation methods that speed up the SVDD classifier. In the second part, we review work on active learning with query synthesis for outlier detection. The first part is extracted from [Eng+20a] (Section 3) and the second part is extracted from [EB20] (Section 2).

3.1. Speed up the SVDD Classifier

Training SVDD requires solving a quadratic problem (QP). The time complexity of solving SVDD is in $O(N^3)$ [CTK04]. Thus, training does not scale well to large data sets. However, the time complexity for classification is only linear in the number of support vectors $O(|SV|)$. So for large N, training time is much larger than classification time. Still, long classification times may be an issue, e.g., in time-critical applications. So curbing the runtimes has long become an important topic in the SVDD literature. In Section 3.1.1, we categorize existing approaches that focus on SVDD speedup, see Figure 3.1 for an overview. In Section 3.1.2, we then turn to *Sampling*, the category our contribution belongs to.

3.1.1. Categorization

We distinguish between *Fast Training* and *Fast Classification*.

Fast Training To speed up training of SVDD, one has two options: reduction of the problem size and optimization of the solver. For *Reduction*, one can distinguish further: A first type reduces the number of observations by *Sampling* [Ala+20; HZH14; Kra+18; Li+18; Li+19; Li11; Qu+19; Sun+16; Xia+14; Zhu+14]. A second type reduces the size of the *Kernel matrix*, e.g., by approximation [AMS02; FS02; NHJ08; Sch+00]. Examples are the Nyström-method [WS01] and choosing random Fourier features [Yan+12].

Optimization, on the other hand, decomposes the QP into smaller chunks that can be solved efficiently. Literature features methods that decompose with clustering [Kim+07] and with multiple random subsets [Cha+18]. The most widely used decomposition methods are sequential minimal optimization (SMO) [Pla98] and its variants. These methods iteratively divide SVDD into small QP sub-problems and solve them analytically. Finally, there is a core-set method that expands the decision boundary by iteratively updating an SVDD solution [CL19; CTK04]. Core-set approaches are $(1 + \varepsilon)$ approximations, i.e., they may not find the exact decision boundary, given training data.

Reduction and Optimization are orthogonal to each other. Thus, one can use problem-size reduction in a *pre-processing* step before solving SVDD efficiently.

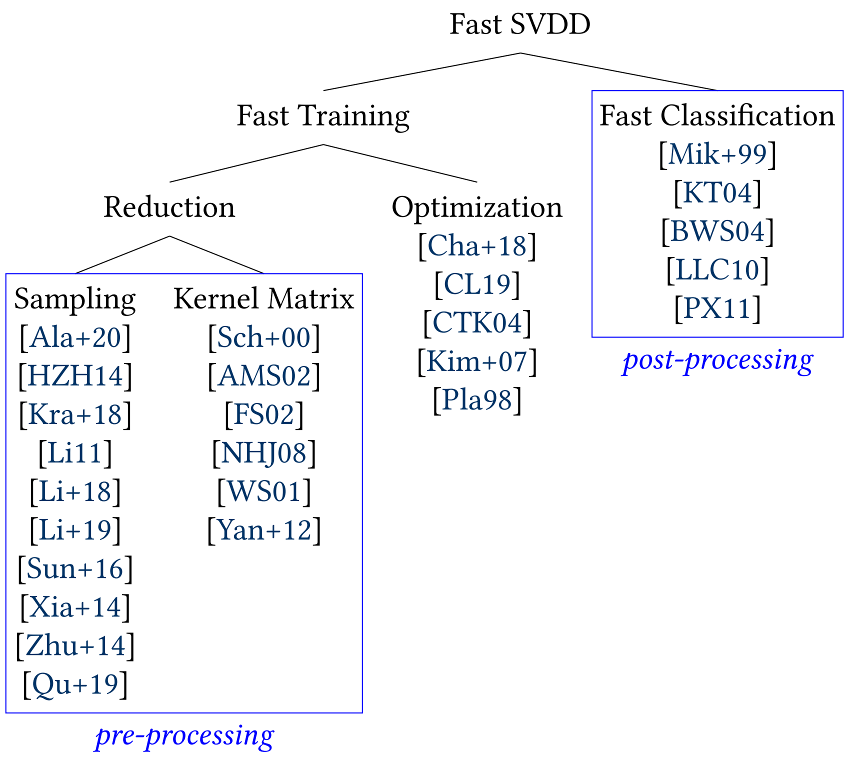


Figure 3.1.: Categorization of literature on SVDD speedup.

Fast Classification When SVDD uses a non-linear kernel, one cannot compute the pre-image of the center a. Instead, one must compute the distance of an observation to a by a linear combination of the support vectors in the kernel space, see Equation 2.7. However, literature proposes several approaches to approximate the pre-image of a [BWS04; KT04; LLC10; Mik+99; PX11]. With this, classification no longer depends on the support vectors, and is in $O(1)$. Fast Classification is orthogonal to Fast Training, i.e., it can come as a *post-processing* step, after training.

3.1.2. Sampling Methods

Sampling methods take the original data set $\mathbf{X}$ as an input and produce a sample $\mathbf{S}$. All existing sampling methods for SVDD assume the *target-only scenario*, i.e., all observations in $\mathbf{X}$ are from the target class. This is equivalent to a supervised setting where one knows the ground truth and $\mathbf{Y} = \langle \text{in}, \text{in}, \ldots, \text{in} \rangle$. Thus, one must modify these methods to apply them in the outlier scenario, see Section 4.1.1. We categorize them into different types: *Edge-point* detectors, *Pruning* methods, and *Others*. Table 3.1 provides an overview.

Edge-point Most sampling approaches focus on selecting observations that demarcate the estimated inlier density from the outlier one, and therefore are expected to be support vectors. Such observations are called “edge points” or “boundary points”. Literature proposes various ways to identify edge points. One idea is to use the angle between an

Table 3.1.: Sampling methods proposed for SVDD.

Method	Publication	Year	Exogenous Parameters*
BPS	Li [Li11]	2011	k= $\lfloor 10 \ln N \rfloor$, ε=0.05
DAEDS	Hu, Zhou, and Hu [HZH14]	2014	k=30, ε=0.1, δ=0.3
DBSRSVDD	Li et al. [Li+19]	2019	$minPts$=7, ε=0.5
FBPE	Alam et al. [Ala+20]	2020	n=360
HSR	Sun et al. [Sun+16]	2016	k=20, ε=0.01 $\cdot M$
HSC†	Qu et al. [Qu+19]	2019	k=20
IESRSVDD	Li et al. [Li+18]	2018	ε=0.5
KFNCBD	Xiao et al. [Xia+14]	2014	k=100, ε=0.2
NDPSR	Zhu et al. [Zhu+14]	2014	k=20, ε=10
OCSFLSDE†	Krawczyk et al. [Kra+18]	2018	8 different parameters

* The listed values for the exogenous parameters are the ones used in our experiments in Section 4.2.

† Not included in our experiments, see Section 4.2.1 for details.

observation and its k nearest neighbors [Li11; Zhu+14] as an indication. An observation is selected as edge point if most of its neighbors lie within a small, convex cone with the observation as the apex. One has to specify a threshold for the share of neighbors and the width of the cone [Li11] as exogenous parameters. Others suggest to identify edge points through a farthest neighbor search. For instance, one suggestion is to first sort the observations by decreasing distance to its k-farthest neighbors (KFN) [Xia+14], and then select the top ε percent as edge points. The rationale presented in the paper is that inner points are expected to have a lower KFN distance than edge points. A more recent variant uses angle-based search [Ala+20]. The idea of the paper is to initialize the method by the mean over all observations as the apex and divide the space into a pre-specified number of cones. For each cone, one only keeps the farthest observation as edge point.

Next, there are methods that select edge points by density-based outlier rankings, e.g., DBSCAN [Li+19] and LOF [HZH14]. Here, the assumption is that edge points occur in sparse regions of the data space. A similar idea is to rank observations with a high distance to all other observations [Li+18]. Others have suggested to rank observation highly if they have low density and a large distance to high-density observations [Qu+19]. Naturally, ranking methods require to set a cutoff value to distinguish edge points from other observations.

Pruning The idea of pruning is to iteratively remove observations from high-density regions as long as the sample remains “density-connected”. One way to achieve this is by pruning all neighbors of an observation closer than a minimum distance, starting from the observation closest to the cluster mean [Sun+16]. However, this approach requires to set the minimum distance threshold, and a good choice is data-dependent.

Others There is one method that differs significantly from the other ones [Kra+18]. The basic idea is to generate artificial outliers to transform the problem into a binary classification problem. Based on the augmented data, one can apply conventional sampling methods such as binary instance reduction. The sampling method then relies on an evolutionary algorithm where the fitness function is the prediction quality on the augmented data. Finally, the method only retains the remaining inliers and discards all artificial observations. However, this requires solving many SVDD instances in each iteration.

To summarize, there are many methods to select a sample for SVDD. However, they are based upon some intuition regarding the SVDD and do not come with any formal guarantee for the resulting classification quality when training on the sample. Edge point detectors, in particular, return an insufficient sample in some cases since they do not guarantee the coherence of a selected sample, as we will see in Chapter 4. Further, all existing approaches require to set some exogenous parameter. But the influence of the parameter values on the sample is difficult to grasp. Finally, existing sampling methods are designed for the *target-only scenario*. It is unclear whether they work well with the *outlier scenario*.

3.2. Query Synthesis

In this section, we review work on multi-class query synthesis and one-class active learning. We also discuss how artificial outlier generation, adversarial attacks, and experimental design are related to one-class query synthesis.

Multi-class Query Synthesis Research has proposed several approaches for query synthesis in a binary classification setting. One uses observation pairs that belong to opposite classes to synthesize queries along the decision boundary [Wan+15]. Two others choose queries that shrink the version space of potential decision boundaries [AGZ15; CHK17]. Another option is to cluster the data and synthesize a query between the centroids of the clusters [HWY12; Jos11]. All these approaches require negative examples, which may be missing in a one-class setting. Next, previous work features handcrafted query synthesis strategies, e.g., from biology [Kin+04; Kin+09], that do not generalize to other domains.

One-Class Active Learning As we have discussed in Section 2.4.3, all existing one-class approaches are pool-based query strategies. They compute the informativeness for all unlabeled observations and then query the best one. Following the categorization in [TEB21], there are three types of one-class query strategies. The first type, data-based query strategies, select queries independent of the classifier only based on data characteristics such as densities [Gha+11a; Gha+11b]. The other two types of strategies base on a one-class classifier that learns a decision boundary, e.g., support vector data description (SVDD) [TD04]. Model-based strategies query the unlabeled observation closest to [Gör+09] or farthest from [BBJ15] the learned decision boundary. Without unlabeled observations, it is unclear what a query far away from the decision boundary would be. We will use the idea of querying observations that lie on the decision boundary and combine it with query synthesis.

The last category of query strategies are hybrid strategies that combine the distance to the decision boundary with neighborhood information [Gör+13; YWF18]. Data-based and hybrid query strategies are not suitable for query synthesis. The generated queries are inserted back into the original data set. These queries then misguide classifiers and query strategies that work with densities and neighborhoods.

Artificial Outliers Another area where observations are synthesized in the context of one-class learning is artificial outlier generation. Artificial outliers are used to transform an unsupervised problem to a supervised one [AZL06; BKB07; Dés+13]. They allow to balance a classification problem for, say, outlier detection. One can then use traditional binary classifiers such as a SVM [BKB07]. Literature also suggests to tune the hyper-parameters of one-class classifiers with synthetic outliers [TD01; Wan+09; Wan+18]. In both use cases, the goal is to train a classifier with a high outlier detection rate. However, algorithms for artificial outlier generation are not designed for active learning since they are independent of the classifier.

Adverserial Attacks Query synthesis is also used in the context of adversarial attacks. Here, an attacker seeks to evade the detection by a classifier while changing his malfeasance only minimally. There are approaches that try to reconstruct the decision boundary [LM05; SK18] or search for an attack instance close to the desired malfeasance that the classifier does not detect [Nel+12]. Adversarial query strategies assume a fixed classifier, i.e., they do not consider feedback from an oracle.

Design of Experiments Engineers perform query synthesis to explore the experimental design space. The literature distinguishes between *exploration* of this space, i.e., finding new feasible regions, and *exploitation*, the refinement in areas with existing observations [CF17a]. Several approaches take a bounded design space and perform adaptive sampling [BM08; Bry+06; LJ08; LM12; RP11]. One other approach exists that explores an unbounded design space [CF17a]. Existing work then trains a surrogate model on the obtained labels – commonly a Gaussian process classifier [Bry+06; CF17a; LJ08; LM12] or a SVM [BM08; RP11]. Both models are not applicable since they are binary classifiers and require labels of both classes.

Part II.

Cost-Quality Trade-Offs in One-Class Active Learning

4. Efficient SVDD Sampling with Approximation Guarantees for the Decision Boundary

In the first part of this thesis, we have introduced the basics of one-class active learning. One of the issues that we have discussed is high classification training costs. In this chapter, we take a closer look at the cost-quality trade-off one faces with training Support Vector Data Description (SVDD).[1] SVDD is the most popular and actively research one-class classifier for anomaly and novelty detection [LLC10; TD04; TEB21]. Recall from Section 2.2.3 that SVDD is an unsupervised classifier that fits a tight hypersphere around the majority of observations, the inliers, to distinguish them from irregular observations, the outliers. Despite its resounding success, a downside is that SVDD and its progeny do not scale well with data size [TEB19]. Even efficient solvers like decomposition methods [Cha+18; CTK04; Kim+07; Pla98] result in training times prohibitive for many applications. In these cases, sampling for data reduction is essential [Ala+20; HZH14; Kra+18; Li+18; Li+19; Li11; Qu+19; Sun+16; Xia+14; Zhu+14].

One of the defining characteristics of SVDD is that only a few observations, the support vectors, define a decision boundary. Thus, a good sample is one for which SVDD selects support vectors similar to the original ones, i.e., the ones obtained on the complete data set. This has spurred the design of sampling methods that try to identify support-vector candidates in the original data, to retain them in the sample [Ala+20; HZH14; Li+18; Li+19; Li11; Qu+19; Xia+14; Zhu+14]. A common approach is to select so-called "boundary points" as support-vector candidates, e.g., observations that are dissimilar [Li11; Zhu+14].

But calibrating existing methods such that they indeed identify boundary points is difficult. A reason is that the sample they return depends significantly on the choice of exogenous parameters, and selecting suitable parameter values is not intuitive (see Section 4.2). A further shortcoming is that including all boundary points in a sample does not guarantee SVDD training to indeed yield the original support vectors. The issue is that selection of support vectors hinges on other aspects, such as the ratio between inliers and outliers in the sample and a sufficient number of non-boundary observations in the sample. Disregarding them may, for instance, fragment contiguous inlier regions and yield wrong outlier classifications after sampling, see Figure 4.1. The influence of these aspects on SVDD is known, but their effects on sample selection are not well studied. It is an open question how to select a sample where SVDD indeed approximates the original

[1] The remainder of this chapter bases on the article [Eng+20a]: Adrian Englhardt et al. "Efficient SVDD Sampling with Approximation Guarantees for the Decision Boundary". In: *arXiv preprint arXiv:2009.13853* (2020). We have shortened the text to be less repetitive, applied minor corrections, and changed the formatting and notation so that it is in line with the format and structure of this thesis.

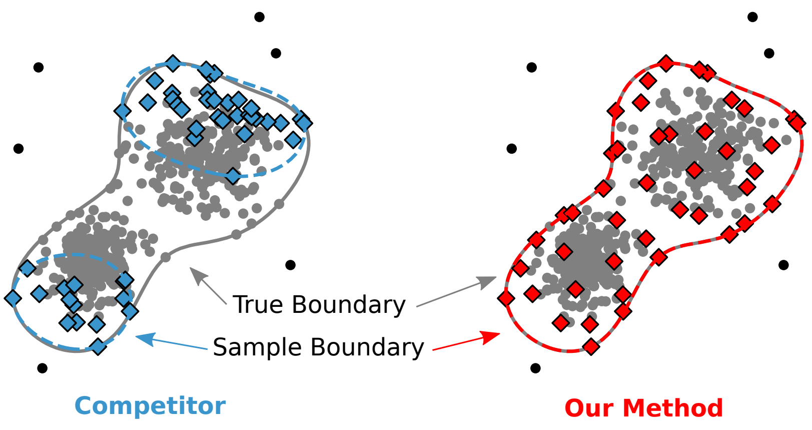


Figure 4.1.: Sample and decision boundary of the state-of-the-art boundary-point method FBPE [Ala+20] and of our method RAPID.

decision boundary. Finally, a point largely orthogonal to these issues is that there also is very limited experimental comparison among competitors. This makes an empirical selection of suitable SVDD sampling methods difficult as well.

Contributions In this chapter, we propose a novel way to SVDD sampling. We make three contributions. First, we reduce SVDD sampling to a decision-theoretic problem of separating data using empirical density values. Based on this reduction, we formulate SVDD sampling as a constrained optimization problem. Its objective is to find a minimal sample where the density of all observations of the data set is close-to-uniform. We provide theoretical justification that a sample obtained in this way i) prevents a fragmentation of the inlier regions, and ii) retains the observations necessary to identify the original support vectors.

Second, we propose RAPID, an efficient algorithm to solve the optimization. RAPID is the first SVDD sampling algorithm with theoretical guarantees on retaining the original decision boundaries. RAPID does not require any parameters in addition to the ones already required by SVDD. This lets RAPID stand out from existing methods, which all hinge on mostly unintuitive, exogenous parameters. RAPID further is easy to implement, and scales well to very large data sets.

Third, we conduct the – by far – most comprehensive comparison of SVDD sampling methods. We compare RAPID against 8 methods on 23 real-world and 85 synthetic data sets. In all experiments, RAPID consistently produces a small sample with high classification quality. Overall, RAPID outperforms all of its competitors in the trade-off between runtime, sample size, and classification accuracy, often by one order of magnitude.

4.1. Density-based Sampling for SVDD

In this section, we present an efficient and effective sampling method for scaling SVDD to very large data sets. In a nutshell, we exploit that an SVDD decision boundary is in fact a level-set estimate [VV06], and that inliers are a super-level set. The idea behind our

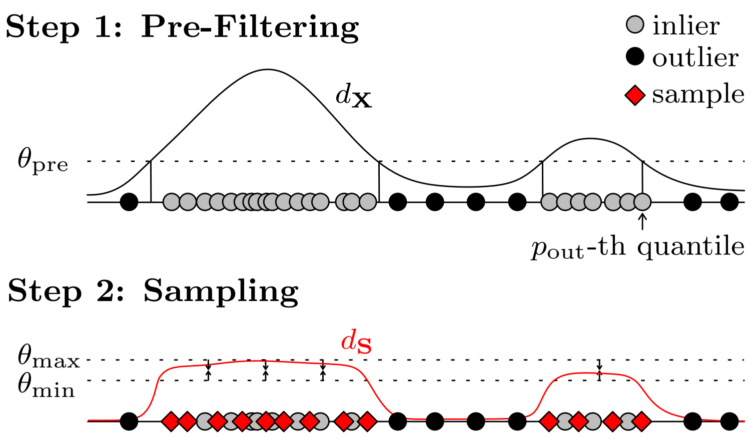


Figure 4.2.: The idea of density-based sampling for SVDD.

Algorithm 1: Pre-filtering

```
Input  : Data set $\mathbf{X} \in \mathbb{R}^{N \times M}$, Kernel function $k(x_i, x_j)$,
         Outlier percentage $p_{\text{out}} \in [0, 1]$
Output : Indices for inliers $\mathcal{I}$ and outliers $\mathcal{O}$, density $d$
1 $d = \langle \sum_{j=1}^{N} k(x_1, x_j), \dots, \sum_{j=1}^{N} k(x_N, x_j) \rangle$        ▹ $\mathcal{O}(N^2)$
2 $\theta_{\text{pre}} = \text{sort-ascending}(d)_{\lfloor p_{\text{out}} \cdot N \rfloor}$        ▹ $\mathcal{O}(N \log N)$
3 $\mathcal{I} = \{i \mid i \in \{1, \dots, N\},\ d_i \geq \theta_{\text{pre}}\}$        ▹ $\mathcal{O}(N)$
4 $\mathcal{O} = \{i \mid i \in \{1, \dots, N\}\} \setminus \mathcal{I}$        ▹ $\mathcal{O}(1)$
5 $d = d - \langle \sum_{j \in \mathcal{O}} k(x_1, x_j), \dots, \sum_{j \in \mathcal{O}} k(x_N, x_j) \rangle$        ▹ $\mathcal{O}(N^2)$
6 return $\mathcal{I}, \mathcal{O}, d$
```

sampling method is to remove observations from a data set such that the inlier super-level set does not change. To this end, we show that for the Gaussian kernel, *the super-level set of inliers does not change as long as not-selected observations have a higher density than the minimum density of selected observations.* If a sample violates the *density rule*, sampling may produce “gaps”, i.e., regions of inliers that become regions of outliers. Such gaps curb the SVDD quality. Thus, we strive for a sample of the minimal size that satisfies the density rule.

Figure 4.2 illustrates our approach. In a first step, we separate the unlabeled data into outlier and inlier regions based on their empirical density, see Section 4.1.1. We then frame sample selection as an optimization problem where the constraints enforce the density rule in Section 4.1.2. In Section 4.1.3 we propose RAPID, an efficient and easy-to-implement algorithm to solve the optimization problem. RAPID returns a small sample with close-to-uniform density, i.e., a small sample that still obeys the density rule, and also contains the boundary points of the original data.

4.1.1. Density-based Pre-Filtering

Any sampling method faces an inherent trade-off: reducing the size of the data as much as possible while maintaining a good classification on the sample. Recall that formally a *sample* is a subset $\mathbf{S} \subseteq \mathbf{X}$ of the data set and that we denote $x \in \mathbf{S}$ as *selected*, and $x \notin \mathbf{S}$ as *not-selected* observations. This allows us to frame sampling an optimization problem

$$\begin{aligned} \underset{\mathbf{S} \subseteq \mathbf{X}}{\text{minimize}} \quad & |\mathbf{S}| \\ \text{subject to} \quad & \mathit{diff}(f^{\mathbf{S}}, f^{\mathbf{X}}) \leq \varepsilon, \end{aligned} \tag{4.1}$$

where *diff* is a similarity between two decision functions and ε a tolerable deterioration in accuracy. Solving Optimization Problem 4.1 requires knowledge of $f^{\mathbf{X}}$. But obtaining this knowledge is infeasible. The reason is that $|\mathbf{X}|$ is too large to solve — SVDD would not need any sampling in the first place otherwise. Thus, one cannot infer which observations $f^{\mathbf{X}}$ classifies as inlier or outlier. However, we know that the SVDD hyperparameter C defines a lower bound on the share of observations predicted as outliers in the training data [TD04]. A special case is if $C = 1$, since $f^{\mathbf{X}}(x;\ C = 1) = \text{in}, \forall x \in \mathbf{X}$. Recall that this is the upper bound of the cost parameter C where SVDD degenerates to a hard-margin classifier, cf. Section 2.2.3. In this case, *diff* is zero if SVDD trained on $\mathbf{S}$, i.e., $f^{\mathbf{S}}$, also includes all observations within the hypersphere. Further, we can make use of the following characteristic of SVDD.

Characteristic 1 (SVDD Level-Set Estimator) *SVDD is a consistent level set estimator for the Gaussian kernel [VV06].*

In consequence, inliers form a super-level set with respect to the decision boundary. Formally, this means that there exists a level set L_θ and a corresponding level-set classifier $g^{\mathbf{X}}_\theta$ such that $g^{\mathbf{X}}_\theta \equiv f^{\mathbf{X}}$. We can exploit this characteristic as follows. First, we *pre-filter* the data based on their empirical density, such that a share of p_{out} observations are outliers. Formally, p_{out} is equivalent to choosing a threshold θ_{pre} on the empirical density, where θ_{pre} is the p_{out}-th quantile of the empirical density distribution. Using this threshold in a level-set classifier separates observations into inliers $\mathbf{I}$ and outliers $\mathbf{O}$.[2]

$$\mathbf{I} = \{x \in \mathbf{X} \colon g^{\mathbf{X}}_{\theta_{\text{pre}}} = \text{in}\} \qquad \mathbf{O} = \{x \in \mathbf{X} \colon g^{\mathbf{X}}_{\theta_{\text{pre}}} = \text{out}\}.$$

Second, we replace $f^{\mathbf{X}}$ with $f^{\mathbf{I}}$ and set $C = 1$. With this, we know that $f^{\mathbf{I}}(x) = \text{in}, \forall x \in \mathbf{I}$, without training $f^{\mathbf{I}}$. Put differently, pre-filtering the data with an explicit threshold allows us to get rid of an implicit outlier threshold C. This in turn allows estimating the level set estimated by SVDD without actually training the classifier. Algorithm 1 is the pseudo code for the pre-filtering.

Pre-filtering does not add any new exogenous parameter, but replaces the SVDD trade-off parameter C with p_{out}. Further, p_{out} is a parameter of SVDD, not of our sampling method. We also deem p_{out} slightly more intuitive than C, since it makes the lower bound defined by C tight, i.e., pre-filtering assumes an exact outlier ratio of $p_{\text{out}} = {}^{|\mathbf{O}|}/_{|\mathbf{X}|}$. This in turn makes the behavior of SVDD more predictable. We close the discussion of pre-filtering with two remarks.

[2] For brevity, we do not use the active learning notation $\mathcal{L}_{in}$ and $\mathcal{L}_{out}$ in this chapter.

Remark 1 *Technically, one may directly use the level-set classifier $g^{\mathbf{X}}_{\theta_{pre}}$ instead of SVDD. However, classification times are very high, since calculating the kernel density of an unseen observation is in $O(N)$. So one would give up fast classification, one of the main benefits of SVDD. Next, one may be tempted to interpret this pre-filtering step as a way to transform an unsupervised problem into a supervised one to train a binary classifier (e.g., SVM) on* **O** *and* **I**. *However, binary classification assumes the training data to be representative of the underlying distributions. This assumption is not met with outlier detection, since outliers may not come from a well-defined distribution. Thus, binary classification is not applicable.*

Remark 2 *Pre-filtering is a necessary step with all sampling methods discussed in related work. In Section 3.1, we have explained that existing sampling methods assume to only have inliers in the data set, i.e.,* $\mathbf{I} = \mathbf{X}$ *and* $\mathbf{O} = \emptyset$. *However, if* $\mathbf{X}$ *contains outliers, this affects the sampling quality negatively and leads to poor SVDD results, see Section 4.2.3.*

4.1.2. Optimal Sample Selection

After *pre-filtering*, we can reduce Optimization Problem 4.1 to a feasible optimization problem. We begin by replacing $f^{\mathbf{X}}$ with $f^{\mathbf{I}}$.

$$\begin{aligned} \underset{\mathbf{S}\subseteq\mathbf{X}}{\text{minimize}} \quad & |\mathbf{S}| \\ \text{subject to} \quad & \mathit{diff}(f^{\mathbf{S}}, f^{\mathbf{I}}) \leq \varepsilon. \end{aligned} \tag{4.2}$$

With Characteristic 1, we further know that both classifiers have equivalent level-set classifiers. We set $g^{\mathbf{I}}_{\theta_{\text{pre}}}$ as the equivalent level-set classifier for $f^{\mathbf{I}}$. For $f^{\mathbf{S}}$, there also exists a level-set classifier $g^{\mathbf{S}}_{\theta'}$, but the level set θ' depends on the choice of $\mathbf{S}$. Thus, we must additionally ensure that θ' indeed is the level set estimated by training SVDD on $\mathbf{S}$. The modified optimization problem is

$$\underset{\mathbf{S}\subseteq\mathbf{X}}{\text{minimize}} \quad |\mathbf{S}| \tag{4.3}$$

$$\text{subject to} \quad \mathit{diff}(g^{\mathbf{S}}_{\theta'}, g^{\mathbf{I}}_{\theta}) \leq \varepsilon \tag{4.3a}$$

$$g^{\mathbf{S}}_{\theta'} \equiv f^{\mathbf{S}}, \tag{4.3b}$$

where $\equiv$ denotes the equivalence in classifying $\mathbf{S}$. Constraint 4.3b is necessary, since one may select a sample that yields a level-set classifier similar to the one obtained from $\mathbf{I}$, but on which SVDD returns another decision boundary. This can, for instance, occur if $\mathbf{S}$ does not contain the boundary points of $\mathbf{I}$. Optimization Problem 4.3 still is very abstract. We will now elaborate on both of its constraints and show how to reduce them so that the problem becomes practically solvable.

Constraint 4.3a We now discuss how to obtain a sample that minimizes $\mathit{diff}(g^{\mathbf{S}}_{\theta'}, g^{\mathbf{I}}_{\theta})$. To this end, we use the following theorem.

Theorem 1 $g^{\mathbf{S}}_{\theta'} \equiv g^{\mathbf{I}}_{\theta}$ *if* $d_{\mathbf{S}}$ *is uniform on* $\mathbf{I}$.

Proof Think of a sample $\mathbf{S} \subseteq \mathbf{I}$ with uniform empirical density $d_\mathbf{S}$. Then $\mathbf{S}$ has exactly one level set $\theta' = \theta_{\min} = \min_{x \in \mathbf{S}} d_\mathbf{S}(x)$. Further, it also holds that $d_\mathbf{S}(x) = \theta_{\min}, \forall x \in \mathbf{I}$. It follows that $\min_{x \in \mathbf{I} \setminus \mathbf{S}} d_\mathbf{S}(x) = \min_{x \in \mathbf{S}} d_\mathbf{S}(x)$, and consequently $g^\mathbf{S}_{\theta_{\min}}(x) = g^\mathbf{I}_\theta(x), \forall x \in \mathbf{I}$. □

Theorem 1 implies that one can satisfy Constraint 4.3a with $\varepsilon = 0$ if one reduces the sample to one with a uniform empirical distribution $d_\mathbf{S}$. However, any empirical density estimate on a finite sample can only *approximate* a uniform distribution. So one should strive for solutions of Optimization Problem 4.3 where ε is small. Put differently, one can interpret the difference between a perfect uniform distribution and the empirical density to assess the quality of a sample. We propose to quantify the fit with a uniform distribution as the difference between the maximum density $\theta_{\max} = \max_{x \in \mathbf{S}} d_\mathbf{S}(x)$ and minimum density $\theta_{\min} = \min_{x \in \mathbf{S}} d_\mathbf{S}(x)$ with

$$\Delta^\mathbf{S}_{\text{fit}} = \theta_{\max} - \theta_{\min}. \tag{4.4}$$

There certainly are other ways to evaluate the goodness of fit between distributions. However, $\Delta^\mathbf{S}_{\text{fit}}$ has some desirable properties of the sample, which we discuss in Theorem 2.

One further consequence of only approximating a uniform density is that there may be some not-selected observations $x \in \mathbf{I} \setminus \mathbf{S}$ with a density value $d_\mathbf{S}(x)$ less than $\theta_{\min}$. Since the level set estimated by $f^\mathbf{S}$ is $L_{\theta_{\min}}$, these not-selected observations would be wrongly classified as outliers. Thus, we must also ensure that $\mathbf{S}$ is selected so that $d_\mathbf{S}(x) \geq \theta_{\min}, \forall x \in \mathbf{I} \setminus \mathbf{S}$. We can now re-formulate Constraint 4.3a as a sample optimization problem SOP.

$$\text{SOP}: \underset{\mathbf{v},\mathbf{w},\theta_{\min},\theta_{\max}}{\text{minimize}} \quad \theta_{\max} - \theta_{\min} \tag{4.5}$$

$$\text{subject to} \quad \underbrace{\sum_{j \in \mathcal{I}} v_j \cdot k(x_i, x_j)}_{d_\mathbf{S}(x_i)} \geq \theta_{\min}, \forall i \in \mathcal{I} \tag{4.5a}$$

$$\sum_{j \in \mathcal{I}} v_j \cdot k(x_i, x_j) \leq \theta_{\max}, \forall i \in \mathcal{I} \tag{4.5b}$$

$$\sum_{j \in \mathcal{I}} w_i \cdot v_j \cdot k(x_i, x_j) \leq \theta_{\min}, \forall i \in \mathcal{I} \tag{4.5c}$$

$$\sum_{j \in \mathcal{I}} v_j > 0; \sum_{j \in \mathcal{I}} w_j = 1; \; v_j \geq w_j, \forall j \in \mathcal{I} \cup \mathcal{O} \tag{4.5d}$$

$$v_j = 0, \forall j \in \mathcal{O}; v_j, w_j \in \{0, 1\}, \forall j \in \mathcal{I} \cup \mathcal{O} \tag{4.5e}$$

where $\mathcal{I} = \{i \mid i \in \{1, \dots, N\}, x_i \in \mathbf{I}\}$, $\mathcal{O} = \{1, \dots, N\} \setminus \mathcal{I}$. The decision variable $v_j = 1$ indicates if an observation x_j is in $\mathbf{S}$, i.e., $\mathbf{S} = \{x_i \in \mathbf{X} \mid v_i = 1\}$. Constraint 4.5b is a technical necessity to obtain the maximum density of $d_\mathbf{S}$. The first constraint in 4.5d rules out the trivial solution $v = \vec{0}$. The first constraint in 4.5e results from the *pre-filtering*, cf. Section 4.1.1. If the solution set of SOP is not singular, we select the solution where $|\mathbf{S}|$ is minimal to minimize training time.

Constraints 4.5a, 4.5c, and 4.5d together guarantee that the density of not-selected observations is at least $\theta_{\min}$, as follows. Only for one observation j we have $w_j = 1$ and for all other observations $i \neq j$, $w_i = 0$. Then for Constraint 4.5c and 4.5d to hold, j must be the observation with the minimum density and $d_\mathbf{S}(x_j) = \theta_{\min}$. Additionally, with $v_j \geq w_j$

it follows that $v_j = 1$, thus observation j is in the sample S. So, for any feasible solution of SOP all not-selected observations have a density of at least the minimum density of the selected observations. From 4.5a, it follows that $d_\mathrm{S}(x) \geq \theta_{\min}, \forall x \in \mathbf{I}$. So any solution of SOP satisfies Inequality 4.3a with a small ε.

Constraint 4.3b We now show that a solution of SOP also satisfies Constraint 4.3b. To this end, we make use of the following characteristic.

Characteristic 2 (Boundary Points) *The set of boundary points are a superset of the support vectors of SVDD [TD04].*

So for Constraint 4.3b to hold, an optimum of SOP must contain boundary points of **I**. We show that a solution with boundary points is preferred over one without boundary points by the following theorem.

Theorem 2 *The set of boundary points does not change when solving SOP iteratively.*

Proof Suppose that there is a sample S which is not a local optimum of SOP. Then there is a boundary point $x_{\min} = \arg\min_{x \in \mathrm{S}} d_\mathrm{S}(x)$, an observation $x_{\max} = \arg\max_{x \in \mathrm{S}} d_\mathrm{S}(x)$ and $x_p \in \mathrm{S}$. Let $\mathrm{S}_p = \mathrm{S} \setminus \{x_p\}$ and $\mathrm{S}_{\max} = \mathrm{S} \setminus \{x_{\max}\}$. If removing x_p from S is an optimal choice, no other observation reduces the objective more than x_p. Thus, the following specific case must hold:

$$\begin{aligned} & \Delta_{\text{fit}}^{\mathrm{S}_p} \leq \Delta_{\text{fit}}^{\mathrm{S}_{\max}} \\ \Leftrightarrow\ & \theta_{\max} - k(x_p, x_{\max}) - (\theta_{\min} - k(x_p, x_{\min})) \\ & \leq \theta_{\max} - k(x_{\max}, x_{\max}) - (\theta_{\min} - k(x_{\max}, x_{\min})) \\ \Leftrightarrow\ & k(x_p, x_{\max}) - k(x_p, x_{\min}) \geq 1 - k(x_{\max}, x_{\min}). \end{aligned} \tag{4.6}$$

We conclude that $x_p = x_{\min}$ is not feasible, because in this case the left hand side of Inequality 4.6 is strictly negative, and right hand side positive. Since boundary points have, per Definition 2, a density close to $\theta_{\min}$, they cannot be a candidate for removal.

Next, under two assumptions that (A1) the locations of the maximum and of the minimum density are distant from each other, and that (A2) the kernel bandwidth is sufficiently small, we have $k(x_{\max}, x_{\min}) \rightarrow 0$, and $k(x_p, x_{\max}) - k(x_p, x_{\min}) \geq 1 \Leftrightarrow x_p = x_{\max}$. So in this case, removing $x_{\max}$ is optimal. From this, it also follows that the minimum density does not significantly change when removing $x_{\max}$. With Definition 2, it follows that also the set of boundary points does not change after removing $x_{\max}$. □

Remark 3 *Our proof hinges on two assumptions:* (A1) A sufficiently large distance between $x_{\max}$ and $x_{\min}$. *This assumption is intuitive since removing an observation with a density close to* $\max_{x \in \mathrm{S}} d_\mathrm{S}(x)$ *improves* Δ_{fit} *more than removing one close to* $\min_{x \in \mathrm{S}} d_\mathrm{S}(x)$. *Generally, the distance between* x_{max} *and* x_{min} *depends on the data distribution. However, we find that this is not a limitation in practice, see Section 4.2.* (A2) A sufficiently small kernel bandwidth. *This assumption is reasonable because when selecting the kernel bandwidth, one strives to avoid underfitting, i.e., avoid a kernels bandwidth that is too wide. This holds empirically as well, see Section 4.2.*

Algorithm 2: RAPID

```
Input  : Data set $\mathbf{X} \in \mathbb{R}^{N \times M}$, Kernel function $k(x_i, x_j)$,
         Outlier percentage $p_{\text{out}} \in [0, 1]$
Output : Sample indices $\mathcal{S}$
   ▷ Pre-filtering, see Algorithm 1
1  $\mathcal{I}, \mathcal{O}, d = \text{pre-filtering}(\mathbf{X}, k, p_{\text{out}})$        ▷ $\mathcal{O}(N^2)$
   ▷ Sampling
2  $\mathcal{S} = \mathcal{I}$
3  for $iter \leftarrow 1 \dots |\mathcal{I}| - 1$ do                                   ▷ $\mathcal{O}(N^2)$
4      $r = \arg\max_{i \in \mathcal{S}} d_i$
5      $d = d - \langle k(x_1, x_r), \dots, k(x_N, x_r) \rangle$
6      $\theta_{\min} = \min_{i \in \mathcal{S}} (d_i)$
7      if $\exists\, i \in \mathcal{I} : d_i < \theta_{min}$ then
8          return $\mathcal{S}$
9      end
10     $\mathcal{S} = \mathcal{S} \setminus \{r\}$
11 end
12 return $\mathcal{S}$
```

Remark 4 *Overfitting the kernel parameter of SVDD affects all sampling methods. When the kernel bandwidth is very small, removing any observations from a sample yields a decision boundary that is different from the one obtained with training on the full data set. For SOP an overfitted kernel bandwidth results in density values of approximately* 1 *for all observations with the Gaussian kernel, i.e., the density is already uniform. The objective function of SOP then is already minimal, with a value of* 0*. Thus, SOP does not remove any observation from the sample and retains the original decision boundary. In practice, one can rely on one of the many heuristics to choose a suitable kernel parameter to avoid overfitting, see for example our choice in Section 4.2.1.*

SOP is theoretically appealing. However, it is a mixed-integer problem with non-convex constraints, and it is hard to solve. Thus, solver runtimes quickly become prohibitive, even for relatively small problem instances – this contradicts the motivation for sampling. We therefore propose RAPID, a fast heuristic to search for a local optimum of SOP.

4.1.3. A RAPID Approximation

The idea of our approximation is to initialize S = **I**, which is a feasible solution to SOP, and remove observations from S iteratively as long as S remains feasible, see Algorithm 2. RAPID is a fast greedy algorithm, i.e., it may not produce the smallest sample with uniformity, cf. objective function of SOP. However, the proofs for SOP that sampling retains the decision boundary also hold for RAPID.

As input parameters RAPID takes the data set **X**, the expected outlier percentage p_{out} and a kernel function k. Line 1 is the *pre-filtering*. RAPID then iteratively selects the most dense observation $x_{\max}$ in the current sample S for removal (Line 4) and updates the densities (Line 5). If $\mathrm{S} \setminus \{x_{\max}\}$ is infeasible, RAPID terminates (Line 6–8). Line 7 checks whether there is an observation $x_i \in \mathbf{I}$ that violates Constraint 4.5a. As required by SOP, RAPID does not remove boundary points. This is because $x_{\max}$ must not be a boundary point, as long as S is not uniform, i.e., $\Delta^{\mathrm{S}}_{\mathrm{fit}} > 0$. Thus, a solution of RAPID satisfies both Constraint 4.3a and Constraint 4.3b. The return in Line 12 is the special case where a single observations remains in the sample. In this case uniformity is achieved with one observations, i.e., all observations are equal.

The overall time complexity of RAPID is in $O(N^2)$, see Algorithm 1 and Algorithm 2 for the step-wise time complexities. Further, RAPID is simple to implement with only a few lines of code. It is efficient, since each iteration (Line 4–8) only requires one pass over the data set to update the densities, compute the new $x_{\max}$, $\theta_{\min}$ and minimum inlier density for the termination criterion. One may further pre-compute the Gram matrix $\mathbf{K}$ for $\mathbf{X}$ to avoid redundant kernel function evaluations.

4.2. Experiments

We now turn to an empirical evaluation of RAPID. Our evaluation consists of two parts. In the first part, we evaluate how well RAPID copes with different characteristics of the data, i.e., with the dimensionality, the number of observations, and the complexity of the data distribution, see Section 4.2.2. The second part is an evaluation on a large real-world benchmark for outlier detection, see Section 4.2.3. We have implemented RAPID and the competitors in an open-source framework written in Julia [Bez+17]. Our implementation, data sets, raw results, and evaluation notebooks are publicly available. [3]

4.2.1. Setup

We first introduce our experimental setup, evaluation metrics, and the parametrization of SVDD and its competitors. Recall that RAPID does not have any exogenous parameter. One must only specify p_{out} instead of the SVDD hyperparameter C, cf. Section 4.1.1.

Metrics Sampling methods trade classification quality for sample size, and one must evaluate this trade-off explicitly. We report the sample size $|\mathrm{S}|$ and sample ratio $|\mathrm{S}|/|\mathrm{X}|$ for each result. For classification quality, we calculate the Matthews Correlation Coefficient (MCC) as introduced in Section 2.5.2. Our experiments do not require a train-test split, since all sampling methods are unsupervised. For non-deterministic methods, we report average values over five repetitions. Our experiments ran on an AMD Ryzen Threadripper 2990WX with 64 virtual cores and 128 GB RAM.

SVDD SVDD requires to set two hyperparameters: the Gaussian kernel parameter γ and the trade-off parameter C. We tune γ with *Scott's Rule* [Sco15] for real-world data. For high-dimensional synthetic data, however, we found that the *Modified Mean Criterion* [Lia+18] is a better choice. Because of *pre-filtering* we set $C = 1$, cf. Section 4.1.1.

[3] `https://www.ipd.kit.edu/ocs`

Competitors We compare our method against 8 competitors, see Table 3.1. The approaches from [Qu+19] and [Kra+18] require to solve several hundreds of SVDDs, resulting in prohibitive runtimes. We do not include them in our evaluation. We initialize the exogenous parameters according to the guidelines in the original publications. In some cases, the recommendations do not lead to a useful sample, e.g., $\mathbf{S} = \emptyset$. To ensure a fair comparison, we mitigate these issues by fine-tuning the parameter values through preliminary experiments.

Next, we compare two variants of each competitor: sampling on $\mathbf{X}$ as in their original version, and sampling on $\mathbf{I}$, i.e., after applying our *pre-filtering*. The *pre-filtering* requires to specify the expected outlier percentage p_{out}. In practice, one can rely on domain knowledge or estimate it [Ach+10]. To avoid any bias when over- or under-estimating the outlier percentage, we set it to the true percentage. Nevertheless, we have run additional experiments where we deliberately deviate from the true percentage. We found that deviating affects the performance of all sampling methods similarly. So, our conclusions do not depend on this variation, and we report the respective results in the supplementary materials.[3]

We also evaluate against random baselines. Each baseline $Rand_r$ returns a random subset with a specified sample ratio r. We report results for a range of sample ratios $r \in [0.01, 1.0]$ to put the quality of competitors into perspective. When choosing the C parameter of SVDD for the random baseline, one must observe that outliers may be part of the selected sample. However, in experiments of ours, we have observed that $C = 1$ generally yields the most competitive baseline even if some outliers are part of the training data. Training a $r = 1$ baseline on the full data set is prohibitive for large data sets. So we only report the values for the smaller data sets.

4.2.2. Evaluation of Sample Characteristics

The first part of our experiments validates various properties of RAPID and its competitors. Our intention is to give an intuition of how a sample is selected, and to explore under which conditions the sampling methods work well. The basis for our experiments are synthetic data sets with controlled characteristics. Specifically, we generate data from Gaussian mixtures with varying number of mixture components, data dimensions, and number of observations. We run these experiments to answer the following two questions.

Q1 How are observations in a sample distributed?

To get an intuition about the sample distribution, we run RAPID and the competitors on a bi-modal Gaussian mixture, see Figure 4.3. The tendencies of the methods to select boundary points and inner points are clearly visible. For instance, BPS only selects a sparse set of boundary points; IESRSVDD only prunes high-density areas. RAPID selects both the boundary points and a uniformly distributed set of inner points. The decision boundary of RAPID matches the one obtained from the complete data set perfectly. Only three competitors (DAEDS, IESRSVDD, and NDPSR) also result in an accurate decision boundary. But all of them produce significantly larger sample sizes than RAPID.

Q2 To what extent do data characteristics influence a sample and the resulting classification quality?

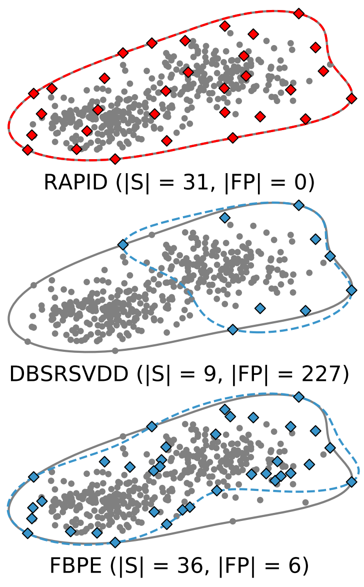
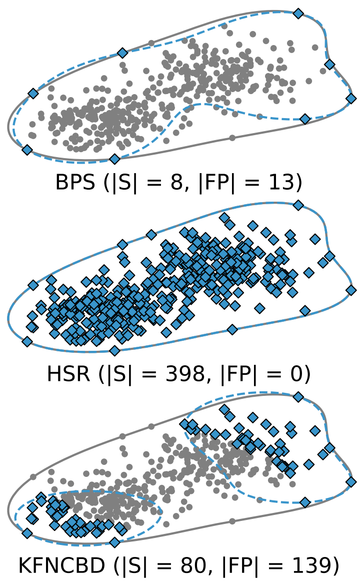
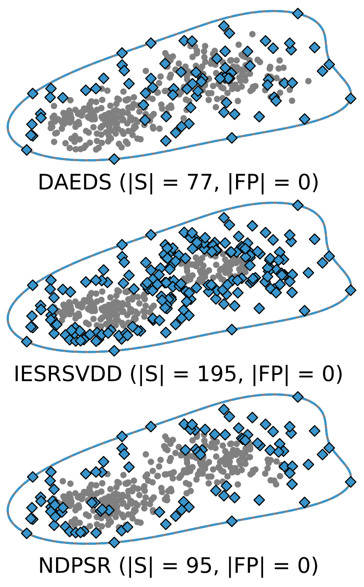




Figure 4.3.: Sampling strategies applied to a synthetic Gaussian mixture with two components and $N = 400$. The gray points are the original data set and the red/blue diamonds the selected observations. The original decision boundary is the gray line and the red/blue one is the boundary trained on the sample. |S| is the sample size and |FP| the number of misclassified inliers.

To explore this question, we individually vary the number of observations, the dimensionality, and the number of the mixture components. In the following visualizations, an optimal sampling always yields a MCC of 1 in the upper row and very small sample sizes in the bottom row, i.e., altering any data characteristic does not influence the sampling. Some values for the competitors are missing since the sample has been empty.

Number of observations: Ceteris paribus, increasing the number of observations should not have a significant impact on the observations selected. This expectation is reasonable since increasing the data size does not change the underlying distribution and the true decision boundary. Figure 4.4a graphs the sample quality and sample size for the different methods. Many competitors (BPS, IESRSVDD, KFNCB, and DAEDS) do not scale well with more observations, i.e., the sample sizes increase significantly. BPS scales worst and only removes a tiny fraction of observations. Further, the sample quality drops significantly with more than 500 observations for some competitors (DBSRSVDD and HSR). RAPID, on the other hand, is robust with increasing data size, for both sample quality and sample size. The sample sizes returned are small, even for large data sets, and the resulting quality is always close to MCC = 1.0.

Dimensionality: The expectation is that the sample quality does not deteriorate with increasing dimensionality. However, sample sizes may increase slightly. This is because learning a decision boundary of a high-dimensional manifold requires more observations than of a low-dimensional one. Figure 4.4b shows the sample quality and size. For some competitors (HSR, NDPSR, and KFNCBD), sample quality decreases with increasing dimensionality. This indicates that they do not select observations in all regions, which

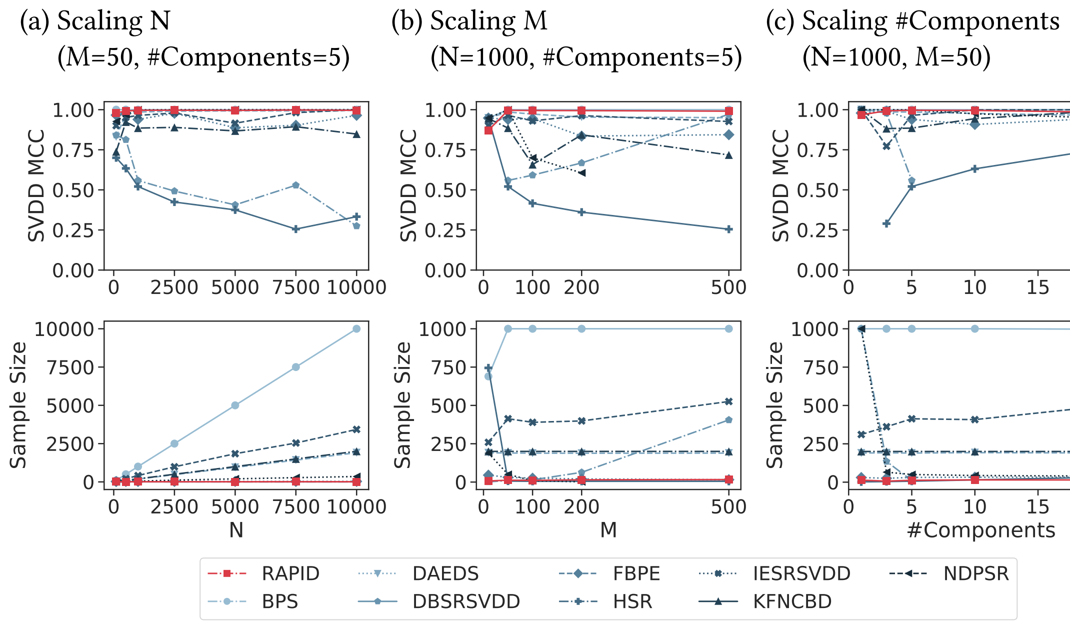


Figure 4.4.: Evaluation on synthetic data with varying data size (N), dimensionality (M), and complexity (#Components). An optimal sampling always yields a MCC of 1 in the upper row and very small sample size in the bottom row, i.e., altering any data characteristics does not influence the sampling. Some values for the competitors are missing due to an empty sample.

leads to misclassification. Even tuning exogenous parameter values does not mitigate these effects. RAPID returns a small sample in all cases, with high classification accuracy.

Number of Mixture Components: Finally, we make the data set more difficult by increasing the number of Gaussian mixture components. Like before, we expect sample sizes to increase slightly since the generated manifolds are more difficult to classify. Figure 4.4c shows the sample quality and size. For HSR and DBSRSVDD, sampling quality fluctuates significantly. NDPSR and DBSRSVDD do not prune any observation with only one component. We think that these effects are due to the sensitivity to the exogenous parameters of the various methods. This is, methods with fluctuating results would require different parameter values for data sets of different difficulties. However, the competitors do not come with a systematic way to choose parameter values to adapt to varying data set difficulty. RAPID in turn, is very robust to changes in difficulty. As expected, the sample size increases only slightly with increasing difficulty. The classification accuracy is close to MCC = 1.0, even for high difficulties.

In summary, our experiments on synthetic data reveal that many competitors are sensitive to data size, dimensionality, and complexity. Different parameter values may mitigate the effects in a few cases, but selecting good values is difficult. RAPID, on the other hand, is very robust. It adapts well to different data characteristics and without any parameter tuning.

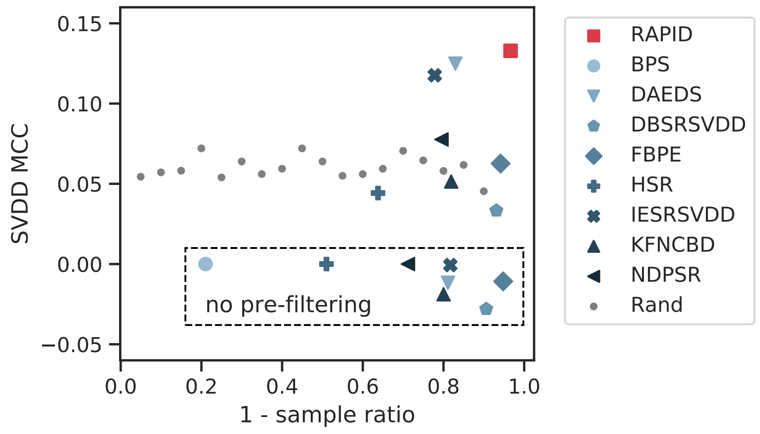


Figure 4.5.: Median MCC and ratio of observations removed by sampling (1 - sample ratio $= {}^{(N-|S|)}/_{|X|}$) over real-world data.[5] Rand is shown for different $r \in [0.01, 1.0]$. BPS with pre-filtering did not solve for large data sets.

4.2.3. Benchmark on Real-World Data

Next, we turn to data sets with real distributions and more diverse data characteristics. The basis for our experiments are 23 standard benchmark data sets for outlier detection [Cam+16] that we have introduced in Section 2.5.1. Recall that Campos et al. constructed this benchmark from classification data where one of the classes is downsampled and labeled as outlier. The data sets have different sizes (80 to 49 534 observations), dimensionality (3 to 1555 dimensions), and outlier ratios (0.2 % to 75.38 %, median 9.12 %[4]). Again, we structure our experiments along two questions.

Q3 How well do methods adapt to real-world data sets?

First, we compare RAPID against competitors without any pre-processing. Figure 4.5 plots the median sample ratio against the SVDD quality over all data sets.[5] Good sampling methods return small sample ratios and yield high SVDD quality, i.e., they appear in the upper right corner of the plot. All of the competitors in their original version, i.e., without pre-filtering, result in poor SVDD quality, much lower than the Rand baselines. The reason is that they expect all observations to be inliers.

With our *pre-filtering*, SVDD qualities of competitors improve considerably, see Figure 4.5 and Table 4.1. Still, RAPID outperforms its competitors; none of them produces a sample with higher SVDD quality or smaller sample size than RAPID. The methods closest to RAPID are DAEDS and IESRSVDD, with similar SVDD quality but significantly larger sample sizes. On average, the sample selected by RAPID even yields the same quality as training a SVDD without sampling.[6]

Q4 What are the runtime benefits of sampling?

[4] Only the data set “Parkinson” has an outlier percentage higher than 40%.

[5] We report the median statistics, but results also hold for mean values and individual comparisons (ranks), see `https://www.ipd.kit.edu/ocs`.

Table 4.1.: Median metrics over real-world data.[5]

	runtimes			sample		quality
	t_{samp}	t_{train}	$t_{\text{class}}{}^*$	size	ratio	MCC
RAPID	0.02	0.02	0.01	21	0.03	0.13
BPS[†]	0.61	0.47	0.15	385	0.60	†
DAEDS	0.59	0.04	0.07	98	0.17	0.12
DBSRSVDD	0.01	0.02	0.01	40	0.07	0.03
FBPE	0.07	0.02	0.01	39	0.06	0.06
HSR	0.47	0.07	0.03	130	0.36	0.04
IESRSVDD	0.02	0.10	0.03	154	0.22	0.12
KFNCBD	0.53	0.06	0.04	100	0.18	0.05
NDPSR	0.51	0.07	0.03	103	0.21	0.08

* time for classification in seconds per 1000 observations.
† did not solve for large data sets.

Finally, we look at the impact of sampling on algorithm runtimes, see Table 4.1. We measure the execution runtimes of the sampling method (t_{samp}), of SVDD training on the sample (t_{train}), and of the classification (t_{class}). Overall, all methods have reasonable runtimes for sampling, with BPS being the slowest with 0.61 s on average. However, RAPID is the fastest method overall. Methods with runtimes similar to RAPID, such as DBSRSVDD, feature significantly lower SVDD quality. Compared to SVDD applied to large original data sets without sampling, RAPID reduces training times from over one hour to only a few seconds.[6]

In summary, RAPID outperforms its competitors on real-world data as well. There is no other method with higher SVDD quality and similarly small sample sizes. RAPID scales very well to very large data sets and reduces overall runtimes by up to one order of magnitude.

4.3. Summary

In this chapter, we have studied the trade-off between classification quality and classification training costs for SVDD. Since SVDD does not scale well to large data sets, working with a sample instead of the original data has received much attention in literature. Various existing sampling approaches guess the support vectors of the original SVDD solution from data characteristics. These methods are difficult to calibrate because of unintuitive exogenous parameters. They also tend to perform poorly regarding outlier detection. One reason is that including support vector candidates in the sample does not guarantee them to indeed become support vectors.

We have addressed these issues in this chapter. We have formalized SVDD sample selection as an optimization problem, where constraints guarantee that SVDD indeed

[6] Based on data sets with non-prohibitive runtime, i.e., $N < 25\,000$, see `https://www.ipd.kit.edu/ocs` for details.

yields the correct decision boundaries. We have achieved this by reducing SVDD to a density-based decision problem, which gives way to rigorous arguments why a sample indeed retains the decision boundary. To solve this problem effectively, we have proposed a novel iterative algorithm named RAPID. RAPID does not rely on any additional parameter tuning beyond the one already required by SVDD. RAPID is efficient and consistently produces a small high-quality sample. Our experiments have shown that our way of framing sampling as an optimization problem allows us to substantially outperform existing methods with respect to runtimes, sample sizes, and classification accuracy.

5. Batch Selection for One-Class Active Learning

In the last chapter, we have proposed a new approach for *sampling* to reduce the classification training costs of the one-class classifier SVDD. In this chapter, we propose an orthogonal extension to *sampling* to further reduce the classifier training costs. We investigate how we can use the concept of *delaying* classifier updates (cf. Section 2.3.3) to reduce costs.

As discussed in Section 2.4.3, query strategies for one-class active learning are *sequential*.[1] Once the oracle has provided a class label for a single observation, the active learning system retrains the classifier. But there are cases when the sequential mode is unfavorable, e.g., when classifier retraining is slow [TEB19], if several annotators are available in parallel [Set11], or with changeover costs of experiments and simulations [GK11]. In such cases, it seems natural to ask the oracle to annotate a *batch* of observations at a time. However, the benefit of batch active learning hinges on trade-offs between the cost of active learning, which comprises the costs for query selection, classifier training and annotation, and the increase in classification quality. Literature tends to oversimplify these trade-offs: *Large batches yield slower learning rates, but they reduce the overall cost of active learning because of less frequent classifier training*, see [LG94; OJM17] for example. In reality, however, batch selection is more involved. For example, it is difficult to model the cost terms or to select a good batch size. Further, selecting a good batch requires (i) quantifying the expected utility of a batch, and (ii) a strategy to traverse the space of candidate batches, which is prohibitively large. These issues have not been studied for one-class active learning so far. We are the first to study when exactly batch one-class active learning is useful, compared to the sequential case. We break this motivation down into three specific questions:

(Q1) How can batch utility be quantified in the one-class setting?
(Q2) What are suitable schemes to select candidate batches?
(Q3) Is there a sweet spot between the costs of batch active learning and classification accuracy?

Studying these questions is difficult. First, there are multiple ways to approach batch selection, with varying levels of complexity. One example is the different models for annotation costs. The costs can either be constant per label, decrease with the batch size, or depend on the individual queries of the batch [Set11]. While some of these aspects

[1] The remainder of this chapter bases on the article [Eng+20b]: Adrian Englhardt et al. "Finding the Sweet Spot: Batch Selection for One-Class Active Learning". In: *SIAM International Conference on Data Mining (SDM)*. SIAM. 2020, pp. 118–126. DOI: 10.1137/1.9781611976236.14. We have shortened the text to be less repetitive, applied minor corrections, and changed the formatting and notation so that it is in line with the format and structure of this thesis.

have been mentioned in literature, there neither exists an overview nor a formal model of them. This makes it very hard to understand the trade-offs, to detect simplifications of the batch-selection problem, and to justify the benefit of batches.

Second, literature does not feature a method to calculate the batch utility for one-class active learning. In other domains, such as multi-class classification, the naive alternative to select the top-k sequential queries for a batch often is sub-optimal, since these observations tend to be similar to each other [Set12; She+04; SZ05]. Different ways to approach this issue are conceivable, e.g., by introducing notions like batch diversity and representativeness. But which notions actually are useful in a one-class setting, and how to possibly combine them into one aggregate measure is unclear.

Finally, batch selection is a combinatorial problem: there are $2^N - 1$ potential batches for N unlabeled observations. Current work on batch utility is confined to balanced domains and multi-class classification [Cha+13; RV18; WY15]. However, one-class problems have several distinctive properties, such as a significant class imbalance and undefined densities for the outlier class. These properties require specific sequential selection strategies [TEB21], and we expect this for batch strategies as well.

Contributions This chapter features the first principled approach to batch active learning for one-class classification. We make two specific contributions: (i) We formalize batch selection as a general and comprehensive optimization problem. Our framework trades off batch utility against annotation, batch-selection, and classification costs under varying batch sizes. To our knowledge, this is the first formal framework that makes assumptions and simplifications for batch selection explicit. (ii) We propose several strategies to measure batch utility, specific to one-class classification. We then combine our utility measures with search strategies from four categories: top-k, iterative, partitioning, and filtering strategies. We discuss their theoretical properties and compare them empirically on real-world data against sequential selection and random baselines.

An important takeaway from this chapter is that batch queries with outlier detection are indeed different from their multi-class counterparts. In contrast to the multi-class case, batch diversity is not essential. Instead, selecting the top-k observations by informativeness suffices for good learning rates. There also is a sweet spot, and it is to use the top-k observations ranked by a sequential strategy with batch sizes between 8 and 16 observations. This decreases computational cost by up to one order of magnitude, while retaining the classification accuracy from the sequential case.

5.1. Formalization

In this section, we derive a formal model of batch selection, by framing it as an optimization problem. We start with theoretical considerations and then propose simplifications.

Batch active learning selects a sequence of sets of observations for annotation and retrains a classifier after each set has been annotated. The ultimate objective is that the model has good classification accuracy after the last batch has been annotated. In general, annotations have a positive expected marginal utility. This is because each one provides additional information that helps to improve the classification model. But annotations may introduce a short-term bias, which can deteriorate classification accuracy temporarily [Ber+18]. This bias diminishes when more annotations become available. Next, one further assumes that active learning is subject to some budget restriction as we have discussed in Section 2.3.3. For one, annotations require resources of real entities, such as humans or technical equipment. Next, retraining a classifier and searching for a good batch requires computational resources. One can assign resources specific costs, be they monetary, be they in time equivalents, and restrict the resources to the budget available. Existing cost-sensitive approaches merely focus on the annotation costs [CBP10; Hae+08; TH19; TL19] and lack a comprehensive model.

5.1.1. A Theoretical Model

We have a data set $\mathbf{X}$ with N observations and ground truth labels are $y \in \{\text{inlier}, \text{outlier}\}$. Formally, the objective of batch selection is to find a sequence of batches $\mathbf{B} = (B_1, \ldots, B_l)$ where $B_i \in \mathcal{P}(\mathbf{X}), B_i \cap B_{j \neq i} = \emptyset$ that yields the best possible classification accuracy *acc* on data $\mathbf{X}$ with labels y, with a semi-supervised classifier that trains a *model* on the annotated observations $\bigcup_{B_i \in \mathbf{B}} B_i = \mathcal{L}$ and the remaining observations $\mathcal{U}$, a budget T, and a cost function $c : B_i \mapsto \mathbb{R}$. Formally:

$$\begin{aligned} \underset{\mathbf{B}}{\text{maximize}} \quad & \text{acc}\left(\text{model}\left(\bigcup_{B_i \in \mathbf{B}} B_i \cup \mathcal{U}\right), \mathbf{X}, y\right) \\ \text{subject to} \quad & c(\mathbf{B}) \leq T. \end{aligned}$$

This optimization problem only is of theoretical value, for two reasons. First, calculating *acc* requires access to ground truth labels. Since the very objective of active learning is to obtain these labels, one cannot solve this optimization problem directly. Instead, one must estimate the value of annotating a batch by a utility function u that is independent of a ground truth. One can interpret u as a proxy, i.e., a high utility value of a batch is expected to improve classification accuracy. An important property of u is that it depends on all batches that have already been annotated. This is because annotated batches provide information on the classification problem that the batch selection method in turn, can use when selecting future batches. So the optimal sequence of batches is the one with the maximum cumulative utility, i.e., $\sum_{i=1}^{l} u(B_i | \bigcup_{j=1}^{i-1} B_j)$.

Second, the cost function is difficult to estimate. On the one hand, the various costs depend on each other. For instance, humans may idle during classifier retraining, but this only is cost-relevant when they cannot resort to some intermediary tasks. On the other hand, annotation costs may vary over time, e.g., because of the complexity of the query [Set11; TL19] or because humans become more experienced with annotating.

While this optimization problem is theoretical, it is important to state it to specify the goal of batch active learning. The problem is a basis for deriving another problem that one can solve in practice. Existing literature omits this step and performs batch selection for each iteration of active learning independently.

5.1.2. A Relaxed Model

A simplification is to relax the budget constraint, as follows. First, one only restricts the *number* of annotations. So the constraint becomes $|\bigcup_{i=1}^{l} B_i| \leq T,\ T \in \mathbb{N}$. The second simplification is to include the cost as independent and commensurable terms c_s (select), c_a (annotate), and c_t (train) in the objective function. This may require normalizing cost and utility terms. This gives

$$\begin{aligned} \underset{\mathbf{B}}{\text{maximize}} \quad & \sum_{i=1}^{l} u(B_i | \bigcup_{j=1}^{i-1} B_j) - c_s(B_i) - c_a(B_i) - c_t(B_i) \\ \text{subject to} \quad & |\bigcup_{i=1}^{l} B_i| \leq T, \quad T \in \mathbb{N}, \quad B_0 = \emptyset. \end{aligned}$$

Obtaining a solution to this relaxed problem still is difficult. Namely, as explained earlier, the utility of B_i depends on all previous batches. But the actual annotation of these batches is not available at the time of optimization. At first sight, a remedy would be to calculate the expected utility based on simulating all possible outcomes (inlier or outlier), for all batches. However, the state space to consider is $\mathcal{P}(\mathbf{X} \times \{inlier, outlier\})^l$. Traversing this space is prohibitive for a reasonably large number of observations, since it requires retraining the classifier in each state.

5.1.3. A Practical Model

To make solving the optimization problem feasible, one can use an additional simplification. Instead of optimizing for all batches $(B_i, \ldots, B_l)$, the optimization variable only is the current batch B_i, after $(B_1, \ldots, B_{i-1})$ have already been annotated. This allows to separate the problem into two nested problems. The inner one is to find a good batch for a given batch size k. For this inner problem, only annotation costs are relevant. The reason is that classifier retraining, as well as the search for a batch, are the same for all batch candidates of size k. The outer problem is to find an optimal k. Formally:

$$\max_k \left[\left(\underbrace{\max_{B_i \in \mathcal{P}_{=k}(\mathcal{U})} u(B_i | \bigcup_{j=1}^{i-1} B_j) - \sum_{x \in B_i} c_a(x, B_i)}_{\text{inner problem}} \right) - c_t(k) - c_s \cdot \binom{|\mathcal{U}|}{k} \right] \cdot \frac{T}{k}.$$

Here, c_t depends on k if the classifier is incremental, i.e., retrained k times. Further, there are $\binom{|\mathcal{U}|}{k}$ possible batches of size k to consider. The last term $\frac{T}{k}$ is a technical constraint to adhere to the budget restriction. It implicitly assumes that k is fix for all batches, and that $k \in \{j \in \mathbb{N} | (\exists i \in \mathbb{N})[i \cdot j = T]\}$. However, this restriction is not crucial. In practice, one can repeat the optimization after the first batch is annotated, with the remaining budget $T' = T - k$. The nested problem now gives way to model the cost terms as follows.

Classifier training: There is an essential difference between incremental and non-incremental classifier training. If a classifier features dynamic updates, c_t linearly depends on k. Otherwise, c_t is constant, i.e., there only occurs one full retraining per batch. However, c_t is likely to be much higher for a full retraining than in the incremental case.

Batch Selection: c_s requires to calculate the utility for each possible batch. Thus, c_s is multiplied with the number of possible batches of size k.

Annotation: The costs of annotation are specific for each observation and may depend on the batch [Set11; TL19]. Further, observations may be easier to annotate in sufficiently large batches [Set11]. However, a very common simplification is to assume identical annotation costs for all observations independently of the batch size [CBP10]. In this case, $c_a(\cdot)$ reduces to $\bar{c}_a \cdot k$ and becomes part of the outer problem, since it only depends on k.

With these simplifications, the inner problem depends solely on u. This is convenient since it allows considering utility calculation and cost estimation independently. Put differently, the inner problem, i.e., finding a batch of size k with maximal utility, can be studied independently of any cost function. So the remaining problem to solve under the given simplifications is to find a batch of size k given already annotated observations $\mathcal{L}$ by evaluating a given utility function u. One can then vary k and u to find good batch sizes and a suitable utility function.

5.2. Method

In this section, we propose different batch strategies for one-class active learning. First, we discuss different criteria to measure *batch utility*. Then we present *batch strategies* to select a batch based on these criteria.

5.2.1. Batch Utility

Intuitively, a utility function u quantifies the expected benefit of annotating one or more observations with respect to the classification accuracy. In the sequential case, u is a function $u_{\text{seq}} : \mathbf{X} \rightarrow \mathbb{R}$, i.e., it assigns a utility per observation. For batch selection, the function is of type $u : \mathcal{P}(\mathbf{X}) \rightarrow \mathbb{R}$. So u quantifies the utility of a set. This allows considering inter-observation relationships. For instance, annotating similar observations may have information overlap, and having them in one batch can be suboptimal. So u typically considers three criteria: informativeness, representativeness, and diversity [She+04]. In the following, we elaborate on these criteria, review different possibilities to implement them with one-class classifiers, and describe our choice for this chapter. We focus on general ways to quantify these criteria and do not consider application-specific ones, like remote sensing [SDZ15] or document relevance [XAZ07].

5.2.1.1. Informativeness

Recall from Section 2.3 that informativeness is a function $\tau(x)$ that quantifies how much a classifier is expected to benefit from knowing the label of a single observation $x \in \mathbf{X}$.

There are several categories of informativeness functions, and many of them have been proposed explicitly for one-class classification [TEB21] and we have already discussed them in Section 2.4.3.

Our choice: We choose the two model-based query strategies that work with the decision function f of a one-class classifier (cf. Section 2.2.3). The first one is τ_{DB} and the second one is τ_{HC}, see Section 2.4.3 for details.

5.2.1.2. Representativeness

Representativeness is a function $rep(x)$ that quantifies how well an observation represents the underlying data set, and observations doing this well are preferred. However, there are different interpretations of representativeness. An observation can be representative if it lies in a dense area of the data distribution. This can be quantified by kernel density estimation [HGX11] or by calculating the average distance to the k-nearest neighbors [KCR18; ZLG03]. A different, implicit, approach is to cluster data and select the cluster medoids as representative observations [DPB11; SZ05; XAZ07]. Finally, some approaches quantify representativeness for a batch [CBP15; Cha+13; Du+17; Wan+16; WY15]. They use maximum mean discrepancy (MMD) to measure how well the batch follows the full data distribution.

Literature proposes strategies that combine representativeness and informativeness for sequential query selection, e.g., a linear combination of the distance to the decision boundary and the nearest neighbors [YWF18]. Whenever such a sequential strategy is applicable in the following, we make this clear by writing u_{seq} instead of τ or *rep*.

Our choice: In this thesis, we quantify representativeness using kernel density estimation. We refer to Section 2.1 for an explanation on how to calculate the kernel density. However, we also present two clustering approaches that choose representative queries implicitly, see Section 5.2.2.3.

5.2.1.3. Diversity

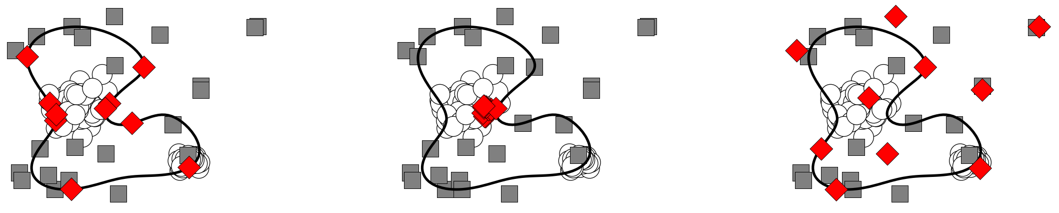
While the first two criteria are defined for individual observations, the diversity criterion is for batches. Intuitively, a batch is diverse if its observations are dissimilar to each other. A high diversity of a batch is good since dissimilar observations are expected to have little information overlap. There are several ways to enforce diversity, either implicitly or explicitly. One explicit method is to maximize pair-wise distances, e.g., in the data or kernel space [Bri03; CBP15]. Cluster-based strategies consider diversity implicitly since they select observations from different clusters, which is likely to yield a diverse batch.

Our choice: We follow previous work to quantify the diversity *div* of a batch B as the minimum pair-wise distance d of two observations [Bri03; Car+17; KCR18]

$$div(B) = \min_{x_i, x_j \in B} d(x_i, x_j). \tag{5.1}$$

We use two distance functions. The first one is the Euclidean distance d_{ED} in the data space. The second one is a distance in the reproducing kernel Hilbert space of a kernel-based classifier $d_{\text{AK}} = -\left|\cos(\angle(\phi(x_i), \phi(x_j)))\right|$. Intuitively, d_{AK} is proportional to the angle of two observations in the kernel space [Bri03].

A key challenge is to combine these so-called batch criteria [She+04] into a single utility measure. Before addressing this, we discuss some theoretical properties of the criteria in the context of one-class classification for outlier detection. First, with outlier detection, a large share of the data space is sparse. Given such a sparsity, diversity may not be useful. This is because it is likely to include observations from sparse regions, which only provide little information for the classifier. Second, with outlier detection, representativeness may not be very useful either. The reason is that it focuses on regions of high density, which are likely to contain only observations from the inlier class. The following example illustrates these properties.



(a) Informativeness (b) Representativeness (c) Diversity

Figure 5.1.: Comparison of batch selection based on different criteria.

Example 5.2.1 *Figure 5.1 shows a 2-dim synthetic data set with inliers (white circles) and outliers (gray squares). We fit a one-class classifier, considering all observations as unlabeled; the decision boundary is the black line. We then compute batches of 10 observations for each criterion independently (red diamonds).*

As expected, representativeness selects the observations where the data density is high. The resulting batch contains a bulk of observations in a small region, with high similarity. With diversity, the observations in the batch are well spread. Yet, some queries lie in very sparse regions where feedback might only influence the classification of very few observations. Informativeness with τ_{DB} selects the 10 observations closest to the decision boundary, see Figure 5.1a. Although the batch does not cover the full data space, visually the observations selected are diverse and representative, without explicit consideration of these criteria.

Given this, we propose the following hypothesis.

Hypothesis 5.2.1 (Batch Criteria) *With one-class outlier detection, the representativeness and diversity criteria are not useful for batch selection.*

Our experiments on real-world data will confirm this hypothesis. The hypothesis also holds for weighted combinations of the criteria, see Section 5.3.

5.2.2. Batch Strategies

In general, many ways to combine batch criteria to select a batch are conceivable. Some strategies have been proposed for binary and multi-class settings [Cha+13; WY15]. We only

Algorithm 3: Random-B Strategy

```
Input  : $k$, $\mathcal{U}$
Output : $B$
1 $B$ = Draw an independent sample without replacement of size $k$ from $\mathcal{U}$
```

Algorithm 4: TopK Strategy

```
Input  : $u_{\text{seq}}$, $k$, $\mathcal{U}$
Output : $B$
1 $\mathcal{U}_{ranked}$ = sort $x$ in $\mathcal{U}$ by $u_{\text{seq}}$, descending
2 $B$ = select first $k$ from $\mathcal{U}_{ranked}$
```

Algorithm 5: Iterative Strategy

```
Input  : $\tau(x)$, $rep(x)$, $div(B)$, $\lambda_{\text{inf}}$, $\lambda_{\text{rep}}$, $\lambda_{\text{div}}$, $k$, $\mathcal{U}$
Output : $B$
1 $B = \left\{ \underset{x_i \in \mathcal{U}}{\arg\max} \Big( \lambda_{inf} \cdot \tau(x_i) + \lambda_{rep} \cdot rep(x_i) \Big) \right\}$
2 for $i \leftarrow 2 \dots k$ do
3     $x_i^* = \underset{x_i \in \mathcal{U} \setminus B}{\arg\max} \Big( \lambda_{\text{inf}} \cdot \tau(x_i) + \lambda_{\text{rep}} \cdot rep(x_i) + \lambda_{\text{div}} \cdot div(B \cup \{x_i\}) \Big)$
4     $B = B \cup \{x_i^*\}$
5 end
```

are aware of one summary for multi-label data [RV18] which requires other approaches than our one-class setting. Further, there does not seem to be any strategy specific to one-class classification. It is unclear whether existing multi-class approaches transfer to the one-class setting. In this section, we, therefore, propose different batch selection strategies for one-class active learning. We classify them into four categories: baseline, iterative, partitioning, and filtering strategies.

5.2.2.1. Baselines

As one baseline, we propose *random batch*, which samples k unlabeled observations independent of any criteria, see Algorithm 3.

The second baseline is *TopK*, a straightforward extension of the sequential mode. The idea is to calculate utility for each observation independently, with a given u_{seq}, and then choose the top k observations, see Equation 2.11 and Algorithm 4.

Algorithm 6: Cluster Strategy

```
Input  : $k, \mathcal{U}$
Output : $B$
1 $C = KMedoidsClustering(\mathcal{U}, k)$
2 $B = \emptyset$
3 foreach cluster $C \leftarrow \mathcal{C}$ do
4     $x_C$ = select medoid from $C$
5     $B = B \cup \{x_C\}$
6 end
```

5.2.2.2. Iterative Strategy

Iterative strategies are heuristics to find a batch that maximizes the weighted sum of batch criteria. Formally, this is

$$u(B) = \sum_{x \in \mathcal{B}} \lambda_{\text{inf}} \cdot \tau(x) + \lambda_{\text{rep}} \cdot rep(x) + \lambda_{\text{div}} \cdot div(B), \tag{5.2}$$

with weight parameters $\lambda_{\text{inf}}, \lambda_{\text{rep}}, \lambda_{\text{div}} \in \mathbb{R}$ to specify the importance of the three criteria. A fundamental difficulty with this approach is that one must calculate u for $\binom{|\mathcal{U}|}{k}$ candidate batches. So to find a good solution, one can instead build the batch greedily [Hoi+06; XAZ07]. Thus, the observation that maximizes the weighted sum of the three criteria is added to the batch in each iteration until the batch contains k observations; see Algorithm 5. If τ, rep and div are submodular, the greedy solution has a lower bound of $(1 - \frac{1}{e}) \cdot u(B^*)$, relative to the optimal utility $u(B^*)$ [Hoi+06].

5.2.2.3. Partitioning Strategies

A different approach is to emphasize diversity. The idea is to divide the data set into k random or disjoint subsets and then select one observation from each subset. To this end, two common approaches are to *cluster* data [DPB11; LGC12; She+04; SZ05; XAZ07] and to train several classifiers on different samples of the data as an *ensemble* [LGC12].

One *cluster* strategy partitions $\mathcal{U}$ into k clusters, for instance by k-medoids clustering. The medoids of each cluster then form the batch, see Algorithm 6. *Cluster* only considers representativeness and diversity, but not informativeness.

An extension is *cluster-TopK*, see Algorithm 7. The idea is to include informativeness to filter unlabeled observations. This is, *cluster-TopK* selects the m highest ranked unlabeled observations according to $\tau(x)$ where $|\mathcal{U}| \gg m \gg k$ and clusters them into k clusters. As before, the medoids of each cluster are the batch B.

An *ensemble* strategy randomly samples k subsets of size $|\mathbf{X}|/k$, see Algorithm 8. For each subset, the strategy trains a classifier and selects the best observation for a given u_{seq}.

Algorithm 7: Cluster-TopK Strategy

```
Input  : $\tau(x)$, $k$, $m \geq k$, $\mathcal{U}$
Output : $B$
1 $\mathcal{M}$ = top $m$ observations from $\mathcal{U}$ ranked by $\tau(x)$
2 $C = KMedoidsClustering(\mathcal{M}, k)$
3 $B = \emptyset$
4 foreach cluster $C \leftarrow C$ do
5     $x_C$ = medoid of $C$
6     $B = B \cup \{x_C\}$
7 end
```

Algorithm 8: Ensemble Strategy

```
Input  : $\mathbf{X}_1, \ldots, \mathbf{X}_k$, $C_1, \ldots, C_k$, $u_{\text{seq}}$, $\mathcal{U}$
Output : $B$
1 $B = \emptyset$
2 foreach sample $\mathbf{X}_i$ with classifier $C_i$ do
3     $x_i^* = \arg\max_{x_i \in \mathbf{X}_i, x_i \notin B} u_{\text{seq}}(x_i)$
4     $B = B \cup \{x_i^*\}$
5 end
```

5.2.2.4. Filtering Strategies

Finally, there are filter strategies that proceed top-down to select a batch. The idea is to start with all unlabeled observations and step-wise apply filter criteria until only k observations are left. We see two types of *filter* strategies.

The *filter similar* strategy searches for the two most similar observations and removes the one with less informativeness [Sad+17], see Algorithm 9.

The *filter hierarchical* strategy filters for each of the three criteria, step by step [Jia+14], see Algorithm 10. This is, *filter hierarchical* first selects the $4 \cdot k$ most representative observations and from these the $2 \cdot k$ most informative ones. From them, it greedily selects the batch, similarly to the iterative strategy.

All batch strategies presented have the same objective, increasing classification accuracy. But the realizations differ significantly. Based on plausibility arguments, there is no single, superior approach. While literature has proposed strategies for the multi-class setting, it is unclear which conclusions transfer to the one-class setting. We now derive two hypotheses. If they hold, they help with the selection of a suitable approach in the one-class setting.

The first hypothesis extends Hypothesis 5.2.1 to batch strategies.

Hypothesis 5.2.2 (Batch Selection Strategy) *With one-class outlier detection, a strategy solely based on informativeness is expected to outperform more sophisticated strategies that explicitly incorporate representativeness and diversity.*

Algorithm 9: Filter Similar Strategy

```
Input  : $\tau(x)$, $d(x_i, x_j)$, $k$, $\mathcal{U}$
Output : $B$
1 $B = \mathcal{U}$
2 while $|B| > k$ do
3     $a, b = \arg\min_{x_i, x_j \in B} d(x_i, x_j)$
4     $B = B \setminus \left\{ \arg\min_{x \in \{a,b\}} \tau(x) \right\}$
5 end
```

Algorithm 10: Filter Hierarchical Strategy

```
Input  : $\tau(x)$, $rep(x)$, $div(B)$, $k$, $\mathcal{U}$
Output : $B$
1 $\mathcal{M} = 4 \cdot k$ highest ranked $x$ in $\mathcal{U}$ by $rep(x)$
2 $\mathcal{M} = 2 \cdot k$ highest ranked $x$ in $\mathcal{M}$ by $\tau(x)$
3 $B = \{$select highest ranked $x_i$ in $\mathcal{M}$ by $\tau(x)\}$
4 for $i \leftarrow 2 \ldots k$ do
5     $x_i^* = \arg\max_{x_i \in \mathcal{M} \setminus B} div(B \cup \{x_i\})$
6     $B = B \cup \left\{ x_i^* \right\}$
7 end
```

Based on this hypothesis, we expect *TopK* to perform well, compared to more sophisticated approaches.

The second hypothesis concerns the cost trade-offs, see Section 5.1. Batch selection strategies have different complexity. When assuming constant evaluation costs for u_{seq} and τ, we derive the following complexities: constant for *random* batches, $O(n)$ for *TopK*, $O(kn)$ for *Iterative*, approximately $O(kndi)$ with dimensionality d and a number i of clustering steps for *cluster-based*, $O(k\,(n/k)^2)$ for *Ensemble* where k classifiers are learned on n/k large partitions, $O(k+n^2)$ for *Filter Similar* where n^2 is the complexity of computing the similarity matrix, and $O(kn)$ for *Filter Hierarchical.* With this, we expect the runtime cost of the batch selection strategies to be low compared to the classifier training, which requires solving a quadratic optimization problem for standard one-class classifiers, like SVDDneg. All this motivates the following hypothesis.

Hypothesis 5.2.3 (Cost Trade-Offs) *The classifier cost c_t dominates the batch selection costs c_s during active learning with batches in most cases.*

When this hypothesis holds, batch selection costs c_s are not relevant, and we can simplify the optimization model, see Section 5.1.3. An exception is *Ensemble*, which requires training k classifiers on n/k partitions of the data set. Our experiments on real world data confirm both hypotheses, see Section 5.3.

5.3. Experiments

We now present our empirical findings and discuss them in the context of the three hypotheses introduced earlier. Our implementation, raw results, and notebooks are available at `https://www.ipd.kit.edu/bocal`.

5.3.1. Setup

We use 21 standard data sets for outlier detection [Cam+16], introduced in Section 2.5.1. We use three resampled versions, with an outlier ratio of 5%, and up to 1000 observations.

Classifier: Our base classifier is SVDDneg [TD04] with the Gaussian kernel (cf. Equation 2.8), tuned as proposed in [Wan+18], and the cost parameter set as proposed in [TD04]. We also use the obtained kernel parameter for kernel density estimation and for the kernel angle distance d_{AK}.

Active Learning: We choose τ_{DB} as the informativeness criterion, since it has yielded the best results in preliminary experiments of ours. For diversity, we use d_{AK} unless stated differently. We set $m = 10k$ for *Cluster-TopK*. We start with no labels, a budget of $T = 128$ and evaluate $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$.

Evaluation Metrics: We evaluate classification accuracy with the Matthews Correlation Coefficient (MCC), see Section 2.5.2 for details. We report the end quality (EQ), i.e., the accuracy after the budget is exhausted, as the median over all data sets. Our experiments run on an AMD Ryzen Threadripper 2990WX with 64 virtual cores. We measure the total runtime t and the query selection time t_s in seconds.

5.3.2. Results

We first evaluate the usefulness of the three batch criteria in the one-class setting. We then compare the batch strategies presented in Section 5.2. We compare all strategies against the sequential mode and evaluate the sensitivity to varying batch sizes. Finally, we present runtime measurements and discuss them with resepect to the trade-offs introduced in Section 5.1.

Batch Criteria We evaluate the usefulness of batch criteria by varying the weights of the *iterative* strategy. Figure 5.2 shows the median EQ for the different combinations with a batch size of $k = 4$. The EQ is highest for $\lambda_{\text{rep}} = 0$, as indicated by the red dots on the bottom line of the triangle. Results with different batch sizes or τ_{HC} as the informativeness criterion are similar. With d_{ED} as diversity, a small value $\lambda_{\text{rep}} > 0$ does not reduce EQ as much as for d_{AK} but $\lambda_{\text{rep}} = 0$ also results in the highest EQ. On a per-data-set level, there are some instances where setting $\lambda_{\text{div}} > 0$ has a positive effect on EQ. However, tuning the weight parameters per data set is unrealistic. Namely, this would require a labeled training set. We conclude that combining the three criteria with strictly positive weights does not increase the model quality in the one-class setting. Informativeness has a dominating influence on EQ – this supports Hypothesis 5.2.1. So $\lambda_{\text{inf}} = 1, \lambda_{\text{rep}} = 0, \lambda_{\text{div}} = 0$ is the best choice, which corresponds to the *TopK* strategy. This is a strong difference to the balanced and multi-class domain [KCR18].

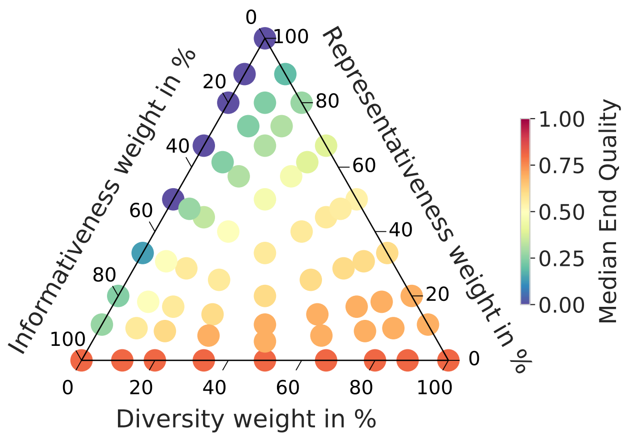


Figure 5.2.: Influence of batch criteria on end quality.

Table 5.1.: Median EQ over all data sets in comparison to a sequential baseline.

k	rand-B	TopK	Clust	ClustTopK	Ensemble	FiltSim	FiltHrch	Seq
1	0.49	**0.81**	0.00	0.70	**0.81**	0.70	0.00	**0.81**
2	0.43	**0.81**	0.28	**0.81**	**0.81**	**0.81**	0.64	-
4	0.49	**0.81**	0.35	0.79	**0.81**	0.79	0.78	-
8	0.49	**0.81**	0.54	0.76	**0.81**	0.78	0.79	-
16	0.49	**0.80**	0.57	0.68	**0.80**	0.78	0.77	-
32	0.49	**0.79**	0.60	0.62	†	0.75	0.75	-
64	0.49	**0.79**	0.57	0.60	†	0.70	0.74	-
128	0.49	**0.75**	0.57	0.57	†	0.70	0.71	-
Rank‡	5.10	**1.99**	4.72	3.42	-	2.79	2.98	-

†Optimization problem infeasible for $k > 16$. Sequential and ensemble strategy are excluded.
‡ Mean of the rank calculated for each data set and batch size.

Batch Strategies Next, we compare the performance of the different batch strategies proposed in this chapter. See Table 5.1 for the median EQ. *TopK* outperforms all other strategies. Up to a batch size of $k = 8$, the EQ is 0.81 and hence equal to the sequential strategy. Up to $k = 64$, the accuracy loss is small compared to the sequential strategy. The more complex partitioning or filtering strategies generally yield results similar to or worse than *TopK*. This supports Hypothesis 5.2.2. These results are again contrary to multi-class batch strategies, where partitioning and iterative strategies outperform *TopK* [SZ05; XAZ07], and where end quality increases with the batch size in some cases [RV18].

Trade-offs Table 5.2 shows the t and t_s for varying batch sizes. The sequential strategy has a runtime of 178s where 5.9% of the time is spent on batch selection. In all cases, t decreases with increasing batch size. As expected, batch selection makes up only a

Table 5.2.: Median experiment runtime t in seconds and ratio t/t_s of time spent for query selection in %.

k	rand-B	TopK	Clust	ClustTopK	Ensemble	FiltSim	FiltHrch
1	129s/0.1%	158s/6.7%	179s/1.9%	213s/6.3%	344s/49.2%	180s/12.4%	158s/13.4%
2	70s/0.1%	71s/6.2%	99s/1.8%	79s/5.4%	117s/34.7%	99s/10.3%	93s/7.5%
4	41s/0.0%	45s/5.3%	58s/0.3%	44s/5.9%	53s/34.2%	55s/11.0%	40s/9.1%
8	18s/0.1%	24s/5.1%	23s/0.1%	19s/7.1%	35s/45.9%	22s/11.0%	23s/7.4%
16	10s/0.0%	12s/5.0%	13s/0.2%	14s/5.8%	25s/59.6%	14s/10.4%	12s/6.6%
32	6s/0.0%	5s/5.4%	7s/0.1%	6s/6.1%	-	8s/13.4%	6s/8.8%
64	3s/0.0%	4s/3.5%	3s/0.3%	4s/4.7%	-	4s/11.0%	4s/5.6%
128	2s/0.0%	2s/3.6%	3s/0.4%	3s/2.9%	-	3s/7.8%	3s/6.4%

fraction of the overall runtime, except for *ensemble*. The experiment runtime is roughly proportional to $1/k$. The dominating factor is the number of classifier trainings T/k.

Overall, the runtime costs of classifier training are two magnitudes higher than the ones of batch selection, i.e., $c_t \gg c_s$. This confirms Hypothesis 5.2.3. We conclude that, in a one-class setting, computational costs can decrease by a factor of 10 with *TopK* without sacrificing accuracy, compared to the sequential case. More sophisticated batch selection strategies do not improve results in the one-class setting (Q2). So the sweet spot between active learning costs and classification accuracy is to use *TopK* batches with decision boundary informativeness (Q1) and setting k to a value in $[8, 16]$ (Q3). The sweet spot relies on the assumption that annotation costs are fix, see Section 5.1. However, our conclusions also hold if annotation costs decrease with the batch size until $k = 8$. Namely, increasing the batch size does not affect classification accuracy.

5.4. Summary

In this chapter, we have studied how we can reduce costs with batch active learning. Batch active learning gives way to annotate observations in parallel when classifier retraining is slow or experimental changeover costs are high. To utilize computational and annotation resources most efficiently, we strive to find a sweet spot between the costs of one-class batch active learning and the improvement in classification quality. To this end, we have presented a formal framework for batch query selection. Based on it, we have proposed several batch selection methods tailored towards one-class classification. Our general considerations and experiments have shown that selecting the top-k observations according to a sequential query strategy is a dominant choice, compared to more sophisticated strategies. This finding is different from the situations in multi-class and binary active learning. Batch one-class active learning achieves the classification quality of a sequential strategy while reducing computational costs for classifier training by one order of magnitude.

6. Query Synthesis in One-Class Active Learning

In the previous chapter, we have studied how to efficiently select queries in batches from an unlabeled pool of observations. However, there are scenarios where no unlabeled observations are available because data collection is limited or expensive. One must then synthesize queries to collect feedback. Query synthesis is a difficult task in a high-dimensional data space. Random queries or structured queries with grids are inefficient because they do not take acquired annotations into account. So, the trade-off we will investigate in this chapter is how one can reduce the costs to generate a synthetic query with high informativeness against how the query improves the classification quality.[1]

In this chapter, we study how we can use query synthesis to expand the classifier knowledge beyond an initial sample of inliers that form a single connected region in an unbound data space. An example application is a compute-intensive simulation for design-space exploration [CF17a; CF17b; LM12], where the collected data set often is small and biased, i.e., does not represent the real data distribution well. We call this the *domain expansion problem*. Figure 6.1 is an illustration. We start with a small initial sample of mostly inliers and seek to expand the classifier boundaries of a one-class classifier towards the real boundaries.

Existing approaches for query synthesis are for the multi-class setting and require observations from all classes [Jos11; Wan+15]. They cannot synthesize queries in directions where no observations exist, as it is the case with domain expansion. Next, methods for design space exploration [CF17a; CF17b; LM12] and reliability-based design optimization [LJ08] use the prediction uncertainty of a binary classifier for query generation. But this uncertainty may not be obtainable from a one-class classifier. So we target a method for domain expansion that performs query synthesis in one-class active learning.

Challenges Query synthesis in the one-class settings is difficult for three reasons.

High dimensionality: Existing one-class query strategies use the unlabeled observations as query candidates and query the one with the highest informativeness [TEB21]. Without such observations, the volume of a high-dimensional data space is too large for a random candidate generation. This calls for a more selective candidate placement, e.g., near the current observations.

[1] The remainder of this chapter bases on the article [EB20]: Adrian Englhardt and Klemens Böhm. “Exploring the Unknown – Query Synthesis in One-Class Active Learning”. In: *SIAM International Conference on Data Mining (SDM)*. SIAM. Jan. 2020, pp. 145–153. DOI: 10.1137/1.9781611976236.17. We have shortened the text to be less repetitive, applied minor corrections, and changed the formatting and notation so that it is in line with the format and structure of this thesis.

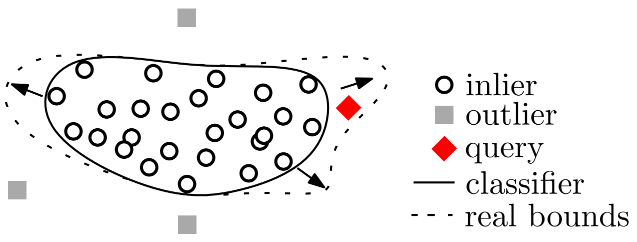


Figure 6.1.: Domain expansion with query synthesis.

Data Set Pollution: During query synthesis, artificially generated queries and their feedback are inserted into the original data set. This affects the data density and the neighborhoods. To illustrate, querying in the same region several times increases the data density where it may be low in reality. However, several one-class classifiers such as Nearest Neighbor Description [Tax02] and pool-based query strategies [Gha+11a; Gha+11b; Gör+13; YWF18] rely on these data characteristics. Query synthesis may then misguide the classifiers and query strategies.

Evaluation standards: Evaluation standards from pool-based one-class active learning do not carry over to query synthesis. On the one hand, an evaluation needs a simulated annotator, a so-called oracle. It is unclear how to simulate an oracle that can provide an answer for an arbitrary query with query synthesis. On the other hand, one must compare a new query synthesis strategy to a baseline. But the existing baseline to select a random query [Gör+13] is infeasible, and sampling from a potentially unbounded data space is ineffective. In this chapter, we study how to construct a more realistic baseline, by deriving boundaries from the available data.

Contributions This chapter features an approach for query synthesis for one-class active learning. It performs domain expansion based on initial observations from one class. Our contributions are as follows:

Framework: We generalize pool-based one-class active learning and propose a framework for query synthesis in one-class active learning (SYNOCAL). The framework is defined by the initial labels, a one-class classifier and a query synthesis strategy. We frame query synthesis as an optimization problem. This allows for efficient query synthesis, given any query strategy that quantifies the informativeness of a candidate. We then use meta-heuristics to solve this problem even for high-dimensional data.

Domain Expansion Stragegy (DES): We propose a new query-synthesis strategy to expand the classifier knowledge beyond the initial, potentially biased, data sample by leveraging work on adaptive data shifting [Wan+18]. Our novel strategy allows us to learn in areas of uncertainty where no observations exist.

Evaluation standards: We propose evaluation standards for one-class query synthesis. We introduce three ways how to define an oracle using synthetic and real-world data. Additionally, we derive two realistic baseline strategies for query synthesis from existing artificial outlier generation algorithms [AZL06; Dés+13].

In the evaluation, we perform extensive experiments for different domain expansion problems. We show that our new query-synthesis strategy outperforms all alternatives in settings with few inlier labels. We publish the implementation of our framework to reproduce our results.[2]

6.1. Framework

In this section, we present our *framework*. We formalize one-class query synthesis and present several query strategies that serve as baselines later. The framework is the fundament for our novel domain expansion strategy. Yet, it is general and facilitates research on one-class query synthesis strategies.

Definition 10 (Query synthesis strategy) *Given a data set X with label pools $\mathcal{U}$, $\mathcal{L}$, and a classifier C, a query synthesis strategy QSS is a function of type $QSS : C, \mathcal{U}, \mathcal{L} \rightarrow Q$ where $\forall q \in Q : q \in \mathcal{X}$ is an artificial query for feedback collection.*

Definition 11 (Query synthesis) *Query synthesis is a method to improve the classification by a classifier C on data X by acquiring feedback from an oracle O on queries generated by a query synthesis strategy QSS.*

In a one-class setting, classifier and query synthesis strategy must cope with imbalanced class distributions. To this end, we propose a new framework called SYNOCAL to perform query SYNthesis in One-Class Active Learning. SYNOCAL consists of three elements: (1) active learning scenario, (2) one-class classifier, and (3) query synthesis strategy.

6.1.1. Active Learning Scenario

Previous research on one-class active learning relies on assumptions that affect the interaction of a user with the active learning system or confine the choice of the classifier and the query strategy [TEB19; TEB21]. The active learning scenario specifies initial class label availability. We have already introduced the three scenarios: unsupervised, semi-supervised, and supervised in Section 2.2.1 Pool-based query strategies only work with the unsupervised and semi-supervised scenario, while query synthesis works with any setup.

6.1.2. One-Class Classifier

The second element of our framework is the one-class classifier (OCC). A one-class classifier outputs a decision function, see Definition 4.

6.1.2.1. Requirements

We derive two requirements that a one-class classifier must meet for query synthesis:

- *Semi-supervised:* The classifier must be able to learn from feedback. So the classifier must be semi-supervised to use labeled and unlabeled observations.

[2] https://www.ipd.kit.edu/des

- *Boundary-based:* To avoid *data set pollution*, the classifier must not use densities or neighborhoods. This calls for a classifier that learns tight enclosing boundaries around the inliers.

6.1.2.2. One-Class Classifiers Choice

Research proposed various one-class classifiers [KM14] with many specific variants, like classification with weights [Zha+09] or in subspaces [TB19]. Following our requirements leaves us with classifiers based on SVDD [TD04]. The original SVDD is unsupervised and requires additional help to learn from feedback, see our discussion in Section 2.4.2. SVDDneg and SSAD are semi-supervised extensions of SVDD that can use label information. SVDDneg and SSAD meet all our requirements for one-class query synthesis.

6.1.3. Query Synthesis Strategy

The third element of our framework is the query synthesis strategy. Such a strategy uses the classifier, available observations, and labels to generate an artificial query. We focus on sequential query synthesis strategies, i.e., where $|\mathcal{Q}| = 1$, and consider batch query synthesis future work. We differentiate between direct and indirect *QSS*.

Direct QSS The first type directly synthesizes the query given the classifier, available observations, and labels. Baseline strategies that randomly generate a query are direct strategies. To define these baselines, we follow the approaches for artificial outlier generation [AZL06; Dés+13; TD01] and bound the data space with a hyper-rectangle that encloses all observations. A hyper-rectangle H is defined by two boundary vectors $a, b \in \mathbb{R}^d$, usually the minima and maxima along each dimension. H_ε stands for such an expanded hyper-rectangle where $\varepsilon \in \mathbb{R}^+$ is the expansion, e.g., $H_{0.1}$ for a 10% expansion. This gives way to our two baselines.

- $DQSS_{\text{rand}}$: Samples a random query $\mathcal{Q} \in H_\varepsilon$.
- $DQSS_{\text{rand-o}}$: Samples a query $\mathcal{Q} \in H_\varepsilon$ with $f(\mathcal{Q}) > 0$, i.e., classified as outlier by C.

Indirect QSS The second type, on the other hand, first defines the informativeness for an arbitrary observation which is then optimized. τ quantifies the expected information gain, i.e., how valuable an arbitrary observation $x \in \mathcal{X}$ is, see Definition 8. One can then optimize $\tau(x)$ to obtain the optimal synthetic query:

Definition 12 (Query synthesis optimizer) *Given an* indirect *query synthesis strategy with a function $\tau(x)$, a query synthesis optimizer (QSO) yields the optimal query $\mathcal{Q}$ by computing $\mathcal{Q} = \arg\max_{x \in \mathcal{X}} \tau(x)$.*

There is a variety of optimization algorithms, and the optimal choice depends on $\tau(x)$. Preliminary experiments of ours have shown that the derivative-free meta-heuristic optimizer DXNES [Fuk+11] offers high solution quality in reasonable compute time.

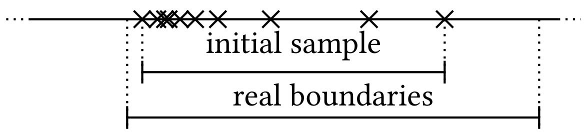


Figure 6.2.: 1-dimensional domain expansion problem.

$IQSS_{DB}$: Querying observations close to the decision boundary [Gör+09] is a pool-based strategy (cf. Equation 2.12) that we can adapt as an indirect query synthesis strategy. Given the decision function f of a classifier, $IQSS_{\mathrm{DB}}$ defines $\tau_{\mathrm{DB}} = -|f(x)|$. The strategy gives high informativeness to queries on the margin between the two classes where the classification uncertainty is highest.

6.2. Domain Expansion

This section deals with the *domain expansion* problem. We first formalize it and then present our novel query synthesis strategy.

6.2.1. Formalization

We define C^* as the optimal one-class classifier for the data space $\mathcal{X}$. One starts with a sample of inliers $\mathbf{X}$ drawn from the data space $\mathcal{X}$ and seeks to learn a classifier C that matches C^*. To do so, we use query synthesis and perform one-class active learning. The challenge now is the definition of a query synthesis strategy to achieve this. We split this problem into two subproblems.

Identification of exploration direction (IED) The first challenge is the identification of an exploration direction. Figure 6.2 gives a 1-dimensional example with an initial sample and the real boundaries. Here, one might prefer exploring to the left, due to the higher density of the sample in this area. However, first exploring to the right is more beneficial. The space uncovered by the initial sample is larger to the right than to the left when looking at the hidden real boundaries. Generally, the initial sample does not give any information on the direction to choose. This is because it can be biased and may not represent the actual distribution of the data. Initially, one has to rely on guessing. In high dimensional data spaces where the boundaries of the initial sample match the real boundaries well, and only a few directions would yield an improved boundary, it becomes more difficult to guess the "improving" directions. So more queries are necessary to check all directions. When an oracle has labeled an outlier in a direction d_{out}, one should either explore in different areas or decrease the vector length of d_{out} to query between the outlier and the inliers.

Identification of exploration magnitude (IEM) One has to define how far to explore in a given direction. Initially, one only has inlier samples. One option again is to guess the exploration magnitude. But different value ranges require different magnitudes. To illustrate, exploring a 1-dimensional data set with a hidden value range of $[1, 100]$ and initial sample value range of $[5, 80]$ with an exploration magnitude of 0.001 requires many queries. So one should infer the magnitude from the initial sample. If one continuously retrieves inlier feedback for queries in one direction, one could gradually increase the magnitude to explore the direction more quickly. When outlier labels become available, one should decrease the magnitude in the outlier direction to query between the outlier and the known inliers.

6.2.2. Our Solution

We now present our solution and say how we address the two subproblems. We propose Domain Expansion Stragegy (DES) that performs query synthesis. We briefly describe the intuition behind our approach first and then present its components.

Intuition Our new query strategy is an indirect query synthesis strategy with an informativeness function. Following our previous discussion, all exploration directions initially have the same weight. Since the dimensionality and the shape of the inlier region vary between data sets, we derive these directions directly from the SVDD-based classifier. Without loss of generality, all exploration directions are orthogonal to the decision boundary. When restricting them to length ε, they form a hull around the initial sample. This hull corresponds to the decision boundary shifted away from the initial sample by some ε. Additionally, we must consider any labeled outliers and decrease the exploration magnitude in directions with outliers, i.e., shift the boundary less far. To this end, we propose SVDDnegEps that learns such a shifted decision boundary while respecting outlier labels. We propose a parametrization method to compute the exploration magnitude, i.e., the shift ε. Finally, we introduce a method to quickly prune the search space of the exploration directions and magnitudes by decreasing the informativeness in areas with outliers. This pruning enforces query diversity over multiple active learning cycles.

SVDDnegEps Our novel one-class classifier called Support Vector Data Description with negative Examples and Epsilon Shift (SVDDnegEps) learns a shifted decision boundary where outliers stay outside of the hypersphere. SVDDnegEps enforces an extended boundary by injecting the exploration magnitude, i.e., the shift amount ε, into the constraints of the SVDDneg optimization problem (cf. Equation 2.9):

$$\begin{aligned} \underset{R,a,\xi}{\text{minimize}} \quad & R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l \\ \text{subject to} \quad & \|\Phi(x_i) - a\|^2 + \varepsilon \le R^2 + \xi_i, \ \forall i \\ & \|\Phi(x_l) - a\|^2 \le R^2 - \xi_l, \ \forall l \\ & \xi_i, \ge 0, \ \forall i \\ & \xi_l, \ge 0, \ \forall l \end{aligned} \tag{6.1}$$

The index i refers to observations in $\mathcal{U} \cup \mathcal{L}_{in}$ and l to labeled outliers in $\mathcal{L}_{out}$, ξ_i and ξ_l are the corresponding slack variables and $C_1, C_2 \in [0, 1]$ the cost parameters. After solving the problem, we have a fixed center a and radius R that define the enclosing hypersphere. We can then use Equation 2.4 to get a decision function.

Parametrization We use the artificial outlier generation approach from [Wan+18] to infer ε. The approach generates artificial outliers to tune the hyper-parameters of a SVDD. The generation process places artificial outliers around the data by shifting boundary observations along their negative data density estimated from the neighborhoods. So the approach adapts to arbitrary value ranges. Our idea is to use these artificial outliers to quantify the maximum magnitude of how far one should explore. We can measure the distance of these outliers to the decision boundary of an OCC with the decision function f of the classifier. Given f and artificial outliers $\mathbf{X}_{\text{out}}$, the shift is:

$$\varepsilon = \max_{x_i \in \mathbf{X}_{\text{out}}} f(x_i) \tag{6.2}$$

This bounds the maximum exploration to the farthest artificial outlier generated from the initial sample.

Domain Expansion Strategy Our novel query strategy $IQSS_{\text{DES}}$ is similar to the decision boundary strategy $IQSS_{\text{DB}}$, except that it queries on the shifted decision boundary of a SVDDnegEps. f_ε stands for this decision function, to make the difference to f explicit. The informativeness of $IQSS_{\text{DES}}$ is:

$$\tau_{\text{DES}} = -|f_\varepsilon(x)|. \tag{6.3}$$

We can then use our framework to perform query synthesis with an optimizer.

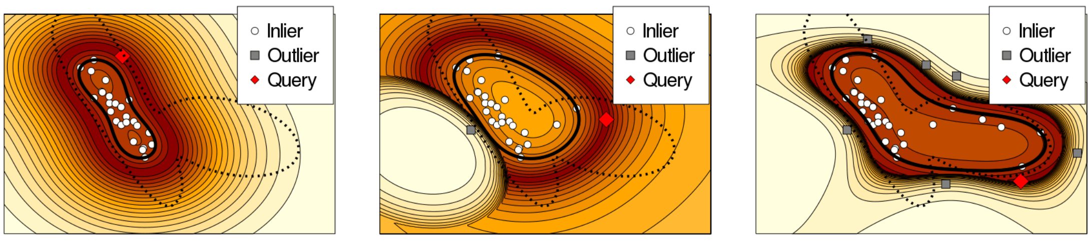
Search space pruning In high dimensional data spaces, one should punish areas with negative feedback and should encourage exploration elsewhere. To this end, we propose a method to quickly prune the search space of the exploration directions and magnitudes. This extension is general and works with an arbitrary indirect query synthesis strategy (IQSS) that defines an informativeness. The idea is to reduce the informativeness in areas with outliers automatically. To this end, we train a binary SVM on the available data. By using a binary SVM, one can quantify the distance to the decision boundary in the same kernel space as for the SVDD-based classifier. So no value-range adjustment is needed, and we can directly combine the distance to the SVM and the OCC. The decision function for an SVM is f_{SVM}. Again, an outlier x yields $f_{\text{SVM}}(x) > 0$, and an inlier x gives a value $f_{\text{SVM}}(x) \leq 0$. Given an IQSS with informativeness τ, we then define a combined informativeness function τ^*:

$$\tau^*(x) = \tau(x) - \max(f_{\text{SVM}}(x), 0) \tag{6.4}$$

This modification of τ punishes areas with negative feedback (IED) and reduces the exploration magnitude in the direction of outliers (IEM).

6.2.3. Example

We now illustrate how our query synthesis strategy with search space pruning, dubbed $IQSS^*_{\text{DES}}$, works, cf. Figure 6.3. Figure 6.3a shows the start of the active learning cycle. Here, the black line is the decision boundary fitted by a SVDDneg, and the dashed line is the oracle, i.e., the target boundary that we want to learn. The initial sample does not



(a) Iteration 1 (b) Iteration 5 (c) Iteration 15

Figure 6.3.: Expanding the decision boundary of a SVDDneg with our new $IQSS^{*}_{\mathrm{DES}}$ with search space pruning.

cover the full valid space and misses a large area to the right. The heat map indicates the informativeness – the darker, the higher. After five queries, we have expanded the boundary and have acquired the first outlier. The pruning then punishes the area around the outlier, see Figure 6.3b, and the query strategy starts exploring other areas. The state after 15 queries is visualized in Figure 6.3c. Our method has already approximated the real decision boundary quite well.

6.3. Evaluation

Previous work uses benchmark data sets with ground truth and performs pool-based active learning. However, an evaluation is more difficult in the context of query synthesis because the oracle must provide a label for arbitrary queries. We see three options how to evaluate one-class query synthesis.

Synthetic Data The first option relies on synthetic data. Here, one defines a function that acts as the oracle. For instance, one can use a multivariate distribution. The oracle function then labels an observation as an outlier if the density falls under a threshold. Additionally, the distribution allows to generate an arbitrary volume of test data to evaluate the classifier.

Ground Truth Fitting The second option relies on existing one-class data sets. One can fit a classifier to the ground truth data to obtain an oracle. The fitted classifier can then serve as an oracle to answer arbitrary queries in the active learning cycle. Example classifiers are SVDD or also basic classifiers such as SVM or kNN classifier. There are two problems with this approach. First, the oracle is limited by the capabilities of what the underlying classifier can learn. So one may not achieve a perfect fit to the data, and queries may yield wrong feedback. Second, one-class outlier benchmark sets are unbalanced. The oracle classifier has high uncertainty in these sparse areas. Additionally, we also have a very limited number of outliers for testing. Active learning may improve a classifier, but there is no test data to reflect this improvement.

Hybrid Query Synthesis The third option avoids crafting an oracle by only allowing queries for existing observations. With query synthesis, one can use a *hybrid approach*:

One first uses any strategy to synthesize a query Q. Then the observation closest to Q under some distance function *dist* becomes the query:

$$Q^* = \arg\min_{x \in \mathrm{X}} dist(x, Q) \tag{6.5}$$

This hybrid approach generally avoids the problem of awkward queries [BL92] and has already been used to evaluate multi-class query synthesis [Wan+15]. However, we see a problem in the one-class setting with this option. The selected unlabeled observation may be far away from the synthesized query due to data sparsity. Then, a query strategy may have preferred a completely different observation when computing the informativeness of the unlabeled observations, compared to the one chosen by the hybrid approach.

6.4. Experiments

In this section, we evaluate our framework and our new query synthesis strategy on various domain expansion problems. In the first part, we use synthetic data of different dimensionality. In the second part, we assess the performance of our method on common real-world data sets for outlier detection [Cam+16]. We have implemented the query synthesis framework and the benchmark setup in Julia [Bez+17]. Our implementation, the raw results of all settings and notebooks to reproduce experiments and evaluation are publicly available at `https://www.ipd.kit.edu/des`.

6.4.1. Setup

We first present the setup of our experiments.

Data and Oracle We use synthetic and real-world data sets for our experiments. We present the data sets and how we define the oracle.

Synthetic data: Gaussian mixtures allow to flexibly generate synthetic domain expansion problems. We generate 100 random Gaussian Mixtures for each combination of 3, 5 and 7 components and 2, 4, 6, 8 and 10 dimensions. To bias the initial sample, we then choose the initial sample only from one of the components. Given the Gaussian mixture with density function $p(x)$, we use a threshold $t = 0.1$ to define the oracle: Observations are inliers if $p(x) \geq t$ and outliers otherwise. To evaluate the classifier, we generate 1000 inliers and outliers each in addition to the initial sample.

Real-world data: For the real-world experiments, we use the publicly available outlier benchmark data sets from [Cam+16], see Section 2.5.1. We use the normalized and deduplicated version of 15 different data sets. They have different sizes (80–7129 observations) and outlier ratios (4–75%). To limit the search space, we perform feature selection to extract the 5 most meaningful features using mutual information. This is in line with the discussion in Section 6.2, where we argue that guessing an exploration direction at the start of the learning becomes more difficult with increasing dimensionality. To choose the initial sample, we perform a neighborhood walk: We first randomly select one observation and iteratively add the nearest neighbor to the initial sample until we reach the desired

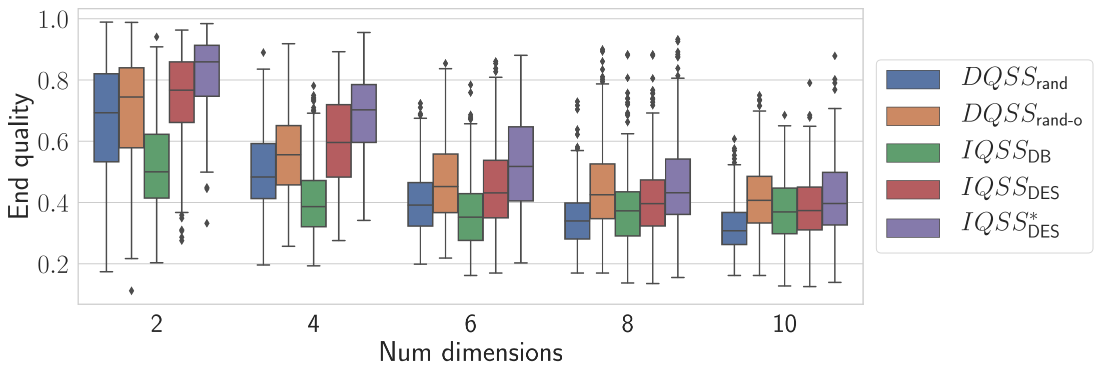


Figure 6.4.: End quality for synthetic data sets with different dimensionality.

sample size. In this way, we bias the initial sample into one area. We run 10 resamples of the initial sample for each data set. For the oracle, we perform *ground truth fitting*. Preliminary experiments have shown that a binary SVM performs better as an oracle than a one-class classifier when using all labels. We additionally generate synthetic inliers and outliers with the method proposed in [Wan+18] to tune the kernel parameter of the oracle SVM with cross-validation.

Parametrization To simulate a small biased sample to expand from, we set the number of initial observations to 25. We then run 100 active learning iterations with our framework. As a model, we use the SVDDneg [TD04] with a hard margin $C_1, C_2 = 1.0$ and tune parameter of the Gaussian kernel (cf. Equation 2.8) with the method proposed in [Wan+18]. We use the same parameters for the SVDDnegEps of $IQSS_{\text{DES}}$. For the baseline query strategies $DQSS_{\text{rand}}$ and $DQSS_{\text{rand-o}}$, we set the hyper-rectangle expansion to $\varepsilon = 0.1$ as in [AZL06]. For the indirect query synthesis strategies we use *distance-weighted exponential natural evolution strategy* [Fuk+11] (DXNES) with the default parameters implemented in [Fel18]. We fix the optimization boundaries to an extended hyper-rectangle $H_{1.0}$ with 100% expansion beyond the labeled inliers.

Evaluation metrics To evaluate the results, we calculate the Matthews Correlation Coefficient (MCC) to compare the classifier prediction for the test data with the ground truth for each iteration. We then calculate the *End Quality* (EQ) [TEB21] to quantify the performance of the classifier after the active learning. See Section 2.5.2 for details on how to calculate MCC and EQ.

6.4.2. Results

We now present and discuss the results of our experiments to show the superiority of our novel query strategy on different domain expansion problems.

Table 6.1.: Median end quality on real world data sets.

Data set	rand	rand-o	DB	DES	DES*
Annthyroid	0.05	0.07	0.03	0.04	**0.08**
Arrhythmia	0.18	0.24	0.17	0.18	**0.27**
Cardio	0.09	0.14	0.06	0.07	**0.20**
Glass	0.13	0.16	0.11	0.20	**0.24**
HeartDisease	0.22	0.27	0.18	0.26	**0.28**
Hepatitis	0.28	**0.31**	0.27	0.19	0.21
InternetAds	0.22	0.65	0.13	0.14	**0.73**
Ionosphere	0.22	0.26	0.22	0.27	**0.49**
PageBlocks	0.10	0.17	0.05	0.06	**0.22**
Parkinson	0.62	0.62	**0.64**	0.60	0.42
Pima	0.14	0.13	**0.15**	0.14	0.14
SpamBase	0.19	**0.46**	0.13	0.12	0.44
Stamps	0.15	**0.26**	0.11	0.14	0.22
WPBC	-0.03	-0.06	-0.02	-0.05	**0.01**
Wilt	-0.08	-0.06	-0.06	**-0.05**	-0.08

Synthetic Data We first run our framework with two baselines $DQSS_{\text{rand}}$, $DQSS_{\text{rand-o}}$, indirect query synthesis strategies $IQSS_{\text{DB}}$, $IQSS_{\text{DES}}$, and with search space pruning $IQSS^*_{\text{DES}}$ on the synthetic data. Figure 6.4 shows the end quality for different dimensionalities. Our framework allows to improve the classifier from a small initial sample even in the absence of unlabeled observations. The end quality decreases with increasing data dimensionality. This is because finding a helpful exploration direction becomes more difficult and thus requires more queries – here we are using a fixed number of 100 iterations. Our proposed approach with search space pruning $IQSS^*_{\text{DES}}$ outperforms or is at least on par with all other strategies. The baseline strategy $DQSS_{\text{rand-o}}$ proposed earlier yields competitive results. This is similar to the results in [TEB21] where a random baseline for pool-based one-class learning outperforms other strategies on one-third of the data sets. Querying directly on the decision boundary with $IQSS_{\text{DB}}$ does not yield any improvement. $IQSS_{\text{DB}}$ mostly queries inliers near the initial sample and does not gain insights beyond the sample.

Real-World Data We run the same setup on the real-world data sets. Table 6.1 shows the median end quality for all the proposed query synthesis strategies on different data sets. The end quality with active learning varies from data set to data set. For the WPBC and Wilt data set, no method yields any considerable improvement, resulting in a low end quality. Except for them, our proposed method $IQSS^*_{\text{DES}}$ achieves the best results overall, winning in 9 out of 13 data sets. Model quality significantly increases from the initial set with $IQSS^*_{\text{DES}}$. Similarly, to the synthetic evaluation, our method benefits from search space pruning. One can see this by comparing the results of $IQSS_{\text{DES}}$ and $IQSS^*_{\text{DES}}$.

6.5. Summary

This chapter studies the domain expansion problem where one strives to improve a one-class classifier by extending a small initial data sample with additional training data. To this end, we have proposed a general framework that formalizes query synthesis in the one-class setting as an optimization problem. We have then proposed a novel domain expansion strategy that explores the data space and does away with the bias of small samples. The method includes pruning of the considered exploration space that diversifies the queries considerably. Evaluating query synthesis strategies requires an oracle that can provide an answer for arbitrary queries. In this chapter, we derive three ways to use synthetic and real-world data to simulate such an oracle. Comprehensive experiments on synthetic and real-world domain expansion problems demonstrate that our method efficiently expands the knowledge of a classifier beyond a biased sample and outperforms realistic baselines.

Part III.

Conclusions

7. Conclusions

An active learning system consists of several components that work together in an iterative process. Because the components are strongly intertwined, it is difficult to configure an active learning system in practice. With most applications, there are budgets that limit certain parts of the system, e.g., the runtime of a classifier or the number of queries. When the system exhausts budgets, the operator has to reconfigure the system and lower the costs. This often implies a loss of classification quality. So, when configuring an active learning system in a new application, the operator has to trade off costs against quality. In this thesis, we have discussed the various costs of active learning and have reviewed approximation methods to trade off costs against quality in Chapter 2. We have then taken a closer look at three important trade-offs and have proposed conceptual and technical contributions for each of them. We summarize our contributions in the following and state how they facilitate the configuration of a one-class active learning system.

First, we have analyzed how one can reduce classification training costs of the one-class classifier SVDD with sampling in Chapter 4. Sampling selects a subset of the data for training and thereby lowers the runtime. Existing sampling methods are heuristics and do not come with any guarantee on the resulting classification quality. In this thesis, we have taken a more principle approach where sampling does not reduce the classification quality compared to an SVDD trained on the complete data set. Therefore, we have formalized sampling as an optimization problem and have added constraints that guarantee the original decision boundary. To solve this optimization problem efficiently, we have then proposed the iterative algorithm RAPID. RAPID has three key advantages compared to sampling methods from literature. First, it does not require any additional parameters beyond the ones already required by SVDD. Second, the constraints guarantee that RAPID yields the same classification quality as a classifier trained on the complete data set. Third, RAPID is efficient and consistently produces small samples. With comprehensive experiments, we have shown that RAPID substantially reduces classification training costs by one order of magnitude compared to training an SVDD on the complete data set.

Second, we have studied querying in batches instead of single observations for one-class active learning in Chapter 5. Batch queries reduce the classification training costs because they delay the classifier update. Batch queries also have an advantage over sequential querying when multiple oracles are available in parallel or the changeover costs between experiments are high. We have studied how to construct batch queries for one-class active learning. In a first step, we have formalized batch queries as an optimization problem. Our formalization enables us to study the cost-quality trade-offs of batch selection. In a second step, we have proposed different strategies to construct batches based on the informativeness, representativeness, and diversity criteria of a batch. Afterwards, we have compared these heuristics in an empirical study. Our evaluation has shown that a simple top-k batch selection outperforms all other more sophisticated strategies. Tuning the batch size k gives the operator control over the costs of active learning. We have identified a sweet spot where we can retain the classification quality of a sequential strategy and reduce classification training costs by one order of magnitude.

Third, we have proposed query synthesis for one-class active learning in Chapter 6. Query synthesis allows to collect feedback at any point in the data space without being restricted by a pool of query candidates. However, finding queries that improve the classification quality is difficult, in particular in a unbound data space. To enable efficient query synthesis for one-class active learning, we have made two contributions. First, we have proposed a framework for query synthesis that efficiently searches the data space with black-box optimizers. Second, we have established evaluation standards to simulate oracles with synthetic and real-world data to allow an experimental comparison of query synthesis strategies. Based on our two contributions, we have then studied one specific one-class active learning problem: the domain expansion problem. To this end, we have developed a novel query synthesis strategy DES to expand the decision boundary of a one-class classifier. With DES and our framework, one can improve the generalization of a one-class classifier to new data efficiently.

8. Outlook

In this thesis, we focus on the cost-quality trade-offs an operator has to take when configuring an active learning system for outlier detection. Understanding these trade-offs is crucial to realize one-class active learning systems in practice. Our contributions focus on a few trade-offs, but there are other, still unexplored, cost-quality trade-offs. In the following, we elaborate on open research questions that are closely related to our contributions.

Annotation cost dependencies between queries In Chapter 5, we have studied how batch queries reduce classification training costs. We have shown that a high diversity of the queries in a batch yields high classification quality. However, there are scenarios where similarity between queries reduces costs. Query similarity can, for example, reduce the change-over costs between experiments [GK11]. Our theoretical formalization of batch queries in Section 5.1.3 already considers this cost dependency. But none of the methods we have proposed for batch construction considers the annotation cost dependencies. They assume that the annotation step is independent for each query, i.e., there is no annotation cost dependency between the observations of one batch. It is an open question of how to extend batch-construction methods to be more cost-sensitive. Technically, one has to identify the cost dependencies between queries in real-world applications and model them. The follow-up question is then how one can incorporate such a dependency model into the batch construction.

Multiple noisy oracle Throughout this thesis, we have assumed that there is a single oracle that correctly annotates all the queries. Several challenges arise when annotations are incorrect and when multiple annotators are available.

First, one has to study how to model imperfect oracles that produce incorrect annotations in the active learning system. A simple model is that oracles return incorrect annotations based on some fixed probability [Set11]. But there can be other reasons why annotation quality deteriorates. For example, fatigue reduces the annotation quality of human oracles. The annotation quality may also depend on the complexity of the query, and sparse regions of the data set where outliers reside may be more difficult to annotate. It is an open question if one can cope with imperfect oracles by learning an *uncertainty* model of an oracle from the queries and their annotations. Existing studies [CS17; SPI08] are mostly theoretical, and it is yet unclear if they apply to one-class active learning. So a follow-up question is how one can use an oracle uncertainty model in an active learning system. One option is to use the uncertainty to weight the observations during classifier training. Another idea is to incorporate oracle uncertainty into the query selection. On the one hand, one can query areas with high oracle uncertainty multiple times to receive more reliable annotations. On the other hand, querying close to labeled observations can improve the oracle uncertainty model. In both cases, the additional queries increase the costs. Therefore, new cost-quality trade-offs arise with imperfect oracles.

Second, the configuration of an active learning system becomes even more difficult if there are multiple oracles that differ in their annotation quality and cost. This is relevant when one seeks to collect robust annotations with crowdsourcing [KOS14]. Another scenario is choosing between a cheap simulation with low annotation quality or an expensive real-world experiment with high annotation quality. One must then jointly consider the individual oracle uncertainty and annotation costs when choosing an oracle during query selection. Multiple imperfect oracles with varying annotation quality and costs have not yet been studied with one-class active learning.

Query presentation costs for human oracles In Section 2.3.3 we have discussed that *query presentation costs* arise when working with human oracles. Providing auxiliary information to the query in the annotation interface may reduce the annotation costs and increase annotation quality. This idea has sparked research on Visual Interactive Analytics (VIA) and Visual Inter-Active Labeling (VIAL). These research areas study which kinds of visualization assist a human to interactively explore and label a data set, e.g., to find outliers [Ber+18; Bög+17; Lei+18; Lin+17; Wil18]. However, computing these visualization requires additional computational resources. The cost-quality trade-off of a sophisticated query presentation is still unexplored for active learning. We have proposed a prototype implementation and a road map to study these cost-quality trade-offs for one-class active learning with a user study [TEB19]. The insights from such a study will help understand how to support human annotations efficiently.

Streaming-based OCAL In this thesis, we have studied one-class active learning with pool-based querying and query synthesis. As explained in Section 2.3.2, the third query scenario is *stream-based selective sampling*. It is unexplored how the existing solutions for one-class active learning transfer to a stream setting. Cost budgets may be particularly tight in a stream setting, which challenges all parts of an active learning system: classifier updates, query selection, and annotation. It is an open question if existing algorithms yield a sufficiently high classification quality when trading costs against quality to adhere to these budgets.

To conclude, this thesis advances the state-of-the-art on one-class active learning for outlier detection. Our conceptual and technical contributions help to trade off costs against quality when realizing one-class active learning systems in practice. Finally, we have discussed several open research questions that emerge from our research. We deem them interesting, and they may contribute to the overarching goal of simplifying the configuration of robust and efficient one-class active learning systems.

Bibliography

[AAR96] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. “A Linear Method for Deviation Detection in Large Databases”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 1996, pp. 972–981.

[Ach+10] Elke Achtert, Hans-Peter Kriegel, Lisa Reichert, Erich Schubert, Remigius Wojdanowski, and Arthur Zimek. “Visual Evaluation of Outlier Detection Models”. In: *Database Systems for Advanced Applications (DASFAA)*. Springer, 2010, pp. 396–399. DOI: 10.1007/978-3-642-12098-5_34.

[AFR97] Emin Aleskerov, Bernd Freisleben, and Bharat Rao. “Cardwatch: A neural network based database mining system for credit card fraud detection”. In: *Computational Intelligence for Financial Engineering (CIFEr)*. IEEE, 1997, pp. 220–226. DOI: 10.1109/cifer.1997.618940.

[Agg15] Charu C Aggarwal. *Outlier analysis*. Springer, 2015, pp. 237–263. DOI: 10.1007/978-3-319-47578-3.

[AGZ15] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. “Efficient Active Learning of Halfspaces via Query Synthesis.” In: *Conference on Artificial Intelligence (AAAI)*. 2015.

[Ala+20] Shamshe Alam, Sanjay Kumar Sonbhadra, Sonali Agarwal, P. Nagabhushan, and M. Tanveer. “Sample reduction using farthest boundary point estimation (FBPE) for support vector data description (SVDD)”. In: *Pattern Recognition Letters* 131 (2020), pp. 268–276. DOI: 10.1016/j.patrec.2020.01.004.

[Alp20] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, 2020. ISBN: 9780262043793.

[AMS02] Dimitris Achlioptas, Frank McSherry, and Bernhard Schölkopf. “Sampling techniques for kernel methods”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2002.

[AZL06] Naoki Abe, Bianca Zadrozny, and John Langford. “Outlier detection by active learning”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2006. DOI: 10.1145/1150402.1150459.

[BB08] Enrico Blanzieri and Anton Bryl. “A survey of learning-based techniques of email spam filtering”. In: *Artificial Intelligence Review* 29.1 (2008), pp. 63–92. DOI: 10.1007/s10462-009-9109-6.

[BBJ15] V. Barnabé-Lortie, C. Bellinger, and N. Japkowicz. “Active Learning for One-Class Classification”. In: *International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2015. DOI: 10.1109/icmla.2015.167.

[Ber+18] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. “VIAL: a unified process for visual interactive labeling”. In: *The Visual Computer* 34.9 (2018), pp. 1189–1207. DOI: 10.1007/s00371-018-1500-3.

[Bez+17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM Review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671.

[BKB07] András Bánhalmi, András Kocsor, and Róbert Busa-Fekete. “Counter-Example Generation-Based One-Class Classification”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD).* Springer. 2007, pp. 543–550. DOI: 10.1007/978-3-540-74958-5_51.

[BL92] Eric Baum and Kenneth Lang. “Query learning can work poorly when a human oracle is used”. In: *International Joint Conference on Neural Networks (IJCNN).* 1992.

[BM08] Anirban Basudhar and Samy Missoum. “Adaptive explicit decision functions for probabilistic design and optimization using support vector machines”. In: *Computers & Structures* 86.19-20 (2008), pp. 1904–1917. DOI: 10.1016/j.compstruc.2008.02.008.

[Bög+17] Markus Bögl, Peter Filzmoser, Theresia Gschwandtner, Tim Lammarsch, Roger A Leite, Silvia Miksch, and Alexander Rind. “Cycle Plot Revisited: Multivariate Outlier Detection Using a Distance-Based Abstraction”. In: *Computer Graphics Forum.* Vol. 36. 3. Wiley Online Library. Wiley, 2017, pp. 227–238. DOI: 10.1111/cgf.13182.

[Bre+00] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. “LOF: identifying density-based local outliers”. In: *SIGMOD.* ACM, 2000, pp. 93–104. DOI: 10.1145/342009.335388.

[Bri03] Klaus Brinker. “Incorporating Diversity in Active Learning with Support Vector Machines”. In: *International Conference on Machine Learning (ICML).* ACM. 2003.

[Bry+06] Brent Bryan, Robert C Nichol, Christopher R Genovese, Jeff Schneider, Christopher J Miller, and Larry Wasserman. “Active learning for identifying function threshold boundaries”. In: *Advances in Neural Information Processing Systems (NIPS).* 2006.

[BSM19] M. Bunse, A. Saadallah, and K. Morik. “Towards Active Simulation Data Mining”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) Workshops.* 2019.

[BWS04] Gökhan H Bakır, Jason Weston, and Bernhard Schölkopf. “Learning to find pre-images”. In: *Advances in Neural Information Processing Systems (NIPS).* 2004.

[Cam+16] Guilherme Campos, Arthur Zimek, Jörg Sander, Ricardo J. G. B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”. In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 891–927. DOI: 10.1007/s10618-015-0444-8.

[Car+17] Thiago Cardoso, Rodrigo M Silva, Sérgio Canuto, Mirella M Moro, and Marcos A Gonçalves. “Ranked batch-mode active learning”. In: *Information Sciences* 379 (2017), pp. 313–337. DOI: 10.1016/j.ins.2016.10.037.

[CBP10] Shayok Chakraborty, Vineeth Balasubramanian, and Sethuraman Panchanathan. “An optimization based framework for dynamic batch mode active learning”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2010.

[CBP15] Shayok Chakraborty, Vineeth Balasubramanian, and Sethuraman Panchanathan. “Adaptive Batch Mode Active Learning”. In: *Transactions on Neural Networks and Learning Systems* 26.8 (2015), pp. 1747–1760. DOI: 10.1109/tnnls.2014.2356470.

[CF17a] Wei Chen and Mark Fuge. “Active expansion sampling for learning feasible domains in an unbounded input space”. In: *Structural and Multidisciplinary Optimization* 57.3 (2017), pp. 925–945. DOI: 10.1007/s00158-017-1894-y.

[CF17b] Wei Chen and Mark Fuge. “Beyond the Known: Detecting Novel Feasible Domains Over an Unbounded Design Space”. In: *Journal of Mechanical Design* 139.11 (2017). DOI: 10.1115/1.4037306.

[Cha+13] Rita Chattopadhyay, Zheng Wang, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. “Batch Mode Active Sampling Based on Marginal Probability Distribution Matching”. In: *Transactions on Knowledge Discovery from Data (TKDD)* 7.3 (2013), pp. 1–25. DOI: 10.1145/2513092.2513094.

[Cha+18] Arin Chaudhuri, Deovrat Kakde, Maria Jahja, Wei Xiao, Seunghyun Kong, Hansi Jiang, and Sergiy Percdriy. “Sampling method for fast training of support vector data description”. In: *Reliability and Maintainability Symposium (RAMS)*. IEEE. IEEE, 2018, pp. 1–7. DOI: 10.1109/ram.2018.8463127.

[CHK17] Lin Chen, Seyed Hamed Hassani, and Amin Karbasi. “Near-Optimal Active Learning of Halfspaces via Query Synthesis in the Noisy Setting.” In: *Conference on Artificial Intelligence (AAAI)*. ACM. 2017, pp. 1798–1804.

[CL19] Yunhong Chen and Shuming Li. “A lightweight anomaly detection method based on SVDD for wireless sensor networks”. In: *Wireless Personal Communications* 105.4 (2019), pp. 1235–1256. DOI: 10.1007/s11277-019-06143-1.

[Cor08] Gordon V Cormack. “Email Spam Filtering: A Systematic Review”. In: *Foundations and Trends® in Information Retrieval* 1.4 (2008), pp. 335–455. DOI: 10.1561/1500000006.

[CS17] Adrian Calma and Bernhard Sick. “Simulation of Annotators for Active Learning: Uncertain Oracles.” In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) Workshops*. 2017, pp. 49–58.

[CTK04] Calvin S Chu, Ivor W Tsang, and James T Kwok. “Scaling up support vector data description by using core-sets”. In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE. IEEE, 2004. DOI: 10.1109/ijcnn.2004.1379943.

[Das+16] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. “Incorporating expert feedback into active anomaly discovery”. In: *International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 853–858. DOI: 10.1109/icdm.2016.0102.

[Dés+13] Chesner Désir, Simon Bernard, Caroline Petitjean, and Laurent Heutte. “One class random forests”. In: *Pattern Recognition* 46.12 (2013), pp. 3490–3506. DOI: 10.1016/j.patcog.2013.05.022.

[Dom+18] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. “A comparative evaluation of outlier detection algorithms: Experiments and analyses”. In: *Pattern Recognition* 74 (2018), pp. 406–421. DOI: 10.1016/j.patcog.2017.09.037.

[DPB11] Begüm Demir, Claudio Persello, and Lorenzo Bruzzone. “Batch-Mode Active-Learning Methods for the Interactive Classification of Remote Sensing Images”. In: *Transactions on Geoscience and Remote Sensing* 49.3 (2011), pp. 1014–1031. DOI: 10.1109/tgrs.2010.2072929.

[Du+17] Bo Du, Zengmao Wang, Lefei Zhang, Liangpei Zhang, Wei Liu, Jialie Shen, and Dacheng Tao. “Exploring Representativeness and Informativeness for Active Learning”. In: *Transactions on Cybernetics* 47.1 (2017), pp. 14–26. DOI: 10.1109/tcyb.2015.2496974.

[EB20] Adrian Englhardt and Klemens Böhm. “Exploring the Unknown – Query Synthesis in One-Class Active Learning”. In: *SIAM International Conference on Data Mining (SDM)*. SIAM. 2020, pp. 145–153. DOI: 10.1137/1.9781611976236.17.

[Emm+13] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. “Systematic construction of anomaly detection benchmarks from real data”. In: *International Conference on Knowledge Discovery and Data Mining (KDD) Workshops*. ACM, 2013, pp. 16–21. DOI: 10.1145/2500853.2500858.

[Eng+20a] Adrian Englhardt, Holger Trittenbach, Daniel Kottke, Bernhard Sick, and Klemens Böhm. “Efficient SVDD Sampling with Approximation Guarantees for the Decision Boundary”. In: *arXiv preprint arXiv:2009.13853* (2020).

[Eng+20b] Adrian Englhardt, Holger Trittenbach, Dennis Vetter, and Klemens Böhm. “Finding the Sweet Spot: Batch Selection for One-Class Active Learning”. In: *SIAM International Conference on Data Mining (SDM)*. SIAM. 2020, pp. 118–126. DOI: 10.1137/1.9781611976236.14.

[Fel18] Robert Feldt. *BlackBoxOptim.jl*. Github. 2018. URL: https://github.com/robertfeldt/BlackBoxOptim.jl.

[FLS06] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. CRC press, 2006. ISBN: 1584885467.

[FP99] Tom Fawcett and Foster Provost. “Activity monitoring: Noticing interesting changes in behavior”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 1999, pp. 53–62. DOI: 10.1145/312129.312195.

[FS02] Shai Fine and Katya Scheinberg. “Efficient SVM training using low-rank kernel representations”. In: *Journal of Machine Learning Research (JMLR)* 2 (2002), pp. 243–264.

[Fuk+11] Nobusumi Fukushima, Yuichi Nagata, Sigenobu Kobayashi, and Isao Ono. “Proposal of distance-weighted exponential natural evolution strategies”. In: *Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 164–171. DOI: 10.1109/cec.2011.5949614.

[FYM05] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. “An approach to spacecraft anomaly detection problem using kernel feature space”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2005, pp. 401–410. DOI: 10.1145/1081870.1081917.

[FZL12] Yifan Fu, Xingquan Zhu, and Bin Li. “A survey on instance selection for active learning”. In: *Knowledge and Information Systems* 35.2 (2012), pp. 249–283. DOI: 10.1007/s10115-012-0507-8.

[Gha+11a] A. Ghasemi, M. T. Manzuri, H. R. Rabiee, M. H. Rohban, and S. Haghiri. “Active one-class learning by kernel density estimation”. In: *International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2011, pp. 1–6. DOI: 10.1109/mlsp.2011.6064627.

[Gha+11b] A. Ghasemi, H. R. Rabiee, M. Fadaee, M. T. Manzuri, and M. H. Rohban. “Active Learning from Positive and Unlabeled Data”. In: *International Conference on Data Mining (ICDM) Workshops*. IEEE, 2011, pp. 244–250. DOI: 10.1109/icdmw.2011.20.

[GK11] Rayid Ghani and Mohit Kumar. “Interactive learning for efficiently detecting errors in insurance claims”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2011, pp. 325–333. DOI: 10.1145/2020408.2020463.

[Gor+09] Dirk Gorissen, Karel Crombecq, Ivo Couckuyt, and Tom Dhaene. “Automatic Approximation of Expensive Functions with Active Learning”. In: *Studies in Computational Intelligence*. Springer, 2009, pp. 35–62. DOI: 10.1007/978-3-642-01082-8_2.

[Gör+09] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. “Active learning for network intrusion detection”. In: *Conference on Computer and Communications Security (CCS) Workshops*. ACM, 2009, pp. 47–54. DOI: 10.1145/1654988.1655002.

[Gör+13] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. “Toward Supervised Anomaly Detection”. In: *Journal of Artificial Intelligence Research (JAIR)* 46 (2013), pp. 235–262. DOI: 10.1613/jair.3623.

[Hae+08] Robbie Haertel, Kevin D Seppi, Eric K Ringger, and James L Carroll. “Return on investment for active learning”. In: *Advances in Neural Information Processing Systems (NIPS) Workshops*. 2008.

[Haw80] Douglas M Hawkins. *Identification of Outliers*. Springer, 1980. DOI: 10.1007/978-94-015-3994-4.

[HGX11] Timothy Hospedales, Shaogang Gong, and Tao Xiang. “Finding rare classes: Active learning with generative and discriminative models”. In: *Transactions on Knowledge and Data Engineering (TKDE)* 25.2 (2011), pp. 374–386. DOI: 10.1109/tkde.2011.231.

[Hoi+06] Steven Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. “Batch mode active learning and its application to medical image classification”. In: *International Conference on Machine Learning (ICML)*. ACM, 2006. DOI: 10.1145/1143844.1143897.

[HWY12] Xuelei Hu, Liantao Wang, and Bo Yuan. “Querying representative points from a pool based on synthesized queries”. In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–6. DOI: 10.1109/ijcnn.2012.6252607.

[HZH14] Chenlong Hu, Bo Zhou, and Jinglu Hu. “Fast Support Vector Data Description training using edge detection on large datasets”. In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 2176–2182. DOI: 10.1109/ijcnn.2014.6889718.

[JD03] Piotr Juszczak and Robert P.W. Duin. “Selective Sampling Methods in One-Class Classification Problems”. In: *Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP)*. Springer, 2003, pp. 140–148. DOI: 10.1007/3-540-44989-2_18.

[Jia+14] Yang Jiao, Pengpeng Zhao, Jian Wu, Yujie Shi, and Zhiming Cui. “A Multicriterion Query-Based Batch Mode Active Learning Technique”. In: *Foundations of Intelligent Systems* (2014), pp. 669–680. DOI: 10.1007/978-3-642-54924-3_63.

[Jos11] Ajay Jayant Joshi. “Image classification with minimal supervision”. PhD thesis. University of Minnesota, 2011.

[KCR18] Seho Kee, Enrique del Castillo, and George Runger. “Query-by-committee improvement with diversity and density in batch active learning”. In: *Information Sciences* 454-455 (2018), pp. 401–418. DOI: 10.1016/j.ins.2018.05.014.

[Kim+07] Pyo Jae Kim, Hyung Jin Chang, Dong Sung Song, and Jin Young Choi. “Fast support vector data description using k-means clustering”. In: *International Symposium on Neural Networks (ISNN)*. Springer. Springer Berlin Heidelberg, 2007, pp. 506–514. DOI: 10.1007/978-3-540-72395-0_64.

[Kin+04] Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. “Functional genomic hypothesis generation and experimentation by a robot scientist”. In: *Nature* 427.6971 (2004), pp. 247–252. DOI: `10.1038/nature02236`.

[Kin+09] Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. “The automation of science”. In: *Science* 324.5923 (2009), pp. 85–89. DOI: `10.1126/science.1165620`.

[KLC02] Eamonn Keogh, Stefano Lonardi, and Bill’Yuan-chi’ Chiu. “Finding surprising patterns in a time series database in linear time and space”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2002, pp. 550–556. DOI: `10.1145/775047.775128`.

[KM14] Shehroz S. Khan and Michael G. Madden. “One-class classification: taxonomy of study and review of techniques”. In: *The Knowledge Engineering Review* 29.3 (2014), pp. 345–374. DOI: `10.1017/s026988891300043x`.

[KOS14] David R Karger, Sewoong Oh, and Devavrat Shah. “Budget-optimal task allocation for reliable crowdsourcing systems”. In: *Operations Research* 62.1 (2014), pp. 1–24. DOI: `10.1287/opre.2013.1235`.

[Kra+18] Bartosz Krawczyk, Isaac Triguero, Salvador García, Michał Woźniak, and Francisco Herrera. “Instance reduction for one-class classification”. In: *Knowledge and Information Systems (KAIS)* 59.3 (2018), pp. 601–628. DOI: `10.1007/s10115-018-1220-z`.

[KSZ08] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. “Angle-based outlier detection in high-dimensional data”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2008, pp. 444–452. DOI: `10.1145/1401890.1401946`.

[KT04] J.T.-Y. Kwok and I.W.-H. Tsang. “The Pre-Image Problem in Kernel Methods”. In: *Transactions on Neural Networks* 15.6 (2004), pp. 1517–1525. DOI: `10.1109/tnn.2004.837781`.

[Laz+03] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. “A comparative study of anomaly detection schemes in network intrusion detection”. In: *SIAM International Conference on Data Mining (SDM)*. SIAM. 2003, pp. 25–36. DOI: `10.1137/1.9781611972733.3`.

[LC94] David D Lewis and Jason Catlett. “Heterogeneous Uncertainty Sampling for Supervised Learning”. In: *Machine Learning* (1994), pp. 148–156. DOI: `10.1016/b978-1-55860-335-6.50026-x`.

[Lei+18] Roger A Leite, Theresia Gschwandtner, Silvia Miksch, Simone Kriglstein, Margit Pohl, Erich Gstrein, and Johannes Kuntner. “EVA: Visual Analytics to Identify Fraudulent Events”. In: *Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 330–339. DOI: `10.1109/tvcg.2017.2744758`.

[LG94] David Lewis and William Gale. “A Sequential Algorithm for Training Text Classifiers”. In: *Special Interest Group on Information Retrieval (SIGIR)*. Springer, 1994, pp. 3–12. DOI: 10.1007/978-1-4471-2099-5_1.

[LGC12] Xianglin Li, Runqiu Guo, and Jun Cheng. “Incorporating Incremental and Active Learning for Scene Classification”. In: *International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2012. DOI: 10.1109/icmla.2012.51.

[Li+18] DongDong Li, Zhe Wang, Chenjie Cao, and Yu Liu. “Information entropy based sample reduction for support vector data description”. In: *Applied Soft Computing* 71 (2018), pp. 1153–1160. DOI: 10.1016/j.asoc.2018.02.053.

[Li+19] Zhaozhao Li, Lide Wang, Yueyi Yang, Xiaomin Du, and Hui Song. “Health Evaluation of MVB Based on SVDD and Sample Reduction”. In: *IEEE Access* 7 (2019), pp. 35330–35343. DOI: 10.1109/access.2019.2904600.

[Li11] Yuhua Li. “Selecting training points for one-class support vector machines”. In: *Pattern Recognition Letters* 32.11 (2011), pp. 1517–1522. DOI: 10.1016/j.patrec.2011.04.013.

[Lia+18] Yuwei Liao, Deovrat Kakde, Arin Chaudhuri, Hansi Jiang, Carol Sadek, and Seunghyun Kong. “A new bandwidth selection criterion for using SVDD to analyze hyperspectral data”. In: *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery*. Vol. 10644. SPIE, 2018, pp. 463–474. DOI: 10.1117/12.2314964.

[Lin+05] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. “Approximations to Magic: Finding Unusual Medical Time Series”. In: *Symposium on Computer-Based Medical Systems (CBMS)*. IEEE. 2005, pp. 329–334. DOI: 10.1109/cbms.2005.34.

[Lin+17] Hanfei Lin, Siyuan Gao, David Gotz, Fan Du, Jingrui He, and Nan Cao. “RCLens: Interactive Rare Category Exploration and Identification”. In: *Transactions on Visualization and Computer Graphics* 24.7 (2017), pp. 2223–2237. DOI: 10.1109/TVCG.2017.2711030.

[LJ08] Tae Hee Lee and Jae Jun Jung. “A sampling technique enhancing accuracy and efficiency of metamodel-based RBDO: Constraint boundary sampling”. In: *Computers & Structures* 86.13-14 (2008), pp. 1463–1476. DOI: 10.1016/j.compstruc.2007.05.023.

[LLC10] Yi-Hung Liu, Yan-Chen Liu, and Yen-Jen Chen. “Fast Support Vector Data Descriptions for Novelty Detection”. In: *Transactions on Neural Networks* 21.8 (2010), pp. 1296–1313. DOI: 10.1109/tnn.2010.2053853.

[LM05] Daniel Lowd and Christopher Meek. “Adversarial learning”. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2005, pp. 641–647. DOI: 10.1145/1081870.1081950.

[LM12] Brad J Larson and Christopher A Mattson. “Design space exploration for quantifying a system model’s feasible domain”. In: *Journal of Mechanical Design* 134.4 (2012). DOI: 10.1115/1.4005861.

[Mik+99] Sebastian Mika, Bernhard Schölkopf, Alex J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. “Kernel PCA and De-noising in feature spaces”. In: *Advances in Neural Information Processing Systems (NIPS)*. 1999.

[MS10] Prem Melville and Vikas Sindhwani. “Recommender systems”. In: *Encyclopedia of machine learning* 1 (2010), pp. 1056–1066. DOI: 10.1007/978-1-4899-7687-1_964.

[Nel+12] Blaine Nelson, Benjamin IP Rubinstein, Ling Huang, Anthony D Joseph, Steven J Lee, Satish Rao, and JD Tygar. “Query Strategies for Evading Convex-Inducing Classifiers”. In: *Journal of Machine Learning Research (JMLR)* 13.44 (2012), pp. 1293–1332.

[NHJ08] XuanLong Nguyen, Ling Huang, and Anthony D Joseph. “Support Vector Machines, Data Reduction, and Approximate Kernel Matrices”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Springer. 2008, pp. 137–153. DOI: 10.1007/978-3-540-87481-2_10.

[OJM17] Jack O’Neill, Sarah Jane Delany, and Brian MacNamee. “Model-Free and Model-Based Active Learning for Regression”. In: *Advances in Intelligent Systems and Computing (AISC)* 513 (2017), pp. 375–386. DOI: 10.1007/978-3-319-46562-3_24.

[Pap+03] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. “LOCI: fast outlier detection using the local correlation integral”. In: *International Conference on Data Engineering (ICDE)*. IEEE. 2003, pp. 315–326. DOI: 10.1109/icde.2003.1260802.

[Pla98] John Platt. *Sequential minimal optimization: A fast algorithm for training support vector machines*. Tech. rep. Microsoft Research, 1998.

[PX11] Xinjun Peng and Dong Xu. “Efficient support vector data descriptions for novelty detection”. In: *Neural Computing and Applications* 21.8 (2011), pp. 2023–2032. DOI: 10.1007/s00521-011-0625-3.

[Qu+19] Hua Qu, Jianlong Zhao, Jihong Zhao, and Dingchao Jiang. “Towards support vector data description based on heuristic sample condensed rule”. In: *Chinese Control And Decision Conference (CCDC)*. IEEE. 2019. DOI: 10.1109/ccdc.2019.8833182.

[RP11] Yi Ren and Panos Y Papalambros. “A Design Preference Elicitation Query as an Optimization Process”. In: *Journal of Mechanical Design* 133.11 (2011). DOI: 10.1115/1.4005104.

[RV18] Oscar Reyes and Sebastián Ventura. “Evolutionary Strategy to Perform Batch-Mode Active Learning on Multi-Label Data”. In: *ACM Transactions on Intelligent Systems and Technology* 9.4 (2018), pp. 1–26. DOI: 10.1145/3161606.

[Sad+17] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. “Active preference-based learning of reward functions”. In: *Robotics: Science & Syst.* 2017. DOI: 10.15607/rss.2017.xiii.053.

[Sch+00] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. “New Support Vector Algorithms”. In: *Neural Computation* 12.5 (2000), pp. 1207–1245. DOI: 10.1162/089976600300015565.

[Sch+01] B Schölkopf, J C Platt, J Shawe-Taylor, A J Smola, and R C Williamson. “Estimating the Support of a High-Dimensional Distribution”. en. In: *Neural Computation* 13.7 (2001), pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965.

[Sco15] David W Scott. *Multivariate Density Estimation.* Wiley, 2015. DOI: 10.1002/9781118575574.

[SDZ15] Qian Shi, Bo Du, and Liangpei Zhang. “Spatial Coherence-Based Batch-Mode Active Learning for Remote Sensing Image Classification”. In: *Transactions on Image Processing* 24.7 (2015), pp. 2037–2050. DOI: 10.1109/tip.2015.2405335.

[Set11] Burr Settles. “From theories to queries: Active learning in practice”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS).* 2011, pp. 1–18.

[Set12] Burr Settles. “Active Learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012). DOI: 10.2200/s00429ed1v01y201207aim018.

[She+04] Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. “Multi-criteria-based active learning for named entity recognition”. In: *Association for Computational Linguistics (ACL).* ACL, 2004. DOI: 10.3115/1218955.1219030.

[Sil+17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. In: *arXiv preprint arXiv:1712.01815* (2017).

[SK18] Tegjyot Singh Sethi and Mehmed Kantardzic. “Data driven exploratory attacks on black box classifiers in adversarial domains”. In: *Neurocomputing* 289 (2018), pp. 129–143. DOI: 10.1016/j.neucom.2018.02.007.

[SM11] Michael R Smith and Tony Martinez. “Improving classification accuracy by identifying and removing instances that should be misclassified”. In: *International Joint Conference on Neural Networks (IJCNN).* IEEE, 2011, pp. 2690–2697. DOI: 10.1109/ijcnn.2011.6033571.

[SPI08] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. “Get another label? improving data quality and data mining using multiple, noisy labelers”. In: *International Conference on Knowledge Discovery and Data Mining (KDD).* ACM, 2008, pp. 614–622. DOI: 10.1145/1401890.1401965.

[Sto+08] Jack W Stokes, John Platt, Joseph Kravis, and Michael Shilman. *Aladin: Active learning of anomalies to detect intrusions.* Tech. rep. Microsoft Research, 2008.

[Sun+16] Wenzhu Sun, Jianling Qu, Yang Chen, Yazhou Di, and Feng Gao. "Heuristic sample reduction method for support vector data description". In: *Turkish Journal of Electrical Engineering & Computer Sciences* (2016).

[SZ05] Xuehua Shen and ChengXiang Zhai. "Active feedback in ad hoc information retrieval". In: *Special Interest Group on Information Retrieval (SIGIR)*. ACM, 2005, pp. 59–66. DOI: 10.1145/1076034.1076047.

[Tax02] David M. J. Tax. "One-class classification: Concept learning in the absence of counter-examples". PhD thesis. Delft University of Technology, 2002.

[TB19] Holger Trittenbach and Klemens Böhm. "One-Class Active Learning for Outlier Detection with Multiple Subspaces". In: *International Conference on Information and Knowledge Management (CIKM)*. ACM, 2019, pp. 811–820. DOI: 10.1145/3357384.3357873.

[TBA20] Holger Trittenbach, Klemens Böhm, and Ira Assent. "Active Learning of SVDD Hyperparameter Values". In: *International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2020, pp. 109–117. DOI: 10.1109/dsaa49011.2020.00023.

[TD01] David M. J. Tax and Robert P. W. Duin. "Uniform Object Generation for Optimizing One-class Classifiers". In: *Journal of Machine Learning Research (JMLR)* (2001).

[TD04] David Tax and Robert Duin. "Support Vector Data Description". In: *Machine Learning* 54.1 (2004), pp. 45–66. DOI: 10.1023/b:mach.0000008084.60811.49.

[TEB19] Holger Trittenbach, Adrian Englhardt, and Klemens Böhm. "Validating One-Class Active Learning with User Studies – a Prototype and Open Challenges". In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) Workshops*. 2019.

[TEB21] Holger Trittenbach, Adrian Englhardt, and Klemens Böhm. "An overview and a benchmark of active learning for outlier detection with one-class classifiers". In: *Expert Systems with Applications* 168 (2021), p. 114372. DOI: 10.1016/j.eswa.2020.114372.

[TH19] Ying-Peng Tang and Sheng-Jun Huang. "Self-paced active learning: Query the right thing at the right time". In: *Conference on Artificial Intelligence (AAAI)*. Vol. 33. 2019, pp. 5117–5124. DOI: 10.1609/aaai.v33i01.33015117.

[TL19] Yu-Lin Tsou and Hsuan-Tien Lin. "Annotation cost-sensitive active learning by tree sampling". In: *Machine Learning* 108.5 (2019), pp. 785–807. DOI: 10.1007/s10994-019-05781-7.

[VV06] Régis Vert and Jean-Philippe Vert. "Consistency and Convergence Rates of One-Class SVMs and Related Algorithms". In: *Journal of Machine Learning Research (JMLR)* (2006), pp. 817–854. ISSN: 1532-4435.

[Wan+09] Chi-Kai Wang, Yung Ting, Yi-Hung Liu, and Gunawan Hariyanto. “A novel approach to generate artificial outliers for support vector data description”. In: *International Symposium on Industrial Electronics*. IEEE, 2009. DOI: 10.1109/isie.2009.5214421.

[Wan+15] Liantao Wang, Xuelei Hu, Bo Yuan, and Jianfeng Lu. “Active learning via query synthesis and nearest neighbour search”. In: *Neurocomputing* 147 (2015), pp. 426–434. DOI: 10.1016/j.neucom.2014.06.042.

[Wan+16] Zengmao Wang, Bo Du, Lefei Zhang, and Liangpei Zhang. “A batch-mode active learning framework by querying discriminative and representative samples for hyperspectral image classification”. In: *Neurocomputing* 179 (2016), pp. 88–100. DOI: 10.1016/j.neucom.2015.11.062.

[Wan+18] Siqi Wang, Qiang Liu, En Zhu, Fatih Porikli, and Jianping Yin. “Hyperparameter selection of one-class support vector machine by self-adaptive data shifting”. In: *Pattern Recognition* 74 (2018), pp. 198–211. DOI: 10.1016/j.patcog.2017.09.012.

[Wil18] Leland Wilkinson. “Visualizing Big Data Outliers Through Distributed Aggregation”. In: *Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 256–266. DOI: 10.1109/tvcg.2017.2744685.

[Won+03] Weng-Keen Wong, Andrew W Moore, Gregory F Cooper, and Michael M Wagner. “Bayesian network anomaly pattern detection for disease outbreaks”. In: *International Conference on Machine Learning (ICML)*. 2003, pp. 808–815.

[WS01] Christopher KI Williams and Matthias Seeger. “Using the Nyström method to speed up kernel machines”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2001.

[WY15] Zheng Wang and Jieping Ye. “Querying Discriminative and Representative Samples for Batch Mode Active Learning”. In: *Transactions on Knowledge Discovery from Data (TKDD)* 9.3 (2015), pp. 1–23. DOI: 10.1145/2700408.

[XAZ07] Zuobing Xu, Ram Akella, and Yi Zhang. “Incorporating Diversity and Density in Active Learning for Relevance Feedback”. In: *European Conference on Information Retrieval (ECIR)*. Vol. 4425. Springer, 2007, pp. 246–257. DOI: 10.1007/978-3-540-71496-5_24.

[Xia+14] Yanshan Xiao, Bo Liu, Zhifeng Hao, and Longbing Cao. “A K-Farthest-Neighbor-based approach for support vector data description”. In: *Applied Intelligence* 41.1 (2014), pp. 196–211. DOI: 10.1007/s10489-013-0502-0.

[Yan+12] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. “Nyström method vs random fourier features: A theoretical and empirical comparison”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[YWF18] Lili Yin, Huangang Wang, and Wenhui Fan. “Active learning based support vector data description method for robust novelty detection”. In: *Knowledge-Based Systems* 153 (2018), pp. 40–52. DOI: 10.1016/j.knosys.2018.04.020.

[Zha+09] Yong Zhang, Xiao-Dan Liu, Fu-Ding Xie, and Ke-Qiu Li. "Fault classifier of rotating machinery based on weighted support vector data description". In: *Expert Systems with Applications* 36.4 (2009), pp. 7928–7932. DOI: 10.1016/j.eswa.2008.10.062.

[Zhu+14] Fa Zhu, Ning Ye, Wei Yu, Sheng Xu, and Guobao Li. "Boundary detection and sample reduction for one-class support vector machines". In: *Neurocomputing* 123 (2014), pp. 166–173. DOI: 10.1016/j.neucom.2013.07.002.

[ZLG03] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions". In: *International Conference on Machine Learning (ICML) Workshops*. 2003.