Karlsruhe Institute of Technology

User-Centric Active Learning for Outlier Detection

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Holger Trittenbach

aus Mannheim

Tag der mündlichen Prüfung:	05. Februar 2020
Erster Gutachter:	Prof. Dr. Klemens Böhm
Zweiter Gutachter:	Prof. Dr. Ira Assent

Acknowledgments

A PhD is more than a few pages of writing, and more than the work of only one person. I am very grateful to all who have contributed to my work.

I would like to thank my supervisor Prof. Klemens Böhm, who has guided me in my scientific endeavors. Thank you for the freedom to set and pursue my own research agenda. I greatly appreciate your encouragement to collaborate with scientists from various fields and your tireless support in writing publications.

I would like to thank Prof. Ira Assent, my external supervisor and reviewer. Thank you for your interest in my research and for hosting me for a fruitful research stay in Aarhus. Your suggestions have always been inspiring.

To all my colleagues, and the PIs of my research training group: Thank you for the great fun and lively discussions. You have always challenged and encouraged me to think beyond my field of research. A special thank you goes to Adrian Englhardt and the MEGA group (Michael Vollmer, Simon Bischof, and Dominik Werle), who have devoted much time and effort to our joint research projects. Thank you for your dedication, it has been a blast working with you.

Finally, I would like to thank my family and friends for their unwavering support. Thank you for cheering me up in difficult times and for celebrating with me the good times.

Abstract

Outlier detection searches for unusual, rare observations in large, often high-dimensional data sets. One of the fundamental challenges of outlier detection is that “unusual” typically depends on the perception of a user, the recipient of the detection result. This makes finding a formal definition of “unusual” that matches with user expectations difficult. One way to deal with this issue is active learning, i.e., methods that ask users to provide auxiliary information, such as class label annotations, to return algorithmic results that are more in line with the user input. Active learning is well-suited for outlier detection, and many respective methods have been proposed over the last years. However, existing methods build upon strong assumptions. One example is the assumption that users can always provide accurate feedback, regardless of how algorithmic results are presented to them – an assumption which is unlikely to hold when data is high-dimensional. It is an open question to which extent existing assumptions are in the way of realizing active learning in practice.

In this thesis, we study this question from different perspectives with a differentiated, user-centric view on active learning. In the beginning, we structure and unify the research area on active learning for outlier detection. Specifically, we present a rigorous specification of the learning setup, structure the basic building blocks, and propose novel evaluation standards. Throughout our work, this structure has turned out to be essential to select a suitable active learning method, and to assess novel contributions in this field. We then present two algorithmic contributions to make active learning for outlier detection user-centric. First, we bring together two research areas that have been looked at independently so far: outlier detection in subspaces and active learning. Subspace outlier detection are methods to improve outlier detection quality in high-dimensional data, and to make detection results more easy to interpret. Our approach combines them with active learning such that one can balance between detection quality and annotation effort. Second, we address one of the fundamental difficulties with adapting active learning to specific applications: selecting good hyperparameter values. Existing methods to estimate hyperparameter values are heuristics, and it is unclear in which settings they work well. In this thesis, we therefore propose the first principled method to estimate hyperparameter values. Our approach relies on active learning to estimate hyperparameter values, and returns a quality estimate of the values selected. In the last part of the thesis, we look at validating active learning for outlier detection practically. There, we have identified several technical and conceptual challenges which we have experienced firsthand in our research. We structure and document them, and finally derive a roadmap towards validating active learning for outlier detection with user studies.

Zusammenfassung

Verfahren zur Ausreißererkennung suchen nach ungewöhnlichen, seltenen Beobachtungen in großen, oft hoch-dimensionalen Datenbeständen. Eine typische Anwendung ist die Analyse von industriellen Stromverbrauchsdaten, wo ungewöhnliche Messwerte beispielsweise auf eine falsche Maschinenkonfiguration oder hohen Verschleiß hinweisen können. Eine frühzeitige Erkennung von Auffälligkeiten kann hier hilfreich sein, um Schäden an der Maschine zu vermeiden.

Konventionelle Ausreißererkennung ist jedoch eine Einbahnstraße: Der Erkennungsalgorithmus wird auf den Datenbestand angewandt und produziert einen „Score“ anhand dessen sich Beobachtungen nach Grad der Auffälligkeit sortieren lassen. Der Adressat dieses Ergebnisses, der Nutzer, muss zunächst mit diesem Resultat leben. Gewünschte Anpassungen, zum Beispiel um Fehleinschätzungen des Algorithmus zu korrigieren, sind, wenn überhaupt, nur durch geeignete Manipulation von Trainingsdaten oder von Hyperparameterwerten des Klassifikators möglich. Deren Einfluss auf den Algorithmus ist im Allgemeinen jedoch komplex und schwierig einzuschätzen.

Eine Möglichkeit dem Nutzer direkten Einfluss auf die Ausreißererkennung zu geben sind aktive Lernverfahren. „Aktives Lernen“ bezeichnet dabei die Möglichkeit, dass der Algorithmus gezielt zusätzliche Informationen zu einzelnen Beobachtungen vom Nutzer erfragen kann, bei denen beispielsweise unsicher ist ob sie der Klasse „Ausreißer“ zugeordnet werden sollten. Das dabei übergeordnete Ziel ist es mit möglichst wenig Annotationen eine gute Klassifikationsgenauigkeit zu erreichen.

In einschlägiger Literatur haben sich verschiedene, meist implizite Annahmen etabliert, von denen der Erfolg aktiver Lernverfahren wesentlich abhängt. Das sind zum einen technische Voraussetzungen, wie z. B. die Anzahl bereits annotierter Beobachtungen zu Beginn des Lernprozesses. Zum anderen sind es fundamentale Annahmen zur Qualität und zur Verfügbarkeit von Nutzerannotationen. In der Literatur wird beispielsweise davon ausgegangen, dass Nutzer unabhängig von der Art und der Präsentation der Klassifikationsergebnisse sinnvoll und korrekt annotieren. Dies ist eine starke Vereinfachung, die in der Realität so erfahrungsgemäß nicht zutrifft. Man stelle sich beispielsweise einen 20-dimensionalen Vektorraum vor, der das Ergebnis verschiedener Vorverarbeitungsschritte ist. Hier lässt sich nur schwer argumentieren, dass ein Nutzer die Anfrage „Ist die Beobachtung $\langle x_1, \ldots, x_{20} \rangle$ mit einem Score von 0.74 tatsächlich ein Ausreißer?“ ohne Weiteres beantworten kann. Stattdessen ist davon auszugehen, dass eine Unterstützung des Nutzers notwendig ist, zum Beispiel durch eine Visualisierung der Entscheidungsgrenzen des Klassifikators oder durch die Bereitstellung weiterer, erklärender Informationen.

Darüber hinaus hängt die Klassifikationsgenauigkeit maßgeblich von den Hyperparameterwerten der verwendeten Klassifikatoren ab. Eine schlechte Wahl dieser Werte kann dazu führen, dass entweder fast alle oder gar keine der Beobachtungen als Ausreißer erkannt werden. Selbst für weit verbreitete Klassifikatoren ist die Wahl von Hyperparameterwerten jedoch nicht intuitiv, und hängt stark von den vorliegenden Daten ab. Meist sind zu Beginn des aktiven Lernens auch noch keine Annotationen vorhanden. In diesen Fällen müssen sich Nutzer auf Heuristiken verlassen, die zwar Werte für die Hyperparameter vorschlagen, aber darüber hinaus keine weiteren Anhaltspunkte über deren Qualität und Eignung für das vorliegende Problem geben.

Die angesprochenen Schwierigkeiten sind unserer Ansicht nach maßgebliche dafür, dass trotz einer Vielzahl an Literatur zu aktiven Lernverfahren zur Ausreißererkennung nur sehr wenige praktische Anwendungen bekannt sind. In dieser Thesis stellen wir daher etablierte Annahmen und Voraussetzungen in Frage, und schlagen eine differenziertere Betrachtung von aktiven Lernverfahren zur Ausreißererkennung vor. Die Thesis umfasst vier konkrete Beiträge zur Erweiterung des Stands der Forschung in der Informatik.

Übersicht und Benchmark. Zunächst gehen wir der Frage nach, wie sich existierende aktive Lernverfahren zur Ausreißererkennung kategorisieren und vergleichen lassen. Zu Beginn erarbeiten wir einen systematischen Überblick, der existierende Verfahren aus der Literatur einheitlich darstellt und getroffene Annahmen explizit diskutiert. Hierbei wird deutlich, dass sich, je nach getroffenen Annahmen, verschiedene aktive Lernalgorithmen unterschiedlich gut zur Erkennung von Ausreißern eignen. Aufbauend auf unserer theoretischen Einordnung evaluieren wir die vorhandenen Ansätze empirisch. Dazu schlagen wir neue Evaluationsstandards und ein Rahmenwerk vor, um aktive Lernverfahren zu vergleichen. Ein umfangreicher Benchmark zeigt, dass diese Standards nützlich sind, um ein geeignetes Verfahren für eine spezifische Anwendung auszuwählen.

Aktives Lernen in Teilräumen. Eine wichtige Einsicht aus unserer Übersicht ist, dass man in der Literatur im Allgemeinen annimmt, dass Nutzer immer Annotationen geben können, unabhängig davon wie ihnen die algorithmischen Ergebnisse präsentiert werden. Wir stellen diesbezüglich nun die Frage, ob sich diese Annahme mit einer unserer Ansicht nach realistischeren Annahme ersetzen lässt: Dass Nutzer in niedrigdimensionalen Datenräumen Annotationen vergeben können, diese Fähigkeit aber mit steigender Dimensionalität des Datenraums abnimmt. Um dieser angepassten Annahme gerecht zu werden, schlagen wir einen neuen Klassifikator vor, der Entscheidungsgrenzen in niedrigdimensionalen Projektionen der Daten findet. Durch Variation der betrachteten Projektionen lässt sich zwischen der Komplexität der erkennbaren Ausreißer und der Realisierbarkeit von Nutzerfeedback abwägen. Sind die betrachteten Projektionen beispielsweise zweidimensional, lassen sich zwar nur vergleichsweise einfach Ausreißer erkennen. Dafür sind die Entscheidungsgrenzen des Klassifikators einfach zu visualisieren und es ist zu erwarten, dass Nutzer mit weniger Aufwand Annotationen vergeben können.

Wahl der Hyperparameter. Eine weitere Einsicht aus unserer Übersicht ist, dass die Qualität der Ausreißererkennung maßgeblich von den gewählten Hyperparametern des Klassifikators abhängt. Eine schlechte Wahl kann dazu führen, dass der Klassifikator alle Beobachtungen als Ausreißer klassifiziert, oder überhaupt keine Ausreißer mehr erkennt. Der bisher übliche Weg Hyperparameterwerte zu wählen ist über Heuristiken. Allerdings ist die Auswahl einer geeigneten Heuristik ebenfalls schwierig. Ein Grund ist, dass eine

Vielzahl an Heuristiken zur Auswahl steht, die zum Teil unterschiedliche aber gleichermaßen plausible Ergebnisse hervorbringen. Das motiviert die Fragestellung, ob sich ein Verfahren finden lässt, dass die Wahl der Hyperparameter zuverlässiger und einfacher für den Nutzer macht. Hierzu schlagen wir ein neues Lernverfahren zur Auswahl geeigneter Hyperparameter vor. Die Kernidee unseres Ansatzes ist Hyperparameter ebenfalls über ein aktives Lernverfahren zu wählen. Ein zentraler Vorteil gegenüber heuristischen Verfahren ist dabei, dass sich die Qualität der gewählten Hyperparameterwerte aus den erhobenen Annotationen schätzen lässt. Das vorgeschlagene Verfahren erlaubt dem Nutzer damit Hyperparameterwerte zu wählen, ohne ihre komplexen Zusammenhänge verstehen zu müssen.

Prototyp zur Validierung. Zum Abschluss der Thesis beschäftigen wir uns mit der Validierung aktiver Lernverfahren. Hierbei steht insbesondere die Frage im Vordergrund, was genau erforderlich ist, um aktive Lernverfahren zur Ausreißererkennung in der Praxis umzusetzen. Zur Untersuchung der Frage stellen wir eine Referenzarchitektur und einen Prototyp zur Durchführung von Nutzerstudien vor. Anhand des Prototyps lassen sich mehrere offene Herausforderungen konzeptioneller und technischer Natur erkennen, die zur Realisierung eines solchen Systems in der Praxis zunächst eine Lösung erfordern. In der Thesis werden diese Herausforderungen systematisch aufgearbeitet und relevanten Forschungssträngen aus der Literatur zugeordnet. Abschließend stellen wir einen Leitfaden vor, um schrittweise zu einer Validierung von aktiven Lernverfahren zur Ausreißererkennung mit Nutzerstudien zu gelangen.

Zusammenfassend lässt sich festhalten, dass der Fokus auf den Nutzer ein neuer Schwerpunkt in der Forschung zu aktiven Lernsystemen ist, der nun erstmals umfassend und aus verschiedenen Blickwinkeln bearbeitet wurde. Unsere Arbeit trägt dazu bei, Ausreißererkennung mit aktiven Lernverfahren nutzerorientiert zu gestalten. Sie unterstützen damit die Realisierung komplexer Anwendungen zur Erkennung von Auffälligkeiten, beispielsweise bei der Analyse von industriellen Stromverbrauchsdaten.

Contents

Part I.

Introduction

1. Motivation

Machine learning algorithms have long become a commonplace and an integral part of our society. In our everyday lives, we face the ramifications of data-based decisions, often even without noticing. They relieve us from seemingly cumbersome tasks, like choosing background music for dinner [Cel10]. They help us to stay on top of our daily routines, like by smart watches that monitor our activities [Wei$^+$16], and urge us to take the stairs once in a while to stay healthy. But machine learning algorithms also have decisive power on our way of living. For instance, they facilitate credit scoring [TC10], and thus can be a reason to be denied a house construction loan. Machine learning also has influence on critical societal functions, like predictive policing [Pea10], the identification of individuals and locations at high risk of crime based on historical data.

Machine learning algorithms base on mathematical equations, and obey the rules of logical reasoning. So technically speaking, an algorithm cannot be “wrong”. But results are open to judgment from humans. Simply put, individuals may perceive an algorithmic decision as correct if it coincides with their expectations, or with the broader goals of society. Otherwise, a decision may be perceived to be wrong. One often just overlooks correct decisions, since they are subtle, and in line with our expectations. Wrong decisions, however, are more noticeable. They often have a direct, negative effect on our daily lives. However, the gravity of their consequences varies greatly. For instance, a bad music recommendation can make you listen to some unpleasant background noise during a dinner with friends. In another case however, this recommendation may turn out to be a conversation starter in a tense date. Of far greater severity though is a wrong decision on a loan, which can have an essential impact on the life planning opportunities of an individual.

So algorithms are not isolated entities which exist independent of human judgment and influence. Humans are affected by algorithmic decisions, and must be a critical corrective. As a society, we are starting to understand that algorithms should be able to learn from humans, include them in critical decision making, and provide mechanisms to influence algorithmic results.

One may argue that humans can already influence nearly all machine learning algorithms, for instance by manipulating the training data, or by customizing algorithm parameters. And indeed, these are some of the possibilities to influence algorithm behavior. But they are mainly technical tools for machine learning experts to fine-tune algorithms. They require methodological knowledge and highly specialized technical skills – this applies only to very few, selected individuals. Making the influence on machine learning broadly available requires more simple mechanisms and interfaces between the machine learning system and the human.

A research area that focuses on a simple interaction between humans and machine learning is *active learning*. In brief, active learning is the paradigm to include humans in machine learning by iteratively asking them for auxiliary information. For instance, humans can be asked to provide annotations, like categories or class labels, to individual observations. Another option is to ask a human to weight or to rank input attributes according to their importance to the classification task [RMJ06; DSM09]. The conventional interpretation of active learning is that the machine learning model strives to improve upon some learning objective, like classification accuracy, based on the auxiliary information acquired. However, we deem active learning more than just a mechanism to extract information from human resources. The answers of a human, i.e., the auxiliary information the human provides, can have direct influence on the algorithmic decisions. So in a way, active learning allows a human-in-the-loop to convey preferences and beliefs to the machine learning algorithm. This is of fundamental value when the quality of a machine learning result depends on individual perception, and when humans struggle to provide a formal and complete description of their preferences. With music, for example, users of a streaming platform can “like” or “dislike” specific songs to convey their music taste to the system [ERR14]. The machine learning algorithm incorporates this information, such that future recommendations are more in line with the user input.

Naturally, there is a limit on how much input one can expect from users. Rating all songs ever recorded is a life task. Even annotating a large share of songs is not desirable, since it requires attention and cognitive work. In a different context, say, when active learning is used in a business environment, one can even put a price tag on the labor required to provide input. This motivates a core objective of active learning: improving on some learning objective with minimum user interaction. Active learning realizes this objective by estimating the informativeness of the user input, e.g., how useful annotating a specific observation is for the machine learning task. User inquiries are then limited to the most useful instances only. So in the music example, users would be asked to provide input to a few songs that reveal as much as possible about their music taste. Ideally, users are asked for input as long as its marginal benefit is positive, i.e., as long as the cost of user interaction is lower than the benefit of improving upon the machine learning task.

Active learning has been used with many machine learning tasks, including collaborative filtering [ERR14] and document classification [SC00]. Successful applications come from a variety of domains, including bioinformatics [Liu04] and cyber security [Gör+09; Sto+08]. For each task and domain, the question is how to implement the abstract concept of iterative user interaction, how to update the machine learning model, and how to select the most informative observations.

In this thesis, we focus on an important machine learning task that has received much attention the last years: the detection of outliers, i.e., unusual and very rare observations, in large, multi-dimensional data sets [Agg15]. Popular applications for outlier detection are intrusion detection in computer networks [Gör+09; Sto+08], and the discovery of unusual events in sensor data from machines and industrial plants [YWF18]. Active learning fits particularly well with outlier detection. One reason is that the definition of what constitutes an outlier is relative and subjective. It is *relative*, because the very definition of “unusual” requires a regular, normal counterpart. In contrast to normal observations, outliers do not need to be similar to each other. The only characteristics outliers have in common is that they are, in one way or another, different to a normal reference. But conventional outlier detection methods require to choose one specific definition of unusual, on the basis of which they separate between normal and unusual observations. It often is unclear which definition works well in a specific application. Next, "normal" is an inherently *subjective* concept. What a user perceives as normal depends on individual preferences, past experiences and domain knowledge. Think about a network intrusion system that reports high network traffic that occurs at off-peak hours as suspicious. In this case, a layman may conclude a potential security breach; a local network administrator knows that this is the result of a scheduled maintenance.

On the one hand, humans typically have difficulties to give a complete, formal description of what they consider to be normal. On the other hand, judging whether a specific observation is unusual is comparatively easy. This makes active learning well-suited for outlier detection. Literature is in line with this, since numerous active learning methods have been proposed that are tailored towards outlier detection [Gha+11b; Gör+13; BBJ15]. Surprisingly however, a closer look reveals that beyond the theoretical and conceptual contributions in this field, there only are very few reported cases of successful real-world application, e.g., [BCB18; Ver+18]. This observation suggests that there are difficulties that prevent a practical application. It is an open question where these difficulties come from, and whether there are assumptions and requirements that are in the way of successful applications. In this thesis, we study this question from different perspectives to identify existing difficulties with active learning for outlier detection, and strive for novel methods to overcome them.

1.1. Challenges

Studying active learning for outlier detection is difficult. One reason is that this research field is not well-structured. For one, active learning for outlier detection is at the intersection between different research areas, namely outlier detection, semi-supervised classification and active learning. Further, there are several related areas including Explainable Artificial Intelligence (XAI), and Human-Computer Interaction (HCI) that also play an important role in interactive systems. Over the last years, many approaches have been proposed that address specific facets of active learning for outlier detection, more or less isolated within the different research areas. But there has been no effort so far to align the different approaches holistically. Beyond this general difficulty, there are several, more specific challenges.

(C1) Outlier Properties. As mentioned earlier, outliers are rare and different to normal observations, but not necessarily similar to each other. This has two implications. First, every outlier may be a sample from a unique distribution. Therefore, methods that assume a common underlying distribution, like density estimation techniques, are not applicable without further ado. Second, outliers are rare to non-existent in training data. As a result, the class distribution is very imbalanced. In some cases, training data may not contain any outliers at all. A consequence is that active learning methods that work well with general binary or multi-class classification may not be directly applicable to outlier detection.

(C2) Complex Processing Pipeline. Active learning for outlier detection builds upon semi-supervised one-class classifiers. One-class classifiers learn what constitutes normal observations, and all observations that do not follow the normal pattern are classified as outliers. The methods applicable to active learning are semi-supervised, i.e., they can process both, unlabeled as well as annotated observations. Here, the challenge is that existing one-class active learning methods require static vector data as an input. This is not an issue when raw data already comes as static vectors. For instance, a music song database may consist of different attributes, like song length, key, and genre. A song is described by a static vector in this attribute space. But in many cases, raw data has a different format and must be transformed to static vectors. An example are multivariate time-series, e.g., from smart-meter measurements, see Chapter 2. There, one must rely on comprehensive pre-processing and feature extraction to make one-class active learning applicable. The processing pipeline from raw measurements to the final result typically is complex, and involves many design choices. For instance, one must decide which features to extract. With active learning, features should be human-interpretable. Other pre-processing steps, like dimensionality reduction by principal component analysis, might not be useful if they bog down interpretability. The complexity of the pre-processing pipeline makes comprehending outlier detection results difficult. This in turn makes it more difficult for users to provide annotations.

(C3) Incoherent Standards. There currently is no overview on active learning methods for outlier detection. This is, there is no categorization of existing classifiers and methods to select useful observations, notation is largely inconsistent, and many of the core assumptions made in literature are implicit. Further, there are many different strategies to select observations for annotation, explicitly proposed for semi-supervised one-class classifiers. However, selecting a good strategy is difficult, since they are designed for different objectives, like classification accuracy or proportion of outliers presented to the user. They also often make assumptions on the underlying data, like a common distribution for the outlier class. Next, there are no standards on how to evaluate active learning methods. A reason is that the “quality” of an active learning method might vary significantly, depending on the specific setup. Most commonly, quality is a measure of classification accuracy when a specific number of annotations have been collected. But quality might also refer to other objectives, like how quickly a classifier improves with the first few annotations. There currently are no established ways to quantify the different meanings of quality. All this makes it very difficult to achieve an overview of the research field, and to assess novel contributions.

(C4) Initialization. Initializing one-class classifiers with complex use cases is difficult. A reason is that one-class classifiers require to set several hyperparameter values. Finding good hyperparameter values notoriously difficult, since a good choice depends on the use-case and the data at hand. Further, active learning starts as an unsupervised problem.

So one has to resort to heuristics for hyperparameter value selection. But the conditions under which existing heuristics work well are largely unclear.

(C5) User out of the Loop. One of the primary objectives of active learning is to include humans in the machine learning process. While there also are some other applications, like using the outcome of compute-intensive simulations or of resource-intensive experiments as annotations [BSM19], relying on human input is by far the most common scenario for active learning. However, literature on one-class active learning abstracts almost entirely from the human in the loop. This is, annotations are simulated based on benchmark data where a ground truth is available. This raises the question under which conditions humans can actually provide annotations, if at all, especially after data has been wrangled through a complex processing pipeline. With classical outlier detection, there has been increasing effort in the last years to make methods more user-centric, e.g., by searching for outliers in low-dimensional projections of the data space that are easy to comprehend and to visualize [KMB12; TB19a]. However, it is unclear whether methods from this area also transfer to the active learning setting.

(C6) Implementation. Validating active learning for outlier detection in practice requires to implement methods in a real-world system, and to conduct user studies. However, there are several difficulties that are in the way of such a system, beyond the ones already discussed in this section. For instance, interactive systems have requirements on the runtimes of the learning algorithms. Long runtimes can be an issue with semi-supervised one-class classifiers when data sets are large. Another example is that one has to manage the data flows with such a system, e.g., between an algorithmic back-end and a user interface. This goes beyond the design of machine learning algorithms and also affects other disciplines, like software engineering and HCI. These technical and conceptual challenges are currently in the way of implementing real-world systems, and of validating active learning for outlier detection in practice.

1.2. Contributions

In this thesis, we strive towards a user-centric approach with active learning for outlier detection. A core concern is to identify and to question existing assumptions that are in the way of the real world adaptation of active learning methods. To this end, we take a more differentiated perspective than current literature, and put emphasis on the user in the loop. We make several contributions that extend the current state-of-the-art.

Overview and Benchmark. In the beginning of this thesis, we address the question on how to structure and unify the field of active learning for outlier detection, and how to compare existing methods. To this end, we propose a categorization of existing methods. We distinguish between three building blocks: (i) a *learning scenario* that summarizes basic assumptions and objectives of the method, (ii) a *base learner*, i.e., a semi-supervised classifier, and (iii) a *query strategy* that selects the observations to be annotated by a user. Based on this categorization, we provide a literature overview of existing methods.

Next, we propose several ways to evaluate active learning. We use them as a standard to benchmark existing approaches, as well as to provide guidelines on how to select a suitable active learning method for outlier detection with specific use cases. This contribution addresses *C3 Incoherent Standards.*

Active Learning in Subspaces. One key insight from our overview is that there is a fundamental assumption in literature: users are expected to always provide accurate feedback, regardless of how algorithmic results are presented to them. We deem this assumption unlikely to hold in practice. In particular with high-dimensional data, users may struggle to provide accurate feedback. Thus, we ask the question whether one can replace the existing assumption with a more realistic one: users can provide feedback if the dimensionality of the data space is low, but this ability decreases with increasing dimensionality. In view of this more realistic assumption, we introduce Subspace Support Vector Data Description (SubSVDD), a novel one-class classifier that applies active learning in multiple projections, so-called subspaces, of the data. It is the first approach to bring the fields of subspace outlier detection [KMB12; TB19a] and of active learning together. In a nutshell, SubSVDD allows to compromise between interpretability and the capability of detecting complex outliers. For instance, if projections are two-dimensional, one can provide users with simple visualizations of the data and of classification boundaries. There, we expect users to provide accurate annotations. However, the complexity of the outliers that can be detected in two-dimensional projections is low. By increasing the dimensionality, the detectable outliers become more complex, but users may have difficulty to annotate them accurately. So SubSVDD gives way to trade-off the detection capabilities against annotation quality. This contribution addresses *C5 User out of the Loop.*

Selection of Hyperparameters. Another take-away from our overview is that the selection of good hyperparameter values is vital for good results with one-class classifiers. To this end, one currently has to rely on heuristics. However, selecting a good heuristic is difficult, since they tend to return different but equally plausible results, and do not come with any formal guarantees. This motivates the question how to make the selection of hyperparameter values more reliable and intuitive. To address this question, we propose Local Active Min-Max Alignment (LAMA), the first principled approach to estimate hyperparameter values of Support Vector Data Description (SVDD), one of the most popular one-class classifiers. LAMA is evidence-based, i.e., it uses active learning to obtain class-label annotations which then are used to estimate hyperparameter values. This does away with purely heuristic methods, where results are difficult to validate. In experiments on real-world data, the hyperparameters tuned with LAMA result in good classification accuracy, in several cases close to the empirical upper bound. This contribution address *C4 Initialization.*

Prototype and Validation. Finally, we ask what is required to realize a one-class active learning system in practice. To study this question, we present an architectural sketch of such a system and implement a prototype. Based on our prototype, we describe important characteristics and prerequisites of one-class active learning and how they influence the design of interactive systems. We then discuss several open technical and conceptual challenges that are in the way of realizing such a system. We conclude by proposing a roadmap to validating one-class active learning for outlier detection with user studies. This contribution addresses *C6 Implementation*.

The challenges *C1 Outlier Properties* and *C2 Complex Processing Pipeline* are of a more general nature, and underlie all our contributions. This is, all of the methods presented in this thesis are designed specifically for outlier detection in complex use cases.

1.3. Thesis Outline

The thesis consists of three parts. In the first part, we get into the subject with an exemplary, horizontally complete use case for active learning with outlier detection, see Chapter 2. The use case is from the domain of energy data, and aims at detecting unusual observations from smart meter measurements. Here, horizontally complete means that the use case starts from a collection of raw time series of sensor measurements and ends with a deployable classifier, trained through an active learning cycle. Based on this use case, we illustrate the challenges of the complex processing pipeline with active learning systems (C2 Complex Processing Pipeline). We then focus on the active learning part, the core topic of this thesis, and explain how our contributions fit in the overall process. Chapter 3 then explains fundamentals on outlier detection and on active learning. The second part of this thesis features our specific contributions. Chapter 4 is an overview and a benchmark of the state-of-the-art of active learning for outlier detection with one-class classifiers. In Chapter 5, we present SubSVDD, our novel approach to active learning in subspaces. Chapter 6 introduces LAMA, our novel approach to learn hyperparameter values of SVDD. Finally, we discuss our prototype and the roadmap towards user studies in Chapter 7. In the last part of this thesis, we conclude with a summary (Chapter 8) and a discussion of open questions for future research (Chapter 9).

2. A Horizontal Use Case

Outlier detection is not an end in itself, but a tool to create value by solving a real world problem. There usually is a specific application that motivates to spend effort on data analysis and on interpreting the results. In this section, we introduce a real use case that has been an overarching motivation to the fundamental research presented in this thesis. Our use case is from the energy data domain, a research area that has gained much attention in the last years, see [Sta⁺18; Bar⁺18; Vol⁺19] for examples. More precisely, we are looking at energy consumption data, collected from an industrial production site [Bis⁺18]. There, the objective is to identify unusual energy consumption patterns that indicate unexpected machine behavior or system misconfigurations. Detecting such unusual patterns can, for instance, facilitate the prediction of machine faults, which in turn contributes to the overall robustness of the production.

In the following, we describe our full data processing pipeline, from raw data acquisition to interactive outlier detection. One objective is to illustrate the complexity of active learning with outlier detection – something typically not discussed in literature on active learning. A second objective is to explain how our contributions fit within this pipeline.

2.1. The Use Case

The raw data is collected by smart meters installed in an industrial production site that produces electronic systems [Bis⁺18]. The production is order-related, in small batches for varying customers, including research projects on particle physics and on battery systems. Several machines, like a soldering oven and pick-and-place unit, have been instrumented with high-resolution smart meters that collect measurements on up to three phases with a time resolution of about 5 s between subsequent measurements. The smart meters measure several electrical quantities, including voltages, current, frequency, active and reactive power, and harmonic distortion. This results in a multi-variate time series of sensor measurements for each production machine. Figure 2.1 depicts a subset of the quantities measured at a soldering oven for 20 min.

2.2. The Pipeline

Existing methods on active learning for outlier detection rely on a static vector space as an input, see Chapter 4. This is, each observation is of the format $\langle x_1, x_2, \dots, x_d \rangle$, where x is a numerical or categorical attribute, and d the number of attributes in the data set. In our use case, however, measurements are time series $\langle t_1^s, t_2^s, \dots, x_T^s \rangle$ of numerical measurements for points in time $1 \dots T$, collected for each electrical quantity s. A naive way to transform

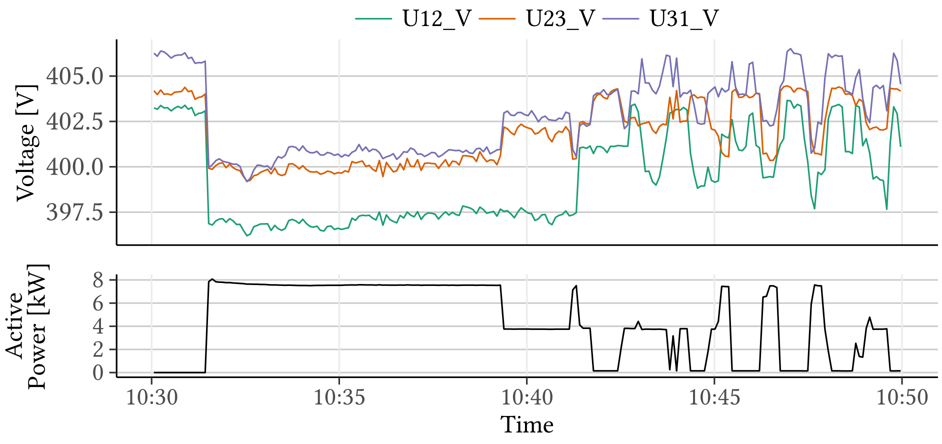


Figure 2.1.: Illustration of smart-meter data for a soldering oven. Graphic reproduced from [Bis⁺18].

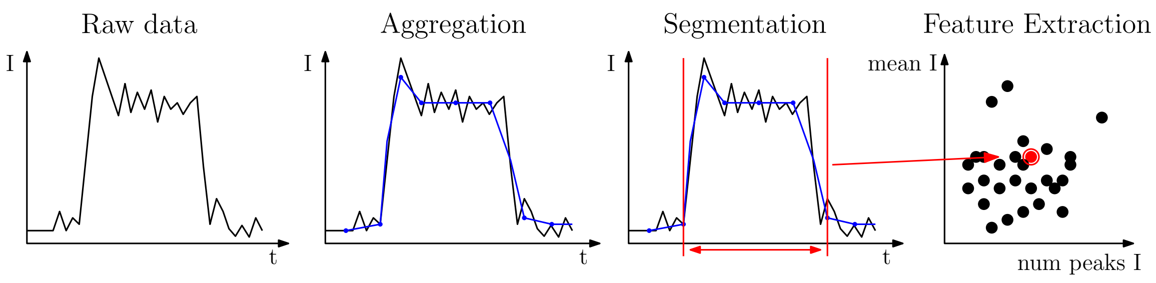


Figure 2.2.: Illustration of pre-processing steps.

the time series to a static vector is to interpret each time step as one dimension. However, this is not viable. For one, time steps are not necessarily equidistant. Another reason is that the dimensionality of the vector space would be increasing with the length of the time series, and quickly becomes prohibitively large. Thus, one has to rely on more complex pre-processing and feature engineering. In the following, we give an overview on the steps required with our use case.

2.2.1. Pre-processing

Our pre-processing consists of three steps, see Figure 2.2 for an illustration. The first step is data aggregation, an optional step to reduce the data volume. Then, one segments the full time series into sequences, and finally uses feature engineering to map the extracted sequence to a static vector.

Aggregation. A high temporal resolution results in large data volumes, and challenges the scalability of downstream analyses. To reduce this volume, one can use temporal aggregation, for instance by summarizing measurements by their average over 15 min intervals. However, we have recently shown that this can affect result quality, depending on the aggregation function and interval [TBB18; TBB19; Wer+19]. Thus, one has to evaluate if such effects occur in a given application, and then carefully balance between the benefits of a reduced volume and the impact on the result quality.

Segmentation. After aggregation, one splits the time series into sequences. Each sequence that has been extracted this way is then mapped to a numeric vector in the so-called feature space. Segmentation requires to decide whether one wants to extract sequences with *variable* or *fixed* length [TBB19]. Fixed length means to decide upon a temporal window length, say 15 min, and split the time series in non-overlapping windows. Variable-length sequences are slightly more difficult to extract. For instance, one may extract a sequence as the cohesive time window where a machine has been consuming energy, i.e., where the active power is above a specified threshold. One may further filter sequences to the ones relevant to the application [Vol+19], e.g., retaining only intervals where the average current is above a certain threshold. In either case, the type of extraction determines the reference group in which outliers can be detected. When sequences are fixed to say 15 min, only a 15 min sequence can be detected as unusual relative to other, normal 15 min intervals. When sequences are of variable length, a sequence can only be detected as unusual relative to other, normal sequences of variable length, e.g., intervals where the machine has been consuming energy. The kind of unusual sequences one is interested in depends on the application.

Feature Extraction. The final step of pre-processing is feature selection, i.e., the mapping of sequences to numeric vectors. A simple example of a feature mapping are statistical summaries, like the mean, variance and extreme values. However, there virtually is no restriction on the possible mappings, and literature has proposed a plethora of feature extraction methods. A comprehensive overview of this field is not focus of this thesis. For energy time series, however, we have provided an extensive overview on available features elsewhere, with guidelines how to select them systematically [Vol+19]. After feature extraction, the data finally has the structure required to apply active learning for outlier detection.

2.2.2. Active Learning Cycle

Active learning involves three steps: initialization, classifier retraining, and query selection. In this section, we focus on an intuition of these steps, and how they relate to our use case and contributions. In Chapter 3, we give a detailed, technical introduction.

Initialization. After transforming the time series to a feature space, the data is a set of vectors, but there are not yet any class label annotations; after all, acquiring annotations is one of the motivations for active learning. So in this case, active learning is a cold start problem, which is challenging for several reasons. First, a suitable classifier must work in an unsupervised setting, i.e., based on observations without class labels. Once annotations are available, the classifier must feature a mechanism to incorporate them during training. As mentioned earlier, existing active learning for outlier detection therefore relies on semi-supervised one-class classifiers.

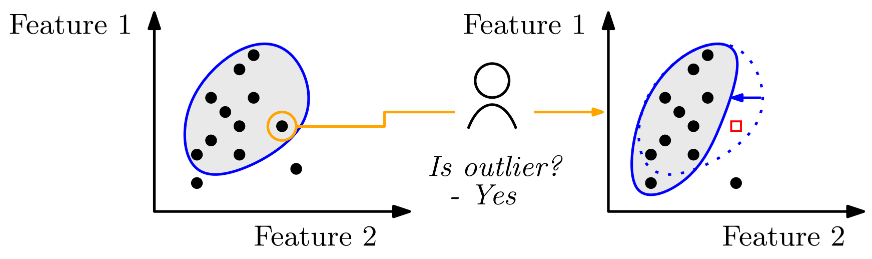


Figure 2.3.: Illustration on adapting decision boundaries with one-class classifiers. Observations classified as normal are inside the shaded area.

Next, for one-class classifiers to work well, one has to choose suitable hyperparameter values. Without annotated observations, methods like cross-validation for hyperparameter tuning are not applicable. Also, setting hyperparameter values is typically not intuitive, and one cannot expect a user to estimate values just based on experience. Thus, one has to rely on heuristics. However, existing heuristics to initialize one-class classifiers do not give any indication of how good the selected parameters are. They still require an elaborate validation by a human. In Chapter 6, we propose a different way to estimate hyperparameter values based on active learning to overcome this issue (Contribution *Selection of Hyperparameters*).

Classifier Training. Once the one-class classifier is parameterized, it is trained on the feature vectors. In a nutshell, training a one-class classifier means to search for a decision boundary in the feature space that separates regions of normal from regions of unusual observations. The classifier then adapts the decision boundary based on annotations. Figure 2.3 is a schematic to illustrate the adaptation of the decision boundary in a 2-dimensional feature space.

Selection. Active learning methods select observations where annotations help the classifier to better distinguish between usual and unusual regions. In Figure 2.3, the selected, highlighted observation is close to the decision boundary. This is a very common strategy with active learning. The rationale is that there is uncertainty about the true class label for observations that are close to the boundary. Also, the decision boundary is more likely to change by annotating observations close to the boundary than the ones far from it.

After a user has annotated an observation, the active learning cycle repeats by retraining the classifier followed by the selection of the next observation to annotate. There are different ways to decide when to terminate the cycle, for instance when a predefined share of observations has been annotated.

On a conceptual level, the active learning cycle is straightforward. However, annotating an observation turns out to be a difficult task for users, considering the different pre-processing steps, and the level of abstraction by modeling the real-world problem by a one-class classifier. To illustrate, think about a simple setting where feature engineering involves only five features, say, some basic statistical summaries. Calculating these features for five electrical quantities results in a 25-dimensional feature vector. In this case, answering the question *"Is the observation $\langle x_1, \ldots, x_{25} \rangle$ an outlier?"* requires a user to

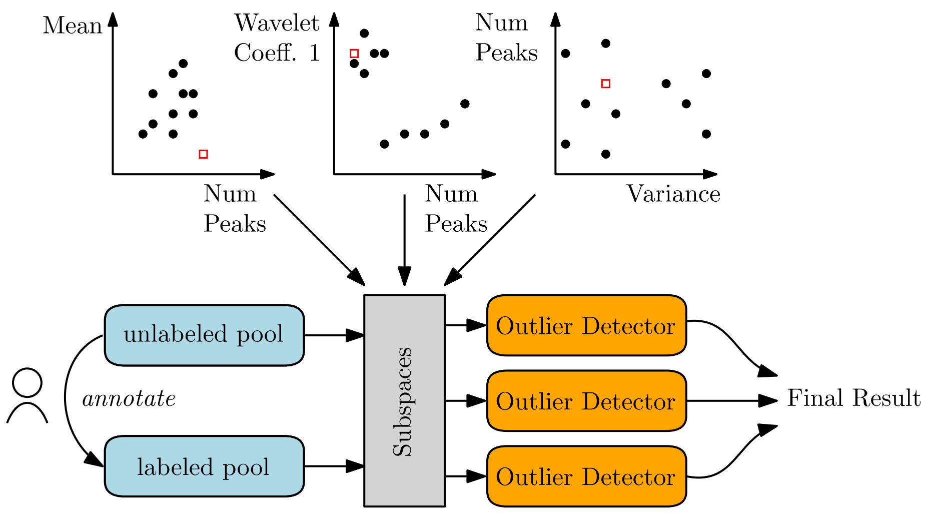


Figure 2.4.: Illustration of an active learning cycle extended to multiple subspaces.

understand the meaning of the derived features, and to have some intuition on how data is distributed in the feature space. From our experience, this is an unrealistic expectation that does not hold up in practice. In a more realistic and complex scenario, with more features and electrical quantities, annotating feature vectors is even harder.

Intuitively, annotating observations is easier in low-dimensional feature spaces, in particular if data can be visualized in two or three dimensions. In fact, there is a branch of outlier detection that focuses on searching for outliers in subspaces, i.e., low-dimensional projections of the data. One motivation for using such methods is to strive for a description of outliers and for an increased interpretability of algorithmic results [Dan+14; Mül+12]. However, subspace outlier detection has only been studied in the context of unsupervised outlier detection, and not in connection with active learning. In this thesis, we bring together the concept of low-dimensional projections, semi-supervised one-class classifiers, and active learning (Contribution *Active Learning in Subspaces*).

On a conceptual level, this results in a more comprehensive cycle, since it now comprises multiple subspaces. Figure 2.4 is a schematic of the active learning cycle with subspace. There, the feature space is split into multiple projections. Each of these projections yields a decision boundary. The final result is a combination of the classifications in the individual projections: an observation is an outlier if it falls outside the decision boundary in at least one of the projections. However, there are several conceptual and technical challenges on how to select and use annotations with multiple subspaces, which we discuss in detail in Chapter 5.

In this section, we have introduced an end-to-end use case to illustrate how our technical contributions fit into the data processing pipeline for outlier detection with active learning. One important takeaway from this use case is that the pipeline from raw data to a final classification result is very complex, and involves many design choices. This motivates to break down the classification into subspaces where providing annotations is less difficult for a user. Another takeaway is that the steps of the pipeline all serve a specific purpose, with well-defined input and outputs. For instance, subspace-based active learning requires a set of relevant subspace projections as input, which one can obtain by subspace search heuristics [KMB12; TB19a]. The input required by subspace search heuristics is data in form of numeric vectors, which is the output of feature engineering. These input-output relationships give way to study each of the steps along our use case individually. Of course, to validate the end-to-end benefit, one must take a holistic view. In Chapter 7, we therefore study active learning for outlier detection from a system perspective, and discuss open issues with validating end-to-end use cases with user studies (Contribution *User Study*).

3. Fundamentals

This chapter introduces fundamentals and notation. The focus is to provide a general background on outlier detection and active learning. We present specifics on one-class classification and its combination with active learning in Chapter 4.

3.1. Notation

Data. Let $\mathcal{X} \subseteq \mathbb{R}^M$ be a data space with M dimensions. We refer to a data space dimension as an *attribute* or as a *feature*. A data set is a sample $\mathbf{X} = \{x_1, x_2, ..., x_N\}$ from $\mathcal{X}$ of size N. So each observation $x_i = \langle x_{i1}, x_{i2}, \dots, x_{iM} \rangle \in \mathbf{X}$ is an M-dimensional, numerical vector. Observations in $\mathbf{X}$ can be separated into two classes: *inlier* and *outlier*. Inliers make up for the majority of observations in $\mathbf{X}$, and have a common but unknown underlying distribution. In contrast, outliers are rare, and each of the observations may come from a separate, independent distribution. We do not make any assumptions about the data distribution other than the class imbalance.

Class Labels. The categorical label $y_i \in \{+1(\text{inlier}), -1(\text{outlier})\}$ denotes the class membership of an observation x_i. This is the *ground truth* or *gold standard*, i.e., the actual class membership. One has to be careful on how to interpret "truth" in this context. For benchmark data, the ground truth has typically been collected manually, i.e., a human went through the trouble of annotating all of the observations in $\mathbf{X}$. As explained earlier, the definition of what constitutes an outlier is subjective, so any ground truth must be treated with caution. We discuss the challenges of using standard benchmark data for outlier detection in Section 3.4.

Label Pools. In real applications, ground truth labels often are not available. In this case, an annotator, the *oracle*, can provide a class label for specific observations. We distinguish between the set of observations without such an annotation, the *unlabeled pool* $\mathcal{U}$, and the *labeled pools* of observations that have been annotated as inlier $\mathcal{L}_{\text{in}}$ and the ones that have been annotated as outlier $\mathcal{L}_{\text{out}}$. The sets $\mathcal{U}$, $\mathcal{L}_{\text{in}}$ and $\mathcal{L}_{\text{out}}$ are pairwise disjoint, and $\mathbf{X} = \mathcal{U} \cup \mathcal{L}_{\text{in}} \cup \mathcal{L}_{\text{out}}$. We use $\mathcal{L} = \mathcal{L}_{\text{in}} \cup \mathcal{L}_{\text{out}}$ as a shorthand. Annotating an observation x_i from $\mathcal{U}$ results in updated sets $\mathcal{U}' = \mathcal{U} \setminus \{x_i\}$ and either $\mathcal{L}_{\text{in}}{}' = \mathcal{L}_{\text{in}} \cup \{x_i\}$ or $\mathcal{L}_{\text{out}}{}' = \mathcal{L}_{\text{out}} \cup \{x_i\}$. Technically, the ground truth label does not have to be the same as the ones provided by an annotator. Further, two different annotators may return different labels for the same observation. In the following, however, we assume that there is only one annotator, and that the annotator always returns the ground truth label.

3.2. Outlier Detection

Outlier detection, often also called anomaly detection, has been a focus of research for decades. The main objective of outlier detection is to identify observations that are "different than the majority of the data and therefore suspicious" [ZS17]. Outlier detection is not an end in itself, and there generally is an overarching, application-specific motivation to it. For instance, identified outliers can be a hint for fraud, indicate system misconfigurations and facilitate medical condition monitoring [HA04]. Another motivation is to remove outliers from the data set to increase the quality of a downstream data analysis task.

There is a plethora of method to detect outliers. The research area is very active, and every year novel methods, like using artificial neural networks [PFB19] or ensemble-based methods [ZCS14], are being proposed. In this section, we focus on properties and concepts which are important to the methods we present in the body of this thesis. For a comprehensive overview and a taxonomy of different outlier detection methods, we refer to the survey in [Agg15].

3.2.1. Outlier Scores and Classification

Per definition, outlier is relative concept, i.e., observations are unusual with respect to normal, regular observations. However, the difference between "unusual" and "normal" is not necessarily dichotomous. The following short example illustrates this.

> **Example 1 (French Fries)** *Imagine you have ordered a bowl of French fries from your favorite burger chain. Once you start eating from this bowl, you find mostly regular fries. But for some reason, a curly fry and a banana happen to be in the bowl as well. A curly fry may not be what you have expected, although you might actually be happy to have one in your bowl. Thus, the curly fry is somewhat unusual with respect to the normal fries; it is an* outlier. *However, a banana in a bowl of fries is way out of the ordinary, and thus even* more *unusual than the curly fry.*

In this example, both of the irregular food items, the curly fry and the banana, are outliers. However, they both have a different degree of unusualness. We explain how outlier detection deals with different degrees of unusualness in the following.

Score-based Methods

Outlier detection methods quantify different degrees of unusualness by returning a *score* for observations [TEB18].

> **Definition 1 (Outlier Score Function)** *An outlier score function is a function of type score*: $\mathbf{X} \rightarrow \mathbb{R}$*, i.e., a function maps each observation of sample* X *to the real line.*

For simplicity we assume $score(x) \in \mathbb{R}_{[0,1]}$, where higher scores indicate more unusual observations.[1]. In our example, this would result in in a value close to 0 for regular fries, a

[1] In general, outlier scores may have any scale, some even on an open interval, but there has been some effort to unify score values [Kri⁺11].

medium score for the curly fry and a high score for the banana. There are many score-based methods that rely on different principles to quantify unusualness. Examples are statistical models, e.g., based on the Mahalanobis distance, and proximity-based methods like LOF [Bre+00] and LOCI [Pap+03].

Scores induce a ranking of observations by their degree of unusualness. In many cases, however, one is only interested in a binary decision, and a ranking is not of relevance. For a binary decision, scores can be transformed back into categorical labels by applying some threshold on the score values. A score above (below) the threshold is assigned to the outlier (inlier) class. However, there is no universal way on how to set such a threshold. In the French Fries example, assume that a person is very picky about fries. In this case, the threshold might be very low to make sure that curly fries, and maybe even overly small or large regular fries are labeled as outliers. A less picky person may just be fine with anything fried to be in the bowl, so they are inclined to apply a higher threshold. So setting the threshold is subjective and application-specific. This has important implications. When one searches for outliers at scale in, say smart meter data, setting a low threshold may flag a lot of observations as unusual, which an analyst in turn must investigate. This might result in a high false positive rate, which entails high effort of investigating many presumed outliers, and eventually may frustrate users. On the other hand, when the threshold is high, one might miss important unusual observations.

To summarize, working with outlier scores allows a nuanced separation between inliers and outliers. However, assigning class labels to observations requires to set a threshold, which is application specific, and requires to trade off between false positives and false negatives.

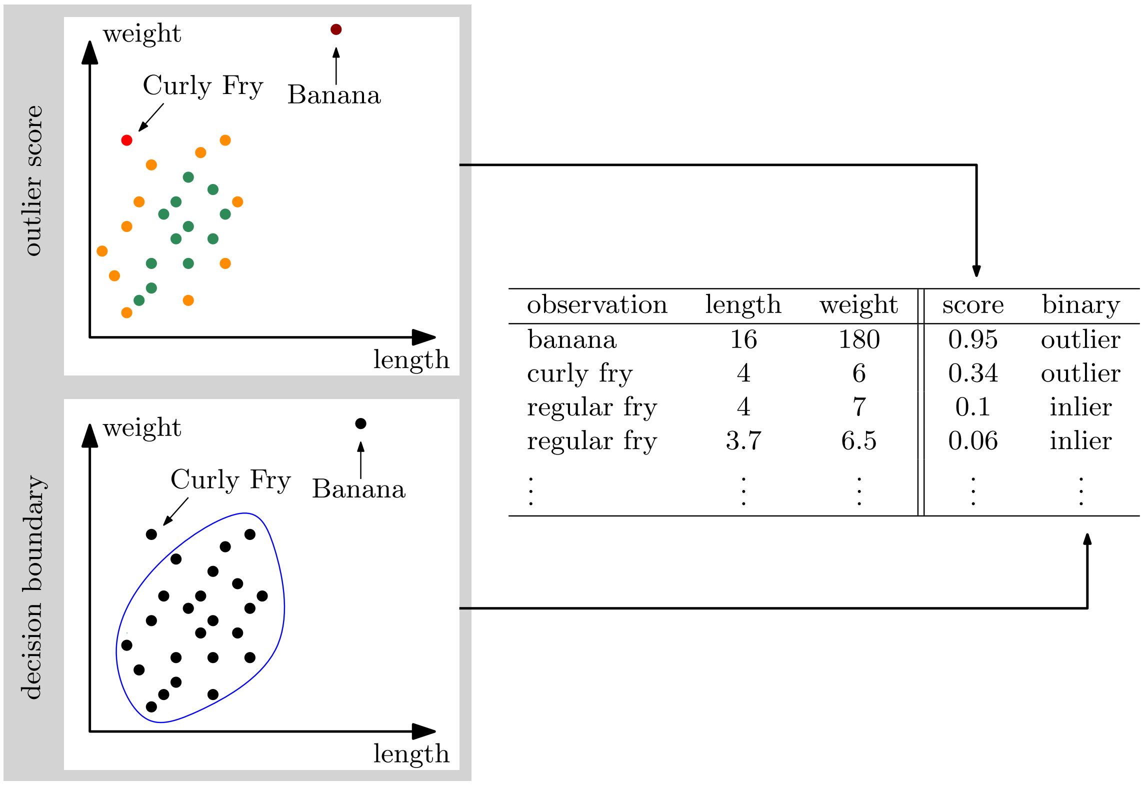
Binary Classifiers

A second type of outlier detection methods directly return a binary classification. The basic idea of binary classifiers for outlier detection is to draw a boundary in the data space that separates inliers from outliers. They are called one-class classifiers, since they focus on learning the concept of the inlier class, and classify any observation that does not fit that concept as outlier. The formal language to describe the inlier concept are decision functions.

> **Definition 2 (Decision Function)** *A decision function f is a function of type $f : \mathcal{X} \rightarrow \mathbb{R}$. An observation is an outlier if $f(x) > 0$ and an inlier otherwise.*

A decision function is different to a threshold with score-based methods. A threshold is not defined on the data space, and thus does generally not induce a function in the data space. Further, thresholds are defined a-posteriori, after the detection has been carried out, while a decision function is intrinsic to the outlier detection method.

Figure 3.1 illustrates score-based outlier detection and binary classification based on the French Fries example. There, two attributes, length and weight, describe the objects in the bowl. This example shows an important advantage of binary decision functions over score-based methods: binary decision classifiers facilitate fast inference. This is, once



observation	length	weight	score	binary
banana	16	180	0.95	outlier
curly fry	4	6	0.34	outlier
regular fry	4	7	0.1	inlier
regular fry	3.7	6.5	0.06	inlier
⋮	⋮	⋮	⋮	⋮

Figure 3.1.: Outlier scoring and binary classification for a two-dimensional example.

the decision function is fit on training data, one can predict the class label for yet unseen observations by applying the decision function to them. The result of score-based methods is typically confined to the training data. This is, when a new observation arrives, they require a full retraining to infer the score of the new observation. There are dynamic variants of score-based methods to address this issue. However, online learning algorithms focus on stream data, i.e., a setting where novel observations arrive continuously at high velocity, and where a classifier must adapt itself to changes in the data distribution. Stream data analysis is a problem class different to our setting, and not a focus of this thesis.

Support Vector Data Description. We now introduce Support Vector Data Description (SVDD) [TD04], a very popular binary classifier for outlier detection. SVDD and some variants of it are the foundation for the active learning methods presented in the body of this work. The basic idea of SVDD is to fit a hypersphere with radius R and center a around the data such that the majority of the observations lies within the hypersphere. One way to interpret this hypersphere is that it defines the support of the inlier distribution, i.e., the regions of the data space with positive inlier density. Another way to interpret the hypersphere is as a description of inliers. Any observation that does not fit this description, i.e., observations that fall outside the hypersphere, are outliers. By decreasing the radius, the description becomes more conservative, and more observations fall outside the description. On the other hand, one can increase the radius such that all observations in the training data are within the hypersphere. Formally, the problem of including all observations in the hypersphere is a Minimum Enclosing Ball (MEB) problem.

$$\begin{aligned} \text{MEB:} \quad & \underset{R,a}{\text{minimize}} && R^2 \\ & \text{subject to} && \|x_i - a\|^2 \leq R^2, \quad \forall i \in \{1 \ldots |\mathbf{X}|\} \end{aligned} \tag{3.1}$$

The optimal solution of MEB can hinge significantly on a few observations in the data space that are distant to all other observations. The reason is that MEB must enclose *all* observations, which is also called a hard-margin problem. By relaxing this requirement, one can transform MEB to a soft-margin problem. The idea is to allow certain observations to fall outside the hypersphere, if this allows to decrease the radius of the hypersphere significantly. Formally, this is realized by introducing the slack variables $\boldsymbol{\xi}$ to the optimization problem. Observations that fall outside the hypersphere have positive slack, $\xi_i > 0$, equivalent to the distance of this observation to the hypersphere. By assigning costs C to positive slack, one can influence the trade-off between the cost of increasing the radius vs. the total cost of having observations with positive slack. This results in the following optimization problem.

$$\begin{aligned} \text{SVDD (Primal):} \quad & \underset{R,a,\boldsymbol{\xi}}{\text{minimize}} && R^2 + C \cdot \sum_i \xi_i \\ & \text{subject to} && \|\Phi(x_i) - a\|^2 \leq R^2 + \xi_i, \ \forall i \\ & && \xi_i \geq 0, \ \forall i \end{aligned} \tag{3.2}$$

Intuitively, if $C \geq 1$, the cost of positive slack is higher than to increase the radius, and SVDD reduces to MEB. Thus, without loss of generality, we restrict C to $[0, 1]$. Choosing a high value for C makes excluding observations from the hypersphere more expensive. Thus, in the optimum, less observations fall outside the description. Vice versa, setting C to low values results in a large proportion of observations that fall outside the description. Choosing a good value for C is difficult and application specific, see Chapter 6.

In many cases, it is not possible to separate inlier from outlier data by a hypersphere. In this case, one can search for a non-linear mapping of the data to some other space where this separation is possible. Technically, this is realized by a map $\Phi\colon \mathcal{X} \rightarrow \mathcal{F}$, where $\mathcal{F}$ is a reproducing kernel Hilbert space, also called feature space. To avoid confusion with the concept of a feature space introduced in Chapter 2, we denote $\mathcal{F}$ as the *kernel feature space.* However, the feature mapping must not be calculated explicitly. This follows from the structure of the Lagrangian dual of SVDD

$$\begin{aligned} \text{SVDD (Dual):} \quad & \underset{\boldsymbol{\alpha}}{\text{maximize}} && \sum_{i,j} \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle + \sum_i \alpha_i \langle \Phi(x_i), \Phi(x_i) \rangle \\ & \text{subject to} && \sum_i \alpha_i = 1 \\ & && 0 \leq \alpha_i \leq C, \ \forall i, \end{aligned} \tag{3.3}$$

with dual variables $\boldsymbol{\alpha}$. The SVDD dual only depends on inner product calculations in the feature space. In this case, one can use Mercer's Theorem to replace the inner product by a kernel function $k\colon \mathcal{X} \times \mathcal{X} \rightarrow \mathbf{R}$. So an explicit mapping of observations to the feature is not necessary. This is also known as the *kernel-trick*.

The kernel-trick gives way to learn arbitrarily shaped decision functions by choosing an appropriate kernel function. To this end, one can choose from many different, sometimes domain-specific kernel functions. Examples are the polynomial kernel $k(x, x') = \langle x, x' \rangle^d$ with parameter d, and the Gaussian kernel

$$k(x, x') = e^{-\gamma \|x - x'\|^2}, \tag{3.4}$$

with parameter γ. The Gaussian kernel sometimes also called the Radial Basis Function (RBF) kernel, and has an alternative formulation $k(x, x') = e^{-\frac{\|x-x'\|}{2\sigma}}$, where $\gamma = \frac{1}{2\sigma}$. It is by far the most popular kernel with SVDD, and it has been shown to work well with a variety of applications. By choosing its parameter value γ, one can influence the complexity of the decision boundary. This becomes apparent when interpreting the kernel function as a similarity between observations. Identical observations take a value of 1 and observations that are orthogonal in the kernel feature space take a value of 0. So when $\gamma = 0$, all kernel functions evaluate to 1, i.e., all observations are projected to the same vector in the kernel feature space. The resulting decision boundary in the data space is a perfect hypersphere. For $\gamma \rightarrow \infty$, all pairwise similarities of non-identical observations approach 0, i.e., the observations are projected to vectors that are orthogonal to each other. In this case, and when there are no duplicates in the data space, the kernel feature space is N-dimensional and the resulting decision boundary highly non-linear. So with increasing values of γ, the decision boundary becomes more and more complex, and fits the shape of the training data distribution more closely. However, this also means that SVDD is more likely to overfit to the training data, i.e., SVDD does not generalize and is likely to result in a high prediction error on unseen observations. Determining which complexity is required to allow sufficiently flexible boundaries without overfitting to the data is a difficult problem, see Chapter 6.

After choosing the two parameters C and γ, solving SVDD gives a solution for R and a. This solution induces the following decision function in the data space

$$f(z) = \|z - a\|^2 - R. \tag{3.5}$$

The interpretation is straightforward. If the distance of an observation to the center is smaller than the radius, the decision function yields a negative value, and the observation is classified as an outlier, cf. Definition 2. Again, the decision function can be expressed by inner products only [TD04]. Substituting a with $\sum_i \alpha_i$ (see Equation 3.3) gives

$$\|z - a\|^2 = \langle z, z \rangle - 2 \sum_i \alpha_i \langle z, x_i \rangle + \sum_{i,j} \alpha_i \alpha_j \langle x_i, x_j \rangle. \tag{3.6}$$

Replacing the inner product with the kernel function results in

$$\|z - a\|^2 = 1 - 2 \sum_i \alpha_i k(z, x_i) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j). \tag{3.7}$$

Interestingly, only the terms with $\alpha_i > 0$ are relevant for the distance calculation, and the respective observations are called support vectors. Further, the third term only depends

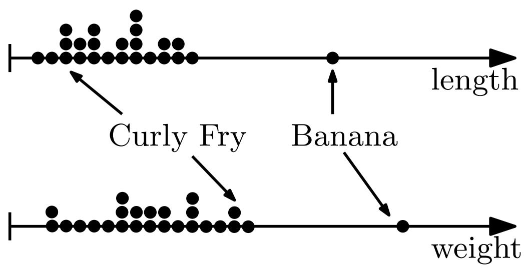


Figure 3.2.: One-dimensional projections of the length and the weight attribute.

on the training data, i.e., is constant over all predictions. Thus, evaluating the decision function for unseen observations is very efficient.

In summary, we have presented two different approaches to outlier detection: score-based methods and binary classifiers. Binary classifiers have an advantage over score-based methods in applications where one has to make predictions for a large number of observations. A further benefit of binary classification is that its result, binary class labels, are more simple to interpret, in particular when decision boundaries can be visualized. In Chapter 4, we will show that binary classifiers are also a good choice with active learning for outlier detection.

3.2.2. Subspace Outliers

Let us return to the French Fries example. So far, we have only considered two attributes of the items in the bowl, length and weight. As we can see from Figure 3.1, these two attributes are sufficient to discern between fries, the inliers, and the other food items. In this specific example, these two attributes also are necessary to identify both of the outliers. This is, the curly fry no longer appears as an outlier when only looking at one of the attributes in isolation, see Figure 3.2. Of course, there are many more attributes to describe the items, like saltiness, nutritional value, color, temperature, and curvature, just to name a few. Among them, different combinations of attributes may discern outliers from inliers. But some attributes may not be relevant for this distinction. For instance, the attribute color may not be relevant, even in combination with other attributes, since all items have a yellow tinge. Also, some combination of attributes may more clearly differentiate between inliers and outliers than others. For instance, a combination of curvature and weight may be better than color and saltiness.

There is a further consequence of using many attributes for outlier detection, which is less intuitive. At first, more attributes seem useful to single out the outliers more easily. However, adding irrelevant attributes to the data set, i.e., attributes where outliers do not deviate from the inliers, actually impedes outlier detection. The reason for this effect is that pairwise distances between observations become more similar with increasing data dimensionality [ZSK12]. This phenomenon is called the *curse of dimensionality*, and has been studied extensively in outlier detection literature. The mathematical reason for this is the *concentration of distances* with increasing dimensionality. A respective theorem [Bey+99] states that when the number of dimensions M of a data space increases, the probability that the minimum D^M_{min} and maximum distance D^M_{max} over all observations in the data set only differ by a factor $(1 + \epsilon)$ is 1. Formally,

$$\lim_{M \to \infty} P(D^M_{max} \leq (1 + \epsilon)\, D^M_{min}) = 1 \quad , \forall \epsilon > 0. \tag{3.8}$$

So distances between observations are negligible in high-dimensional data. The implication of this is that proximity-based outlier detection scores lose meaning in high-dimensional data spaces as well.

Subspace Search

To overcome this issue, literature has proposed *subspace outlier detection* methods that search for outliers in projections of the data. If the dimensionality of these dimensions is sufficiently low, the concentration of distances is insignificant, and proximity-based outlier detection is meaningful. However, finding the relevant low-dimensional projections of the data, i.e., the ones where outliers are apparent, is difficult. The reason is that the number of candidate projections to consider is $2^M - 1$, i.e., the number increases exponentially with the number of dimensions of the data space. An exhaustive search through all possible projections is prohibitive. Therefore, literature has come up with so-called *subspace search* methods that traverse the space of projections heuristically to find the most relevant ones. One can distinguish between two different categories of subspace search approaches: correlation-based and object-based approaches [TB19a].

Correlation-based methods search for subspaces independently of the outlier detection [KMB12; TB19a; NMB13; Ngu+13]. The basic idea is to use some correlation measure to quantify the relevance of a subspace, and use it to select a good set of subspaces. The rationale behind this is that outliers are expected to not follow some of the correlations that one can observe between attributes, and thus are more likely to be apparent in correlated projections. Outlier detection methods are then applied to each of the subspaces selected individually. An example is the subspace with the weight and length attribute, see again Figure 3.1. There, length and weight of regular fries are correlated, and the curly fry falls out of this pattern. To derive an outlier score for an observation, the score-based method is applied to each of the subspaces, and the subspace scores are combined, e.g., by taking the average over a set of subspaces $\mathcal{S}$.

$$score(x) = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} score_S(x) \tag{3.9}$$

A recent contrast-based method is Greedy Maximum Deviation (GMD), which we have proposed elsewhere [TB19a]. GMD quantifies the subspace relevance by the deviation between the joint distribution of its attributes and the marginal distributions. The basic idea is that when attributes $\{s_1, s_2, \ldots, s_d\}, d \leq M$ in a subspace S are correlated, the marginal distribution of an attribute $s_i, i \in 1 \ldots S$ changes when restricting the remaining attributes of that subspace $s_j, j \neq i$ to some interval $[l_j, r_j]$, where $\min s_j \leq l_j < r_j \leq \max s_j$. The deviation is the difference in the cumulative distribution function (CDF) of the restricted and non-restricted marginal distribution, measured by some statistical test. For instance, one can measure the difference by the Kolmogorov Smirnoff test which uses the peak difference between two CDFs.

$$dev^c(s_i, S) = \sup_{x \in \mathbf{X}} |\hat{F}_{s_i}(x) - \hat{F}^c_{s_i}(x)| \tag{3.10}$$

where $\hat{F}_{s_i}$ is the empirical marginal CDF of s_i, and $\hat{F}^C_{s_i}$ the empirical marginal CDF of s_i under the interval restrictions c on all other attributes. The final subspace relevance q for attribute s_i is the average over several randomly selected interval subsets $\{C_1, \ldots, C_K\}$.

$$q(s_i, S) = \frac{1}{K} \sum_{c \in C_1, \ldots, C_K} dev^c(s_i, S) \tag{3.11}$$

GMD uses a greed heuristic to select one subspace for each of the data space dimensions individually. It starts by selecting a two-dimensional projection for dimension s_i that yields the highest relevance, and adds additional dimensions as long as they increase the relevance further. The result is a set $\mathcal{S}$ with $|\mathcal{S}| = M$ subspaces, i.e., one subspace for each dimension of the data space.

The second type of subspace search approaches are *object-based methods*. Here, the basic idea is to search for a set of subspace where a given observation occurs as an outlier [Kri+12; MSS11; ZGW06]. A more recent focus of object-based methods is to select subspaces that are relevant to *describe* the outlier, and to make outlier detection more interpretable [Mic+13; Gup+18]. However, these methods require a set of classified outliers as input. This is orthogonal to the detection of outliers, the focus of our work.

With subspace outlier detection, one distinguishes between a *subspace outlier*, i.e., an outlier that is apparent in a specific subspace of the data, and a *full-space outlier*, i.e., an outlier that is apparent in the full data space. We denote the subspaces where an observation is detected as an outlier as *outlying subspaces*. An important characteristic of subspace outliers is their *multi-view* property [Kel15].

> **Definition 3 (Multi-View Property)** *A subspace outlier can occur in multiple subspaces. The set of outlying subspaces for one observation are independent of the outlying subspaces of any other observation.*

The multi-view property is one of the reasons why adapting binary outlier classification and active learning to subspaces is difficult, see Chapter 5.

3.2.3. Supervised and Unsupervised Detection

A further fundamental distinction of outlier detection methods is between unsupervised, supervised and semi-supervised detection approaches. The level of supervision specifies to which extent annotated observations are used by a classifier. Note that there is a subtle difference between the availability of class labels, and whether they are actually used by a classifier.

Unsupervised outlier detectors cannot make use of any class label annotations. So even if annotations are available, i.e., $|\mathcal{L}| > 0$, they are treated equivalently to observations in $\mathcal{U}$. Consequently, annotating observations to increase the size of $\mathcal{L}$ does not have any effect on the outlier detection method. The only way to influence the classifier is by modifying the data set the method is applied to. For instance, one can train an unsupervised classifier on a sub-sample of $\mathcal{X}$, or only on observations in $\mathcal{L}_{\text{in}}$. In practice, there often is an insufficient number of annotated outliers for supervised methods to work well. So most outlier detection methods, including the ones that we have presented earlier in this section, are unsupervised.

When annotated observations are available for both classes, one can apply *supervised* methods. Supervised outlier detection uses only annotated observations $\mathcal{L}_{\text{in}}$ and $\mathcal{L}_{\text{out}}$ to train a classifier. Generally, supervised methods become more accurate the more representative the annotated observations are. However, to be effective, they require $\mathcal{L}$ to be a sufficiently large. Further, supervised methods often make different assumptions about outliers than the ones we have introduced in the beginning of this chapter. One example for this is rare class detection. There, outliers still are assumed to be rare, but they are also expected to be similar to each other. This is an important difference to the assumption that outliers do not follow a common distribution. Rare class detection requires classifiers that work well with imbalanced data, or pre-processing techniques like over- and undersampling to mitigate class imbalance. An example is rare class detection with Support Vector Machines [HG10]

Finally, there are *semi-supervised* methods. Semi-supervised methods make use of both unlabeled and labeled observations. Typically, one assumes $|\mathcal{U}| \gg |\mathcal{L}|$, i.e., there only are a few annotated observations in addition to a large bulk of non-annotated data. Most semi-supervised outlier detection methods base on an unsupervised model. The core idea is then to bias the model such that it also fits to the available annotations. All this makes semi-supervised methods a good fit for active learning. We will introduce the semi-supervised classifiers used with active learning for outlier detection in Chapter 4.

3.3. Active Learning

Broadly speaking, active learning comprises methods that involves users during the training phase of a machine learning algorithm. The initiator of this involvement is the machine that asks the user to provide some input that the machine expects to be useful to improve upon some objective. Active learning has been used with many machine learning tasks, such as classification and filtering, speech recognition, information extraction and computational biology, see [Set12] for an overview.

Active learning has an intuitive interpretation that bases on the core objective of classification. The core objective of classification is to train a classifier that generalizes well from a training sample to the true underlying distribution. Active learning strives to collect annotations that help to gradually narrow down the space of all possible generalizations to the best one. To illustrate, let us once again return to the French Fries example. Assume

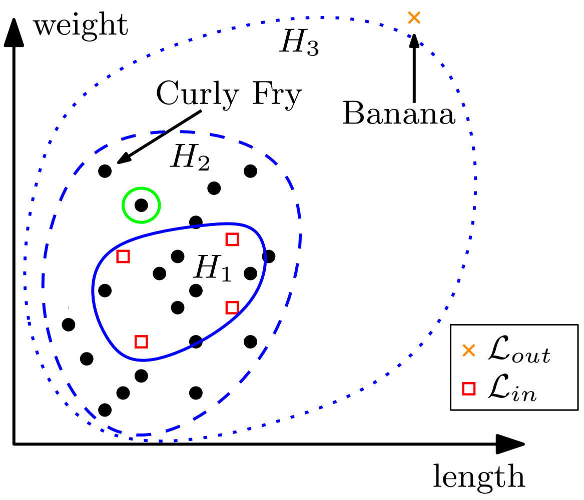


Figure 3.3.: Three different classifiers (solid, dashed and dotted lines) that agree with the annotations available.

that someone has carefully collected the length and weight measurements for the food items in the bowl. Additionally, this person has annotated five observations to be French fries. Figure 3.3 depicts the collected data together with three possible decision boundaries. These boundaries may come from different classifiers, the same classifier with different hyperparameter settings, or the same classifier and hyperparameter values but trained on different subsamples of the data. For instance, the most inner, solid decision boundary might be the result of SVDD trained on $\mathcal{L}_{\text{in}}$, the middle, dashed decision boundary the result of SVDD trained on $\mathcal{L}_{\text{in}} \cup \mathcal{U}$. All of the decision boundaries depicted agree equally well with the annotations, i.e., they have zero classification error. However, they represent different generalizations, of varying quality. One can interpret them as distinct hypotheses (H_1, H_2, H_3) on how to generalize the description of the inliers from the few annotated observations. The idea of active learning is to request additional information, the label annotations, to eliminate as many of the poor hypotheses as possible. For instance, consider to collect the class label for the observation marked with a green circle. If this observation is annotated as an inlier, one could reject H_1, since it does not agree anymore with the training data. Analogously, if the observation is annotated as an outlier, one could reject H_2 and H_2. In this case, the objective of active learning is to identify the observations that minimize the number of requests necessary to infer the correct generalization.

More accurately, however, active learning involves a trade-off between different types of cost: the cost of obtaining annotations, the cost of misclassification, and the cost for the resources of training a classification model and for finding good observations for annotation. Typically, one assumes that the cost of obtaining annotations is very high, and the cost of classifier retraining comparatively low. The cost of misclassification is usually not considered explicitly. The reason is that in an inductive setting, without a specific application, the error rate is a limit value analysis: any error rate larger than zero will cause infinitely high cost. So instead of modeling the cost of misclassification explicitly, literature typically assumes a fixed a budget available for annotation. Active learning then strives to minimize the classification error on this budget in an iterative

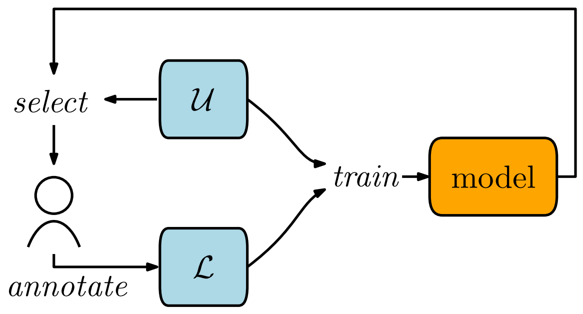


Figure 3.4.: Active learning process.

process. This process consists of three steps: training a machine learning model, selecting the relevant observations, and annotating the selected observations, see Figure 3.4. Once the annotations are collected, the cycle starts over by retraining the machine learning model based on the updated label pools.

In the remainder of this section, we focus on the general fundamentals of active learning independent of a specific machine learning task. Chapter 4 then focuses on the specifics with active learning for outlier detection. In the following, we first introduce some key terms. Then, we discuss different types of active learning scenarios. Finally, we introduce some general concepts for selecting relevant observations.

The user. So far, we have referred to the user rather abstractly as a feedback entity that receives and answers requests from a machine learning algorithm. In our work, we consider the user to be a human that has some domain knowledge on the problem studied. This is, the user familiar with the process that generated the data, and the data attributes. We further assume that the user can, with some effort, judge whether an observation is unusual. But for this judgment, the user requires sufficient information, like visualizations and explanations of the outcomes of the machine learning model. We will come back to this assumption in Chapter 5 and Chapter 7. However, the user may not be familiar with the underlying machine learning and active learning methods. Thus, we do not assume any technical knowledge that allows the user to manipulate the machine learning model by modifying hyperparameter values or the implementation.

The reason to assume a human as the feedback entity is purely pragmatic. For one, it is by far the most common scenario with active learning. Further, it motivates some of our contributions, like to ease the annotation process by active learning in subspaces, see Chapter 5. However, the concept “feedback entity” is more general. Basically, active learning is applicable to any scenario where data generation is inexpensive, and obtaining a class label costly. Think about active learning in research areas that rely heavily on computer simulations [BSM19]. An example are the material sciences, where finite element analysis simulations are used to assess material behavior under stress [Tri+18; Sch+19]. Here, the data are material configuration parameters and the stress applied to the material. Selecting a point in the data space, i.e., choosing some material and stress configuration, comes literally at no cost. However, obtaining a class label for a configuration is costly. The reason is that the class label is the outcome of the simulation, for instance whether the material *breaks* or *endures* the stress applied. Simulating one specific configuration can be very compute intensive, and can take a long time to complete, even on high-performance computing hardware. So in this scenario, the feedback entity is a computer simulation that, upon request, simulates a specific configuration and returns the simulation outcome. Similar considerations apply to real-world experiments, with additional cost for materials and with changeover times.

The Oracle. Research on active learning typically abstracts from a specific feedback entity. Instead, one models the feedback entity as an *oracle* that can answer specific requests. This model typically is realized as a software component within an experimental framework. An advantage of an oracle is that one has full control over its answers. For instance, one can guarantee that the answers are correct with respect to some given gold standard. It also gives way to investigate robustness of learning algorithms, e.g., by adding noise to the answers [DL10]. Simulating feedback by an oracle is a simplification that entails some shortcomings. For instance, an oracle that simulates answers from a ground truth does not incur relevant computational cost. In this case, one would need some additional cost models to study any cost trade-offs in active learning. A further shortcoming of an oracle is that it can, by default, answer any valid feedback request. This might not be true in a real setting, where a human might not be able to answer a request based on the information available, or where an experiment configuration is more difficult to conduct than another. We will elaborate more on this issue in Chapter 7.

The Query. The term *query* has an ambiguous meaning in literature. Most commonly, a query denotes the piece of data that is presented to some oracle with a specific request. However, a query can also refer to the specific format in which the request is presented to an oracle. For instance, the query may be an observation that is visually represented by a plot of the data space, or just by a numerical vector. Unless stated otherwise, we use the first definition without any specific representation. Further, we use the term *feedback* for the response of an oracle to a query.

In most cases, feedback are class-label annotations. However, there are other types of feedback conceivable. For instance, the query can be a feature for which the oracle has to estimate its relevance with respect to the machine learning task [RMJ06; DSM09]. Another example is negative feedback with multi-class classification. This is, when there are more than two classes, the query can be to name classes to which the observation does *not* belong.

Active Learning Scenarios

A fundamental categorization of active learning methods is the availability of observations that can be selected as queries. There are three different scenarios: pool-based learning, stream-based learning and query synthesis [Set12].

With *pool-based* learning, one assumes a closed, fixed set of observations that are available to be selected as queries. Thus, query selection is an optimization problem: finding the query where the machine learning model improves most based on feedback. This is the most common scenario, and also the one that we will take as a basis in this thesis. We have already assumed a pool-based setting in the French Fries example, where the pool is the bowl with a fixed amount of food items.

The second scenario is *stream-based* active learning. In this scenario, unlabeled observations arrive one at a time, and one has to decide on the spot whether this observation should be selected as a query or not. It is not possible to go back to an observation once it has been discarded. This scenario is plausible when observations arrive as a continuous stream, with high volume and velocity, or if the option to select the query only is available for a limited time. Going back to the French Fries example, imagine you want to survey customers in front of the restaurant and want to learn whether they will recommend the place based on socio-economic characteristics and order behavior. There only is a short window between the customer leaving the store and driving away. So the decision whether to survey (query) a customer has to be on the spot, and for each customer individually.

A third scenario is *query synthesis*. Here, one assumes that there are no unlabeled observations available, and one has to generate them artificially. There are many cases where this is plausible. For instance, assume that the entity that provides feedback is a computer simulation that takes some parameter values as input and returns a binary vector (success/error). Given that the domain of the parameters is known, one can synthesize any possible parameter configuration and run the respective simulation. This is reasonable, since there generally is no pre-defined set of experimental configurations to choose from, in particular if the parameter domain is continuous. However, query synthesis entails an important difficulty. In general, one cannot guarantee that all generated queries actually are meaningful. For instance, in the case of material simulations, one may end up generating material configurations as simulation input that are physically infeasible. Thus, one must think carefully about the query generation process [EB20], and enforce additional domain-specific constraints.

Query Strategies

Intuitively, a *query strategy* is a method to select one or more queries during the active learning iterations. In this section, we give an overview over different types of query strategies and discuss their general concepts. In Section 4.1.3, we then formalize and review query strategies specific to one-class classification.

A query strategy quantifies the benefit one can expect from obtaining feedback to one or more observation. Formally, this is realized as a measure of informativeness.

> **Definition 4 (Informativeness [TEB18])** *Let a decision function f, unlabeled observations $\mathcal{U}$ and labeled observations $\mathcal{L}$ be given. Informativeness is a function $x \mapsto \tau(x, \mathcal{U}, \mathcal{L}, f)$ that maps an observation $x \in \mathcal{U}$ to $\mathbb{R}$.*

In the following, we only write $\tau(x)$ for brevity. We call the specific realization of $\tau(x)$ for an observation its *informativeness score.* $\tau(x)$ induces a ranking of observations in $\mathcal{U}$ where high informativeness indicates high expected benefit. Generally, “benefit” relates to the improvement on the classification quality of the machine learning model. Since the actual improvement can only be observed after the oracle has been queried for feedback, $\tau(x)$

estimates the benefit based on the data and annotations available. Based on informativeness scores, a query strategy then selects one or more queries.

> **Definition 5 (Query Strategy [TEB18])** *A query strategy QS is a function of type*
>
> $$QS\colon \mathcal{U} \times \mathbb{R} \rightarrow Q$$
>
> *with* $Q \subseteq \mathcal{U}$.

There are two types of query strategies: *sequential* query strategies, where $|Q| = 1$, and *batch* query strategies, where $|Q| \geq 1$. Sequential query strategies return the observation with the highest informativeness[2]

$$Q = \arg\max_{x \in \mathcal{U}} \tau(x). \tag{3.12}$$

In this thesis, we focus on sequential query strategies. However, batch query strategies can be useful in scenarios where multiple annotators are available in parallel [Set11], or with high context switching cost when answering queries sequentially [GK11a]. Batch query strategies are generally more involved, and often require additional measures that go beyond informativeness. A reason is that the top-k observations ranked by informativeness may be very similar to each other [She+04; SZ05; Set12]. Thus, additional measures like *diversity* and *representativeness* have been proposed to select batch queries [She+04].

There are many strategies to estimate informativeness. Covering them exhaustively is outside the scope of this thesis. Instead, we will focus on some important concepts that existing query strategies build upon.

Version Space Reduction. In the beginning of this section, we have motivated active learning with a simple example of three hypotheses that are consistent with the training data, see Figure 3.3. The space of all consistent hypothesis is called the *version space* $\mathcal{V} \subseteq \mathcal{H}$ [Mit82]. Recall that there are some observations that, if their class label is known, would render some of the hypotheses in $\mathcal{V}$ inconsistent. The basic idea of version space reduction is to select queries that will narrow down (reduce) the version space efficiently.

Estimating how much a query reduces the version space is difficult, for two reasons. First, $\mathcal{H}$ and $\mathcal{V}$ both are uncountable. One can approximate $\mathcal{V}$ by a set of maximally specific hypotheses V_s and maximally general hypotheses V_g. The maximally specific hypotheses are $V_s \subseteq \mathcal{V}$ such that there is no other hypothesis $h' \in \mathcal{V} \setminus V_s$ that is *more* specific than any $h \in V_s$ and also agrees with $\mathcal{L}$. Analogously, the maximally general hypotheses are $V_g \subseteq \mathcal{V}$ such that there is no other hypothesis $h' \in \mathcal{V} \setminus V_g$ that is *less* specific than any $h \in V_g$ and also agrees with $\mathcal{L}$ [HMP04]. Here, "more specific" means that a hypothesis matches a proper subset of all observations [Mit82].

Theoretically, queries that bisect the version space are the highly informative ones, since they eliminate half of the hypotheses. However, identifying such queries is difficult

[2] Some literature assumes that a smaller informativeness score to be better. We adapt the definitions from literature to our notion when necessary.

since the shape of the version space is not known, and it can be non-symmetric. Second, the version space may collapse, i.e., $\mathcal{V} = \emptyset$, when classes are not perfectly separable. Thus, one has to rely on some notion of approximate consistency.

Although theoretically appealing, direct version space reduction turns out to be difficult to realize. Therefore, literature has proposed heuristics that build upon the core idea of version spaces. One common heuristic is to select an ensemble of classifiers $\mathcal{E}$, each representing one hypothesis in $\mathcal{H}$. A *query by committee* strategy then selects the observations where the classifiers in $\mathcal{E}$ disagree most [SOS92]. One example to measure the disagreement is *vote entropy* [Set12]

$$\tau_{VE}(x) = -\sum_{y\in\{1,-1\}} \frac{\sum_{c\in\mathcal{E}} \mathbb{1}_{\{y\cdot f_c(x)>0\}}}{|\mathcal{E}|} \log \frac{\sum_{c\in\mathcal{E}} \mathbb{1}_{\{y\cdot f_c(x)>0\}}}{|\mathcal{E}|} \tag{3.13}$$

where f_c is the decision function of classifier c. Committee-based methods have turned out to work well with many machine learning tasks, like multi-class classification [KW06]. However, a downside is that one hast to maintain a set of $|\mathcal{E}|$ classifiers, i.e., they must be retrained in each iteration.

We end the discussion on version space reduction with a comment on its applicability to SVDD. There are two ways to think about version spaces with SVDD. The first way is to consider the version space spanned by the two hyperparameters C and γ, when using the Gaussian kernel. Each combination of parameter values results in a specific hypothesis, represented as a decision boundary. In this case however, any version space reduction that can be achieved is negligible. The reason is that a good choice of C depends on the selected γ. Put differently, for each given γ, one has to find an optimal C, and this C may only be optimal for the given γ. Since γ can take any positive value, the share of hypotheses that can be reduced by finding an optimal C is negligible.

The second way to think about version spaces is to fix the hyperparameter values to some constant values, and train SVDD on different subsamples of the data set. Each of the subsample classifiers then represents a specific hypothesis. However, SVDD requires to set hyperparameter values for each subsample individually to work well. And the number of possible subsamples grows exponentially with the initial sample size. So multiple subsampling requires to maintain a large set of classifiers, which is computationally expensive. Thus, approaching version spaces through subsamples also is not feasible with SVDD.

Note that there is one approach that uses SVDD ensembles with active learning. However, the idea is to use the SVDD ensemble to annotate observations which then are used to train a binary Support Vector Machine (SVM), and not to improve the SVDD classification [Thi+15].

Uncertainty Sampling. Another, common type of query strategies is to select observations where the classifications are uncertain. For probabilistic classifiers, "uncertain" can have several interpretations. For instance, it can refer to observations where the classifier is least confident about the predicted class. Another way to think about uncertainty is ambiguity in the predicted posterior class probabilities. A common way is to quantify this by information theoretic measures, like Shannon entropy

$$\tau_H(x) = -\sum_{y\in\{1,-1\}} P(y|x)\log P(y|x). \tag{3.14}$$

For classifiers that do not return a posterior probability, one has to rely on proxies to estimate uncertainty. A common proxy is to attribute high informativeness to observations which are close to the decision boundary. Interestingly, one can show that such decision boundary querying is, in some cases, equivalent to version space reduction for margin-based classifiers [Set12], like binary SVM.

Expected Error Reduction. One can also approach query selection from a decision theoretic perspective. Given a trained classifier and some measure of classification error, one has to decide on the observation that reduces the expected future classification error most. The core idea is to simulate the possible feedback options for an unlabeled instance, retrain a classifier for each case, and calculate the difference in an error metric on the unlabeled observations. For instance, with log-loss as the error metric, one can calculate the negative[3] expected log-loss [Set12]

$$\tau_{NLL}(x) = -\sum_{y\in\{1,-1\}} P(y|x) \cdot \left(\sum_{x'\in\mathcal{U}} - \sum_{y'\in\{1,-1\}} P^x(y'|x') \log P^x(y'|x') \right) \tag{3.15}$$

where P^x is the posterior probability after adding x to $\mathcal{L}_{\text{in}}$ or $\mathcal{L}_{\text{in}}$ and retraining the classifier. There are some downsides of expected error reduction. For one, it requires a probabilistic classifier, as well as to choose a suitable error metric. Further, expected error reduction has high runtime complexity with $O(k \cdot |\mathcal{U}|)$ classifier retrainings for each active learning iteration, where k is the number of classes.

Probabilistic Sampling. Probabilistic active learning strives towards an accurate estimate of the true posterior distribution for a probabilistic classifier. In the following, we explain the basic idea of probabilistic active learning based on [KKS14].

The core idea of probabilistic active learning is to model the posterior distribution explicitly, and to select observations where feedback improves the expected gain in classification quality. The respective query strategy takes label statistics into account, i.e., it considers how many instances have already been annotated in a local neighborhood. Intuitively, the more labels are available in a local neighborhood, the more certain the classifier is about the local posterior distribution. Consequently, additional annotations from such a neighborhood provide only little additional information.

One can explicitly model the certainty about the posterior probability p by a beta distribution $\mathbf{X} \sim Beta(\alpha, \beta)$, with the probability density

$$P(p|n,k) = \frac{1}{B(\alpha,\beta)p^k p^{n-k}} \tag{3.16}$$

where n is the total number of samples obtained in the neighborhood and k the number of observations with positive class label. Next, a probabilistic classifier predicts the positive

[3] This is the negative loss since the query strategy takes the maximum informativeness, i.e., the observation that results overall in the smallest expected loss.

class if the estimated posterior probability $\hat{p} > 0.5$. So if the true posterior probability is p, the classification accuracy is

$$acc(p|\hat{p}) = \begin{cases} p & \text{if } \hat{p} > 0.5 \\ 1 - p & \text{otherwise.} \end{cases} \tag{3.17}$$

The expected accuracy over all possible posterior class distributions is

$$\mathbb{E}_p[acc] = \int_0^1 P(p|n,k) \cdot acc(p|\frac{k}{n})dp. \tag{3.18}$$

Similarly to expected error reduction, one can simulate feedback to calculate the expected future accuracy. The difference between current accuracy and expected future accuracy is the expected gain in accuracy *accgain*. An observation has high informativeness if its accgain is high.

$$\tau_{gain}(x) = \mathbb{E}_p[\mathbb{E}_y[accgain(x)]]. \tag{3.19}$$

There are some requirements for probabilistic sampling. For one, probabilistic sampling only is applicable to classifiers that return a posterior class probability. Further, one has to define local neighborhoods for P. For instance, one can estimate P based on a Parzen window, e.g., with a Gaussian kernel [KKS14].

Adaptations. Many of the query strategies consider each of the observations in isolation, i.e., independent of the data distribution. Therefore, literature suggests to weight informativeness with an additional factor that quantifies the density P of the observation. For instance, on can use the average similarity of an observation to all other observations [Set12], or a kernel density estimate [KKS14] as a weighting factor. With this, the final, density-weighted score for a query strategy τ_s is

$$\tau_s^{dw}(x) = P(x) \cdot \tau_s(x) \tag{3.20}$$

3.4. Benchmark Data

A fair assessment of existing and of novel active learning approaches requires standardized benchmark data. For many machine-learning problems, benchmark data is well established and accepted by the community. For outlier detection, however, there still is active research on how to construct a good benchmark suite. One reason for this is that real data sets with actual outliers are rare. Further, what actually makes up an outlier is not clear, and one data set may actually have different interpretations, cf. Section 3.2. There are, however, two emerging approaches on how to construct a benchmark.

The first approach is to start from existing binary or multi-class classification data sets [Cam+16; Emm+13; GU16; Dom+18]. Here, one class is selected as the majority class, and observations from the other class(es) are downsampled to build the outlier class. This approach has systematic shortcomings.

Table 3.1.: Overview on benchmark data sets [Cam+16].

Dataset	Observations (N)	Number of Outliers	Attributes (M)
ALOI	50,000	1,508	27
Annthyroid	7,200	534	21
Arrhythmia	450	206	259
Cardiotocography	2,126	471	21
Glass	214	9	7
HeartDisease	270	120	13
Hepatitis	80	13	19
Ionosphere	351	126	32
KDDCup99	60,632	246	38
Lymphography	148	6	3
Parkinson	195	147	22
PageBlocks	5,473	560	10
PenDigits	9,868	20	16
Pima	768	268	8
Shuttle	1,013	13	9
SpamBase	4,601	1,813	57
Stamps	340	31	5
Waveform	3,443	100	21
WBC	454	10	9
WDBC	367	10	30
WPBC	198	47	33

(i) *Ambiguous labels:* The labeling might not be consistent on the full data set. So two observations may be very similar in the data space, but have diverging labels. A reason for this could be if classes are non-separable, or if the ground truth annotations come from multiple domain experts.

(ii) *Annotations base on state-of-the-art:* Annotations may not be collected on the data set alone, but with help of existing unsupervised algorithms. So the outliers in a data set may have been annotated after an unsupervised outlier detection algorithm has suggested them as interesting candidates. However, this selection is biased. For instance, the specific outlier detection method may have missed some of the outliers. A typical example for this would be subspace outlier detection. There, existing subspace outlier detection methods are relatively recent compared to many of the available benchmark data sets. So it is likely that some of the subspace outliers have not been labeled as such, since the subspace methods have not been available at the time of benchmark construction.

(iii) *Outlier Distribution:* The downsampled observations are sampled from a common, underlying distribution, and thus may have different properties than in real outlier data sets. For instance, the outliers may be clustered with a high pairwise similarity, and the normal class may contain outliers [ZSK12].

The second approach is to generate synthetic benchmark data sets. Generating artificial benchmark data generally is an interesting approach for unsupervised outlier detection [Zim19]. It allows to control the characteristics of outliers directly, and gives way to test detection methods for specific, desired properties. However, existing work on generating artificial outliers mostly uses simple generative models, e.g., for scalability testing [Dom^{+}18].

In our work, we rely on a large body of benchmark data that has been published with an extensive comparative study on outlier detection [Cam^{+}16]. The benchmark data mitigates some of the shortcomings of using binary and multi-class data by applying different pre-processing steps to ensure some diversity in the outlier class. Table 3.1 is an overview on the original data sets before pre-processing. The benchmark suite contains resampled versions of the data sets, that have been normalized, deduplicated, and downsampled to different outlier percentages, see [Cam^{+}16] for details.

Part II.

User-Centric One-Class Active Learning

4. An Overview and a Benchmark

In the first part of this thesis, we have introduced active learning, outlier detection and one-class classification as distinct areas of research. However, they are the corner stones of interactive outlier detection, and have been studied jointly in literature over the last years. In this chapter, we introduce this joint research area, and compare state-of-the-art on *active learning for outlier detection with one-class classifiers* in a comprehensive benchmark.

Since this is a joint research area, it comes as no surprise that there is some obscurity on how concepts and methods from the three different subfields play together. In fact, the huge variety of approaches proposed independently in the subfields is a challenge by itself. It is difficult to translate between different notations, objectives and often implicit assumptions to identify methods that are applicable to the specific problem at hand. For instance, literature has proposed several query strategies specific to one-class classification [BBJ15; Gha⁺11b; Gha⁺11a; JD03b; Gör⁺13]. However, many of these query strategies make implicit assumptions like a common distribution for the outlier class, and thus are not applicable to outlier detection.[1] In addition, evaluation of active learning may lack reliability and comparability [Kot⁺17], in particular with one-class classification. Evaluations often are use-case specific, and there is no standard way to report results. This makes it difficult to identify a learning method suitable for a certain use case, and to assess novel contributions in this field. These observations give way to the following questions, which we study in this chapter:

Categorization	What may be a good categorization of learning objectives and assumptions behind one-class active learning?
Evaluation	How to evaluate one-class active learning, in a standardized way?
Comparison	Which active learning methods perform well with outlier detection?

Answering these questions is difficult for two reasons. First, we are not aware of any existing categorization of learning objectives and assumptions. To illustrate, a typical learning objective is to improve the accuracy of the classifier. Another, different learning objective is to present a high share of observations from the minority class to the user for feedback [Das⁺16]. In general, active learning methods may perform differently with

[1] The remainder of this chapter bases on the article [TEB18] *Holger Trittenbach, Adrian Englhardt, and Klemens Böhm. "An Overview and a Benchmark of Active Learning for Outlier Detection with One-Class Classifiers". In:* arXiv *(2018). arXiv:* `1808.04759`. It has been shortened to be less repetitive. It contains minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

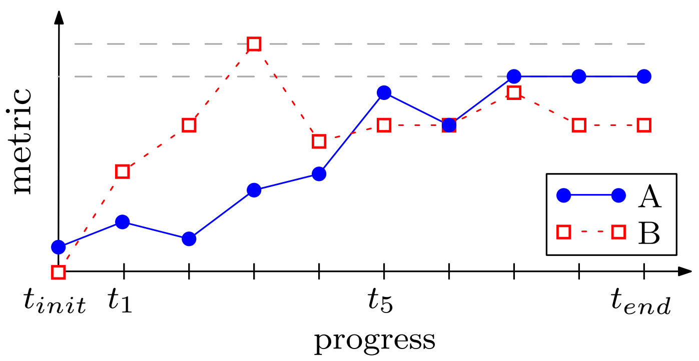


Figure 4.1.: Illustration of an active learning progress curve for active learning methods A and B.

different learning objectives. Next, assumptions limit the applicability of active learning methods. For instance, a common assumption is that some labeled observations are already available before active learning starts. Naturally, methods that rely on this assumption are only applicable if such labels indeed exist. So knowing the range of objectives and assumptions is crucial to assess one-class active learning. Related work however tends to omit respective specifications. We deem this one reason why no overview article or categorization is available so far that could serve as a reference point.

Second, there is no standard to report active learning results. The reason is that "quality" can have several meanings with active learning, as we now explain.

> **Example 2** *Figure 4.1 is a progress curve. Such curves are often used to compare active learning methods. The y-axis is the values of a metric for classification quality, such as the true-positive rate. The x-axis is the progress of active learning, such as the percentage of observations for which the user has provided a label. Figure 4.1 plots two active learning methods A and B from an initial state t_{init} to the final iteration t_{end}. Both methods apparently have different strengths. A yields better quality at t_{init}, while B improves faster in the first few iterations. However, quality increases non-monotonically, because feedback can bias the classifier temporarily. At t_{end}, the quality of B is lower than the one of A.*

The question that follows is which active learning method one should prefer. One might choose the one with higher quality at t_{end}. However, the choice of t_{end} is arbitrary, and one can think of alternative criteria such as the stability of the learning rate. These missing evaluation standards are in the way of establishing comprehensive benchmarks that go beyond comparing individual progress curves.

This chapter contains two parts: an overview on one-class active learning for outlier detection, and a comprehensive benchmark of state-of-the-art methods. We make the following specific contributions.

(i) In the first part of this chapter (Section 4.1), we propose a categorization of one-class active learning methods by introducing learning scenarios. A learning scenario is a combination of a learning objective and an initial setup. One important insight from this categorization is that the learning scenario and the learning objective are decisive for the applicability of active learning methods. In particular, some active learning methods and learning scenarios are incompatible. This suggests that a rigorous specification of the learning scenario is important to assess novel contributions in this field. We then (ii) introduce several complementary ways to summarize progress curves, to facilitate a standard evaluation of active learning in benchmarks (Section 4.2). The evaluation by progress-curve summaries has turned out to be very useful, since they ease the comparison of active-learning methods significantly. As such, the categorization and evaluation standards proposed give way to a more reliable and comparable evaluation.

In the second part of this chapter (Section 4.3), we (iii) put together a comprehensive benchmark with around 84,000 combinations of learning scenarios, classifiers, and query strategies for the selection of one-class active learning methods. To facilitate reproducibility, we make our implementations, raw results and notebooks publicly available.[2] A key observation from our benchmark is that none of the state-of-the-art methods stands out in a competitive evaluation. We have found that the performance largely depends on the parametrization of the classifier, the data set, and on how progress curves are summarized. In particular, a good parametrization of the classifier is as important as choosing a good query selection strategy. We conclude by (iv) proposing guidelines on how to select active learning methods for outlier detection with one-class classifiers.

4.1. Overview on One-Class Active Learning

There are different ways to design one-class active learning systems, and several variants have recently been proposed. Yet we have found that variants follow different objectives and make implicit assumptions. Existing surveys on active learning do not discuss these objectives and assumptions, and they rather focus on general classification tasks [Ram+17; Set12; BKL15; Ols09] and on benchmarks for balanced [Ber+18a] and multi-class classification [JD03b].

In this section, we discuss assumptions for one-class active leaning, structure the aspects where systems differ from each other, and discuss implications of design choices on the active learning system. We structure our discussion into three parts corresponding to the building blocks of a one-class active learning system. Figure 4.2 graphs the building blocks. The first block is the *Learning Scenario*, which establishes assumptions regarding the training data and the process of gathering user feedback. It specifies the initial configuration of the system before the actual active learning starts. The second building block is the *Base Learner*, i.e., a one-class classifier that learns a binary decision function based on the data and user feedback available. The third building block is the *Query Strategy*. In what follows, we explain the blocks and discuss dependencies between them.

4.1.1. Building Block: Learning Scenario

Researchers make assumptions regarding the interaction between system and user as well as assumptions regarding the application scenario. Literature on one-class active learning

2 `https://www.ipd.kit.edu/ocal`

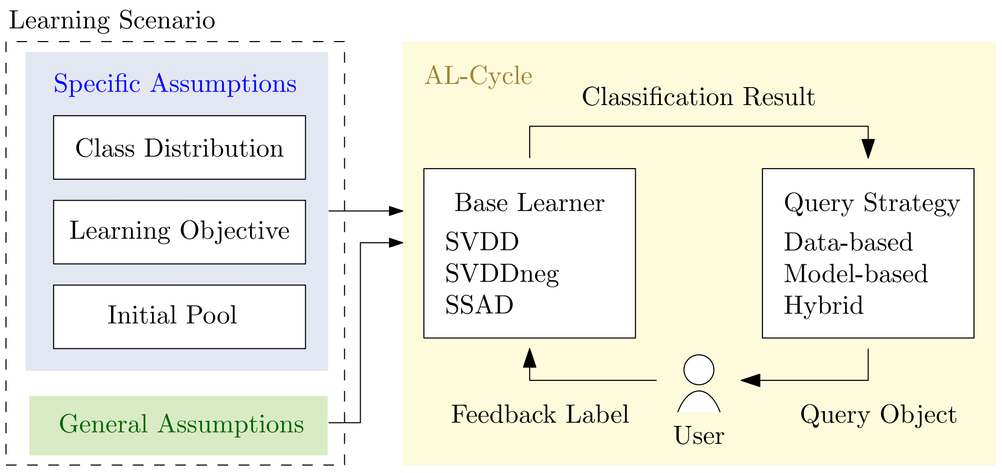


Figure 4.2.: Building blocks of one-class active learning.

often omits an explicit description of these assumptions, and one must instead derive them for instance from the experimental evaluation. Moreover, assumptions often do not come with an explicit motivation, and the alternatives are unclear. We now review the various assumptions found in the literature. We distinguish between two types, general and specific assumptions.

General Assumptions

General assumptions specify modalities of the feedback and impose limits on how applicable active learning is in real settings. These assumptions have been discussed for standard binary classification [Set12], and many of them are accepted in the literature. We highlight the ones important for one-class active learning.

Feedback Type: Existing one-class active learning methods assume that feedback is a class label, i.e., the decision whether an observation belongs to a class or not. However, other types of feedback are conceivable as well, such as feature importance [RMJ06; DSM09]. But to our knowledge, research on one-class active learning has been limited to label feedback. Next, the most common mechanism in literature is sequential feedback, i.e., for one observation at a time. As explained earlier, asking for feedback in batches might have certain advantages, such as increased efficiency of the labeling process, see Section 3.3. But a shift from sequential to batch queries is not trivial and requires additional criteria, such as diversity [Jus06].

Feedback Budget: A primal motivation for active learning is that the amount of feedback a user can provide is bounded. For instance, the user can have a time or cost budget or a limited attention span to interact with the system. Assigning costs to feedback acquisition is difficult, and a budget restriction is likely to be application-specific. In some cases, feedback on observations from the minority class may be costlier. However, a common simplification here is to assume that labeling costs are uniform, and that there is a limit on the number of feedback iterations.

Interpretability: A user is expected to have sufficient domain knowledge to provide feedback purposefully. However, this implies that the user can interpret the classification result in the first place, i.e., the user understands the output of the one-class classifier. This is a strong assumption, and it is difficult to evaluate. For one thing, "interpretation" already has various meanings for non-interactive supervised learning [Lip16], and it has only recently been studied for interactive learning [PCF18; TK18]. Concepts to support users with an explanation of outliers [Mic$^+$13; KMM18] have not been studied in the context of active learning either. In any case, a thorough evaluation would require a user study, see also Chapter 7. As explained in Section 3.3, existing one-class active learning systems bypass the difficulty of interpretation by using an oracle that simulates feedback based on a ground truth.

Specific Assumptions

Specific assumptions confine the learning objective and the data for a particular active learning application. One must define specific assumptions carefully, because they restrict which base learners and query strategies are applicable. We partition specific assumptions into the following categories.

Class Distribution: One-class learning is designed for highly imbalanced domains. There are two different definitions of "minority class". The first one is that the minority class is unusual observations, also called outliers, that are exceptional in a bulk of data. The second definition is that the minority class is the target in a one-vs-all multi-class classification task, i.e., where all classes except for the minority class have been grouped together [JD03a; Gha$^+$11a]. With this definition, the minority class is not exceptional, and it has a well-defined distribution. Put differently, one-class classification is an alternative to imbalanced binary classification in this case. So both definitions of "minority class" relate to different problem domains. The first one is in line with the intent of this chapter, and we stick to it in the following.

Under the first definition, one can differentiate between characterizations of outliers. Recall that the prevalent characterization is that outliers do not follow a common underlying distribution. This assumption has far-reaching implications. For instance, if there is no joint distribution, it is not meaningful to estimate a probability density from a sample of the minority class.

Another characterization of outliers is to assume that it is a mixture of several distributions of rare classes. In this case, a probability density for each mixture component exists. So the probability density for the mixture as a whole exists as well. Its estimation however is hard, because the sample for each component is tiny. The characterization of the outlier distribution has implications on the separation of the data into train and test partitions, as we will explain in Section 4.2.4.

Learning Objective: The learning objective is the benefit expected from an active learning system. A common objective is to improve the accuracy of a classifier. But there are alternatives. For instance, users of one-class classification often have a specific interest in the minority class [Das+16]. In this case, it is reasonable to assume that users prefer giving feedback on minority observations if they will examine them anyhow later on. So a good active learning method yields a high proportion of queries from the minority class. This may contradict the objective of accuracy improvement.

There also are cases where the overall number of available observations is small, even for the majority class. The learning objective in this case can be a more robust estimate of the majority-class distribution [Gha+11a; Gha+11b]. A classifier benefits from extending the number of majority-class labels. This learning objective favors active learning methods that select observations from the majority class.

Initial Pool: The initial setup is the label information available at the beginning of the active learning process. There are two cases: (i) Active learning starts from scratch, i.e., there are no labeled examples, and the initial learning step is unsupervised. (ii) There are some labeled instances available [Jus06]. The number of observations and the share of class labels in the initial sample depends on the sampling mechanism. A special case is if the labeled observations exclusively are from the majority class [Gha+11a]. In our work, we consider different initial pool strategies:

(Pu) Pool unlabeled: All observations are unlabeled.

(Pp) Pool percentage: Stratified proportion of labels for p percent of the observations.

(Pn) Pool number: Stratified proportion of labels for a fixed number of observations n.

(Pa) Pool attributes: As many labeled inliers as number of attributes.

The rationale behind Pa is that the correlation matrix of labeled observations is singular if there are fewer labeled observations than attributes. With a singular correlation matrix, some query strategies are infeasible.

How general and specific assumptions manifest depends on the use case, and different combinations of assumptions are conceivable. We discuss how we set assumptions for our benchmark in Section 4.3.

4.1.2. Building Block: Base Learner

One-class classifiers fall into two categories: support-vector methods and non-support-vector classifiers [KM14]. In this chapter article, we focus on support-vector methods. They are the prevalent choice in the literature on one-class active learning. A reason may be that there are semi-supervised versions of one-class support-vector classifiers that are well-suited for active learning with outlier detection, see our discussions in the following sections. We are not aware of any applications of active learning for outlier detection with other types of one-class classifiers, like Parzen window classifier or one-class decision trees. However, most existing query strategies do not require a specific base learner, as long as it returns a decision function. So using existing strategies with non-support-vector base learners is conceptually feasible. The focus of this chapter however is to review existing methods, not to explore novel approaches. Thus, we restrict our discussion to base learners that have been used in previous work on active learning for outlier detection with one-class classifiers. In particular, we use unsupervised SVDD (cf. Section 3.2), semi-supervised SVDDneg [TD04] with labels from the minority class, and the semi-supervised SSAD [Gör+13] with labels from both classes.

SVDD with Negative Examples (SVDDneg)

SVDDneg [TD04] extends the vanilla SVDD by using different costs C_1 for $\mathcal{L}_{\text{in}}$ and $\mathcal{U}$ and costs C_2 for $\mathcal{L}_{\text{out}}$. An additional constraint places observations in $\mathcal{L}_{\text{out}}$ outside the decision boundary.

$$\begin{aligned} \text{SVDDneg:} \quad \underset{a,R,\xi}{\text{minimize}} \quad & R^2 + C_1 \cdot \sum_{i:x_i \in \mathcal{U} \cup \mathcal{L}_{\text{in}}} \xi_i + C_2 \cdot \sum_{i:x_i \in \mathcal{L}_{\text{out}}} \xi_i \\ \text{subject to} \quad & \|\phi(x_i) - a\|^2 \le R^2 + \xi_i, \;\; i{:}\, x_i \in \mathcal{U} \cup \mathcal{L}_{\text{in}} \\ & \|\phi(x_i) - a\|^2 \ge R^2 - \xi_i, \;\; i{:}\, x_i \in \mathcal{L}_{\text{out}} \\ & \xi_i \ge 0, \;\; i = 1, \ldots, N. \end{aligned} \tag{4.1}$$

Semi-Supervised Anomaly Detection (SSAD)

SSAD [Gör+13] additionally differentiates between labeled inliers and unlabeled observations in the objective and in the constraints. In its original version, SSAD assigns different costs to $\mathcal{U}$, $\mathcal{L}_{\text{in}}$, and $\mathcal{L}_{\text{out}}$. We use a simplified version where the cost for both $\mathcal{L}_{\text{in}}$ and $\mathcal{L}_{\text{out}}$ are C_2. SSAD further introduces an additional trade-off parameter, which we call κ. High values of κ increase the weight of $\mathcal{L}$ on the solution, i.e., SSAD is more likely to overfit to instances in $\mathcal{L}$.

$$\begin{aligned} \text{SSAD:} \quad \underset{a,R,\xi,\tau}{\text{minimize}} \quad & R^2 - \kappa\tau + C_1 \cdot \sum_{i:x_i \in \mathcal{U}} \xi_i + C_2 \cdot \sum_{i:x_i \in \mathcal{L}} \xi_i \\ \text{subject to} \quad & \|\phi(x_i) - a\|^2 \le R^2 + \xi_i, \;\; i{:}\, x_i \in \mathcal{U} \\ & \|\phi(x_i) - a\|^2 \ge R^2 - \xi_i + \tau, \;\; i{:}\, x_i \in \mathcal{L}_{\text{out}} \\ & \|\phi(x_i) - a\|^2 \le R^2 + \xi_i - \tau, \;\; i{:}\, x_i \in \mathcal{L}_{\text{in}} \\ & \xi_i \ge 0, \;\; i = 1, \ldots, N. \end{aligned} \tag{4.2}$$

Under mild assumptions, SSAD can be reformulated as a convex problem [Gör+13].

4.1.3. Building Block: Query Strategy

We now review existing query strategies (cf. Definition 5) from literature that have been proposed for one-class active learning. To this end, we partition them into three categories.

The first category is *data-based query strategies.*[3] These strategies approach query selection from a statistical side. The second category is *model-based query strategies.* These strategies rely on the decision function returned by the base learner. The third category is *hybrid query strategies.* These strategies use both the data statistics and the decision function.

Data-based Query Strategies

The concept behind data-based query strategies is to compare the posterior probabilities of an observation $p(\text{out}|x)$ and $p(\text{in}|x)$. This is well known from binary classification and is referred to as *measure of uncertainty* [Set12]. If a classifier does not explicitly return posterior probabilities, one can use the Bayes rule to infer them. But this is difficult, for two reasons. First, applying the Bayes rule requires knowing the prior probabilities for each class, i.e., the proportion of outliers in the data. It may not be known in advance. Second, outliers do not follow a homogeneous distribution. This renders estimating $p(x|\text{out})$ infeasible. There are two types of data-based strategies that have been proposed to address these difficulties.

The first type deems observations informative if the classifier is uncertain about their class label, i.e., observations with equal probability of being classified as inlier and outlier. The following two strategies quantify informativeness in this way.

Minimum Margin [Gha+11b]: This query strategy relies on the difference between posterior class probabilities

$$\begin{aligned}\tau_{\text{MM}}(x) &= -|p(\text{in}|x) - p(\text{out}|x)| &\text{(4.3a)}\\ &= -\left|\frac{p(x|\text{in}) \cdot p(\text{in}) - p(x|\text{out}) \cdot p(\text{out})}{p(x)}\right| &\text{(4.3b)}\\ &= -\left|\frac{2 \cdot p(x|\text{in}) \cdot p(\text{in}) - p(x)}{p(x)}\right| &\text{(4.3c)}\end{aligned}$$

where Equation 4.3b and Equation 4.3c follow from the Bayes rule. If $p(\text{in})$ and $p(\text{out})$ are known priors, one can make direct use of Equation 4.3c. Otherwise, the inventors of Minimum Margin suggest to take the expected value under the assumption that $p(\text{out})$, i.e., the share of outliers, is uniformly distributed

$$\tau_{\text{EMM}}(x) = \mathbb{E}_{p(\text{out})}\left(\tau_{\text{MM}}(x)\right) = \left(\frac{p(x|\text{in})}{p(x)} - 1\right) \cdot sign\left(0.5 - \frac{p(x|\text{in})}{p(x)}\right). \tag{4.4}$$

We find this an unrealistic assumption, because a share of outliers of 0.1 would be as likely as 0.9. In our experiments, we evaluate both τ_{MM} with the true outlier share as a prior and with τ_{EMM}.

[3] Others have used the term “model-free” instead [ODM17]. However, we deliberately deviate from this nomenclature since the strategies we discuss still rely on some kind of underlying model, e.g., a kernel-density estimator.

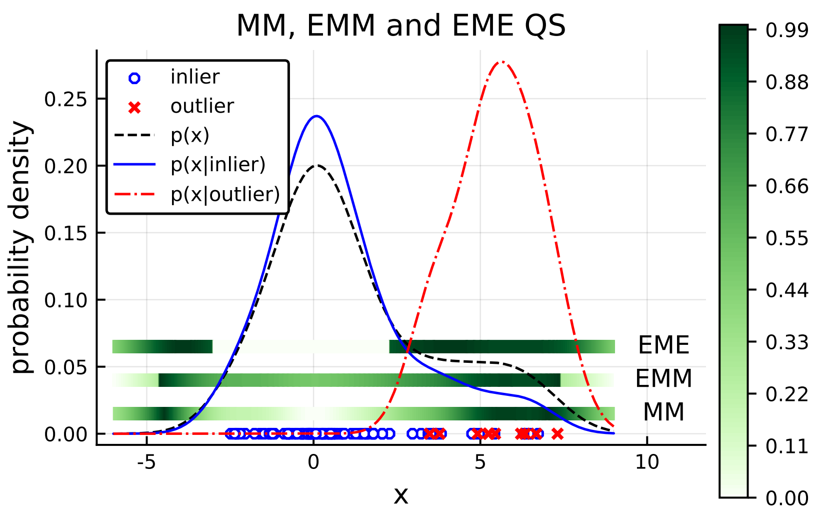


Figure 4.3.: Visualization of informativeness calculated by τ_{MM} (Equation 4.3c), τ_{EMM} (Equation 4.4) and τ_{EME} (Equation 4.6) query strategies (QS). Dark colored regions indicate regions of high informativeness.

Maximum-Entropy [Gha+11b]: This query strategy selects observations where the distribution of the class probability has a high entropy

$$\tau_{\text{ME}}(x) = -[p(\text{in}|x) \cdot \log(p(\text{in}|x)) + p(\text{out}|x) \cdot \log(p(\text{out}|x))]. \tag{4.5}$$

Applying the Bayes rule and taking the expected value as in Equation 4.4 gives

$$\begin{aligned}\tau_{\text{EME}}(x) &= \mathbb{E}_{p(\text{out})}\left(\tau_{\text{ME}}(x)\right) \\ &= \frac{-\left(\frac{p(x|\text{in})}{p(x)}\right)^2 \cdot \log\left(\frac{p(x|\text{in})}{p(x)}\right) + \frac{p(x|\text{in})}{p(x)}}{2 \cdot \frac{p(x|\text{in})}{p(x)}} \\ &+ \frac{\left(\frac{p(x|\text{in})}{p(x)} - 1\right)^2 \cdot \log\left(1 - \frac{p(x|\text{in})}{p(x)}\right)}{2 \cdot \frac{p(x|\text{in})}{p(x)}}.\end{aligned} \tag{4.6}$$

To give an intuition of the Minimum-Margin and the Maximum-Entropy strategy, we visualize the informativeness for Minimum Margin and Maximum Entropy on sample data. Figure 4.3 visualizes τ_{MM}, τ_{EMM} and τ_{ME} for univariate data generated from two Gaussian distributions, with $p(\text{out}) = 0.1$. The authors of τ_{EME} suggest to estimate the densities with kernel density estimation (KDE) [Gha+11b]. However, entropy is defined on probabilities and is not applicable to densities, so just inserting into the formula yields ill-defined results. Moreover, τ_{EME} is not defined for $\frac{p(x|\text{in})}{p(x)} \geq 1$. We set $\tau_{\text{EME}} = 0$ in this case. For τ_{MM}, we use Equation 4.3c with prior class probabilities. Not surprisingly, all three depicted formulas result in a similar pattern, as they follow the same general motivation. The tails of the inlier distribution yield high informativeness. The informativeness decreases slower on the right tail of the inlier distribution where the outlier distribution has some support.

The second type of data-based query strategies strives for a robust estimation of the inlier density. The idea is to give high informativeness to observations that are likely to reduce the loss between the estimated and the true inlier density. There is one strategy of this type.

Minimum-Loss [Gha+11a]: Under the minimum-loss strategy, observations have high informativeness if they are expected to increase the estimate of the inlier density. The idea to calculate this expected value is as follows. The feedback for an observation is either "outlier" or "inlier". The minimum-loss strategy calculates an updated density for both cases and then takes the expected value by weighting each case with the prior class probabilities. Similarly to Equation 4.3c, this requires knowledge of the prior class probabilities.

We now describe Minimum-Loss formally. Let $\hat{g}^{\text{in}}$ be an estimated probability density over all inlier observations $\mathcal{L}_{\text{in}}$. Let $\mathcal{L}_{\text{in}}^{x} = \mathcal{L}_{\text{in}} \cup \{x\}$, and let $\hat{g}^{\text{in},x}$ be its corresponding density. Similarly, we define $\mathcal{L}_{\text{out}}^{x} = \mathcal{L}_{\text{out}} \cup \{x\}$. Then $\hat{g}_{-i}^{\text{in}}$ stands for the density estimated over all $\mathcal{L}_{\text{in}} \setminus x_i$ and $\hat{g}_{-i}^{\text{in},x}$ for $\mathcal{L}_{\text{in}}^{x} \setminus x_i$ respectively. In other words, for $\hat{g}_{-i}^{\text{in},x}(x_i)$, one first estimates the density $\hat{g}_{-i}^{\text{in},x}$ without x_i and then evaluates the estimated density at x_i. One can now calculate how well an observation x matches the inlier distribution by using leave-out-one cross validation for both cases.

Case 1: x is inlier

$$\tau_{\text{ML-in}}(x) = \frac{1}{|\mathcal{L}_{\text{in}}^{x}|} \sum_{i:x_i \in \mathcal{L}_{\text{in}}^{x}} \hat{g}_{-i}^{\text{in},x}(x_i) - \frac{1}{|\mathcal{L}_{\text{out}}|} \sum_{i:x_i \in \mathcal{L}_{\text{out}}} \hat{g}^{\text{in},x}(x_i). \tag{4.7}$$

Case 2: x is outlier

$$\tau_{\text{ML-out}}(x) = \frac{1}{|\mathcal{L}_{\text{in}}|} \sum_{i:x_i \in \mathcal{L}_{\text{in}}} \hat{g}_{-i}^{\text{in}}(x_i) - \frac{1}{|\mathcal{L}_{\text{out}}^{x}|} \sum_{i:x_i \in \mathcal{L}_{\text{out}}^{x}} \hat{g}^{\text{in}}(x_i). \tag{4.8}$$

The expected value over both cases is

$$\tau_{\text{ML}}(x) = p(\text{in}) \cdot \tau_{\text{ML-in}}(x) + (1 - p(\text{in})) \cdot \tau_{\text{ML-out}}(x). \tag{4.9}$$

We illustrate Equation 4.7, Equation 4.8 and τ_{ML} in Figure 4.4. As expected, τ_{ML} yields high informativeness in regions of high inlier density. τ_{ML} gives an almost inverse pattern compared to the Minimum-Margin and the Maximum-Entropy strategies. This illustrates that existing query strategies are markedly different. It is unclear how to decide between them solely based on theoretical considerations, and one has to study them empirically instead.

Model-based Query Strategies

Model-based strategies rely on the decision function f of a base learner. Recall that an observation x is an outlier if $f(x) > 0$ and an inlier for $f(x) \leq 0$. Observations with $f(x) = 0$ are on the decision boundary.

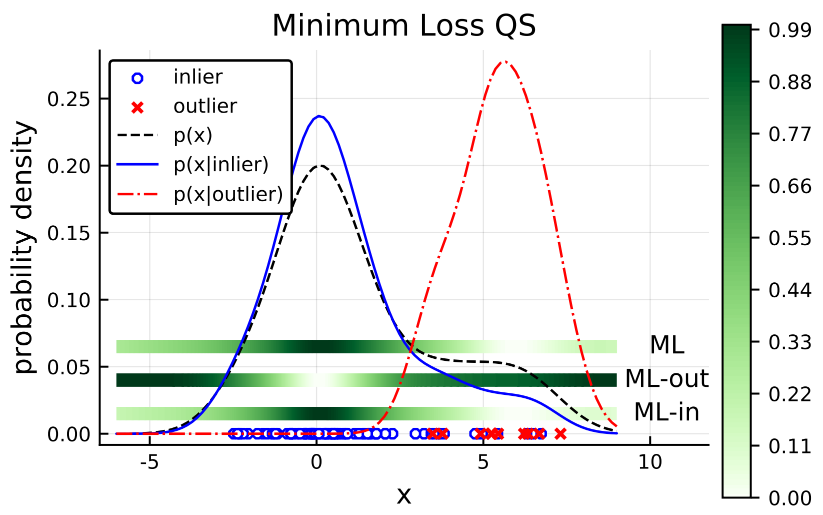


Figure 4.4.: Visualization of informativeness calculated by $\tau_{\text{ML-in}}$ (Equation 4.7), $\tau_{\text{ML-out}}$ Equation 4.8, and τ_{ML} Equation 4.9 query strategies (QS). Dark colored regions indicate regions of high informativeness.

High-Confidence [BBJ15]: This query strategy selects observations that match the inlier class the least. For SVDD this is

$$\tau_{\text{HC}}(x) = f(x). \tag{4.10}$$

Decision-Boundary: This query strategy selects observations closest to the decision boundary

$$\tau_{\text{DB}}(x) = -|f(x)|. \tag{4.11}$$

Hybrid Query Strategies

Hybrid query strategies combine data-based and model-based strategies.

Neighborhood-Based [Gör+13]: This query strategy explores unknown neighborhoods in the feature space. The first part of the query strategy calculates the average number of labeled instances among the k-nearest neighbors

$$\widehat{\tau}_{\text{NB}}(x) = -\left(0.5 + \frac{1}{2k} \cdot |\{x' \in \text{NN}_k(x) \colon x' \in \mathcal{L}_{\text{in}}\}|\right), \tag{4.12}$$

with k-nearest neighbors $NN_k(\cdot)$. A high number of neighbors in $\mathcal{L}_{\text{in}}$ makes an observation less interesting. The strategy then combines this number with the distance to the decision boundary, i.e., $\tau_{\text{NB}} = \eta \cdot \tau_{\text{DB}} + (1 - \eta) \cdot \widehat{\tau}_{\text{NB}}$. Parameter $\eta \in [0, 1]$ controls the influence of the number of already labeled instances in the neighborhood on the decision. The paper does not recommend any specific parameter value, and we use $\eta = 0.5$ in our experiments.

Boundary-Neighbor-Combination [YWF18]: The core of this query strategy is a linear combination of the normalized distance to the hypersphere and the normalized distance to the first-nearest neighbor

$$\begin{aligned}\widehat{\tau}_{\text{BNC}}(x) = (1-\eta)\cdot &\left(-\frac{|f(x)| - \min_{x'\in U}|f(x')|}{\max_{x'\in\mathcal{U}}|f(x')|}\right)\\ + \eta\cdot &\left(-\frac{d(x,\text{NN}_1(x)) - \min_{x'\in\mathcal{U}}(d(x',\text{NN}_1(x')))}{\max_{x'\in\mathcal{U}} d(x',\text{NN}_1(x'))}\right).\end{aligned} \tag{4.13}$$

with a distance function d, and trade-off parameter η. The actual query strategy τ_{BNC} is to choose a random observation with probability p and to use strategy $\widehat{\tau}_{\text{BNC}}$ with probability $(1-p)$. The paper recommends to set $\eta = 0.7$ and $p = 0.15$.

Baselines

In addition to the strategies introduced so far, we use the following baselines.

Random: This query strategy draws each unlabeled observation with equal probability

$$\tau_{\text{rand}}(x) = \frac{1}{|\mathcal{U}|}\ . \tag{4.14}$$

Random-Outlier: This query strategy is similar to Random, but with informativeness 0 for observations predicted to be inliers

$$\tau_{\text{rand-out}}(x) = \begin{cases} \frac{1}{|\mathcal{U}|} & \text{if } f(x) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

In general, adapting other strategies from standard binary active learning is conceivable as well. For instance, one could learn a committee of several base learners and use disagreement-based query selection, see Section 3.3. In this thesis however, we focus on strategies that have been explicitly adapted to and used with one-class active learning.

4.2. Evaluation of One-Class Active Learning

Evaluation of active learning methods is more involved than the one of static methods. Namely, the result of an active learning method is not a single number, but rather a sequence of numbers that result from a quality evaluation in each iteration.

We now address Question *Evaluation* in several steps. We first discuss characteristics of active learning progress curves. We then review common quality metrics (QM) for one-class classification, i.e., metrics that take the class imbalance into account. We then discuss different ways to summarize active learning curves. Finally, we discuss the peculiarities of common train/test-split strategies for evaluating one-class active learning and limitations of the design choices just mentioned.

4.2.1. Progress Curves

The sequence of quality evaluations can be visualized as a *progress curve*, see Figure 4.1. We call the interval from t_{init} to t_{end} an *active learning cycle*. Literature tends to use the percentage or the absolute number of labeled observations to quantify progress on the x-axis. However, this percentage may be misleading if the total number of observations varies between data sets. Next, other measures are conceivable as well, such as the time the user spends to answer a query. While this might be even more realistic, it is very difficult to validate. We deem the absolute number of labeled objects during the active learning cycle the most appropriate scaling. It is easy to interpret, and the budget restriction is straightforward. However, the evaluation methods proposed in this section are independent of a specific progress measure.

The y-axis is a metric for classification quality. There are two ways to evaluate it for imbalanced class distributions: by computing a summary statistic on the binary confusion matrix, or by assessing the ranking induced by the decision function.

4.2.2. One-Class Evaluation Metrics

We use the Matthews Correlation Coefficient (MCC) and Cohen's kappa to evaluate the binary output. They can be computed from the confusion matrix. MCC returns values in $[-1, +1]$, where high values indicate good classification on both classes, 0 equals a random prediction, and -1 is the total disagreement between classifier and ground truth. kappa returns 1 for a perfect agreement with the ground truth and 0 for one not better than a random allocation.

One can also use the distance to the decision boundary to rank observations. The advantage is the finer differentiation between strong and less strong outliers, see Section 3.2. A common metric is the area under the ROC curve (AUC) which has been used in other outlier-detection benchmarks [Cam$^+$16]. An interpretation of the AUC is the probability that an outlier is ranked higher than an inlier. So an AUC of 1 indicates a perfect ranking; 0.5 means that the ranking is no better than random.

If the data set is large, users tend to only inspect the top of the ranked list of observations. Then it can be useful to use the partial AUC (pAUC). It evaluates classifier quality at thresholds on the ranking where the false-positive rate (FPR) is low. An example for using pAUC to evaluate one-class active learning is [Gör$^+$13].

4.2.3. Summary of the Active-Learning Curve

The visual comparison of active learning via progress plots does not scale with the number of experiments. For instance, our benchmark would require to compare 84,000 different learning curves; this is prohibitive. For large-scale comparisons, one should instead summarize a progress curve. Recently, *true performance of the selection strategy (TP)* has been proposed as a summary of increase and decrease of classifier performance over the number of iterations [RAV18]. However, TP is a single aggregate measure, which is likely to overgeneralize and is difficult to interpret. For a more comprehensive evaluation, we therefore propose to use several summary statistics. Each of them captures some characteristic of the learning progress and has a distinct interpretation.

We use $QM(k)$ for the quality metric QM at the active learning progress k. We use $\mathcal{L}^{\text{init}}$ and $\mathcal{L}^{\text{end}}$ to refer to the labeled examples at t_{init} and t_{end}.

Start Quality (SQ): The Start Quality is the baseline classification quality before the active learning starts, i.e., the quality of the base learner at the initial setup

$$SQ = QM(t_{init}).$$

Ramp-Up (RU): The ramp-up is the quality increase after the initial k progress steps. A high RU indicates that the query strategy adapts well to the initial setup

$$RU(k) = QM(t_k) - QM(t_{init}).$$

Quality Range (QR): The Quality Range is the increase in classification quality over an interval $[t_i, t_j]$. A special case is $QR(\text{init}, \text{end})$, the overall improvement achieved with an active learning strategy

$$QR(i, j) = QM(t_i) - QM(t_j).$$

Average End Quality (AEQ): In general, the progress curve is non-monotonic because each query introduces a selection bias in the training data. So a query can lead to a quality decrease. The choice of t_{end} often is arbitrary and can coincide with a temporary bias. So we propose to use the Average End Quality to summarize the classification quality for the final k progress steps

$$AEQ(k) = \frac{1}{k} \sum_{i=1}^{k} QM(t_{end-k}).$$

Learning Stability (LS): Learning Stability summarizes the influence of the last k progress steps on the quality. A high LS indicates that one can expect further improvement from continuing the active learning cycle. A low LS on the other hand indicates that the classifier tends to be saturated, i.e., additional feedback does not increase the quality. We define LS as the ratio of the average QR in the last k steps over the average QR between init and end

$$LS(k) = \begin{cases} \frac{QR(\text{end}-k,\text{end})}{k} / \frac{QR(\text{init},\text{end})}{|\mathcal{L}^{\text{end}} \setminus \mathcal{L}^{\text{init}}|} & \text{if } QR(\text{init}, \text{end}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Ratio of Outlier Queries (ROQ): The Ratio of Outlier Queries is the proportion of queries that the oracle labels as outlier

$$ROQ = \frac{|\mathcal{L}^{\text{end}}_{\text{out}} \setminus \mathcal{L}^{\text{init}}_{\text{out}}|}{|\mathcal{L}^{\text{end}} \setminus \mathcal{L}^{\text{init}}|}.$$

In practice, the usefulness of a summary statistic to select a good active learning strategy depends on the learning scenario. For instance, ROQ is only meaningful if the user has a specific interest in observations from the minority class.

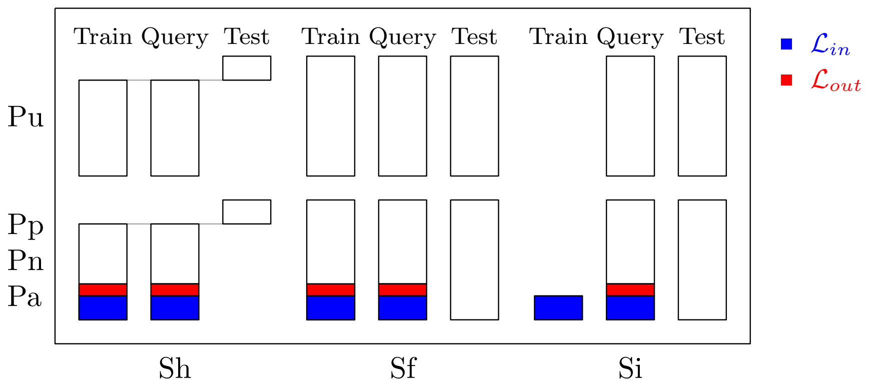


Figure 4.5.: Illustration of initial pool and split strategies. The blue and the red proportion indicate the labeled inliers and outliers in the initial pools that make up for p-% of the full data (Pp), a fixed number of observations (Pn) or the number of attributes (Pa).

We conclude the discussion of summary statistics with two comments. The first comment is on *Area under the Learning Curve* (AULC), which also can be used to summarize active learning curves [Caw11; RAV18]. We deliberately choose to not include AULC as a summary statistic for the following reasons. First, active learning is discrete, i.e., the minimum increment during learning is one feedback label. But since the learning steps are discrete, the “area” under the curve is equivalent to the sum of the quality metric over the learning progress $\sum_{i=\text{init}}^{end} QM(t_i)$. In particular, approximating the AULC by, say, a trapezoidal approximation [RAV18] is not necessary. Second, AULC is difficult to interpret. For instance, two curves can have different shapes and end qualities, but yet result in the same AULC value. We therefore rely on AEQ and SQ, which one can see as a partial AULC, with distinct interpretation.

Our second comment is using summary statistics to select different query strategies for different phases of the active learning cycle is conceivable in principle. For instance, one could start the cycle with a good RU and then switch to a strategy with a good AEQ. However, this leads to further questions, e.g., how to identify a good switch point, that go beyond this work.

4.2.4. Split Strategy

A split strategy specifies how data is partitioned between training and testing. With binary classifiers, one typically splits data into disjoint train and test partitions, which ideally are identically distributed. However, since outliers do not come from a joint distribution, measuring classification quality on an independent test set is misleading. In this case, one may measure classification quality as the resubstitution error, i.e., the classification quality on the training data. This error is an optimistic estimate of classification quality. But we deem this shortcoming acceptable if only a small percentage of the data has been labeled.

The learning objective should also influence how the data is split. For instance, if the learning objective is to reliably estimate the majority-class distribution, one may restrict the training set to inliers (cf. [Gha+11a; Gha+11b]). Three split strategies are used in the literature.

(Sh) Split holdout: Model fitting and query selection on the training split, and testing on a distinct holdout sample.

(Sf) Split full: Model fitting, query selection and testing on the full data set.

(Si) Split inlier: Like Sf, but model fitting on labeled inliers only.

Split strategies increase the complexity of evaluating active learning, since they must be combined with an initial pool strategy. Most combinations of split strategies and initial pool strategies are conceivable. Only no labels (Pu) does not work with a split strategy that fits a model solely on inliers (Si) – the train partition would be empty in this case. Figure 4.5 is an overview of all combinations of an initial pool strategy and a split strategy.

4.2.5. Limitations

Initial setups, split strategies, base learners and query strategies all come with prerequisites. One cannot combine them arbitrarily, because some of the prerequisites are mutually exclusive, as follows.

(i) Pu rules out any data-based query strategy. This is because data-based query strategies require labeled observations for the density estimations.

(ii) Kernel-density estimation requires the number of labeled observations to be at least equal to the number of attributes. A special case is τ_{ML}, Which requires $|\mathcal{L}_{outlier}| \geq M$. As a remedy, one can omit the subtrahend in Equation 4.7 and Equation 4.8 in this case.

(iii) Fully unsupervised base learners, e.g., SVDD, are only useful when the learning objective is a robust estimate of the majority distribution, and when the split strategy is Si. The reason is that feedback can only affect the classifier indirectly, by changing the composition of the training data $\mathcal{L}_{\text{in}}$.

(iv) A combination of Pu and Si is not feasible, see Section 4.2.4.

Table 4.1 is an overview of the feasibility of query strategies. In what follows, we only consider feasible combinations.

4.3. Benchmark

The plethora of ways to design and to evaluate active learning systems makes selecting a good configuration for a specific application difficult. Although certain combinations are infeasible, the remaining options are still too numerous to analyze. This section addresses question *Comparison* and provides some guidance how to navigate the overwhelming design space. We have implemented the base learners, the query strategies and the

Table 4.1.: Overview over the number of labels required by different query strategies; M is the number of attributes. Feasible: ✓, feasible with modification: (✓), not feasible: ×.

Scenario	τ_{MM}	τ_{EME}	τ_{EME}	τ_{ML}	τ_{HC}	τ_{DB}	τ_{NB}	τ_{BNC}	τ_{rand}	$\tau_{\text{rand-out}}$
$\lvert\mathcal{L}_{in}\rvert = 0 \wedge \lvert\mathcal{L}_{out}\rvert = 0$	×	×	×	×	×	✓	✓	✓	✓	✓
$\lvert\mathcal{L}_{in}\rvert \geq M \wedge \lvert\mathcal{L}_{out}\rvert = 0$	✓	✓	✓	(✓)	✓	✓	✓	✓	✓	✓
$\lvert\mathcal{L}_{in}\rvert \geq M \wedge \lvert\mathcal{L}_{out}\rvert \geq M$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

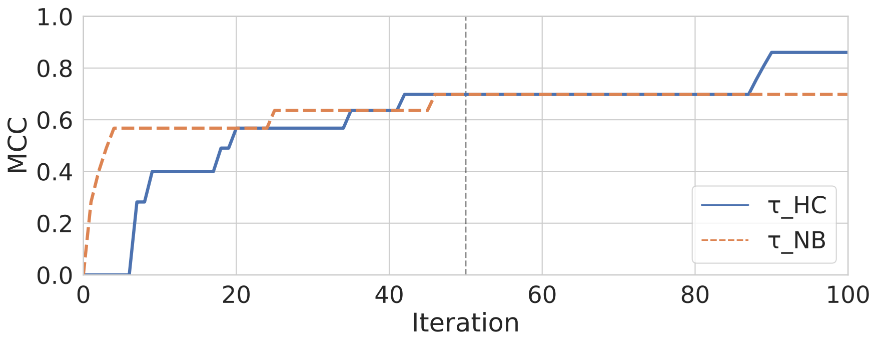


Figure 4.6.: Comparison of the progress curves for two query strategies on Arrhythmia.

benchmark setup in *Julia* [Bez+17]. Our implementation, the raw results of all settings and notebooks to reproduce experiments and evaluation are publicly available.[4]

We begin by explaining our experiments conducted on well-established benchmark data sets for outlier detection [Cam+16]. In total, we run experiments on over 84,000 configurations: 72,000 configurations in Section 4.3.3 to Section 4.3.3 are the cross product of 20 data sets, 3 resampled versions, 3 split strategies, 4 initial pool strategies, 5 models with different parametrization, 2 kernel parameter initializations and 10 query strategies; 12,000 additional configurations in Section 4.3.3 are the cross product of 20 data sets with 3 resampled versions each, 2 models, 2 kernel parameter initializations, 5 initial pool resamples and 10 query strategies. Table 4.2 lists the experimental space, see Section 4.3.2 for details.

Each specific experiment corresponds to a practical decision which query strategy to choose in a specific setting. We illustrate this with the following example.

> **Example 3 (Experiment Run)** *Assume the data set is Arrhythmia, and there are no initial labels, i.e., the initial pool strategy is Pu, and data-based query strategies are not applicable. The classifier is SVDDneg, and we use Sf to evaluate the classification quality. Our decision is to choose τ_{HC} and τ_{NB} as potential query strategies and to terminate the active-learning cycle after 100 iterations.*

[4] `https://www.ipd.kit.edu/ocal`

Table 4.2.: Overview on experimental setup.

Dimension	Configuration
Initial Pools	Pu, Pp (p = 0.1), Pn (n = 25), Pa
Split Strategy	Sf, Sh (80% train, 20% test), Si
Base Learner	SVDD, SVDDneg, SSAD ($\kappa = 1.0, 0.5, 0.1$)
Kernel Initialization	Wang, Scott
Query strategy	$\tau_{\text{MM}}, \tau_{\text{EMM}}, \tau_{\text{EME}}, \tau_{\text{ML}}, \tau_{\text{HC}}, \tau_{\text{DB}}, \tau_{\text{NB}}, \tau_{\text{BNC}}, \tau_{\text{rand}}, \tau_{\text{rand-out}}$

Figure 4.6 graphs the progress curves for both query strategies. A first observation is that it depends on the progress which one is better. For example, τ_{NB} results in a better MCC after 10 iterations, while τ_{NB} is superior after 90 iterations. After 50 iterations, both τ_{HC} and τ_{NB} perform equally well. Until iteration 60, the learning stability (LS) decreases to 0, which speaks for stopping. Indeed, although there is some increase after 60 iterations, it is small compared to the overall improvement.

For a more differentiated comparison, we now look at several progress curve summaries. If only the final classification quality is relevant, i.e., the budget is fixed to 100 observations, τ_{HC} is preferred because of higher EQ and AEQ values. For a fast adaption, one should prefer τ_{NB} with RU(5) = 0.57, compared to a RU(5) = 0.00 for τ_{HC}. Regarding the outlier ratio, both query strategies perform similarly with 7 % *and* 9 %. *Users can now weigh these criteria based on their preferences to decide on the most appropriate query strategy.*

In our benchmark, we strive for general insights and trends regarding such decisions. In the following, we first discuss assumptions we make for our benchmark and the experiment setup. We then report on results. Finally, we propose guidelines for outlier detection with active learning and discuss extensions to our benchmark towards conclusive decision rules.

4.3.1. Assumptions

We now specify the assumptions behind our benchmark.

General Assumptions. In our benchmark, we focus on "sequential class label" as the *feedback type.* We set the *feedback budget* to a fixed number of labels a user can provide. The reason for a fixed budget is that the number of queries in an active learning cycle depends on the application, and estimating active learning performance at runtime is difficult [Kot+19]. There is no general rule how to select the number of queries for evaluation. In our experiments, we perform 50 iterations. Since we benchmark on many publicly available data sets, we do not have any requirements regarding interpretability. Instead, we rely on the ground truth shipped with the data sets to simulate a perfect oracle.

Specific Assumptions. We have referred to specific assumptions throughout this chapter and explained how they affect the building blocks and the evaluation. For clarity, we briefly summarize them. For the *class distribution*, we assume that outliers do not have a joint distribution. The primary learning objective is to improve the accuracy of the classifier. However, we also use the *ROQ* summary statistic to evaluate whether a method yields a high proportion of queries from the minority class. For the initial setup, we do not make any further assumptions. Instead, we compare the methods on all feasible combinations of initial pools and split strategies.

4.3.2. Experimental Setup

Our experiments cover several instantiations of the building blocks. See Table 3.1 for an overview on the benchmark data sets. Table 4.2 lists the experimental space. For each data set we use three resampled versions with an outlier percentage of 5% that have been normalized and cleaned from duplicates. We have downsampled large data sets to $N = 1000$. This is comparable to the size of the data sets used in previous work for active learning for one-class classification. Additionally, one may use sampling techniques for one-class classifiers to scale to large data sets, e.g., [Kra+19; Li11]. However, further studying the influence of the data set size on the query strategies goes beyond the scope of this benchmark.

Parameters: Parameter selection for base learners and query strategies is difficult in an unsupervised scenario. One must rely on heuristics to select the kernel and cost parameters for the base-learners, see Section 4.1.2. We use Scott's rule of thumb [Sco15] and state-of-the-art self-adapting data shifting by Wang et al. [Wan+18] for the kernel parameter γ. For cost C we use the initialization strategy of Tax et al. [TD04]. For SSAD, the authors suggest to set the trade-off parameter $\kappa = 1$ [Gör+13]. However, preliminary experiments of ours indicate that SSAD performs better with smaller parameter values in many settings. Thus, we include $\kappa = 0.1$ and $\kappa = 0.5$ as well. For the query strategies, the selection of strategy-specific parameters is described in Section 4.1.3. The data-based query strategies use the same γ value for kernel density estimation as the base learner.

4.3.3. Results

We now discuss general insights and trends we have distilled from the experiments. We start with a broad overview and then fix some experimental dimensions step by step to analyze specific regions of the experimental space. We begin by comparing the expressiveness of evaluation metrics and the influence of base learner parametrization. Then we study the influence of the split strategy, the initial pool strategy, and the query strategy on result quality.

Evaluation Metric

Recall that our evaluation metrics are of two different types: ranking metrics (AUC and pAUC) and metrics based on the confusion matrix (kappa, MCC). On all settings, metrics of the same type have a high correlation for AEQ, see Table 4.3. So we simplify the evaluation by selecting one metric of each type.

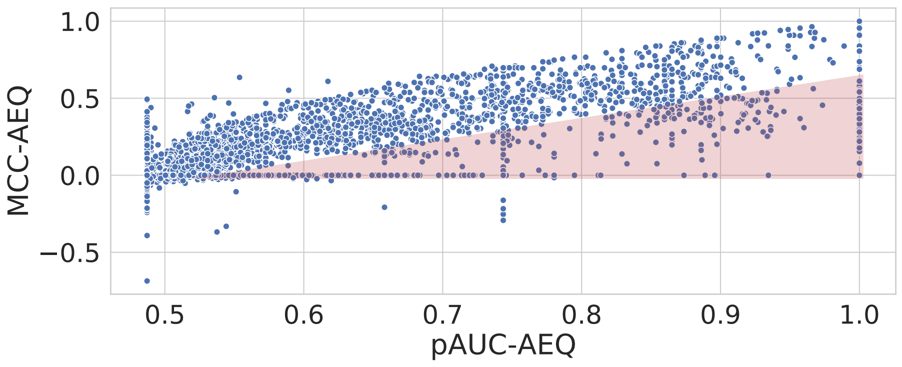


Figure 4.7.: Comparison of pAUC and MCC. Each point corresponds to an experimental run.

Table 4.3.: Pearson correlation of AEQ (k=5) for different evaluation metrics.

	MCC	kappa	AUC	pAUC
MCC	1.00	0.98	0.63	0.78
kappa	0.98	1.00	0.59	0.76
AUC	0.63	0.59	1.00	0.73
pAUC	0.78	0.76	0.73	1.00

Further, there is an important difference between both types. Figure 4.7 depicts the AEQ for pAUC and MCC. For high MCC values, pAUC is high as well. However, high pAUC values often do not coincide with high MCC values, please see the shaded part of the plot. In the extreme cases, there even are instances where pAUC = 1 and MCC is close to zero. In this case, the decision function induces a good ranking of the observations, but the actual decision boundary does not discern well between inliers and outliers. An intuitive explanation is that outliers tend to be farthest from the center of the hypersphere. Because pAUC only considers the top of the ranking, it merely requires a well-located center to arrive at a high classification quality. But the classifier actually may not have fit a good decision boundary.

Our conclusion is that pAUC and AUC may be misleading when evaluating one-class classification. Hence, we only use MCC from now on.

Influence of Kernel Parameter

Recall that the kernel parameter influences the flexibility of the decision boundary; high values correspond to more flexible boundaries. Our hypothesis is that a certain flexibility is necessary for models to adapt to feedback.

Table 4.4 shows the SQ and AEQ for two heuristics to initialize γ. In both summary statistics, Wang strategy outperforms the simpler Scott rule of thumb significantly on the median over all data sets and models. A more detailed analysis shows that there are some data sets where Scott outperforms Wang, e.g., KDD-Cup. However, there are many

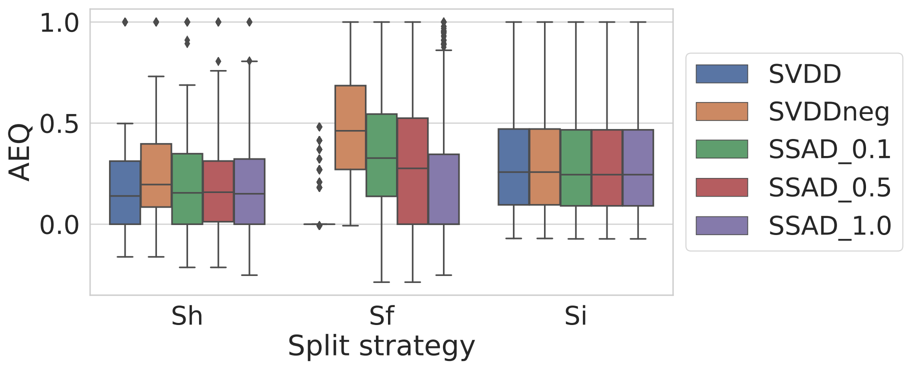


Figure 4.8.: Evaluation of the AEQ (k=5) for different split strategies grouped by base learner.

Table 4.4.: Median AEQ (k=5) and SQ for different gamma initialization strategies.

Model	SQ		AEQ	
	Scott	WangTax	Scott	WangTax
SVDD	0.06	0.06	0.09	0.10
SVDDneg	0.09	0.14	0.21	0.31
SSAD_0.1	0.07	0.11	0.15	0.24
SSAD_0.5	0.06	0.06	0.12	0.22
SSAD_1.0	0.04	0.08	0.10	0.15

instances where Wang performs well, but Scott results in very poor active learning quality. For instance, the AEQ on Glass for Scott is 0.06, and for Wang 0.45. We hypothesize that this is because Scott yields very low γ values for all data sets, and the decision boundary is not flexible enough to adapt to feedback. The average value is $\gamma = 0.77$ for Scott and $\gamma = 5.90$ for Wang.

We draw two conclusions from these observations. First, the choice of γ influences the success of active learning significantly. When the value is selected poorly, active learning only results in minor improvements on classification quality – regardless of the query strategy. Second, Wang tends to select better γ values than Scott, and we use it as the default in our experiments.

Split Strategies

Our experiments show that split strategies have a significant influence on classification quality. Figure 4.8 graphs the AEQ for the different split strategies grouped by base learners.

We first compare the three split strategies. For Sh, the AEQ on the holdout sample is rather low for all base learners. For Sf, SVDDneg and SSAD_0.1 achieve high quality. Some of this difference may be explained by the more optimistic resubstitution error in Sf. However, the much lower AEQ in Sh, for instance for SVDDneg, rather confirms that

Table 4.5.: Comparison of SQ and AEQ (k=5) for different initial pool strategies, Pp = 10 %, Pn = 20 observations.

Data set	Initial Pool	n	Initially labeled	SQ	AEQ
ALOI	Pn	1000	20	0.00	0.14
	Pp	1000	100	0.17	0.22
WBC	Pn	200	20	0.31	0.74
	Pp	200	20	0.31	0.72

outliers do not follow a homogeneous distribution (cf. Section 4.1.1). In this case, the quality on the holdout sample is misleading.

For Si, all classifiers yield about the same quality. This is not surprising. The classifiers are trained on labeled inliers only. So the optimization problems for the base learners coincide. The average quality is lower than with Sf, because the training split only contains a small fraction of the inliers. Based on all this, we question whether Si leads to an insightful evaluation, and we exclude Si from now on.

Next, we compare the quality of the base learners. For Sf, SVDD fails because it is fully unsupervised, i.e., cannot benefit from feedback. For SSAD, the quality fluctuates with increasing κ. Finding an explanation for this is difficult. We hypothesize that this is because SSAD overfits to the feedback for high κ values. For Sf, $\kappa = 0.1$ empirically is the best choice.

In summary, the split strategy has a significant effect on classification quality. SVDDneg and SSAD_0.1 for Sf yield the most reasonable results. We fix these combinations for the remainder of this section.

Initial Pool Strategies

The initial pool strategy specifies the number of labeled observations at t_{init}. Intuitively, increasing it should increase the start quality, as more information on the data is available to the classifier. If the initial pool is representative of the underlying distribution, little benefit can be expected from active learning.

Our results confirm this intuition. Figure 4.9 shows the SQ for the initial pool strategies grouped by SVDDneg and SSAD_0.1. For Pu, there are no labeled observations, and the corresponding SQ is low. When labeled data is available, Pp tends to yield a better SQ than Pn. However, the figure is misleading, because the actual number of labels depends on the data set. This becomes clear when looking at ALOI and WBC, see Table 4.5. For WBC, Pp and Pn result in a similar number of initial labels. For ALOI however, the number of labels with Pp is five times larger than with Pn. So the SQ on ALOI is higher for Pp, but AEQ is only slightly higher than SQ. This means that active learning has comparatively little effect. Pa has a technical motivation, i.e., it is the minimal number of labels required by the data-based strategies. This strategy is not feasible for data sets where the number of

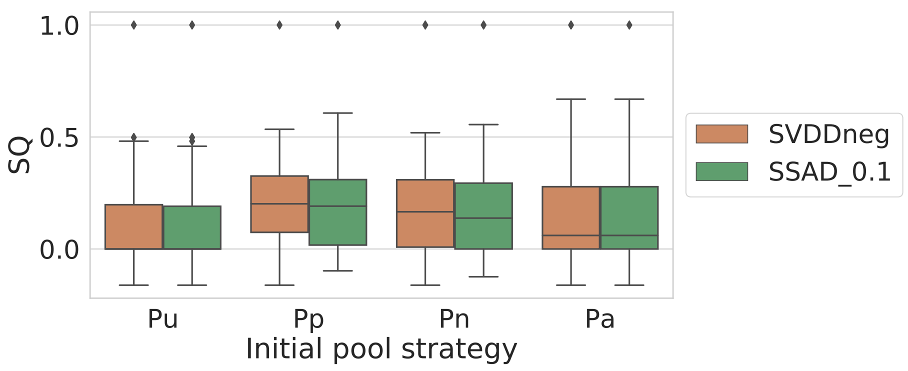


Figure 4.9.: Evaluation of the different initial pool strategies.

attributes is larger than the number of observations. Other than this, the interpretation of Pa is similar to Pp with $p = \frac{M}{N}$.

In summary, different initial pool strategies lead to substantially different results. We deem Pn more intuitive than Pp when reporting results, since the size of the initial sample, and hence the initial labeling effort is explicit. In any case, one must carefully state how the initial sample is obtained. Otherwise, it is unclear whether high quality according to AEQ is due to the query strategy or to the initial pool.

Query Strategy

We have arrived at a subset of the experimental space where comparing different query strategies is reasonable. To do so, we fix the initial pool strategy to Pn with $n = 25$. In this way, we can include the data-based query strategies which all require initial labels. We obtain the initial pool by uniform stratified sampling. Additionally, we exclude the Hepatitis data set because it only contains 60 observations; this is incompatible with 20 initially labeled observations and 50 iterations. We repeat each setting 5 times and average the results to reduce the bias of the initial sample.

Table 4.6 shows the median QR(init, end) grouped by data set. By design of the experiment, SQ is equal for all query strategies. This means that AEQ coincides with QR. On some data sets (indicated by " - "), data-based query strategies fail. The reason is that the rang of the matrix of observations, on which the kernel density is estimated, is smaller than M. For the remaining data sets, we make two observations. First, the QR achieved differs between data sets. Some data sets, e.g., Annthyroid and PageBlocks, seem to be more difficult and only result in a small QR. Second, the quality of a specific query strategy differs significantly between data sets. For instance, τ_{ML} is the best strategy on Lymphography, but does not increase the classification quality on PageBlocks. In several cases, $\tau_{\text{rand-out}}$ clearly outperforms the remaining strategies. There neither is a query strategy category nor a single query strategy that is superior on all data sets. This also holds for other metrics like RU and ROQ.

Next, runtimes for τ_{ML} are an order of magnitude larger than for all other strategies. For PageBlocks, the average runtime per query selection for τ_{ML} is 112 s, compared to 0.5 s for τ_{NB}.

Table 4.6.: Median QR on Pn for different query strategies over 30 settings; highest values per data set in bold.

	data-based				model-based		hybrid		baselines	
Data set	τ_{MM}	τ_{EMM}	τ_{EME}	τ_{ML}	τ_{HC}	τ_{DB}	τ_{NB}	τ_{BNC}	τ_{rand}	$\tau_{\text{rand-out}}$
ALOI	-	-	-	-	**0.30**	**0.30**	0.28	0.00	0.00	0.00
Annthyroid	-	-	-	-	**0.28**	0.20	**0.28**	0.24	0.09	0.14
Arrhythmia	-	-	-	-	**0.73**	**0.73**	0.53	0.40	0.00	0.70
Cardiotocography	-	-	-	-	0.42	**0.46**	**0.46**	0.05	0.08	0.10
Glass	0.0	0.56	0.14	0.67	**0.67**	**0.67**	0.55	0.55	0.24	0.24
HeartDisease	0.0	0.00	0.28	0.34	0.23	0.23	0.32	0.29	0.16	**0.36**
Ionosphere	-	-	-	-	0.65	**0.72**	0.53	0.12	0.21	0.35
KDDCup99	-	-	-	-	0.36	**0.53**	**0.53**	0.33	0.14	0.14
Lymphography	0.0	0.00	0.51	**0.67**	0.40	0.41	0.41	0.43	0.41	0.34
PageBlocks	0.0	0.00	0.01	0.00	0.09	0.22	**0.24**	0.11	0.01	0.08
PenDigits	0.0	0.00	0.00	0.22	0.19	0.19	0.50	0.09	0.16	**0.78**
Pima	0.0	0.00	0.00	0.39	0.28	0.35	0.35	0.19	0.19	**0.50**
Shuttle	0.0	0.00	0.38	0.37	0.57	0.57	0.58	0.61	0.27	**0.75**
SpamBase	-	-	-	-	0.28	0.28	**0.29**	0.10	0.00	0.17
Stamps	0.0	0.00	0.24	0.62	0.61	**0.65**	0.56	0.47	0.24	0.00
WBC	0.0	0.00	0.58	**0.69**	**0.69**	**0.69**	0.39	0.13	0.31	**0.69**
WDBC	-	-	-	-	0.58	0.58	0.58	0.23	0.39	**0.64**
WPBC	-	-	-	-	**0.26**	**0.26**	**0.26**	0.20	0.00	0.14
Waveform	0.0	0.00	0.14	0.00	**0.28**	**0.28**	**0.28**	0.21	0.06	0.00

To summarize, there is no one-fits-all query strategy for one-class active learning. The requirements for data-based query strategies may be difficult to meet in practice. If the requirements are met, all model-based and hybrid strategies we have evaluated except for τ_{BNC} may be a good choice. In particular, τ_{DB} and $\tau_{\text{rand-out}}$ are a good choice in the majority of cases. They result in significant increases over 50 iterations for most data sets and scale well with the number of observations. Even in the few cases where other query strategies outperform them, they still yield acceptable results.

4.3.4. Guidelines and Decision Rules

The results from previous sections are conclusive and give way to general recommendations for outlier detection with active learning. We summarize them as guidelines for the selection of query strategies and for the evaluation of one-class active learning.

Guidelines

(i) Learning scenario: We recommend to specify general and specific assumptions on the feedback process and the application. This narrows down the design space of building-block combinations. Regarding research, it may also help others to assess novel contributions more easily.

(ii) Initial Pool: The initial pool strategy should either be Pu, i.e., a cold start without labels, or Pn with an absolute number of labels. It is important to make explicit if and how an initial sample has been obtained.

(iii) Base Learner: A good parametrization of the base learner is crucial. To this end, selecting the bandwidth of the Gaussian kernel by self-adaptive data shifting [Wan+18] works well. When parameters are well-chosen, SVDDneg is a good choice across data sets and query strategies.

(iv) Query Strategies: Good choices across data sets are τ_{DB} and $\tau_{\text{rand-out}}$. One should give serious consideration to random baselines, as they are easy to implement and outperform the more complex strategies in many cases.

(v) Evaluation: Progress curve summaries yield a versatile and differentiated view on the performance of active learning. We recommend to use them to select query strategies for a specific use case. As the quality metric, we suggest to use MCC or kappa. Calculating this metric as a resubstitution error based on a Sf split is reasonable for outlier detection.

Beyond Guidelines

From the results presented so far, one may also think about deriving a formal and strict set of rules to select an active learning method that are even more rigorous than the guidelines presented. However, this entails major difficulties, as we now explain. Addressing them requires further research that goes beyond the scope of a comparative study.

(i) One can complement the benchmark with additional real-world data sets. But they are only useful to validate whether rules that have already been identified are applicable to other data as well. So, given our current level of understanding, we expect additional real-world data sets to only confirm our conclusion that formal rules currently are beyond reach.

(ii) One may strive for narrow rules, e.g., rules that only apply to data with certain characteristics. This would require a different kind of experimental study, for instance with synthetic data. This also is difficult, for at least two reasons. First, it is unclear what interesting data characteristics would be in this case. Even if one can come up with such characteristics, it still is difficult to generate synthetic data with all these interesting characteristics. Second, reliable statements on selection rules would require a full factorial design of these characteristics. This entails a huge number of combinations with experiment runtimes that are likely to be prohibitive. To illustrate, even just 5 characteristics with 3 manifestations each result in a $3^5 = 243$ data sets instead of 20 data sets, and a total of 874,800 experiments – an order of magnitude larger than the experiments presented here. Yet our experiments already have a sequential run time of around 482 days.

(iii) One could strive for theoretical guarantees on query strategies. But the strategies discussed in Section 4.3.3 are heuristics and do not come with any guarantees. A discussion of the theoretical foundations of active learning may provide further insights. However, this goes beyond the scope of this comparative study as well.

To conclude, deriving a set of formal rules based on our results is not within reach. So one should still select active learning methods for a use case individually. Our systematic approach from the previous sections does facilitate such a use-case specific selection. It requires to carefully define the learning scenario and to use summary statistics for comparisons.

4.4. Summary

In this chapter, we have studied active learning for outlier detection with one-class classifiers. We have identified and explained three building blocks: the learning scenario, a base learner, and a query strategy. While the literature features several approaches for each of the building blocks, finding a suitable combination for a particular use case is challenging. We have approached this challenge in two steps. First, we provide a categorization of active learning for one-class classification and propose methods to evaluate active learning beyond progress curves. Second, we have evaluated existing methods, using an extensive benchmark. Our experimental results show that there is no one-fits-all strategy for one-class active learning. Thus, we have distilled guidelines on how to select a suitable active learning method with specific use cases. Our categorization, evaluation standards and guidelines give way to a more reliable and comparable assessment of active learning for outlier detection with one-class classifiers.

5. One-Class Active Learning in Multiple Subspaces

In our overview and benchmark in the last chapter, we have identified fundamental assumptions on one-class active learning. These assumptions are widely accepted in literature, and have become such a commonplace that they often are neither explicitly discussed nor questioned. However, it is unclear if these assumptions hold up in real applications. In this chapter, we take a closer look at *interpretability*, a "general assumption" which we have introduced earlier, see Section 4.1.1. To recap, interpretability assumes two things of a user: (i) sufficient domain knowledge to provide feedback purposefully, and (ii) the ability to understand the outputs of a one-class classifier. The reason why this assumption is so common in literature is that it does away with the complications of real user interaction, and gives way to conduct comparative studies based on ground truth data. However, this is a far-reaching simplification. Think about our illustration from earlier, where we question whether a user can answer the following query: *Is the real vector $\langle x_1, ..., x_{20} \rangle$ with an outlier score of 0.74 unusual?* An educated answer to this query requires the user to have an intuition of the outlier scoring function and of the multi-dimensional data distribution. From collaborations with scientists from other domains, we know that getting an answer to such a query is unrealistic.[1] In a way, literature also is in line with this: There is a plethora of active learning strategies for one-class classifiers [Gha⁺11b; BBJ15; TEB18; Gör⁺13], but only few references report on real applications [BCB18; Ver⁺18; TEB19]. All this suggests that comprehensiveness and interpretability are important prerequisites for educated feedback.

In this chapter, we study one-class active learning under more realistic assumptions. We focus on semi-supervised one-class classification in low-dimensional subspaces, to facilitate interpretability and consequently high-quality feedback. We replace the assumption that users can give feedback on any classification result with a more differentiated and realistic one: We assume that *users can give feedback on observations whose contexts, i.e., subspaces where the observations are outlying, are low-dimensional, but this ability decreases with increasing dimensionality of the contexts.*

As before, we assume users to provide feedback in form of class-label annotations ("outlier" or "inlier"). However, there is an additional distinction that we have to make, which is between global and subspace-specific feedback. So far, feedback has been global,

[1] The remainder of this chapter bases on the article [TB19b] *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`. It has been shortened to be less repetitive. It contains minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

since we have only considered data in the full space. Subspace-specific feedback is to collect annotations for each subspace individually. Because of the multi-view property of outliers, (cf. Definition 3), subspace-specific annotations only are valid for the specific subspace they have been collected for. Collecting subspace-specific feedback is difficult because the annotation effort grows linearly with the number of subspaces. Existing benchmark data also does not feature a subspace-specific ground truth – this would require to label each observation in all subspaces. This is infeasible, because the number of subspaces increases exponentially by $2^d - 1$ with data-set dimensionality d. Thus, class-label feedback must be global. So a first requirement on a one-class classifier for active learning is that it must feature an update mechanism based on this kind of feedback. A second requirement is that users must be able to interpret the algorithmic result. – Current methods fulfill only one of these requirements. In line with this, one can now approach the development of a new method in two ways:

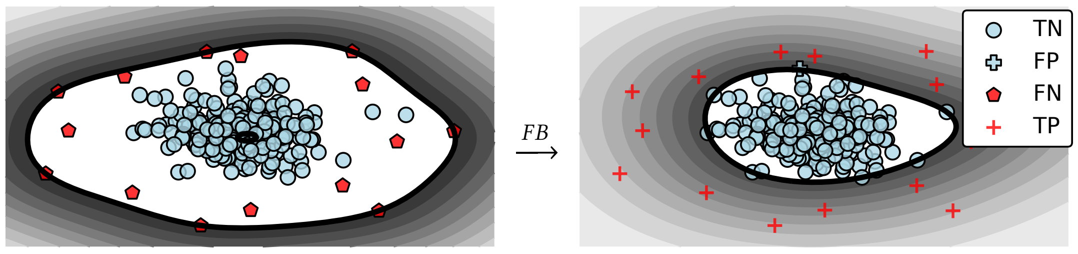
Update Mechanisms for Unsupervised Methods: The first alternative is to start with a subspace outlier detection method and extend it with an update mechanism. However, subspace outlier detection methods are unsupervised and thus cannot use class-label feedback, by definition. So adjusting the model is only possible by modifying hyper-parameter values, e.g., the neighborhood size with methods that rely on local densities. This is notoriously difficult, since hyper-parameters are interdependent, and estimating the effect of changes of their values is hard. We conclude that striving towards such an update mechanism is intricate and not promising.

Interpretable Semi-supervised Methods: The alternative is to borrow the notion of interpretability from unsupervised methods and to apply semi-supervised one-class classifiers to subspaces. But a characteristic we call *outlier asymmetry* is in the way of this. Outlier asymmetry means that an observation is unusual if it is classified as such in *any* subspace, and it is inlying if it is classified as such in *all* subspaces. On the one hand, this is plausible, because outliers may occur only in certain subspaces [Agg15]. On the other hand, this complicates the design of the approach envisioned, in two ways.

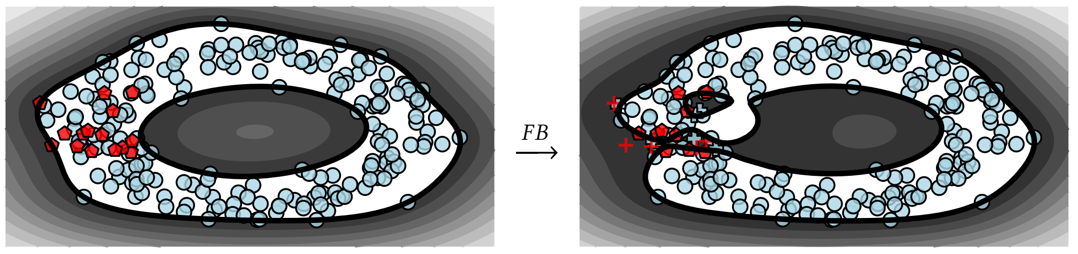
1) Outlier Ratio: One-class classifiers have parameters, e.g., cost parameters in the underlying optimization problems [Gör+13; TD04], that are related to the expected ratio of outliers in the data. However, that ratio in a subspace generally differs from the overall ratio and varies between subspaces; some subspaces may not contain outliers at all. So parameterization is hard, since there is no reasonable way to determine outlier ratios per subspace a priori. Thus, simply training one-class classifiers in multiple subspaces is not feasible.

2) Interpretation of Global Feedback: Recall that class-label feedback is global, i.e., not subspace-specific. Intuitively, the global feedback "outlier" should only affect subspaces where this observation is indeed outlying. This mismatch between global feedback and local outlier detection is challenging. Section 5.1 illustrates this.

We make two contributions in this chapter. (i) We propose SubSVDD, a novel semi-supervised classifier for one-class active learning. In a nutshell, SubSVDD takes a set of subspaces as an external parameter and learns the decision boundaries in these subspaces. It builds upon support vector data description (SVDD), but features modifications of it to overcome the issues caused by outlier asymmetry. (ii) We propose an active learning framework for query selection with multiple subspaces. The idea is to build upon existing



(a) S1 – before feedback. (b) S1 – after feedback.



(c) S2 – before feedback. (d) S2 – after feedback.

Figure 5.1.: Two subspaces with global feedback (FB); true positives (TP) are observations correctly classified as outlier.

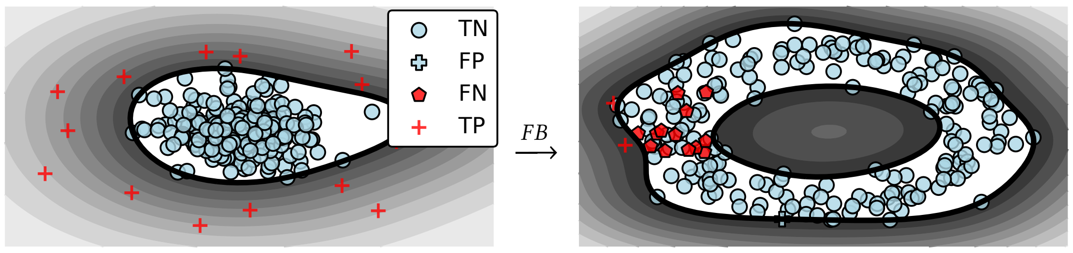
query strategies, apply them to individual subspaces, and to combine their results. This facilitates active learning with SubSVDD.

Active learning with SubSVDD has three advantages over existing one-class active learning methods: The first one is *interpretability*: Together with a binary classification, SubSVDD yields projections explaining each outlier. In other words, this is a concise result description, it maps observations to their contexts. The second one is that SubSVDD allows to *trade the effort users spend on interpreting results for classification quality*: When subspaces are two-dimensional, subspace-specific decision boundaries can be visualized easily. In this case however, one would not detect any outliers which occur only in subspaces with, say, three or more dimensions. Allowing for higher-dimensional subspaces can improve the overall detection, but this also makes the algorithmic result more difficult to comprehend. The third advantage is an *increased classification quality with active learning*. In a competitive benchmark, SubSVDD achieves good classification accuracy and outperforms SVDDneg and SSAD.

5.1. Illustrations

We first illustrate the mismatch between global feedback and classification in subspaces.

Example 4 (Outlier Asymmetry) *Figure 5.1 depicts two subspaces, S1 and S2, of a multi-dimensional data set. In S1, several observations are outliers. In S2, these observations are in a dense area, i.e., are inliers. We now have trained a semi-supervised classifier in each subspace, both before and after a user has provided feedback on individual observations. The feedback is global, i.e., the user has labeled an observation*



(a) S1 – after feedback. (b) S2 – after feedback.

Figure 5.2.: SubSVDD with global feedback.

as outlying if it is unusual in any one subspace. The classifier in S1 responds to the feedback as desired, by shrinking the decision boundary to exclude the outliers, cf. Figure 5.1 (b). In S2, the classifier excludes these observations as well, although they are local inliers, i.e., inliers within S2. This leads to an odd subspace-specific decision boundary and ultimately to false predictions.

The effect just illustrated is common, since an outlier likely is a local inlier in some projection. The effect size varies with the data distribution and with the location and ratio of inlying outliers in the subspace. As explained earlier, using subspace-specific feedback to avoid the issue is not an option. – We now apply SubSVDD to the setting from Example 4.

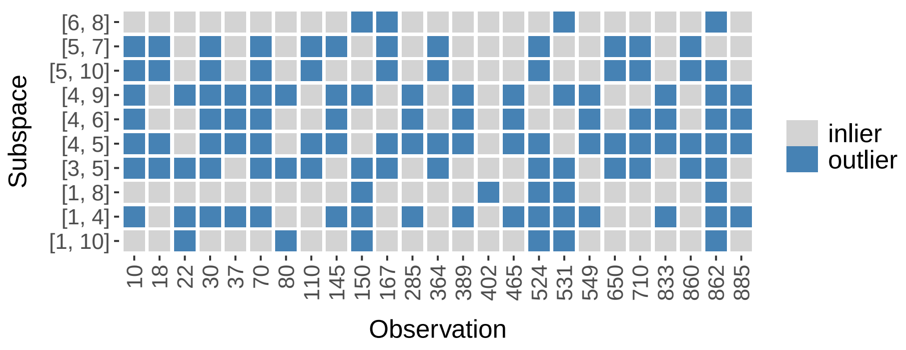
Example 5 (SubSVDD) *In the setting from Example 4, SubSVDD yields the expected decision boundaries in both projections, see Figure 5.2. In S_1, all outliers are excluded from the hypersphere; in S_2, only two observations are classified as outlying, and they are close to the decision boundary and separable from the inliers. In particular, local inliers in S_2 fall inside the decision boundary.*

Section 5.2.1 explains how SubSVDD can achieve this result by using a weighting scheme to interpret global feedback in subspaces. Regarding result representation, SubSVDD extends the algorithmic result with a compact description, the mapping of outliers to their contexts. The following example illustrates its usefulness.

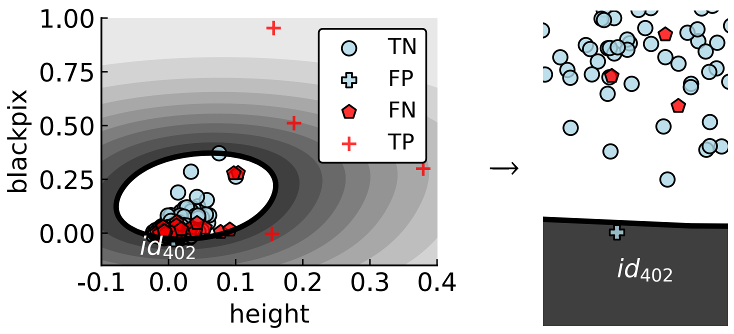
Example 6 *Figure 5.3a is an example result description of SubSVDD, for real-world data. From this output, one can infer that id_{402} has been classified as outlier, but only in subspace $[1, 8]$. The visualization of this subspace, in Figure 5.3b, shows that observation id_{402} lies at the border of a dense area, and, based on this visual inspection, should rather be classified as inlier in this subspace. A comparison with the ground truth reveals that this observation is indeed a false positive.*

5.2. Method

In this section, we introduce SubSVDD, our semi-supervised one-class classifier to detect outliers in multiple subspaces. In Section 5.2.2, we then propose a framework for active



(a) Compact result description.



(b) Subspace $[1, 8]$ with highlighted false positive id_{402} at $(0.0, 0.0)$.

Figure 5.3.: Overview of subspace classifications for the Page data set after 50 active learning iterations.

learning with multiple subspaces. Throughout this chapter, we rely on the following active learning assumptions:

Learning Objective: The learning objective is the primary reason to apply active learning. Our objective is higher classification accuracy. An alternative objective we consider is to have more outliers presented to the user during feedback iterations [Das⁺16; Sid⁺18].

Feedback Type: Active learning with one-class classifiers has been limited to class-label feedback that is provided sequentially, i.e., for one observation at a time. Studying batch queries is beyond the scope this chapter.

Class Distribution: As before, we assume that the class distribution is imbalanced, and that outliers do not come from a joint distribution.

Initial Setup: We assume that no class-label information is available for classifier training initially. This requires the classifier to first work in an unsupervised mode and to switch to a semi-supervised mode when feedback is available.

5.2.1. SubSVDD

We now introduce SubSVDD. We begin by introducing the primal optimization problem, then derive its dual, and finally say how to classify observations.

Primal Problem

The core idea of SubSVDD is to learn several hyperspheres in a set of low-dimensional projections of the data. Formally, this is the following optimization problem.

$$\begin{aligned} \text{SubSVDD}: \quad & \underset{\mathbf{R},\mathbf{a},\xi}{\text{minimize}} && \sum_k R_k^2 + C \cdot \sum_i v_i \cdot \xi_i \\ & \text{subject to} && \left\|\Phi_k(x_{i,k}) - a_k\right\|^2 \leq R_k^2 + \xi_i, \ \forall i, k \\ & && \xi_i \geq 0, \ \forall i \end{aligned} \tag{5.1}$$

with the vector of radii $\mathbf{R}$, the vector of hypersphere centers $\mathbf{a}$, slack variables ξ, and subspaces $S_1, \dots S_K$. The objective is to minimize the sum of the radii and the costs of placing observations outside of the hypersphere. Like SVDDneg and SSAD, SubSVDD requires a kernel function Φ, evaluated in each subspace, and a global trade-off parameter C. SubSVDD also has weight parameters v, which will be the core of our active learning update mechanism. To address the issues of outlier asymmetry, SubSVDD features two important differences compared to SVDD and SVDD_{neg}.

Global Slack: First, while SubSVDD learns the decision boundary simultaneously in multiple subspaces, it uses a slack variable ξ which is global per observation. Intuitively, the slack variable is strictly positive when the observation is outlying and gets larger the farther away it is from one of the decision boundaries. The rationale behind a global slack is that an observation x_j should be excluded from the hypersphere if it is a strong outlier in a subspace S_k, i.e., ξ_j is large because of S_k, or when x_j is a weak outlier in multiple subspaces, i.e., ξ_j is small but allows to decrease the radii in multiple subspaces. We achieve this by binding the slack to observations and not to individual subspaces. This allows to set a single, global cost parameter C. This addresses issue *Outlier Ratio*.

Weight parameter v: Second, we introduce weight parameter v to address the challenge of outlier asymmetry with global feedback. Weighting schemes have been used before with SVDD, their intent has been to improve the robustness of SVDD, e.g., with local densities [CKB14] or fuzzy clustering [Zha+09]. We for our part use weights to update SVDD based on class-label feedback, as follows.

We initialize $v = \langle 1, 1, \dots, 1 \rangle_N$. v_{in} and v_{out} are the hyperparameters of the weight update strategy. When feedback is available, we update $v_i = v_{\text{in}}$ if $x_i \in \mathcal{L}_{\text{in}}$ and $v_i = v_{\text{out}}$ if $x_i \in \mathcal{L}_{\text{out}}$. Intuitively, $v_{\text{out}} \ll 1$ since this means that excluding observation x_i from the hypersphere is cheap. Thus, x_i is unlikely to increase the hypersphere radius in *some* subspaces S_k where x_i is a local outlier. At the same time, x_i is not forced outside the hypersphere in a subspace $S_{k' \neq k}$ where x_i is a local inlier. This is because the cost of excluding the surrounding inliers and unlabeled observations is much higher than the benefit of excluding x_i. So x_i is classified as inlying in $S_{k'}$. On the other hand, $v_{\text{in}} \gg 1$ implies that excluding x_i from a hypersphere is expensive in *all* subspaces. Thus, a large v_i is likely to force the decision boundary to include x_i, in all subspaces. In conclusion, this weighting scheme addresses issue *Interpretation of Global Feedback*. We discuss how to set v_{in} and v_{out} in Section 5.3.2.

Since v depends only on the pool of the observation, i.e., $\mathcal{U}$, $\mathcal{L}_{\text{in}}$ or $\mathcal{L}_{\text{out}}$, one could reformulate the SubSVDD objective function by using three different C values, similarly to Equation 4.1. However, our current formulation with weights is more flexible, since it allows observation-specific updates. For instance, it gives way to update the weights differently based on how certain users are on the feedback they provide.

Dual Problem

To solve the SubSVDD optimization problem, we derive its Lagrangian dual. For better readability, we first derive the dual without the kernel mapping.

$$\begin{aligned}\mathbf{L}(\mathbf{a}, \mathbf{R}, \xi, \beta, \gamma) = \sum_k R_k^2 + C \cdot \sum_i v_i \cdot \xi_i - \sum_i \gamma_i \cdot \xi_i - \\ \sum_{i,k} \beta_{i,k} \left(R_k^2 + \xi_i - \langle x_{i,k}, x_{i,k}\rangle + 2\langle x_{i,k}, a_k\rangle - \langle a_k, a_k\rangle\right)\end{aligned}$$

with dual variables $\gamma_i \geq 0, \beta \geq 0$ which is to be minimized with respect to primal variables $\mathbf{a}, \mathbf{R}, \xi$, and maximized with respect to β, γ. Setting the partial derivatives of $\mathcal{L}$ to zero gives

$$\begin{aligned}\frac{\partial \mathbf{L}}{\partial R_k} = 0: \quad & 2R_k - 2\sum_i \beta_{i,k} R_k = 0 \\ & \Leftrightarrow \sum_i \beta_{i,k} = 1 \text{ with } R_k > 0 \qquad (5.2a)\\ \frac{\partial \mathbf{L}}{\partial a_k} = 0: \quad & -\sum_i \beta_{i,k}\left(2x_{i,k} - 2a_k\right) \\ & \Leftrightarrow a_k = \frac{\sum_i \beta_{i,k} x_{i,k}}{\sum_i \beta_{i,k}} \overset{(5.2a)}{=} \sum_i \beta_{i,k} x_{i,k} \qquad (5.2b)\\ \frac{\partial \mathbf{L}}{\partial \xi_i} = 0: \quad & C \cdot v_i - \sum_k \beta_{i,k} - \gamma_i = 0 \\ & \Leftrightarrow \sum_k \beta_{i,k} = C \cdot v_i - \gamma_i \qquad (5.2c)\end{aligned}$$

where from $\beta_{i,k} \geq 0, \ \forall i, k$ and $\gamma_i \geq 0, \ \forall i$ follows

$$0 \leq \sum_k \beta_{i,k} \leq C \cdot v_i. \qquad (5.3)$$

Substitution back into $\mathbf{L}$ gives the dual problem.

$$\begin{aligned}\underset{\beta}{\text{maximize}} \quad & \sum_{i,k} \beta_{i,k}\langle x_{i,k}, x_{i,k}\rangle - \sum_{i,j,k} \beta_{i,k}\beta_{j,k}\langle x_{i,k}, x_{j,k}\rangle \\ \text{subject to} \quad & \sum_i \beta_{i,k} = 1, \ \forall k \\ & \beta_{i,k} \geq 0, \ \forall i, k \\ & \sum_k \beta_{i,k} \leq C \cdot v_i, \ \forall i\end{aligned} \qquad (5.4)$$

where subspaces are indexed by $k \in \{1, \ldots, |\mathrm{S}|\}$ observations by $i, j \in \{1, \ldots, N\}$. The dual only depends on inner products of the $x_{i,k}$. So we can solve the dual in the kernel space by replacing inner products $\langle x, x' \rangle$ with $\langle \Phi(x), \Phi(x') \rangle$ or by the pairwise (N, N)-kernel matrix K. This changes the objective function in Equation 5.4 to

$$\underset{\beta}{\text{maximize}} \sum_{i,k} \beta_{i,k} K_k(i,i) - \sum_{i,j,k} \beta_{i,k} \beta_{j,k} K_k(i,j)$$

where K_k is the kernel matrix in Subspace S_k.

Since the dual is a quadratic program, one can use standard QP solvers in principle to obtain a solution β^*. However, the number of decision variables as well as the size of the kernel matrix increase with the number of subspaces and observations. This is in the way of scaling SubSVDD to very large problem instances. However, we see several ways to mitigate this issue in practice. For instance, one may reduce the problem instance with sampling methods that are tailored to support vector data descriptions [Li11; Sun+16; Kra+19] and use decomposition methods like sequential minimal optimization [Sch+01]. Despite this scalability challenge, QP solvers have turned out to be sufficient in our experiments.

Classification

Classifying observations with SubSVDD requires two steps. The first one is calculating the distance of observations to the decision boundary. A positive distance classifies an observation as outlier in a subspace, i.e., the distance between center of the hypersphere and the observation is larger than the radius of the hypersphere. The second step is combining classifications from subspaces to a final prediction.

To calculate the distance from the hypersphere in S_k, one has to compute R_k, by calculating the distance of any observation on the hypershpere to the center a_k. Objects on the hypersphere satisfy the primal constraint with equality without slack, i.e., $R_k^2 = \left\| x_{i,k} - a_k \right\|^2$. We use the necessary condition of complementary slackness to derive R_k for the dual problem. It states that, for any feasible dual variable λ_i, inequality constraint g_i and optimal point z^* it holds that:

$$\lambda_i \cdot g_i(z^*) = 0$$

There are three different cases under which complementary slackness holds for an observation x_i.

Case 1: Observation x_i is *inlier* in subspace k', i.e., lies inside the hypersphere. In this case, it holds that

$$\left\| x_{i,k'} - a_{k'} \right\|^2 < R_{k'}^2$$

To fulfill complementary slackness, it follows that $\beta_{i,k'} = 0$.

Case 2: Observation x_i is outlier in Subspace $S_{k'}$, i.e., lies outside of the hypersphere. $\xi_i > 0$ follows to satisfy the inequality constraint. With complementary slackness, it further follows that

$$\xi_i > 0 \Rightarrow \gamma_i = 0$$

and

$$\sum_k \beta_{i,k} = v_i \cdot C.$$

Solving for $\beta_{i,k'}$ gives

$$\beta_{i,k'} = v_i \cdot C - \sum_{k \neq k'} \beta_{i,k}$$

Case 3: Observation x_i lies on the hypersphere. This is the remaining case where

$$0 < \beta_{i,k'} < v_i \cdot C - \sum_{k \neq k'} \beta_{i,k}$$

It follows that $\gamma_i > 0$, $\xi_i = 0$, and consequently

$$\left\|\Phi_k(x_{i,k}) - a_k\right\|^2 = R_k^2$$

The center of the hypersphere a_k is a linear combination of all observations with $\beta_{i,k} > 0$, see Equation 5.2b. The distance of any observation u_k to the center a_k is:

$$\|u_k - a_k\|^2 = \langle u_k, u_k\rangle + 2\sum_i \beta_{i,k}\langle u_k, x_{i,k}\rangle + \underbrace{\sum_{i,j} \beta_{i,k}\beta_{j,k}\langle x_{i,k}, x_{j,k}\rangle}_{const.}$$

The third term depends only on the training data. One can cache it to speed up distance calculations, similarly to SVDD [CLL13]. The set of objects that lie on the hypersphere in S_k is HS_k Then the radius R_k can be derived from the distance of any observation $u'_k \in HS_k$.

$$R_k = \left\|u'_k - a_k\right\|$$

Thus, the distance to the decision boundary in S_k is

$$p_k(x) = \|S_k(x) - a_k\| - R_k. \tag{5.5}$$

From the distance to the decision boundary, one can derive the classification function.

Definition 6 (Subspace Classification) *Let a Subspace S_k and a hypersphere with Center a_k, and Radius R_k be given. A subspace classifier is a function*

$$f_k(x) = \begin{cases} 1 & \textit{if } p_k(x) > 0 \\ 0 & \textit{otherwise.} \end{cases}$$

We call x a *subspace outlier* in S_k if $f_k(x) = 1$ and a *subspace inlier* otherwise.

Definition 7 (Global Classification) *Let a set of subspace classifiers $\{f_1, \ldots, f_k\}$ be given. A global classifier for these subspace classifiers is a function*

$$f(x) = \begin{cases} 1 & \textit{if } \sum_k f_k(x) > 0 \\ 0 & \textit{otherwise.} \end{cases}$$

Observation x is a *global outlier* if $f(x) = 1$, i.e., x is outlier in one subspace, and *global inlier* if x is inlier in all subspaces.

5.2.2. Active Learning with Subspaces

We now present the update mechanism of SubSVDD. Recall that SubSVDD incorporates feedback on observation x_i by adjusting the weight parameter v_i. Intuitively, when the feedback on x_i is outlier (inlier), v_i is decreased (increased). v influences the trade-off between cost and radius in SubSVDD. Choosing a good update value depends on the value of C, which in turn depends on the data. We discuss how to update v values in Section 5.3.2. In this section, we focus on how SubSVDD chooses observations for feedback. We first explain why active learning with subspaces is different from conventional "non-subspace" active learning.

With *conventional active learning*, one calculates an *informativeness score* $\tau(x; p)$ for each observation based on a prediction function p, and selects observations for feedback based on the score values. Many of the existing query strategies for one-class classifiers depend on the distance of observations to the decision boundary, on local neighborhoods, or on a combination of both. For *subspace active learning* however, one cannot use these strategies directly, for two reasons. First, existing strategies rely on a single decision boundary and on one neighborhood. With SubSVDD, there are several data distributions, neighborhoods, and decision boundaries – one in each subspace. Second, feedback on an observation may only impact classifiers in some subspaces. For instance, the feedback that an observation is inlying is unlikely to affect the decision boundary in subspaces where this observation is already classified as inlier by a large margin. So, to quantify the expected impact on SubSVDD, i.e., the overall informativeness of an observation, one has to consider all subspaces.

To this end, we propose an apply-and-combine query strategy to select observations across subspaces. An informativeness score is first calculated per subspace. The overall informativeness results from combining these scores from individual subspaces. Scaling is necessary to make scores comparable across subspaces. The scores are scaled per subspace by a function g

$$\begin{aligned}\tau(x_i, p_k)_{\text{scaled}} &= g_k(\tau(x_i, p_k)) \\ &= g\left(\tau(x_i, p_k); \langle \tau(x_1, p_k), \ldots, \tau(x_N, p_k)\rangle\right),\end{aligned}$$

i.e., scaling of $\tau(x_i, p_k)$ depends on the distribution of scores in S_k. Examples for scaling functions are *min-max normalization* or *softmax*. As combination functions, one can use aggregates like the sum of scores or the maximum. A query strategy then selects observations, as follows.

Definition 8 (Subspace Query Strategy) *Let informativeness score function τ, prediction functions $p_k, \mathbb{R} \to \mathbb{R}, k \in \{1, \ldots, K\}$, scaling function g, and combination function h be given. A subspace query strategy returns the singleton Q with the maximum combined informativeness.*

$$Q = \underset{x \in \mathcal{U}}{\arg\max}\ h\left(g_1(\tau(x, p_1)), \ldots, g_K(\tau(x, p_K))\right) \tag{5.6}$$

For the informativeness per subspace, one can use any function from the literature. In this chapter, we rely on three of the strategies introduced earlier in Chapter 4:

Distance to decision boundary: An observation has high informativeness in S_k if the distance to the decision boundary is small:

$$\tau_{DB}(x, p_k) = -|p_k(x)|$$

High Confidence: An observation has high informativeness in Subspace S_k if the observation matches the inlier class the least [BBJ15].

$$\tau_{HC}(x, p_k) = p_k(x)$$

Random-Outlier: Uniform distributed informativeness for observations that are classified as outlier.

$$\tau_{rand-out}(x, p_k) = \begin{cases} \frac{1}{|\mathcal{U}|} & \text{if } p_k(x) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Figure 5.4 graphs the active learning cycle for outlier detection in subspaces, and Algorithm 1 is an overview in pseudo code. At all stages, the user has access to subspace-level information. This includes predictions and informativeness for the query in each subspace. When subspaces are 2-dimensional, this information can be visualized to assist users in providing feedback, see Section 5.1.

Design alternatives. We conclude the section with comments on two modifications of query selection in subspaces.

Committee-based methods: Query selection with Query-by-committee is popular with binary classifiers [Set12]. The idea is to compare the classifications of a set of classifiers, to select observations for feedback. The assumption is that, in a perfect case, all classifiers agree and predict the correct class. Observations where they do not agree are promising candidates for feedback. But this assumption does not hold with outlier detection, because of outlier asymmetry. In contrast to binary classification, classifications from different subspaces are *expected* to differ. Thus, query-by-committee methods are not applicable to select queries for outlier detection in subspaces.

Extensions for subspace queries: One can also think of extensions to bias the final score towards some of the subspaces, for instance with weighted aggregates. Further, one may also generalize subspace query strategies to return multiple observations, like the top-k informativeness scores, or with different weights per subspace. These generalizations could give way to ask for more complex feedback, e.g., the importance of a subspace, or to ask for feedback on batches of observations. Although these modifications are conceivable, studying them goes beyond the scope of this work.

5.2.3. Implementation Hints

There are some pitfalls when implementing one-class active learning methods.

Kernel Matrix: The one-class classifiers used in this article rely on solving a quadratic program that requires a symmetric positive semi-definite matrix as input. In practice, some parameter configurations can lead to very small negative eigenvalues in the kernel matrix. To solve this issue, we use an eigendecomposition of the kernel matrix and set these eigenvalues to zero.

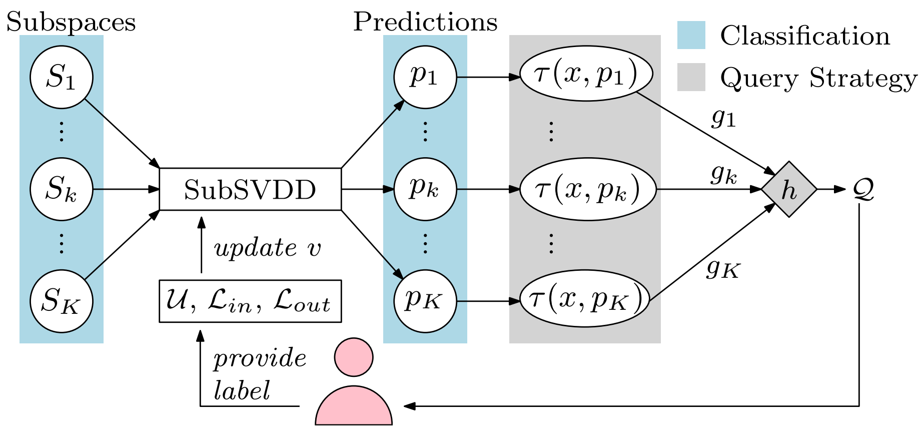


Figure 5.4.: Overview on active learning with subspaces, with Subspaces S_k, distances to decision boundary p_k, informativeness scores τ, scaling g_k and combination function h.

Numerical Comparisons: Further, an implementation requires several comparisons of real vectors, e.g., to find observations that lie on the hypersphere. Thus, one has to choose a threshold for these numerical comparisons, which we have set to 10^{-7}.

Selection of Observations on the Hypersphere: The optimal radius of support vector data description is not necessarily unique [CLL13]. Further, the numerical comparison may result in several observations that lie on the hypersphere within a threshold. In our implementation, we calculate the distance to the center for each of these observations and select the maximum as the radius.

5.3. Experiments

We have designed and conducted experiments to demonstrate the working of SubSVDD and to compare it with other approaches on benchmark data. We first describe the experiment setup and our evaluation method. We then report on experiments and discuss the benefits of subspace classifications. Finally, we turn to model parametrization.

We make our implementations as well as our benchmark setups publicly available.[2] Further, we provide pre-processing and experiment scripts, notebooks, as well as raw result files to reproduce the figures and tables.

5.3.1. Setup

In our experiments, we use publicly available benchmark data, see Section 3.4. Since our experiments cover a very broad range of parameter configurations, we downsample large data sets randomly to keep experimental runtimes reasonable. In practice, one can also use more advanced sampling methods to scale support vector data description to large data sets [Li11; Sun+16; Kra+19].

[2] `https://www.ipd.kit.edu/mitarbeiter/subsvdd/`

Algorithm 1: Active Learning with SubSVDD

```
Data      : $\mathbf{X} = \langle x_1, x_2, \ldots, x_N \rangle$, S = $\{S_1, S_2, \ldots, S_K\}$
Parameter : $C, \Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_k\}$                 // SubSVDD
            $v_{\text{in}}, v_{\text{out}}, \tau, g, h$                          // Active Learning
Output    : $\{p_1, p_2, \ldots, p_K\}$

Procedure updateWeight(model, q, $\mathcal{L}_{out}$, $\mathcal{L}_{in}$):
    if q $\in \mathcal{L}_{out}$ then
        model.$v$[q] $\leftarrow v_{\text{out}}$
    else
        model.$v$[q] $\leftarrow v_{\text{in}}$
    end

$\mathcal{L}_{\text{in}}, \mathcal{L}_{\text{out}} \leftarrow \emptyset$; $\mathcal{U} \leftarrow \{1, \ldots, N\}$; $v \leftarrow \langle 1, \ldots, 1 \rangle_N$
model $\leftarrow$ SubSVDD (X, S, C, $\Phi$, $v$)
while $\neg$terminate do
    $\{p_1, p_2, \ldots, p_K\} \leftarrow$ solveQP (model)
    q $\leftarrow \underset{x \in \mathcal{U}}{\arg\max}\ h\,(g_1(\tau(x, p_1)), \ldots, g_K(\tau(x, p_K)))$
    if askOracle (q) == "outlier" then
        $\mathcal{L}_{\text{out}} \leftarrow \mathcal{L}_{\text{out}} \cup \{q\}$
    else
        $\mathcal{L}_{\text{in}} \leftarrow \mathcal{L}_{\text{in}} \cup \{q\}$
    end
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{q\}$
    updateWeight (model, q, $\mathcal{L}_{\text{out}}$, $\mathcal{L}_{\text{in}}$)
end
```

Datasets: We use three normalized versions of each data set with an outlier ratio of 5 %. We downsample to $N = 1000$ observations for large data sets. To avoid duplicates in subspace projections, we further add random noise to each attribute sampled from $\mathcal{N}(0, 0.01)$. Although this is not required technically, we found that it reduces variance in classifier learning rates, and it eases comparisons in benchmark experiments. For SubSVDD, we further run experiments on three random subspace selections. In total, for each query strategy and data set there are three experimental runs for SVDDneg and SSAD, and nine experimental runs for SubSVDD.

Active Learning: We run experiments for 50 iterations, i.e., a fixed number of feedback queries. To measure classification quality, we rely on summary statistics of the learning curves. As before, we rely on a variant of the resubstitution error, i.e., the classification quality on the training data with unlabeled and labeled instances, cf. Chapter 4. We use the Average End Quality (AEQ) of the Matthews Correlation Coefficient (MCC), averaged over the last 5 iterations, as well as the Ratio of Outlier Queries (ROQ).

	SSAD	SubSVDD(10)	SubSVDD(20)	SVDDneg
Thyroid	**0.24**	0.20	0.18	0.20
Cardio	0.37	0.48	**0.51**	0.48
Heart	0.53	**0.74**	0.57	0.60
Page	0.51	0.57	**0.59**	0.56
Spam	0.28	0.25	0.25	**0.34**
Pima	0.34	**0.41**	0.33	0.33
Stamps	0.73	0.64	0.60	**0.82**

Table 5.1.: Median average end quality (AEQ) after 50 iterations with maximum 8-dim subspaces; best results per data set are in bold.

SSAD	SubSVDD(10)	SubSVDD(20)	SVDDneg
0.08	0.09	**0.12**	0.04
0.19	0.30	**0.35**	0.32
0.08	**0.08**	0.07	**0.08**
0.31	**0.48**	0.41	0.43
0.08	**0.13**	**0.13**	0.12
0.08	**0.10**	0.08	0.06
0.19	0.18	0.20	**0.22**

Table 5.2.: Median Ratio of Outlier Queries (ROQ) after 50 iterations with maximum 8-dim subspaces; best results per data set are in bold.

5.3.2. Parametrization

We now discuss ways to select good values for hyper-parameters C, the kernel mapping, and v, as well as on the input set of subspaces.

The *kernel mapping* influences the smoothness of the decision boundary. Here, we use the radial basis function kernel $K(x, x') = \exp(-\gamma \, \|x - x'\|^2)$. Based on our findings from Chapter 4, we use self-adaptive data shifting [Wan+18], a method based on artificial data generation, to select the kernel bandwidth γ. Specifically, we choose a γ_k for each Subspace S_k individually to account for different dimensionalities and data distributions in subspaces. For experiments based on subspaces with more than four dimensions, we use a search range of $[10^{-2}, 10^2]$. For experiments based on subspaces with four or less dimensions, and for the v comparison, we use a search range of $[0.1, 20]$.

The *trade-off parameter* C bounds the share of observations that are classified as outliers. A good choice of C depends on several factors, such as an assumption on the expected share of outliers in the data. In preliminary experiments of ours, we have observed that the choice of C may also depend on the active learning strategy.

All this makes selecting a good value for C difficult, not only for SubSVDD, but also for its competitors. In our experiments, we select $C = 0.45$ as a default. This choice has worked well with τ_{DB} and τ_{HC} in our preliminary experiments. However, we have found SubSVDD to also work well with suboptimal choices of C in many cases. We hypothesize

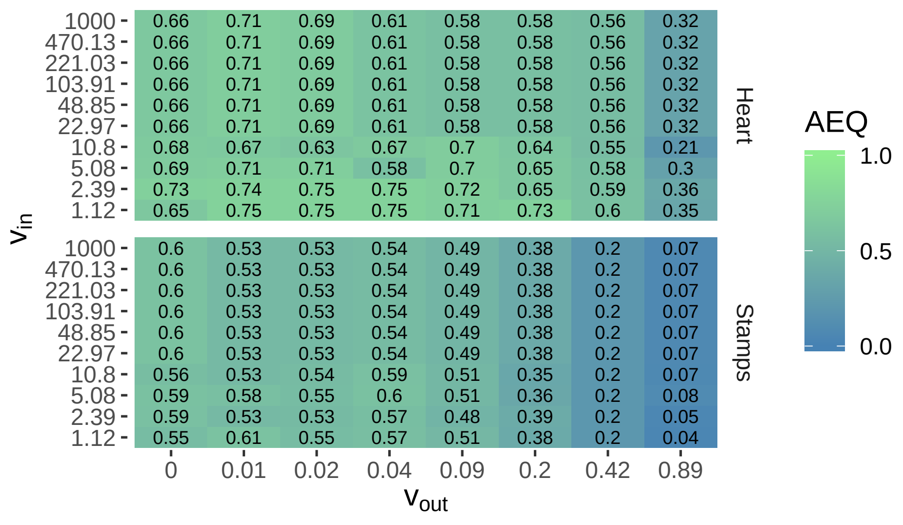


Figure 5.5.: Median average end quality for different combinations of v_{in} and v_{out} on Heart and Stamps.

that this is because the learning weights v compensate an initially bad choice of C over time.

SubSVDD further introduces a *weight parameter* v, and we must decide how to update its value based on user feedback. Recall that decreasing v reduces the cost of excluding an observation from the hypersphere, and vice-versa. Large changes to v have a stronger effect on the model, but may also lead to overfitting.

We initialize our model with $v = 1$ and use two update strategies depending on the feedback: We set v to v_{out} when the feedback is outlier and to v_{in} when it is inlier. We have experimented with different settings of v_{in} and v_{out} to evaluate the robustness of SubSVDD. Figure 5.5 shows the median average end quality on Heart and Stamps for different choices of v_{in} and v_{out}. Only values of v_{out} larger than 0.1 make classification much worse. Based on this, we have set $v_{in} = 10$ and $v_{out} = 0.01$.

SubSVDD further requires a *set of subspaces* as an input. The achievable outlier detection quality depends on it. This is intuitive: Any outlier detection method that relies on subspaces cannot detect outliers that only occur in attribute combinations that are not part of any subspace. In general, one can use any subspace search method as a preprocessing step to SubSVDD. However, its output depends on several aspects, such as data set characteristics and further hyper-parameter settings. Controlling for these factors is difficult and leads to an unreasonable complexity of any evaluation. We have decided to use random sampling of subspaces, a common lower baseline to subspace selection.

We have found our parameter choices to work well with a variety of data sets. Nevertheless, one could reconsider these choices based on the feedback obtained during active learning. This would even give way to use supervised parameter tuning, e.g., through cross-validation. However, we expect these optimizations to be application specific, and we do not consider them in our evaluation.

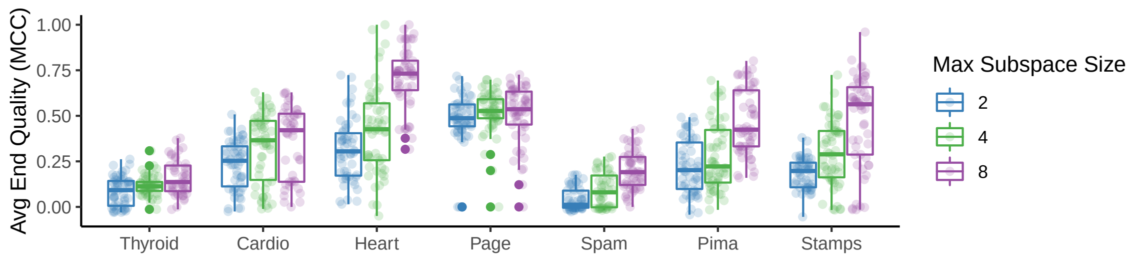


Figure 5.6.: Comparison of AEQ for different subspace sizes.

Competitors: SVDDneg and SSAD require to set the kernel function and the cost parameter C. We use the radial basis function kernel and use self-adaptive data shifting to select the kernel bandwidth [Wan⁺18] within $[10^{-2}, 10^{2}]$. To select C, we use the upper bound estimation from [TD04] with the true proportion of outliers in the data set. SSAD further requires to set an additional trade-off parameter κ, but there is no rule how to choose a good value. We set $\kappa = 0.1$ based on earlier benchmark results, see Chapter 4.

5.3.3. Benchmark Results

In a competitive benchmark, we compare SubSVDD to SVDDneg and SSAD. For SubSVDD, we differentiate based on the number of subspaces used (10 and 20), and the maximum dimensionality of the subspaces in the set (2-dim, 4-dim and 8-dim). For query strategies, we use τ_{DB}, τ_{HC}, and $\tau_{rand-out}$, and use *sum* and *min-max-normalization* with the subspace query strategies.

Table 5.1 shows the median AEQ over τ_{DB} and τ_{HC} for different data sets, and Table 5.2 shows the ROQ. In both summaries, SubSVDD outperforms its competitors on several data sets. Changing the query strategy to $\tau_{rand-out}$ improves the results on Heart and Pima significantly for SubSVDD and SVDDneg, but bogs them down on most of the remaining data sets.

Comparing different subspace sizes yields a more differentiated view on SubSVDD, see Figure 5.6. On most data sets, restricting the maximum subspace dimensionality reduces classification quality. This is expected, since selecting all relevant attribute combinations is more difficult if subspaces are small. However, this also indicates a trade-off between interpretability and classification quality. On the one hand, SubSVDD yields high classification accuracy in large subspaces. On the other hand, it is more difficult for the user to provide feedback in such subspaces. This in turn makes it difficult in practice to achieve the quality observed with large subspaces in our current experiments. By restricting subspaces to two dimensions, a system can visualize classification results, to help users to give feedback. But the classification quality with this restriction tends to be lower than without it. In line with our initial assumption, we deem the lower classification quality a realistic figure.

5.4. Further Related Work

This chapter has already established connections to some related work. Several less directly related publications remain, as follows.

Interpretability and Explanations: There are generic approaches to explain queries in interactive learning tasks [TK18; PCF18] with a focus on model-agnostic explanations. Their applicability to one-class active learning has not been studied yet. Other approaches refer users to external resources, e.g., additional data bases [BCB17; Qi⁺18]. Such explanations are very application-specific. They require availability of external data and assume that this additional information is indeed useful to interpret classification results. Others have proposed to provide additional diagnostic information on one-class classification results [MRW14]. However, their focus is not on explanation, but on finding a good threshold to transform a continuous scoring function into binary classifications.

There is one approach for outlier detection with active learning to detect micro clusters in subspaces [PS11]. It uses active learning to find out which anomalies a user is interested in, and not to update an underlying detection model – an objective similar to [Das⁺16; Sid⁺18].

Next, there are concepts to increase interpretability of unsupervised outlier detection beyond subspaces. For instance, significant work has attempted to make outlier scores comparable and interpretable [Kri⁺11] and to quantify the influence of attributes on score values [PM15]. These methods are tailored towards outlier scores and hence not applicable in our case.

SVDD Modifications: SVDD has been applied with *single* low-dimensional projections [Soh⁺18; GJ17]. This is similar to using dimensionality reduction as a preprocessing step. As explained earlier, outliers may only occur in specific attribute combinations, i.e., they are likely to be hidden in a single projection. So conventional dimensionality reduction is not an alternative to subspace outlier detection.

There also are proposals to combine several SVDD classifiers as an unsupervised ensemble [CT11], or ones that target at multi-class classification [KW12]. Both are not applicable to one-class active learning, since they do not address outlier asymmetry.

Another modification is multi-sphere SVDD. The idea is to partition observations into groups and to train several hyperspheres in the full space [Le⁺10; LSF14]. Although they also learn several decision boundaries, these boundaries all are the in the same, high-dimensional space. So this is orthogonal to subspace methods.

5.5. Summary

Comprehensiveness and interpretability are important to facilitate feedback from human annotators. Current approaches on one-class active learning for outlier detection do not address this issue. Instead, they assume that users can provide feedback, regardless of how results are presented to them.

In this chapter, we rely on a more realistic assumption: Users can give educated feedback on observations whose contexts, i.e., subspaces where they are outlying, are low-dimensional. But this ability decreases with increasing dimensionality of the contexts.

To facilitate feedback with low-dimensional contexts, we introduce SubSVDD, a novel semi-supervised active learning method to detect outliers in multiple subspaces. SubSVDD yields concise result descriptions, i.e., a set of projections that explain each outlier. Further, SubSVDD allows to trade between the effort users spend on interpreting results and classification quality. Comprehensive experiments demonstrate the effectiveness of our approach.

6. Active Learning of Hyperparameter Values

In the previous chapters, we have presented various active learning experiments based on several one-class classifiers. There is one important aspect to these experiments that we have given only little attention so far: the selection of suitable hyperparameter values of one-class classifiers. Until now, whenever applicable, we have compared different existing heuristics to select hyperparameters for SVDD, see Chapter 4. In other cases, we have evaluated the influence of hyperparameter values experimentally, like v on SubSVDD and κ for SSAD.

The selection of good hyperparameter values is important, and can have significant impact on the classification quality, see Section 4.3.3. In this chapter, we therefore take a closer look at the selection of hyperparameter values. More specifically, we strive for a hyperparameter selection method that is easy to apply, even with non-machine-learning experts. We focus on SVDD with the Gaussian kernel, since it is the most widely used one-class classifiers for outlier detection.[1] Recall that SVDD requires to specify two hyperparameter values: a kernel function to allow for non-linear decision boundaries and the cost trade-off C that regulates the share of observations that fall outside the hypersphere. With SVDD, the predominant choice is the Gaussian kernel, which is parameterized by γ. In this case, an optimal γ reflects the actual complexity of the data, and an optimal C excludes the true share of outliers from the hypersphere. However, finding out the true complexity and outlier share is challenging, which makes choosing good hyperparameter values difficult. Moreover, SVDD is sensitive to changes in hyperparameter values [Gha⁺18]. It easily over- or underfits the data, which in turn can deteriorate classification quality significantly.

SVDD is usually applied in an unsupervised setting, i.e., the selection of hyperparameter values cannot rely on class label information. There is a great variety of heuristics for hyperparameter estimation that use data characteristics [Kha⁺11; Gha⁺18], synthetic data generation [Wan⁺18; BKB07], and properties of the fitted SVDD [ABN18; Kak⁺17] to select a good γ. However, these heuristics do not come with any validation measures or formal guarantees, making it difficult to validate if estimated hyperparameters are indeed a good fit. Moreover, selecting a suitable heuristic is difficult in the first place, since the intuition of different heuristics may be equally plausible. This leaves the user with the cumbersome and difficult task of validating the choice of the heuristic and the estimated hyperparameter values. Therefore, we strive for a principled method for SVDD

[1] The remainder of this chapter bases on the article [TBA19] *Holger Trittenbach, Klemens Böhm, and Ira Assent. "Active Learning of SVDD Hyperparameter Values". In:* arXiv *(2019). arXiv:* `1912.01927`. It has been shortened to be less repetitive. It contains minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

hyperparameter estimation. To do away with purely heuristic methods, our idea bases on active learning, i.e., asking users to provide class labels for a few observations that provide grounding for the estimation.

Developing an active learning method for selecting SVDD hyperparameter values is challenging. On the one hand, labels give way to using supervised methods for selecting kernel parameters, such as kernel alignment [Cri+02]. However, a reliable and stable alignment calculation requires a sufficient number of labeled observations [ARM12]. With active learning, there are only very few labels available, in particular during the first iterations. Next, current kernel alignment assumes that observations from the same class are similar to each other. This assumption may not hold with outlier detection, since outliers are rare, do not have a joint distribution, and may be dissimilar to each other. So kernel alignment is not applicable without further ado; Section 6.2.2 illustrates this. A further challenge is that most conventional active learning strategies are not applicable since they rely on an already parameterized classifier (see Chapter 4), or focus on fine-tuning of an already parameterized classifier [Gha+11b]. However, an active learning strategy to estimate hyperparameter values should select observations that are informative of the full data distribution.

In this chapter, we propose Local Active Min-Max Alignment (LAMA), an active learning method to select both SVDD hyperparameters γ and C. To our knowledge, this is the first active learning method to estimate hyperparameters of SVDD. It is a principled, evidence-based method and yields a quality score based on the actual class labels obtained by active learning. This is a key advantage over existing heuristics: LAMA does not require manual validation, since its estimations base on labeled observations.

For γ, we address the challenges in two steps. First, we propose *locally optimal alignment*, an adapted kernel alignment method based on local neighborhoods. It confines the calculation to regions where class labels are available. Second, we propose a novel active learning strategy to explore regions of the data space where class labels are likely to contribute to a reliable alignment calculation. Estimating γ is efficient and widely applicable, since it solely relies on the kernel matrix, and not on any specific model, such as SVDD. For C, we propose a scheme to estimate a feasible lower and upper bound, and then use a grid search to estimate its value. Empirically, LAMA outperforms state-of-the-art heuristics QMS [Gha+18], DFN [Xia+14] and ADS [Wan+18] in extensive experiments on real world data. On several data sets, LAMA even yields results close to the empirical upper bound.

6.1. Related Work

Both SVDD hyperparameters depend on each other, i.e., a good value for C depends on the choice of the kernel. C influences the share of observations that are classified as outlier, and a good value depends on the specific application. Literature has produced several heuristics to select an appropriate γ, but the choice of C is often left to the user. In some cases, the heuristics to select a kernel even require users to initialize C – a requirement unlikely to be met in practice.

The bulk of methods we present in this section focuses on selecting γ. There are three types of heuristics. The first type is *data-based* selection, which solely relies on data characteristics to estimate γ, often in a closed formula. The second type is *model-based* selection, which optimizes for criteria based on the trained model, and thus requires solving SVDD, often multiple times. The third type of selection heuristics generates *synthetic data* in combination with supervised selection schemes to fit a decision boundary. In rare cases, when labeled training data is available, one can use plain *supervised selection*, e.g., by using cross validation.

Data-based Selection

The simplest data-based estimation methods are formulas to directly calculate γ. There are two rules of thumb by Scott [Sco15] and by Silverman [Sil18]. They use the number of observations and lower-order statistics to calculate γ in a closed formula. Others propose to estimate γ by using the distances between the centers of the outlier and inlier class [Kha+11]. Recent approaches use changes in the neighborhood density of training observations to derive closed formulas for γ and C [Gha+18; Gha+16].

A different approach is to define desired properties of the kernel matrix, and optimize for them by modifying the kernel parameter. Several such objectives have been proposed: to maximize the coefficient of variance of non-diagonal kernel matrix entries [EES07], to ensure that the kernel matrix is different from the identity matrix [Cha+17], and to maximize the difference between distances to the nearest and to the farthest neighbors of training observations [Xia+14].

Model-based Selection

Changes in hyperparameter values modify the optimal solution of the SVDD optimization problem, and the properties of this solution. Model-based selection strategies fit SVDD for several γ values, and select the model which has the desired properties. A common approach is to define desired geometric properties of the decision boundary. For instance, one can define criteria on the tightness of a decision boundary, e.g., by estimating whether the decision function is a boundary on a convex set [Xia+14]. A good kernel parameter leads to a decision boundary that is neither too tight nor too loose. Variants of this approach are to first detect edge points of the data sample [XWX14; AKW18]. Intuitively, interior points should be far from the decision boundary, and edge points close to the decision boundary. Thus, one can maximize the difference between the maximum distance of an interior point to the decision boundary and the maximum distance of an edge-point to the decision boundary to balance between tight and loose boundaries.

Others have suggested optimization criteria based on the number of support vectors, i.e., the observations that define the decision boundary. The number of support vectors tends to increase with more complex decision boundaries. So one can search for the smallest γ such that the number of support vectors are the lower bound imposed by C [GK11b]. A variant of this idea is to decrease γ until all support vectors are edge points [ABN18]. A different approach is to select the kernel parameter by training SVDD on multiple resamples of the data and then select the γ that results in the smallest average number of support vectors over all samples [BBD06].

One can also derive objectives directly from the dual objective function of SVDD. For instance, empirical observations suggest that one can set the second derivative of the dual objective with respect to the kernel parameter to zero to obtain a parameter estimate [Kak+17; PKC17]. Also combinations of support vector count and objective function maximization have been proposed as objectives [Wan+13].

Selection with Synthetic Data

The core idea of parameter tuning by synthetic data generation is to enhance the training data with labeled artificial observations. One can then apply supervised parameter tuning, such as grid search and cross-validation, to select parameters that fit best the artificially generated data set. A benefit is that many of the synthetic data generation methods also provide an estimate for C. However, the success of these methods depends on how well the artificial observations are placed, and whether this placement works well for the data at hand is unclear. A poor placement can yield parameter values that have very poor classification quality, see Section 6.3.

The basic variants generate outliers either uniformly [TD01] or from a skewed distribution [DX07] and estimate the false negative rate for the outliers generated. To generate outliers more reliably in high-dimensional data, there are adaptations that decompose this problem into first detecting the edge points of the sample and then generating the artificial outliers based on them [Wan+18; BKB07].

Supervised Selection

If labeled training data is available, one can use supervised hyperparameter tuning [TM04; TLD05; TD13]. However, these methods are not relevant for this work since with active learning, there initially is no labeled training data available.

To conclude, there is a plethora of heuristics available to set the hyperparameter values of SVDD. However, selecting a suitable heuristic is difficult for several reasons. For one, there is no objective criterion to compare heuristics. They do not come with any formal guarantees on their result quality, but offer different intuitions on SVDD, and on motivations for particular estimation strategies. Respective articles generally do not discuss the conditions under which the heuristics work well. Next, existing experimental evaluations comprise only a few of the heuristics, and in many cases only a very limited body of benchmark data. A further important downside of many existing heuristics is that they require to set C manually. This makes both a competitive comparison and the application in practice difficult.

6.2. LAMA

In this section, we propose an active learning method to learn hyperparameters values of SVDD. We first present some preliminaries on kernel learning. Then we focus on cases when only a few labeled observations are available. We then present a query strategy to identify observations that are most informative for learning the kernel parameter value. Finally, we propose a strategy to estimate the cost parameter C based on the set of labels acquired by active learning.

6.2.1. Kernel Learning Fundamentals

Kernel learning methods construct a kernel matrix or a kernel function from labeled training data, or from pairwise constraints. The idea is to identify a good kernel given the training data, independent of the classifier. There are multiple approaches to learn a kernel, e.g., by directly learning the Kernel Matrix (Non-Parametric Kernel Learning) [ZTH11] or to learn an optimal combination of multiple kernels, which may differ by type or by parameterization [GA11].

With SVDD, the Gaussian kernel is the predominant choice for the kernel. The Gaussian kernel is parametric, which gives way to learning good parameter values by so-called kernel alignment [Cri+02]. The idea of kernel alignment is to define an ideal kernel matrix

$$K_{\text{opt}} = yy^{\intercal} \tag{6.1}$$

using class labels y. The entries of K_{opt} are $+1$ if observations have the same class label, and -1 otherwise. The alignment between an empirical and ideal kernel matrix is

$$A(K_\gamma, K_{\text{opt}}) = \frac{\langle K_\gamma, K_{\text{opt}} \rangle_F}{\sqrt{\langle K_\gamma, K_\gamma \rangle_F \langle K_{\text{opt}}, K_{\text{opt}} \rangle_F}} \tag{6.2}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product. Kernel alignment has some desirable theoretical properties [WZT15]: it is *computationally efficient*, i.e., the computation only depends on the number of labeled observations $O(|\mathcal{L}|^2)$; it is *concentrated* around its expected value, i.e., the empirical alignment deviates only slightly from to the true alignment value; it *generalizes* well to a test set.

Kernel alignment is useful for finding a good kernel parameter. By using the kernel alignment as an objective, one can search for an optimum [WZT15]

$$\gamma_{\text{opt}} = \underset{\gamma}{\arg\max}\ A(K_\gamma, K_{\text{opt}}). \tag{6.3}$$

With outlier detection, calculating the alignment is more difficult, since the class distributions are highly imbalanced. In this case, the sensitivity of the alignment measure may drop [WZT15]. One remedy is to adjust y by the relative class frequency [KSC02]. Another method to deal with unbalanced classes is to center the kernel matrix [CMR12]. Preliminary experiments indicate that relative class frequency adjustment does not improve the alignment calculation in our setting. We therefore rely on kernel matrix centering.

6.2.2. Alignment on Small Samples

One difficulty of kernel alignment is that it generally requires a large set of labeled examples to define the ideal kernel matrix [ARM12]. However, with active learning, only very few labels are available, in particular during the first few iterations. A second difficulty is that user labels may be noisy, i.e., the actual label may differ from the user-provided label. A reason is that labeling is a subjective assessment, and that users may misjudge and provide a wrong label. In general, this issue may be negligible, in particular when feedback is correct in most of the cases. However, noisy labels may impact the kernel alignment significantly when the amount of labeled data is small.

In the following, we propose a method that creates a local alignment to mitigate both of these difficulties. The idea is to include the local neighborhood of labeled observations in the alignment calculation. Our method consists of two steps. In the first step, we re-label observations based on a majority vote of the labels in their local neighborhood. The reason for this is two-fold. On the one hand, this step reduces the influence of noisy labels. On the other hand, this creates pseudo labels for observations in $\mathcal{U}$ and increases the number of observations for the alignment calculation. In the second step, we define a locally optimal kernel matrix for the alignment. That is, we limit the comparison between K_γ and K_{opt} to the relevant entries.

Preliminaries

We first introduce some useful definitions.

> **Definition 9 (Nearest Neighbors)** *$NN_k(x)$ are the k closest observations of an observation x. We set $NN_1(x) = \{x\}$.*

> **Definition 10 (Reverse Nearest Neighbors)** *$RNN_k(x)$ is the set of observations that have x as one of their k-nearest neighbors.*
>
> $$RNN_k(x) = \{l \mid x \in NN_k(l)\} \tag{6.4}$$

> **Definition 11 (Symmetric Nearest Neighbors)** *$SNN_k(x)$ is the set of observations that are k-nearest neighbors of x as well as reverse nearest neighbors of x.*
>
> $$SNN_k(x) = \{l \in NN_k(x) \mid x \in NN_k(l)\} \tag{6.5}$$

Relabeling

We propose to relabel observations based on their local neighborhood to increase the number of labeled observations, and to reduce the influence of noisy labels. More specifically, when a user labels an observation x_i, this label is propagated to the local neighborhood of x_i. We propose an asymmetric propagation scheme. When x_i is inlier, the label propagates to the k-nearest neighbors of x_i. So the nearest neighbors of an inlier are deemed inliers as well. When x_i is outlier, the label propagates to the symmetric nearest neighbors of x_i. The rationale behind this propagation scheme is that the nearest neighbor of an outlier may well be an inlier – this holds with certainty if there is only one outlier in the data space. But the nearest neighbors of inliers are likely to also be inliers. So asymmetric propagation mitigates wrong label propagation.

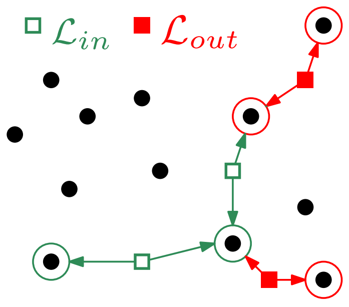


Figure 6.1.: Relabeling with local neighborhoods. The arrows indicate the propagation of class labels to NN_2 (green) and SNN_2 (red) neighborhoods of the labeled observations.

After relabeling, one can count how often x occurs as a reverse k-nearest neighbor of labeled inliers, i.e.,

$$n_{\text{in}}(x) = \sum_{l \in \mathcal{L}_{\text{in}}} \mathbb{1}_{\mathrm{NN}_k(l)}(x). \tag{6.6}$$

Analogously, the number of times x occurs in symmetric nearest neighbors of outliers is

$$n_{\text{out}}(x) = \sum_{l \in \mathcal{L}_{\text{out}}} \mathbb{1}_{\mathrm{SNN}_k(l)}(x). \tag{6.7}$$

Based on these counts, neighborhoods are relabeled based on a majority vote. The relabeled pools are

$$\begin{aligned} \mathcal{L}'_{\text{in}} &= \{x \mid \frac{n_{\text{in}}(x)}{n_{\text{in}}(x) + n_{\text{out}}(x)} > 0.5\}, \text{and} \\ \mathcal{L}'_{\text{out}} &= \{x \mid 0 < \frac{n_{\text{in}}(x)}{n_{\text{in}}(x) + n_{\text{out}}(x)} \leq 0.5\}. \end{aligned} \tag{6.8}$$

The set $\mathcal{U}'$ contains the remaining observations, i.e., the ones that do not occur in neighborhoods of labeled observations. Figure 6.1 illustrates the relabeling.

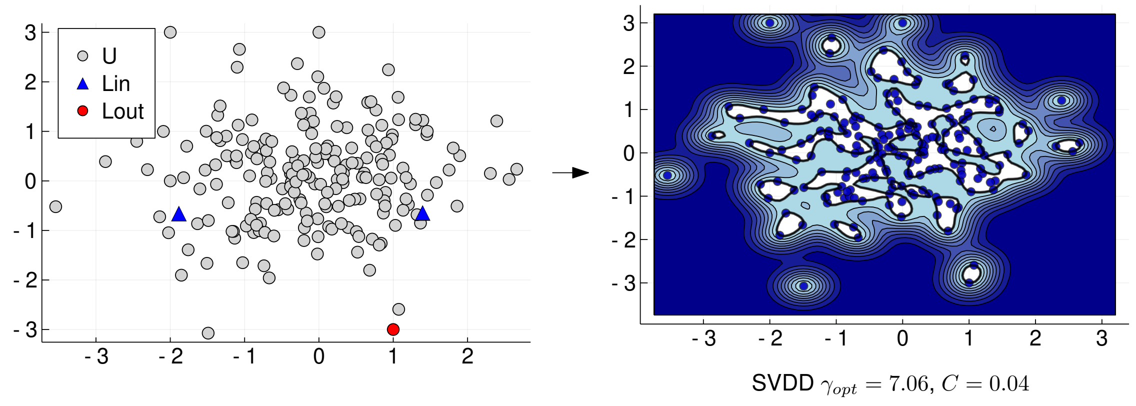
The optimal kernel matrix based on relabeled observations is

$$K'_{\text{opt}} = y'(y')^\intercal, \tag{6.9}$$

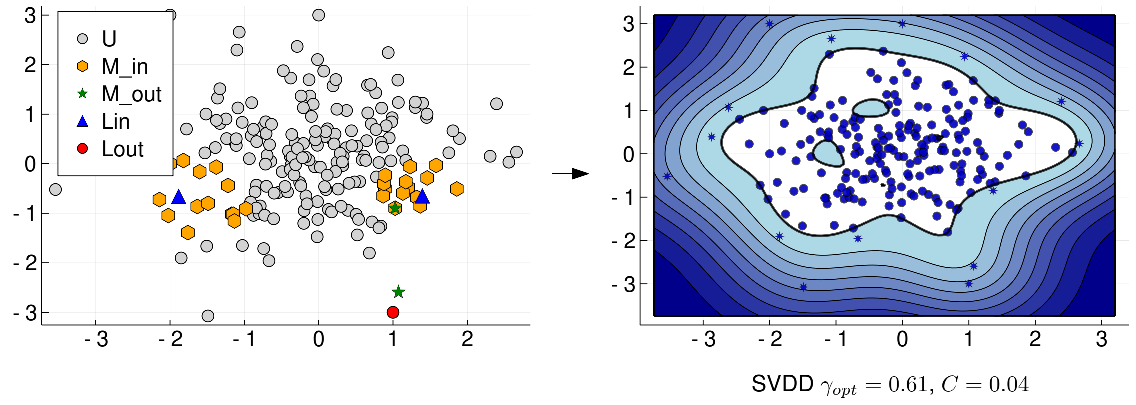
where y' is the label vector after relabeling, cf. Equation 6.1.

Locally Optimal Alignment

The global kernel alignment relies on all entries of the kernel matrix, see Equation 6.2. This is problematic because, when the sample size is small and biased towards some area of the data space, γ_{opt} may be far off the true optimum. Figure 6.2 illustrates this issue on data sampled from a Gauss distribution with two labeled inliers and one labeled outlier. In Figure 6.2a, the alignment is “global”, i.e., does not rely on neighborhood information. It results in a large value for γ_{opt} and causes the SVDD classifier to overfit. In Figure 6.2b,



(a) Global alignment γ_{opt} based on L_{in} and L_{out}.



(b) Locally optimal alignment with $k = 15$.

Figure 6.2.: Comparison of global and local alignment and fitted SVDD with $|\mathcal{L}_{\text{in}}| = 2$ and $|\mathcal{L}_{\text{out}}| = 1$.

the alignment is "local", i.e., includes the local neighborhood of labeled observations in the alignment calculation. The result is a small γ_{opt} – a good choice for the data.

We now explain how to calculate the alignment on a subset of the kernel matrix entries. In general, an inlier should be similar to its nearest neighbors. However, inliers may not be similar to all other inliers. If there only are two distant observations x_i and x_j with $x_i, x_j \in \mathcal{L}_{\text{in}}$, a global kernel alignment would result in a large γ_{opt}, such that $k(x_i, x_j)$ is close to 1. In this case, γ_{opt} overfits to the labeled observations. To avoid this issue, we only expect inliers to be similar to their nearest neighbors that are also labeled as inliers. Next, inliers should be dissimilar to nearest neighbors that are labeled as outliers. Formally, this means to select the kernel matrix entries

$$M_{\text{in}} = \{(i, j) \,|\, i \in \mathcal{L}_{\text{in}},\ j \in \mathcal{L}' \cap \text{NN}_k(i)\} \tag{6.10}$$

With outliers, one cannot assume similarity to their nearest neighbors, since the nearest neighbor of an outlier may often be an inlier. Thus, we assume that outliers are similar only to their symmetric nearest neighbors. Further, outliers should be dissimilar to the nearest inliers that are not their reverse nearest neighbors. Formally, this means to select the kernel matrix entries

$$M_{\text{out}} = \{(i,j) \mid i \in \mathcal{L}_{\text{out}},\ j \in \left(\mathcal{L}'_{\text{out}} \cap \text{SNN}_k(i)\right) \\ \cup \left(\mathcal{L}'_{\text{in}} \cap \text{NN}_k(i) \setminus \text{RNN}_k(i)\right)\} \tag{6.11}$$

Figure 6.2b highlights M_{in} and M_{out}. To calculate an alignment on these subsets, we set the remaining kernel matrix entries to 0, i.e., they do not have any impact on the alignment calculation.

$$K'_{\text{opt}}(i,j) \leftarrow 0, \quad \forall\,(i,j) \notin M_{\text{in}} \cup M_{\text{out}} \tag{6.12}$$

$$K_\gamma(i,j) \leftarrow 0, \quad \forall\,(i,j) \notin M_{\text{in}} \cup M_{\text{out}} \tag{6.13}$$

We denote the alignment on this subset as

$$a_{\text{local}} := A(K_\gamma, K'_{\text{opt}}). \tag{6.14}$$

6.2.3. Query Strategy

Active learning in combination with kernel learning has only been studied for non-parametric kernels [HJ08]. Conventional active learning methods are also not useful for learning hyperparameter values. Most conventional query strategies are not applicable since they rely on already parameterized classifiers, see Chapter 4. Other query strategies rely only on data characteristics and select observations in the margin between classes [Gha+11b], i.e., they select border cases for fine-tuning an already parameterized classifier, which tend to not be representative of the underlying distribution. Hyperparameter estimation requires observations that are informative of both classes and of the data distribution. To our knowledge, there currently is no query strategy with the objective to estimate hyperparameter values.

In our scenario, an observation is informative if its label contributes towards finding γ_{opt}. Intuitively, these are the observations that fit least to the current alignment, and thus lead to large changes. The rationale is that this query strategy results is explorative at first, which leads to large changes in the alignment. Over time, the changes become smaller, and the parameter estimation more stable. Thus, we propose to estimate informativeness of an instance by calculating how much the alignment changes when the label for a yet unlabeled instance would become available.

Min-Max Alignment Query Strategy Given a current γ_{opt}, and the respective alignment a_{local} both derived by Equation 6.14, for each potential query $x \in \mathcal{U}$, there are two cases. If x is an inlier, the updated sets are $\mathcal{L}''_{\text{in}} = \mathcal{L}'_{\text{in}} \cup \{x\}$, otherwise $\mathcal{L}''_{\text{out}} = \mathcal{L}'_{\text{out}} \cup \{x\}$. One must then update M_{in} and M_{out} respectively to calculate an updated alignment. If x is inlier, the updated alignment is $a^{\text{in}}_{\text{local}}$, otherwise it is $a^{\text{out}}_{\text{local}}$. We define the informativeness as the minimum change in the alignment over both cases

$$\tau_{\text{MMA}}(x) = \min\{|a_{\text{local}} - a^{\text{in}}_{\text{local}}|, |a_{\text{local}} - a^{\text{out}}_{\text{local}}|\}. \tag{6.15}$$

So q is the unlabeled observation where τ_{MMA} is maximal. Algorithm 2 is an overview of our proposed active learning method to estimate the kernel parameter.

Algorithm 2: Active Learning of Kernel Parameter

```
Data       : $\mathcal{X} = \langle x_1, x_2, \dots, x_N \rangle$
Parameter  : $k$
Output     : $\gamma_{\text{opt}}$
Function $\tau(x; a, \mathcal{L}'_{in}, \mathcal{L}'_{out})$:
    $L''_{\text{in}} \leftarrow L'_{\text{in}} \cup \{x\}$
    $L''_{\text{out}} \leftarrow L'_{\text{out}} \cup \{x\}$
    calculate $a^{\text{in}}_{\text{local}}, a^{\text{out}}_{\text{local}}$              // Equation 6.14
    return $\min(|a_{\text{local}} - a^{\text{in}}_{\text{local}}|, |a_{\text{local}} - a^{\text{out}}_{\text{local}}|)$    // Equation 6.15
$\mathcal{L}_{\text{in}}, \mathcal{L}_{\text{out}} \leftarrow$ drawInitialSample
$\mathcal{U} \leftarrow \mathcal{X} \setminus \mathcal{L}_{\text{in}} \cup \mathcal{L}_{\text{out}}$
while $\neg$terminate do
    $\mathcal{L}'_{\text{in}}, \mathcal{L}'_{\text{out}} \leftarrow$ relabel $(\mathcal{L}_{\text{in}}, \mathcal{L}_{\text{out}})$        // Equation 6.8
    calculate $M_{\text{in}}, M_{\text{out}}$                  // Equation 6.10, Equation 6.11
    calculate $K'_{\text{opt}}$                       // Equation 6.9, Equation 6.12
    calculate $\gamma_{\text{opt}}$                       // Equation 6.3, Equation 6.14
    // Min-Max Alignment Strategy
    $a_{\text{local}} \leftarrow A(K_{\gamma_{\text{opt}}}, K'_{\text{opt}})$
    $s \leftarrow 0$
    for $x \in \mathcal{U}$ do
        $s' \leftarrow \tau(x; a_{\text{local}}, \mathcal{L}'_{\text{in}}, \mathcal{L}'_{\text{out}})$
        if $s' > s$ then
            q $\leftarrow x$
            $s \leftarrow s'$
        end
    end
    // Update pools
    if askOracle (q) == "outlier" then
        $\mathcal{L}_{\text{out}} \leftarrow \mathcal{L}_{\text{out}} \cup \{q\}$
    else
        $\mathcal{L}_{\text{in}} \leftarrow \mathcal{L}_{\text{in}} \cup \{q\}$
    end
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{q\}$
end
return $\gamma_{\text{opt}}$
```

For efficiency, we calculate τ_{MMA} on a candidate subset $\mathcal{S} \subseteq \mathcal{U}$ with sample size $|\mathcal{S}|$, which we select randomly in each iteration. In our experiments, we have found a sample of size $|\mathcal{S}| = 100$ to work well.

Dataset	LAMA	LAMA-Sample	DFN-Fix	DFN-Sample	QMS	ADS-default	ADS-ext	Emp. UB
Annthyroid	0.02	**0.03**	0.00	-0.00	0.00	0.00	0.00	0.04
Cardio	**0.25**	0.23	0.00	0.22	–	0.00	0.00	0.24
Glass	**0.15**	0.09	0.04	0.03	–	0.00	0.00	0.25
Heart	**0.13**	0.10	0.00	0.10	0.00	0.00	-0.02	0.13
Hepatitis	0.05	0.15	0.00	**0.16**	0.05	0.08	0.05	0.21
Ionosphere	**0.66**	**0.66**	0.00	0.55	–	0.59	0.00	0.78
Lymph	0.47	0.41	0.47	0.39	–	**0.48**	**0.48**	0.51
PageBlocks	**0.42**	0.35	0.10	**0.42**	0.00	0.00	0.00	0.52
Pima	0.08	**0.16**	0.02	0.07	0.00	0.00	0.00	0.14
Shuttle	0.06	**0.19**	0.13	0.10	0.00	0.00	0.00	0.31
SpamBase	**0.01**	-0.01	0.00	0.01	0.00	0.00	0.00	0.04
Stamps	**0.18**	0.17	0.08	**0.18**	0.00	0.00	0.01	0.21
WBC	**0.53**	0.50	**0.53**	0.46	0.00	0.00	0.45	0.59
WDBC	**0.38**	0.31	0.34	0.15	0.00	0.00	-0.01	0.45
WPBC	0.01	**0.04**	-0.03	0.01	0.00	-0.02	-0.05	0.08
Wave	**0.05**	0.04	-0.00	0.02	0.00	0.00	0.04	0.11

Table 6.1.: Result on real world benchmark data; average kappa coefficient over five repetitions; best per data set in bold.

6.2.4. Estimating Cost Parameter C

Active learning results in a ground truth of size $|\mathcal{L}| = k$ after k iterations. The sample obtained through Min-Max Alignment gives way to a grid search for C, as follows. First, there is a lower bound C_{LB} and an upper bound C_{UB} on the feasible region of C. Recall that, with decreasing C, more observations may fall outside of the hypersphere. To obtain C_{LB}, we use binary search for the smallest C where the SVDD optimization problem still has a feasible solution. To obtain C_{UB}, we search for the smallest C where all observations, regardless of their label, are classified as inlier. We then use a grid search to find C_{opt}. We train several classifiers in $[C_{\text{LB}}, C_{\text{UB}}]$ and compare their classification accuracy on $\mathcal{L}$ based on a suitable metric, e.g., Cohen's Kappa. This is the *quality score* that assesses the current parameter estimates. C_{opt} is the value that yields the highest score.

6.3. Experiments

We evaluate our method on an established set of benchmark data for outlier detection, see Section 3.4. Large data sets are sub-sampled to $N = 2000$. Our implementations, raw results, and notebooks to reproduce our results are publicly available.[2] We evaluate our active learning approach against several state-of-the-art methods to estimate SVDD hyperparameter values, and compare against a random baseline and an empirical upper bound. We repeat each experiment five times and report the average results unless stated differently.

Active Learning. As an initial labeled pool, we randomly draw a sample of size $|\mathcal{L}_{\text{in}}| = 2$ and $|\mathcal{L}_{\text{out}}| = 2$. This is a relaxed version of a cold start, and not a limitation in practice. We apply Min-Max Alignment until $|\mathcal{L}| = 50$. To speed up query selection, we only calculate

[2] `https://www.ipd.kit.edu/mitarbeiter/lama`

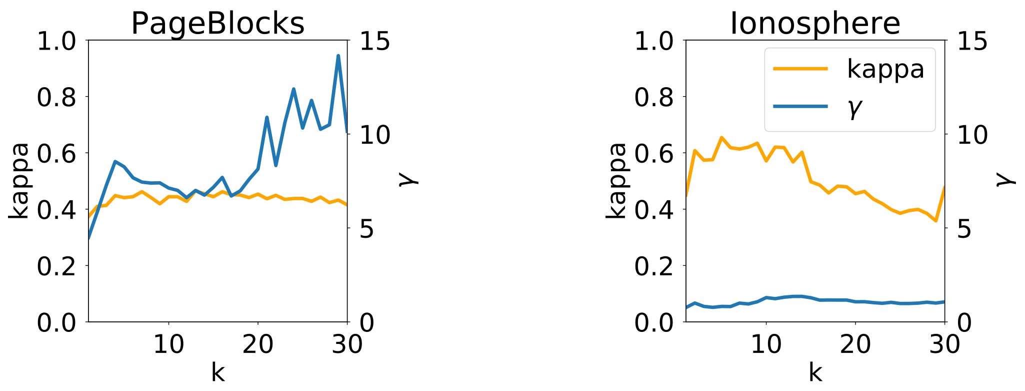


Figure 6.3.: Results with varying k.

τ_{MMA} on a subset of size $\mathcal{S} = 100$ in each iteration, see Section 6.2.3. The locality parameter for relabeling and local alignment is $k = 5$; we will discuss the impact of k later. To estimate C, we split $[C_{\text{LB}}, C_{\text{UB}}]$ by a grid of size 20.

Competitors. We use several state-of-the-art heuristics that have outperformed other competitors in experiments conducted in the respective papers. The first heuristic is *QMS* [Gha⁺18]. We follow the recommendation in the paper to set its parameter $k = \lceil f \cdot N \rceil$, where f is an a-priori estimate of the outlier ratio, which we set to 0.05. The second heuristic is *DFN* [Xia⁺14]. We use two variants: *DFN-Fix* with $C = 0.05$ as recommended in the paper, and *DFN-Sample* where we query the label for 50 randomly selected observations and apply grid search. The third heuristic is *ADS* [Wan⁺18], which uses synthetic observations. We use the grid size recommended in the paper (*ADS-default*) and a variant with a larger grid (*ADS-ext*).

Empirical Bounds. As a lower baseline for the effectiveness of our query strategy, we replace Min-Max Alignment with random sampling (*LAMA-Sample*). Note that the other components of our approach, i.e., selecting γ by local alignment, and C by grid search remain the same as with LAMA. As an empirical upper bound, we search for hyperparameter values based on the ground truth via grid search *(Emp. UB)*. This is an unfair comparison; it merely sets results into perspective. Note that this is an *empirical* upper bound, i.e., instances may occur where one of the competitors yields better results, e.g., for values between the grid steps.

One has to be careful when evaluating classification on outlier benchmark data. This is because measuring classification quality on a holdout split assumes that the train split is representative of the data distribution; this might not hold for outliers. We therefore suggest to evaluate on the full data set, i.e., a variant of the resubstitution error. This a good compromise as long as there are only a few labeled observations, see also Chapter 4. In addition, labels are only used for parameter tuning, the final classifier training is unsupervised, i.e., it does not use the obtained labels. As evaluation metric we use Cohen's kappa, which is well suited for imbalanced data. It returns 1 for a perfect prediction, 0 for random predictions, and negative values for predictions worse than random.

LAMA obtains very good results on the majority of the data sets, see Table 6.1. In several cases, LAMA is even close to the empirical upper bound. This shows that the quality score calculation on a labeled sample aligns very well with the classification quality on the full data set. The local alignment with LAMA-Sample also yields good results. This means that our local alignment works well even on random samples. This is in line with literature, i.e., random selection sometimes scores well against sophisticated alternatives.

LAMA outperforms its competitors on most data sets. Overall, the competitors do not perform well at all, and there are only few instances where they produce useful hyperparameter values. In many cases, the resulting classification accuracy is 0, i.e., the competitors do not produce useful estimates in these cases. Reasons for this might be that neither a direct estimation (DFN-Fix and QMS) nor an estimation with artificial data (ADS) works well with outlier data. For QMS, we found that the closed formula to calculate C returns values that are far from the empirical optimum. In these cases, the classifier either classifies too many or too few observations as outliers. Further, QMS sometimes does not return valid γ values because of duplicates ("–"). DFN-Sample is the closest competitor to LAMA. One reason is that it uses a random sample to estimate C, which tends to be more effective than a closed formula. On most data sets, however, classification results are still worse than LAMA-Sample, which relies on a random sample as well, but uses local kernel alignment for γ instead of a closed formula.

We found LAMA to return good parameters for small values of k. Figure 6.3 shows the result quality and the estimated γ value with k averaged over 10 repetitions for two data sets with increasing k. There, the estimated γ as well as the result quality are stable for $k \in [5, 10]$. This means that our method is not sensitive to k for small values of k. In practice, we recommend to set $k = 5$ since this value has worked well in our benchmark, see Table 6.1.

6.4. Summary

The usefulness of SVDD largely depends on selecting good hyperparameter values. However, existing estimation methods are purely heuristic and require a cumbersome and difficult validation of estimated values.

In this chapter, we propose LAMA, a principled approach to SVDD hyperparameter estimation based on active learning. Its core idea is to refine kernel alignment to small sample sizes by considering only local regions of the data space. LAMA provides evidence-based estimates for both SVDD hyperparameters and eliminates the need for manual validation. LAMA outperforms state-of-the-art competitors in extensive experiments. It provides estimates for both SVDD hyperparameters that result in good classification accuracy, in several cases close to the empirical upper bound.

7. Validating One-Class Active Learning with User Studies

Literature on one-class active learning relies on benchmark data to evaluate the effectiveness of methods. For one, such standardized benchmarks are useful, since algorithmic results and their evaluation are comparable between different experiments. Using benchmark data also is convenient, since there is no need to implement an end-to-end active learning system. This is, one does not need to create user interfaces, or to design and conduct elaborate user studies. So benchmark data facilitates technical contributions to one-class active learning when user studies are out of scope. However, simulating feedback from benchmark data also is a strong simplification. It relies on two fundamental assumptions. The first assumption is that users can always provide accurate feedback, regardless of the presentation of the classification result and of the query. In Chapter 5, we have already discussed that this assumption often is unrealistic, for instance if queries are high-dimensional numerical vectors. The second assumption is that users have endless motivation to provide feedback, even if they do not understand how their feedback affects the algorithmic results. We argue that this also is unlikely to hold in practice. Instead, users may quickly lose interest if they do not see any benefit of spending time and effort on providing annotations.

In literature on one-class active learning, we have observed an over-reliance on these two assumptions. This has led to a peculiar situation. On the one hand, the main value promise of active learning is to allow users to influence the machine learning algorithm, and to contribute information that are not yet included in the training data. On the other hand, there currently is no validation if the value promise can actually be realized in practice. Another consequence of the over-reliance is that only little effort has been spent on the implementation of one-class active learning systems. In fact, the requirements for implementing such a system are still largely unclear.

In our research, we have experienced many of the challenges of realizing a one-class active learning system first hand.[1] There are several *conceptual issues* that are in the way of implementing one-class active learning systems. One issue is that the design space of one-class active learning systems is huge. It requires to define a learning scenario, to choose a suitable classifier and a learning strategy, as well as selecting multiple hyperparameter

[1] The remainder of this chapter bases on the article [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.* It has been shortened to be less repetitive. It contains minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

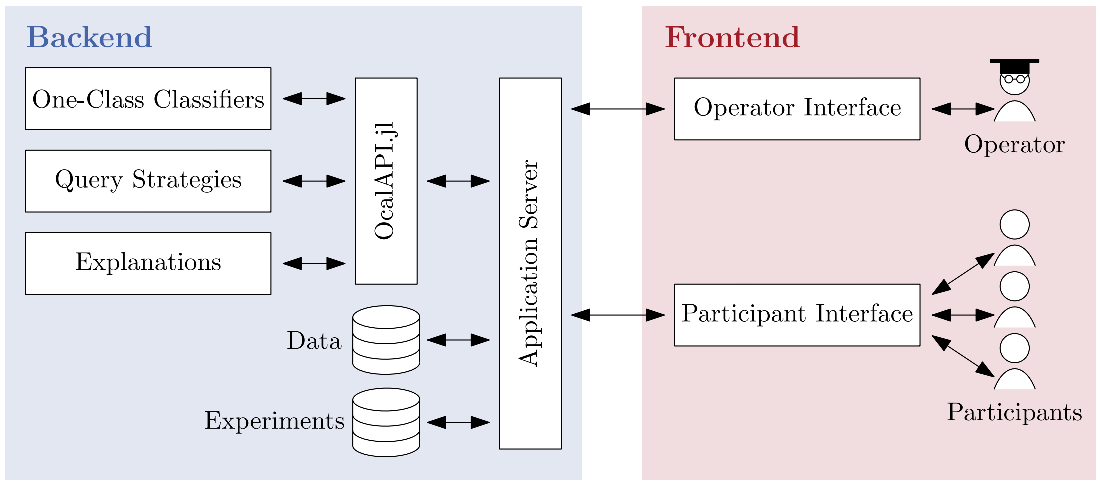


Figure 7.1.: System overview.

values. In addition, there may be several conflicting objectives: One may strive to improve classification accuracy. Another objective may be to use one-class active learning as an exploratory tool to present users with as many interesting instances as possible. A further issue is that objectives of a user study are diverse. One may want to collect a reliable ground truth for a novel data set, or to evaluate specific components of the active learning system, e.g., how well users respond to a particular visualization. Next, there are *technical issues*. For instance, runtimes of training state-of-the-art classifiers may be too long for interactivity. Another example is that it is unclear how to visualize decision boundaries in multi-dimensional data sets, or in subspaces with more than three dimensions.

Although there are many difficulties, we deem user studies imperative to understand the determining factors behind realizing the value of one-class active learning. These factors can serve as guidelines for data mining research and can eventually lead to a more differentiated evaluation of novel query strategies and classifiers. The objective of this chapter is to point out important characteristics and prerequisites of one-class active learning and how they influence the design of interactive systems. To our knowledge, this is the first overview on conceptual and technical challenges regarding one-class active learning systems. We derive these challenges based on an architectural sketch on the components of an existing one-class active learning system, which we have implemented as a prototype. We conclude this chapter by proposing a roadmap towards validating one-class active learning with user studies.

7.1. System Architecture

The purpose of a one-class active learning system is to facilitate experiments with several users. An experiment is a specific technical configuration, i.e., a data set, a classifier, a query strategy, and one or more users, the participants, who provide feedback.

A one-class active learning system consists of several modules. Participants interact with the system through a *participant interface* that visualizes information on active learning iterations, such as the classification result and the progress of the experiment.

The training of the classifier, query selection, and the preparation of additional information such as visualizations and explanations take place in an *algorithm backend*. Finally, there is a human operator who configures, monitors and evaluates the experiments through an *operator interface*. This typically is the researcher who conducts the experiments. Figure 7.1 is an overview of the system architecture. In the following, we describe the different modules and link them to our prototype implementation.

Algorithm Backend

On a technical level, the algorithm backend consists of a classifier module *SVDD.jl*[2] and a module *OneClassActiveLearning.jl*[3], which implements active learning components such as the query strategies. A third module provides additional information, e.g., classifier visualizations. For our prototype, we have implemented the classifiers, query strategies and basic visualization information in *OcalAPI.jl*[4], a ready-to-use JSON REST API. This decoupling allows to re-use the algorithm backend independent of the participant and operator interface.

Operator Interface

The operator interface allows an operator to configure so-called experiment *setups*. A setup consists of a data set, a parameterized classifier and a query strategy. Depending on the research question, the operator may also configure which information is displayed in the participant interface. This gives way to A/B tests, to, say, validate if a certain visualization has an effect on feedback quality. Operators can invite several users to participate in an *experiment run*, i.e., an instantiation of an experiment setup. They can monitor and inspect the experiment runs in an overview panel and export experiment data for further analysis.

Participant Interface

The participant interface has two functions. First, it is an input device to collect feedback during the experiment. Second, it provides the participants with information that supports them to provide educated feedback. For instance, this may be a visualization of a classifier, a view on the raw data or a history of classification accuracy over the past iterations. The participant then provides feedback for some observations. During this process, the interface captures user interactions, e.g., mouse movement and selection. When the query budget or time limit is not exhausted, the participant proceeds with the next iteration.

Our implementation of the interfaces is preliminary, since there are several open challenges, both conceptual and technical (see Section 7.2). We plan to make it publicly available in the future as well. An important takeaway from this section is an intuition about how one-class active learning systems can be designed, on an architectural level. This intuition may be useful to understand the following discussions on the design space of such systems and on the challenges related to the three modules.

[2] `https://github.com/englhardt/SVDD.jl`

[3] `https://github.com/englhardt/OneClassActiveLearning.jl`

[4] `https://github.com/englhardt/OcalAPI.jl`

7.2. Design Decisions

The design and implementation of one-class active learning systems are inherently interdisciplinary and require expertise from several areas, including data mining, human-computer interaction, UX-design, and knowledge of the application domain. Although all disciplines are important, we now focus on the data mining perspective. We first discuss different types of interaction and elaborate on the design options for one-class classifiers and query strategies. We then present different options to prepare information for users during the learning iterations. Finally, we elaborate on several technical challenges.

7.2.1. Type of Interaction

The common definition of active learning is that a query strategy selects one or more observations for feedback. So, strictly speaking, a user does not have the option to also give feedback on other observations not selected by the system. However, there are related disciplines that do away with this restriction. For instance, one research direction is Visual Interactive Analytics (VIA) [Bög+17; Wil17; Lei+17], where a user interactively explores outliers in a data set. VIA systems provide different kinds of visualization to assist users in identifying outliers, in particular with high-dimensional data sets. The unification of active learning and VIA is Visual Inter-Active Labeling (VIAL) [Lin+17; Ber+18b]. VIAL combines active learning with user-supporting visualizations from the VIA community. Variants of VIAL and active learning are conceivable as well. For instance, instead of asking for labels of specific observations, the query strategy could provide a set of observations from which users can select one or more to label.

It is an open question in which cases one should use VIAL or active learning. A user study in [Ber+17] indicates that users label more observations if they are free to choose the observations. However, the resulting classifier accuracy is higher with an active learning query strategy. It is unclear whether these insights transfer to outlier detection where classes are unbalanced. In fact, we see this as one of the overarching questions to answer with user studies.

7.2.2. Type of Feedback

As in the previous chapters, we assume feedback is binary, i.e., users decide whether an observation belongs to the inlier or outlier class. However, recall that other types of feedback are conceivable as well. For instance, in multi-class settings, the system may ask users to state to which classes an observation does *not* belong [CRL12]. Another example is to ask users for feedback on features, as opposed to instances [DSM09]. Existing one-class active learning approaches in turn focus on binary feedback. It is an open question if and how one-class active learning can benefit from allowing for different types of feedback.

7.2.3. Design Space

A one-class active learning system consists of three building blocks: the learning scenario, the classifier, and the query strategy, cf. Chapter 4. Navigating the design space of the building blocks is challenging, and it is generally not feasible to consider and evaluate all possible design alternatives. A good configuration is application-specific and may require fine-tuning of several components.

7.2.4. Preparation of Information

Classifier training and query selection produce a lot of data. On a fine-granular level, this includes the parameterized decision function for the classifier and informativeness scores for the query strategy. After processing this data, query strategies select the most informative instances and predict a label for each observation. In general, this data can be processed and enriched in many ways before presenting it to a user. On a coarse level, one can provide users with additional information, such as explanations of the classifier or contextual information on the learning progress. We now discuss several types of information to present during an active learning iteration: the query, the result, black-box explanations and contextual information.

Query presentation

In general, there are two representations of a query. First, the query has a *raw-data representation.* Examples are text documents, multimedia files, multi-dimensional time series of real-valued sensors, or sub-graphs of a network. Second, the data often is pre-processed to a *feature representation*, a real-valued vector that the classifier can process. In principle, queries can be presented to users in either representation. Our experience is that domain experts are more familiar with raw data and demand it even if the feature representation is interpretable.

Next, one can provide context information for queries. For an individual instance, one can show the nearest neighbors of the query or a difference to prototypes of both classes. Another approach is to use visualization techniques for high-dimensional data [Sac+16; Liu+16] to highlight the query. One can also visualize the score distribution over all candidate queries. Depending on the type of the query strategy, it also is possible to generate heatmaps that indicate areas in the data space with high informativeness [YL18] together with the query.

Result presentation

The presentation of a classification result largely depends on the one-class classifier. A natural presentation of the one-class classifiers used with active learning is a contour plot that shows distances to the decision boundary. However, when data has more than two dimensions, contour plots are not straightforward. The reason is that contour plots rely on the distance to the decision boundary for a two-dimensional grid of observations (x_1, x_2). However, the distance depends on the full vector $(x_1, x_2, \dots, x_n)$ and thus cannot be computed for low-dimensional projections. One remedy would be to train a classifier for each of the projections to visualize. However, the classifier trained on the projection may differ significantly from the classifier trained on all dimensions. So a two-dimensional contour plot may have very little benefit. With common implementations of one-class classifiers, one is currently restricted to present results as plain numeric values, raw data, and predicted labels. A remedy can be active learning in multiple subspaces, when the subspaces selected have less than three dimensions, see Chapter 5.

Black-Box Explanations

Orthogonal to inspecting the queries and the classification result, there are several approaches to provide additional explanations of the classification result. The idea is to treat the classifier, or more generally any predictive model, as a black box, and generate post-hoc explanations for the prediction of individual observations. This is also called *local explanation*, since explanations differ between instances. Recently, CAIPI, a local explainer based on the popular explanation framework LIME [RSG16], has been proposed to explain classification results in an active learning setting [TK19]. The idea behind CAIPI is to provide the user with explanations for the prediction of a query and ask them to correct wrong explanations. Another application of LIME is to explain why an observation has been selected as a query [PCF18]. The idea behind this approach is to explain the informativeness of a query by its neighborhood. The authors use uncertainty sampling, and this approach may also work with other query strategies, such as high-confidence sampling [BBJ15]. However, with more complex query strategies, for instance ones that incorporate local neighborhoods [YWF18] or probability densities [Gha+11b], applying LIME may not be straightforward. For outlier detection, there exist further, more specific approaches to generate explanations. An example is to visualize two-dimensional projections for input features that contribute most to an outlier score [Gup+18]. Other examples are methods from the VIA community that allow users to explore outliers interactively [Bög+17; Wil17; Lei+17].

Contextual Information

The participant interface can also provide additional information that spans several active learning iterations. For instance, the interface can give users access to the classification history, allow them to revisit their previous responses, and give them access to responses of other users, if available. This can entail several issues, such as how to combine possibly diverging responses from different users, and the question whether users will be biased by giving them access to feedback of others. Studying such issues is focus of collaborative interactive learning [Cal+16]. Others have proposed to give users access to 2D scatter plots of the data, the confusion matrix and the progress of classification accuracy on labeled data [LSD19]. In this case, accuracy measures may be biased. For instance, after collecting a ground truth for the first few labels, accuracy may be very high. It may decrease when more labels become available, and the labeled sample covers a larger share of the data space. So it remains an open question whether contextual information will indeed support users to provide accurate feedback.

To conclude, one faces many options in the design of one-class active learning systems. In particular, there are many approaches to support users with information so that they can make informed decisions on the class label. However, the approaches discussed have not yet been evaluated by means of user studies. Instead, they are limited to a theoretical discussion, simulated feedback based on benchmark data, or pen and paper surveys [TK19]. It is largely unclear which methods do enable users to provide feedback and indeed improve the feedback collected.

7.2.5. Technical Challenges

Active learning induces several technical requirements to make systems interactive, and to collect user feedback. Most requirements are general for active learning systems. But their realization with one-class classifiers is difficult.

Cold Start

In most cases, active learning starts with a fully unsupervised setting, i.e., there is no labeled data available. This restricts the possible combinations of classifiers and query strategies in two cases. First, some query strategies, e.g., sampling close to the decision boundary, require a trained one-class classifier to calculate informativeness. In this case, the classifier must be applicable both in an unsupervised and a supervised setting. Second, some query strategies rely on labeled data, e.g., when estimating probability densities for the inlier class [Gha+11a; Gha+11b]. In this case, one cannot calculate informativeness without labels. Current benchmarks mostly avoid this issue by simply assuming that some observations from each class are already labeled. In a real system, one must think about how to obtain the initially labeled observations [Kot+17; AP11]. One option would be to start with a query strategy that does not require any label, such as random sampling, and switch to a more sophisticated strategy once there are sufficiently many labels. Another option is to let users pick the observations to label in the beginning, and then switch to an active learning strategy [AP11; Ber+18b]. However, deciding when to do switches between query strategies is an open question.

Batch Query Selection

Currently, query selection for one-class classifiers is sequential, i.e., for one observation at a time. However, this sequentiality may have several disadvantages, such as frequent updating and re-training of the one-class classifier. Further, it might be easier for users to label several observations in a batch than one observation at a time [Set11]. This may be the case when showing a diverse set of observations helps a user to develop an intuition regarding the data set. There exist some approaches to select multiple observations in batches with one-class classifiers[Eng+20]. However, it is an open question how to embed these strategies in active learning systems.

Incremental Learning

The runtime for updating a classifier constrains the frequency of querying the user. In particular, excessive runtimes for classifier training result in long waiting times and do away with interactivity. Intuitively, there is an upper limit that users are willing to wait, but the specific limit depends on the application.

Several strategies are conceivable to mitigate runtime issues. First, one can rely on incremental learning algorithms [Kef⁺19]. However, state-of-the-art one-class classifiers like SSAD have been proposed without any feature for incremental learning. Second, one can sub-sample to reduce the number of training observations. Several strategies have been proposed explicitly for one-class classifiers [Kra⁺19; Sun⁺16; Li11]. But to our knowledge, there are no studies that combine sub-sampling with one-class active learning. Finally, one can use speculative execution to pre-compute the classifier update for both outcomes (inlier or outlier) while the user is deciding on a label [Spe⁺18]. While such a strategy requires additional computational resources, it might reduce waiting times significantly and improve interactivity. The open question is how to proceed with pre-computing when the look-ahead l is more than one feedback iteration. This is a combinatorial problem, and pre-computing all 2^l learning paths is intractable. Instead, one may use conditional probabilities to pre-compute only the most likely search paths. However, there currently is no method to plan pre-computation beyond $l = 1$. If users select observations to label by themselves, pre-computation would require to compute classifier updates for all observations and outcomes, which is infeasible. Thus, there is a trade-off between giving users flexibility to decide freely on which observations to label, and the capabilities of pre-computation.

Evaluation at Runtime

Without a good quality estimate, it is impossible to know whether the feedback obtained from a user already is sufficient [AP11], i.e., the one-classifier has converged, and additional feedback would not alter the decision boundary any further. However, evaluating the classification quality of active learning at runtime is difficult [Kot⁺19]. This issue exists in both, when benchmarking with simulated feedback, and in real systems – here, we focus on the latter. Users may become frustrated if they face periods where their feedback does not have any effect.

However, showing users any estimated classification quality is difficult for two reasons. First, there might be a short term bias, i.e., the classifier performance might fluctuate significantly. This may be irritating, and it may be difficult to assess for the user. Second, the number of observations in the ground truth increases over time. With only a few labeled observations, the quality estimates may have a large error. This error may reduce with more iterations. So the open question is how to estimate classification quality reliably, and how to adapt these quality estimates during learning. One conceivable option is to switch between exploration and exploitation, i.e., switch from querying for examples that improve classification quality to selection strategies that improve the quality estimate of the classifier. However, there currently is no such switching method for one-class active learning.

Management of Data Flows

Developing an active learning system also requires a sound software architecture. Although this is not a research challenge per se, there are several aspects to consider when implementing one-class active learning systems. One key aspect is the management of data flows. In particular, with a distributed application, see Section 7.1, there are several locations where one has to retain the data set, the classifier, the predictions, and the informativeness scores. For large data sets in particular, transferring data between a client and a backend or loading data sets from disc may affect runtimes significantly. This calls for efficient data caching. Further, one must decide where computations take place. For instance, to visualize contour plots, one must predict the decision boundary for a grid of observations, possibly in multiple projections of the data. In this case, transferring the model over the network may be very little overhead. This can be an efficient strategy when evaluating the model for an observation is cheap. This is the case with SVDD, since the model consists of only a few support vectors. With multi-user studies, one may even reuse trained classifiers and informativeness scores from other user sessions with an equivalent feedback history. In this case, it might be more efficient to pre-compute grid predictions in the backend. So there are several trade-offs and factors that determine an efficient data flow. There currently is no overview on these trade-offs. It also is unclear how they affect design decisions for one-class active learning systems.

7.3. Validation with User Studies

There are a few active learning user studies which have been conducted for special use cases, such as text corpus annotation [Rin+08; ANR09; Cho+19] and network security [BCB18]. However, it is unclear how findings relate to outlier detection – the previous sections illustrate the peculiarities of this application. Further, the plethora of design options make user studies with one-class active learning systems particularly challenging.

Addressing all of the design options at once is not feasible, since there are too many combinations of classifiers, query strategies and ways to prepare information for users. So we propose to start with a narrow use case and to increase the complexity of the one-class active learning system step-wise. Specifically, we have identified the following steps towards a validation in real applications.

(i) *Simplified Use Case:* Much of the value of active learning is in domains where obtaining labels is difficult, even for domain experts. However, we argue that one should identify a use case that many people can easily relate to. This has several advantages. First, we deem reproducibility more important than to obtain sophisticated insights on very special use cases. User studies are easier to reproduce when they do not depend on specific domain expertise. Further, when relationships in data are well understood, one can more easily judge whether the presentation of queries and results is accurate. So we argue to base a validation of one-class active learning on standard benchmark data, for instance the hand-written digit image data set MNIST[5]. Such a simplification also includes to fix the details of the feedback process, for instance to “sequential feedback” and “no initial labels”. If necessary, one should downsample data sets so that runtimes of classifiers and query strategies are not a bottleneck.

[5] http://yann.lecun.com/exdb/mnist/

(ii) *Validation of Information Presented:* The next step is to identify situations when users can give accurate feedback. Since the focus is to validate a learning system with users, one should start with a data set with available ground truth and select the best combination of classifier and query strategy in an experimental benchmark. This might seem counter-intuitive at first sight. In a real application, there generally are not sufficiently many labels available to conduct such a benchmark – in fact, this may even be the motivation for active learning in the first place [AP11; Set12]. However, we argue that this is a necessary step to break the mutual dependency between selecting a good setup and collecting labels. Given a combination of classifier and query strategy, one can then apply different query and result presentations and work with explanations and contextual information. By evaluating this step with user experiments, one can derive assumptions which, if met, enable users to provide accurate feedback.

(iii) *Validation of Classifier and Learning Strategy:* Based on these assumptions, one can vary the dimensions that have been fixed beforehand. This is, one fixes the information presented to the user and varies the query strategies and classifiers. Further, one may validate specific extensions such as batch query strategies.

(iv) *Generalization:* The first step of generalization is to scale the experiments to a large number of observations, using the techniques discussed in Section 7.2.5. Finally, one can then validate the approach on similar data sets, e.g., on different image data.

We expect the findings from these steps to be two-fold. On the one hand, we expect insights that are independent from the use case. For instance, whether scalability techniques are useful is likely to be use-case independent. On the other hand, many findings may depend on the type of data at hand. Explanations based on image data may be very different from the ones for, say, time series data.

Our prototype already includes different classifiers and query strategies, see Section 7.1. So, in general, any researcher can already build a system based on our prototype to conduct Step (i) and the pre-selection of the query strategy and classifier information required for Step (ii). Regarding our prototype, the next steps are to select and implement a working set of query and result presentations, as well as to include black-box explainers and contextual information.

7.4. Summary

Validating one-class active learning through user studies is challenging. One reason is that there are several open conceptual and technical challenges in the design and implementation of interactive learning systems. This chapter features a systematic overview of these challenges, and we have pointed out open research questions with one-class active learning. Next, we have sketched an architecture of a one-class active learning system, which we have implemented as a prototype. Based on it, we propose a roadmap towards validating one-class active learning with user studies.

Part III.

Conclusions

8. Summary

In this thesis, we have studied active learning for outlier detection. Throughout our work, we have followed a user-centric paradigm, i.e., we have put focus on the human in the loop. We deem this focus essential to implement complex end-to-end use cases with active learning, like the one presented in Chapter 2. At the core of our work, we identify and challenge existing assumptions from literature that are in the way of realizing active learning for outlier detection. To overcome the limitations these assumptions entail, we make several conceptual and technical contributions. In the following, we summarize them and highlight how they contribute to a user-centric approach.

In the beginning of our work, we have addressed the question how to categorize and compare existing active learning methods for outlier detection (Chapter 4). There, we present a unified view on literature by structuring one-class active learning into three core building blocks: the learning scenario, the base learner and the query strategy. In particular the specification of the learning scenario, i.e., the underlying assumptions and the initial setup of an active learning approach, has turned out to be crucial to select good base learners and query strategies. A rigorous specification further is important to help users in comparing existing methods, and in selecting a good method for their use case. This is an important take-away which has not received much attention in literature so far.

We have further introduced summary statistics of active learning curves that can serve as an evaluation standard for one-class active learning. Summary statistics make comprehensive benchmarks of active learning methods feasible, and thus support users in a reliable and comparable assessment of novel approaches. In fact, these summary statistics have turned out to be very useful throughout our work, especially in the experimental evaluation of Chapter 5. We have further used them to evaluate the quality of existing methods in a comprehensive benchmark with different base learners, query strategies and learning scenarios. Our results show that there is no one superior one-class active learning method. Instead, a good choice is use-case specific. Thus, to support users in selecting an active learning method nevertheless, we propose guidelines for a use-case specific selection.

Based on our overview and benchmark, we have identified two of the key difficulties that currently are in the way of realizing one-class active learning in practice. The first difficulty is the widely accepted assumption that users can provide feedback, regardless of how algorithmic results are presented to them. For one, this assumption simplifies an empirical evaluation of novel methods, since one can just simulate user feedback based on a ground truth. This simplification has facilitated significant advancements in one-class active learning methods over the last years. However, it is unlikely to hold in practice. One can easily construct counter examples, e.g., a high-dimensional data space, where users may struggle to provide feedback. Therefore, we have asked the question how one can replace this assumption with a more realistic one: that users can provide feedback if the dimensionality of the data space is low, but that this ability decreases with increasing dimensionality of the data space. We have addressed this question by introducing SubSVDD, a novel one-class classifier (Chapter 5). In a nutshell, SubSVDD learns decision boundaries in multiple low-dimensional projections of the data. Users benefit from our method in two ways. (i) When subspaces are two or three-dimensional, one can visualize the decision boundary in scatter plots of the data. There, users can visually analyze the data distribution and the classifier decision boundaries to provide feedback. (ii) SubSVDD yields a binary classification for each observation in all of the subspaces. This gives way to a concise summary on the subspaces where observations are outlying, which in turn can support users in comprehending the classification result.

The second key difficulty we have identified is the selection of hyperparameter values for one-class classifiers. A successful application of one-class classifiers hinges on the choice of their hyperparameter values – a poor choice may deteriorate results significantly. However, selecting hyperparameter values requires to estimate non-obvious problem characteristics, such as the complexity of decision boundaries and the true share of outliers in the data. To this end, state-of-the-art has been to use heuristics to initialize one-class classifiers. This is because there often is no labeled training data available to estimate hyperparameters in the beginning of the active learning cycle. Although a variety of heuristics for hyperparameter estimation exist, selecting a suitable one is difficult. A reason is that they all come with a plausible intuition, but they do not come with any formal guarantees, and the hyperparameter values they return are diverse. This has motivated the question whether one can make parameter selection more reliable and intuitive, by a more principled approach. Our approach is to take a new perspective on hyperparameter estimation, by using active learning (Chapter 6). To this end, we introduce LAMA, an active learning method for kernel alignment with small sample sizes. One of the key benefits is that LAMA only requires binary feedback on a few observations to achieve a good estimate. Based on the sample obtained through active learning, one can further estimate the quality of the hyperparameter values selected. This simplifies hyperparameter selection significantly, since users do not need to understand or validate any hyperparameter values.

Finally, we have have looked at validating one-class active learning with user studies (Chapter 7). In particular, we have asked what is required to realize a one-class active learning system in practice. To study this question, we have designed and implemented a prototype of a one-class active learning system. During implementation, several conceptual and technical issues crystallized. Together with our experience from the previous chapters, we have formulated and categorized them systematically, and have given ideas on how to address these issues. One key take-away is that some of the existing issues, such as batch-mode learning, can be studied independently. However, thinking about how to derive and to prepare information that helps users in providing feedback entails far more difficult and complex questions. Studying them requires a comprehensive approach which involves related research areas such as XAI, visual analytics, and human-computer interaction. We finally conclude with a roadmap on how to make some strides towards validating one-class active learning with user studies.

9. Outlook

This thesis emphasizes the role of the user in the active learning process. We deem this a promising direction for future research on active learning for outlier detection that entails many more interesting research questions. In this section, we give an outlook on questions that are closely related to our work.

Realistic Oracles. In Chapter 7, we have explained that there still are challenges in the way of conducting experiments with users in the loop. Here, we see several intermediate steps to make simulated oracles more realistic. Studying them can smoothen the path to real applications.

One question is to ask what happens when users provide feedback that is incorrect, i.e., that differs from the ground truth. The motivation for this question is straightforward. Under certain circumstances, users may not answer correctly, be it because they do not know better, be it that they simply make a mistake. One can study this issue by introducing imperfect oracles, i.e., oracles that, in some cases, return feedback different to a ground truth [DL10; CS17]. A naive approach would be to use noisy oracles with a fixed probability for correct feedback. However, one can also make assumptions on the conditions under which users are more likely to give correct feedback. For instance, one may simply correlate the data space dimensionality to the probability of correct feedback. When the dimensionality increases, the probability of correct feedback decreases. Another example would be to have a low probability for correct feedback for hidden outliers [SB17], i.e., outliers that only occur in very few, specific subspaces. Such application-specific noisy oracles can facilitate the evaluation of classifier and query strategy robustness under different conditions.

Another consequence of incorrect feedback is that one has to think about query strategies that are sensitive to uncertainty in user feedback. For instance, a respective query strategy might query areas more often where feedback is not reliable. One may approach this by leveraging probabilistic active learning methods [KKS14] to model uncertainty. However, how existing probabilistic sampling methods transfer to the one-class setting is yet unclear.

Multiple Oracles. Existing work currently only considers a single oracle. However, several questions arise when multiple annotators are available. For instance, one may study whether one can exploit the wisdom of multiple users to increase robustness. To this end, one may compare feedback from several oracles, and identify regions of agreement and disagreement between them. Incorrect feedback may be apparent when it differs to the majority vote in regions of high agreement. One can then further exploit this property in a query strategy that strives to increase robustness of classification under noisy oracles.

Another question is whether one can improve the overall cost of active learning by querying multiple users in parallel, so-called batch-mode active learning. A naive way to query batches is to simply select the top-k observations, ranked by informativeness. However, this may result in inefficient queries when the top-k observations are similar to each other [She+04; SZ05; Set12]. There are different ways to approach this issue, e.g., by also considering the representativeness and the diversity of a batch [Eng+20]. It is largely unclear to which extent insights from related disciplines, such as multi-class classification and crowdsourcing, transfer to outlier detection.

Method Extensions. Throughout this thesis, we have hinted at non-trivial extensions to the methods we propose. It is an open question whether these extensions further improve on result quality, or some other metric of interest. For instance, we have briefly discussed that one may adjust v parameter of SubSVDD according to how certain users are with their feedback, see Section 5.2.1. This may be one way to improve robustness of a classifier with realistic oracles, see above. However, it is unclear what a good update rule would be.

Another extension is to use advanced subspace search methods to derive the input set of subspaces for SubSVDD. This is not a straightforward extension, since there are many methods to choose from. In most cases, one cannot easily control the subspace size or even the number of subspaces the methods return. So one has to think carefully about the selection of good subspaces.

A third extension relates to LAMA, where we have focused on the Gaussian kernel. This choice has been pragmatic, since it is the most popular kernel with SVDD. However, other domains use different kind of kernels, such as string kernels [Lod+02] or graph kernels [NSV19]. It is an open question whether the local alignment by active learning also works well with parametric kernels from these domains. However, this also requires to study the more general question whether alternative kernels work well in combination with active learning for outlier detection, which has not been a focus of literature so far.

Complex Query Strategies. The results from our benchmark have revealed that different query strategies work well on different data sets. Another observation is that query strategies may perform differently well in different stages of active learning. For instance, random sampling may work very well in the beginning, while other, more sophisticated strategies may be more effective when a sufficient number of observations is available. To this end, it is an open question when one should switch between different query strategies to exploit their specific advantages.

In summary, this thesis advances the state-of-the-art on active learning for outlier detection. Our methods give way to influence algorithmic decisions in a user-centric way. Thus, they help to realize complex end-to-end use cases with one-class active learning. There are several questions that emerge from our research results. Addressing them may improve the effectiveness of one-class active learning even further, and can ultimately contribute to robust and realistic active learning systems.

Bibliography

[ABN18] Ali Anaissi, Ali Braytee, and Mohamad Naji. “Gaussian Kernel Parameter Optimization in One-Class Support Vector Machines”. In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489383.

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

[AKW18] Ali Anaissi, Nguyen Lu Dang Khoa, and Yang Wang. “Automated Parameter Tuning in One-class Support Vector Machine: An Application for Damage Detection”. In: *International Journal of Data Science and Analytics* 6.4 (2018), pp. 311–325. ISSN: 2364-4168. DOI: 10.1007/s41060-018-0151-9.

[ANR09] Shilpa Arora, Eric Nyberg, and Carolyn P Rosé. “Estimating Annotation Cost for Active Learning in a Multi-annotator Environment”. In: *NAACL Workshop on Active Learning for Natural Language Processing*. ACL. 2009, pp. 18–26. DOI: 10.3115/1564131.1564136.

[AP11] Josh Attenberg and Foster Provost. “Inactive Learning?: Difficulties Employing Active Learning in Practice”. In: *SIGKDD Explorations Newsletter* 12.2 (2011), pp. 36–41. ISSN: 1931-0145. DOI: 10.1145/1964897.1964906.

[ARM12] M Ehsan Abbasnejad, Dhanesh Ramachandram, and Rajeswari Mandava. “A Survey of the State of the Art in Learning the Kernels”. In: *Knowledge and Information Systems* 31.2 (2012), pp. 193–221. DOI: 10.1007/s10115-011-0404-6.

[Bar+18] Lukas Barth, Nicole Ludwig, Esther Mengelkamp, and Philipp Staudt. “A Comprehensive Modelling Framework for Demand Side Flexibility in Smart Grids”. In: *Computer Science-Research and Development* 33.1-2 (2018), pp. 13–23. DOI: 10.1007/s00450-017-0343-x.

[BBD06] Amit Banerjee, Philippe Burlina, and Chris Diehl. “A Support Vector Method for Anomaly Detection in Hyperspectral Imagery”. In: *Geoscience and Remote Sensing* 44.8 (2006), pp. 2282–2291. DOI: 10.1109/TGRS.2006.873019.

[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, pp. 390–395. DOI: 10.1109/ICMLA.2015.167.

[BCB17] Anaël Beaugnon, Pierre Chifflier, and Francis Bach. “ILAB: An Interactive Labelling Strategy for Intrusion Detection”. In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer. 2017, pp. 120–140. DOI: 10.1007/978-3-319-66332-6_6.

[BCB18] Anaël Beaugnon, Pierre Chifflier, and Francis Bach. "End-to-end Active Learning for Computer Security Experts". In: *Conference on Artificial Intelligence (AAAI) Workshops.* 2018.

[Ber+17] Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Fellner, and Michael Sedlmair. "Comparing Visual-interactive Labeling with Active Learning: An Experimental Study". In: *Visualization and Computer Graphics* 24.1 (2017), pp. 298–308. DOI: 10.1109/tvcg.2017.2744818.

[Ber+18a] Jürgen Bernard, Matthias Zeppelzauer, Markus Lehmann, Martin Müller, and Michael Sedlmair. "Towards User-Centered Active Learning Algorithms". In: *Computer Graphics Forum* 37.3 (2018), pp. 121–132. DOI: 10.1111/cgf.13406.

[Ber+18b] 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.

[Bey+99] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. "When is "nearest neighbor" meaningful?" In: *International Conference on Database Theory (ICDT).* Springer. 1999, pp. 217–235. DOI: 10.1007/3-540-49257-7_15.

[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.

[Bis+18] Simon Bischof, Holger Trittenbach, Michael Vollmer, Dominik Werle, Thomas Blank, and Klemens Böhm. "HIPE: An Energy-Status-Data Set from Industrial Production". In: *International Conference on Future Energy Systems (e-Energy) Workshops.* ACM. 2018, pp. 599–603. DOI: 10.1145/3208903.3210278.

[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.

[BKL15] Christian Beyer, Georg Krempl, and Vincent Lemaire. "How to Select Information That Matters: A Comparative Study on Active Learning Strategies for Classification". In: *International Conference on Knowledge Technologies and Data-driven Business.* ACM, 2015, pp. 1–8. DOI: 10.1145/2809563.2809594.

[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* 36.3 (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 Record* 29.2 (2000), pp. 93–104. DOI: 10.1145/335191.335388.

[BSM19] Mirko Bunse, Amal Saadallah, and Katharina 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, p. 104.

[Cal⁺16] Adrian Calma, Jan Marco Leimeister, Paul Lukowicz, Sarah Oeste-Reiß, Tobias Reitmaier, Albrecht Schmidt, Bernhard Sick, Gerd Stumme, and Katharina Anna Zweig. “From Active Learning to Dedicated Collaborative Interactive Learning”. In: *International Conference on Architecture of Computing Systems (ARCS).* VDE. 2016, pp. 1–8.

[Cam⁺16] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB 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`.

[Caw11] Gavin C Cawley. “Baseline Methods for Active Learning”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS) Workshops.* 2011, pp. 47–57.

[Cel10] Oscar Celma. “Music Rrecommendation”. In: *Music Recommendation and Discovery.* Springer, 2010, pp. 43–85. DOI: `10.1007/978-3-642-13287-2_3`.

[Cha⁺17] Arin Chaudhuri, Deovrat Kakde, Carol Sadek, Laura Gonzalez, and Seunghyun Kong. “The Mean and Median Criteria for Kernel Bandwidth Selection for Support Vector Data Description”. In: *International Conference on Data Mining (ICDM) Workshops.* IEEE. 2017, pp. 842–849. DOI: `10.1109/ICDMW.2017.116`.

[Cho⁺19] Minsuk Choi, Cheonbok Park, Soyoung Yang, Yonggyu Kim, Jaegul Choo, and Sungsoo Ray Hong. “AILA: Attentive Interactive Labeling Assistant for Document Classification through Attention-Based Deep Neural Networks”. In: *Conference on Human Factors in Computing Systems.* ACM. 2019, p. 230. DOI: `10.1145/3290605.3300460`.

[CKB14] Myungraee Cha, Jun Seok Kim, and Jun-Geol Baek. “Density Weighted Support Vector Data Description”. In: *Expert Systems with Applications* 41.7 (2014), pp. 3343–3350. DOI: `10.1016/j.eswa.2013.11.025`.

[CLL13] Wei-Cheng Chang, Ching-Pei Lee, and Chih-Jen Lin. *A Revisit to Support Vector Data Description.* Tech. rep. Taiwan University, 2013.

[CMR12] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. “Algorithms for Learning Kernels Based on Centered Alignment”. In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 795–828. URL: `http://dl.acm.org/citation.cfm?id=2503308.2188413`.

[Cri⁺02] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. “On Kernel-Target Alignment”. In: *Advances in Neural Information Processing Systems.* 2002, pp. 367–373. DOI: `10.7551/mitpress/1120.003.0052`.

[CRL12] Nicolas Cebron, Fabian Richter, and Rainer Lienhart. ““I can tell you what it’s not”: Active Learning from Counterexamples”. In: *Progress in artificial intelligence* 1.4 (2012), pp. 291–301. DOI: 10.1007/s13748-012-0023-9.

[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.

[CT11] Veronika Cheplygina and David M. J. Tax. “Pruned Random Subspace Method for One-Class Classifiers”. In: *Multiple Classifier Systems*. Springer, 2011, pp. 96–105. DOI: 10.1007/978-3-642-21557-5_12.

[Dan+14] Xuan Hong Dang, Ira Assent, Raymond T Ng, Arthur Zimek, and Erich Schubert. “Discriminative Features for Identifying and Interpreting Outliers”. In: *International Conference on Data Engineering (ICDE)*. IEEE. 2014, pp. 88–99. DOI: 10.1109/icde.2014.6816642.

[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.

[DL10] Jun Du and Charles X Ling. “Active Learning with Human-Like Noisy Oracle”. In: *International Conference on Data Mining (ICDM)*. 2010, pp. 797–802. DOI: 10.1109/ICDM.2010.114.

[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.

[DSM09] Gregory Druck, Burr Settles, and Andrew McCallum. “Active Learning by Labeling Features”. In: *Conference on Empirical Methods in Natural Language Processing*. ACL. 2009, pp. 81–90. DOI: 10.3115/1699510.1699522.

[DX07] Hongmei Deng and Roger Xu. “Model Selection for Anomaly Detection in Wireless Ad Hoc Networks”. In: *Symposium on Computational Intelligence and Data Mining*. IEEE. 2007, pp. 540–546. DOI: 10.1109/CIDM.2007.368922.

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

[EES07] Paul F Evangelista, Mark J Embrechts, and Boleslaw K Szymanski. “Some Properties of the Gaussian kernel for One Class Learning”. In: *International Conference on Artificial Neural Networks (ICANN)*. Springer. 2007, pp. 269–278. DOI: 10.1007/978-3-540-74690-4_28.

[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+20] Adrian Englhardt, Holger Trittenbach, Dennis Vetter, and Klemens Böhm. "Finding the Sweet Spot: Batch Selection for One-Class Active Learning". In: *International Conference on Data Mining (SDM)*. SIAM. 2020.

[ERR14] Mehdi Elahi, Francesco Ricci, and Neil Rubens. "Active Learning in Collaborative Filtering Recommender Systems". In: *E-Commerce and Web Technologies*. Springer, 2014, pp. 113–124. DOI: 10.1007/978-3-319-10491-1_12.

[GA11] Mehmet Gönen and Ethem Alpaydın. "Multiple Kernel Learning Algorithms". In: *Journal of Machine Learning Research (JMLR)* 12 (2011), pp. 2211–2268.

[Gha+11a] Alireza Ghasemi, Mohammad T Manzuri, Hamid R Rabiee, Mohammad H Rohban, and Siavash 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] Alireza Ghasemi, Hamid R Rabiee, Mohsen Fadaee, Mohammad T Manzuri, and Mohammad 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.

[Gha+16] Zahra Ghafoori, Sutharshan Rajasegarar, Sarah M Erfani, Shanika Karunasekera, and Christopher A Leckie. "Unsupervised Parameter Estimation for One-Class Support Vector Machines". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer. 2016, pp. 183–195. DOI: 10.1007/978-3-319-31750-2_15.

[Gha+18] Zahra Ghafoori, Sarah M Erfani, Sutharshan Rajasegarar, James C Bezdek, Shanika Karunasekera, and Christopher Leckie. "Efficient Unsupervised Parameter Estimation for One-Class Support Vector Machines". In: *Neural Networks and Learning Systems* 29.10 (2018), pp. 5057–5070. DOI: 10.1109/TNNLS.2017.2785792.

[GJ17] Mohsen Ghazel and Nathalie Japkowicz. "Improving Active Learning for One-Class Classification Using Dimensionality Reduction". In: *Canadian Conference on Artificial Intelligence*. Springer. 2017, pp. 39–44. DOI: 10.1007/978-3-319-57351-9_4.

[GK11a] 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.

[GK11b] Prudhvi Gurram and Heesung Kwon. “Support-Vector-Based Hyperspectral Anomaly Detection Using Optimized Kernel Parameters”. In: *Geoscience and Remote Sensing Letters* 8.6 (2011), pp. 1060–1064. DOI: 10.1109/LGRS.2011.2155030.

[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.

[GU16] Markus Goldstein and Seiichi Uchida. “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data”. In: *PloS one* 11.4 (2016), e0152173. DOI: 10.1371/journal.pone.0152173.

[Gup+18] Nikhil Gupta, Dhivya Eswaran, Neil Shah, Leman Akoglu, and Christos Faloutsos. “Beyond Outlier Detection: LOOKOUT for Pictorial Explanation”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD).* Springer. 2018, pp. 122–138. DOI: 10.1007/978-3-030-10925-7_8.

[HA04] Victoria Hodge and Jim Austin. “A Survey of Outlier Detection Methodologies”. In: *Artificial intelligence Review* 22.2 (2004), pp. 85–126. DOI: 10.1023/b:aire.0000045502.10941.a9.

[HG10] He He and Ali Ghodsi. “Rare Class Classification by Support Vector Machine”. In: *International Conference on Pattern Recognition (ICPR).* IEEE. 2010, pp. 548–551. DOI: 10.1109/ICPR.2010.139.

[HJ08] Steven CH Hoi and Rong Jin. “Active Kernel Learning”. In: *International Conference on Machine Learning (ICML).* ACM. 2008, pp. 400–407. DOI: 10.1145/1390156.1390207.

[HMP04] Haym Hirsh, Nina Mishra, and Leonard Pitt. “Version Spaces and the Consistency Problem”. In: *Artificial Intelligence* 156.2 (2004), pp. 115–138. DOI: 10.1016/j.artint.2003.04.003.

[JD03a] Piotr Juszczak and Robert PW Duin. “Selective Sampling Methods in One-Class Classification Problems”. In: *Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003.* Springer, 2003, pp. 140–148. DOI: 10.1007/3-540-44989-2_18.

[JD03b] Piotr Juszczak and Robert PW Duin. “Uncertainty Sampling Methods for One-Class Classifiers”. In: *International Conference on Machine Learning (ICML) Workshops.* Vol. 3. 2003.

[Jus06] Piotr Juszczak. “Learning to Recognise: A Study on One-Class Classification and Active Learning”. PhD thesis. Delft University of Technology, 2006.

[Kak+17] Deovrat Kakde, Arin Chaudhuri, Seunghyun Kong, Maria Jahja, Hansi Jiang, and Jorge Silva. “Peak Criterion for Choosing Gaussian Kernel Bandwidth in Support Vector Data Description”. In: *International Conference on Prognostics and Health Management (ICPHM)*. IEEE. 2017, pp. 32–39. DOI: 10.1109/ICPHM.2017.7998302.

[Kef+19] Takoua Kefi-Fatteh, Riadh Ksantini, Mohamed-Bécha Kaâniche, and Adel Bouhoula. “A Novel Incremental One-Class Support Vector Machine Based on Low Variance Direction”. In: *Pattern Recognition* 91 (2019), pp. 308–321. DOI: 10.1016/j.patcog.2019.02.027.

[Kel15] Fabian Keller. “Attribute Relationship Analysis in Outlier Mining and Stream Processing”. PhD thesis. Karlsruhe Institute of Technology, 2015. DOI: 10.5445/IR/1000048790.

[Kha+11] S Khazai, S Homayouni, A Safari, and B Mojaradi. “Anomaly Detection in Hyperspectral Images Based on an Adaptive Support Vector Method”. In: *Geoscience and Remote Sensing Letters* 8.4 (2011), pp. 646–650. DOI: 10.1109/LGRS.2010.2098842.

[KKS14] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. “Probabilistic Active Learning: Towards Combining Versatility, Optimality and Efficiency”. In: *International Conference on Discovery Science*. Springer. 2014, pp. 168–179. DOI: 10.1007/978-3-319-11812-3_15.

[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.

[KMB12] Fabian Keller, Emmanuel Muller, and Klemens Bohm. “HiCS: High Contrast Subspaces for Density-Based Outlier Ranking”. In: *International Conference on Data Engineering (ICDE)*. IEEE. 2012, pp. 1037–1048. DOI: 10.1109/icde.2012.88.

[KMM18] Jacob Kauffmann, Klaus-Robert Müller, and Grégoire Montavon. “Towards Explaining Anomalies: A Deep Taylor Decomposition of One-Class Models”. In: *arXiv* (2018). arXiv: 1805.06230.

[Kot+17] Daniel Kottke, Adrian Calma, Denis Huseljic, GM Krempl, and Bernhard Sick. “Challenges of Reliable, Realistic and Comparable Active Learning Evaluation”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) Workshops*. 2017, pp. 2–14.

[Kot+19] Daniel Kottke, Jim Schellinger, Denis Huseljic, and Bernhard Sick. “Limitations of Assessing Active Learning Performance at Runtime”. In: *arXiv* (2019). arXiv: 1901.10338.

[Kra+19] Bartosz Krawczyk, Isaac Triguero, Salvador Garcia, Michał Woźniak, and Francisco Herrera. “Instance Reduction for One-Class Classification”. In: *Knowledge and Information Systems* 59.3 (2019), pp. 601–628. DOI: 10.1007/s10115-018-1220-z.

[Kri$^+$11] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. "Interpreting and Unifying Outlier Scores". In: *International Conference on Data Mining (ICDM)*. SIAM. 2011, pp. 13–24. DOI: 10.1137/1.9781611972818.2.

[Kri$^+$12] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. "Outlier Detection in Arbitrarily Oriented Subspaces". In: *International Conference on Data Mining*. IEEE. 2012, pp. 379–388. DOI: 10.1109/ICDM.2012.21.

[KSC02] Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. *On the Extensions of Kernel Alignment*. Tech. rep. University of Southhampton, 2002. URL: https://eprints.soton.ac.uk/id/eprint/259745.

[KW06] Christine Körner and Stefan Wrobel. "Multi-class Ensemble-Based Active Learning". In: *European Conference on Machine Learning (ECML)*. Springer Berlin Heidelberg, 2006, pp. 687–694. DOI: 10.1007/11871842_68.

[KW12] Bartosz Krawczyk and Michał Woźniak. "Combining Diverse One-Class Classifiers". In: *International Conference on Hybrid Artificial Intelligence Systems*. Springer. 2012, pp. 590–601. DOI: 10.1007/978-3-642-28931-6_56.

[Le$^+$10] Trung Le, Dat Tran, Wanli Ma, and Dharmendra Sharma. "A Theoretical Framework for Multi-Sphere Support Vector Data Description". In: *International Conference on Neural Information Processing*. Springer. 2010, pp. 132–142. DOI: 10.1007/978-3-642-17534-3_17.

[Lei$^+$17] 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 (2017), pp. 330–339. DOI: 10.1109/TVCG.2017.2744758.

[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.

[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.

[Lip16] Zachary C Lipton. "The Mythos of Model Interpretability". In: *International Conference on Machine Learning (ICML) Workshops*. 2016. DOI: 10.1145/3233231.

[Liu$^+$16] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. "Visualizing High-Dimensional Data: Advances in the Past Decade". In: *Visualization and Computer Graphics* 23.3 (2016), pp. 1249–1268. DOI: 10.1109/TVCG.2016.2640960.

[Liu04] Ying Liu. "Active Learning with Support Vector Machine Applied to Gene Expression Data for Cancer Classification". In: *Journal of Chemical Information and Computer Sciences* 44.6 (2004), pp. 1936–1941. DOI: 10.1021/ci049810a.

[Lod+02] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. “Text Classification using String Kernels”. In: *Journal of Machine Learning Research (JMLR)* 2.Feb (2002), pp. 419–444.

[LSD19] Phil Legg, Jim Smith, and Alexander Downing. “Visual Analytics for Collaborative Human-Machine Confidence in Human-Centric Active Learning Tasks”. In: *Human-centric Computing and Information Sciences* 9.1 (2019), p. 5. DOI: 10.1186/s13673-019-0167-8.

[LSF14] Sheng Li, Ming Shao, and Yun Fu. “Low-Rank Outlier Detection”. In: *Low-Rank and Sparse Modeling for Visual Analysis*. Springer, 2014, pp. 181–202. DOI: 10.1007/978-3-319-12000-3_9.

[Mic+13] Barbora Micenková, Raymond T Ng, Xuan-Hong Dang, and Ira Assent. “Explaining Outliers by Subspace Separability”. In: *International Conference on Data Mining (ICDM)*. IEEE. 2013, pp. 518–527. DOI: 10.1109/ICDM.2013.132.

[Mit82] Tom M Mitchell. “Generalization as Search”. In: *Artificial intelligence* 18.2 (1982), pp. 203–226. DOI: 10.1016/0004-3702(82)90040-6.

[MRW14] Benjamin Mack, Ribana Roscher, and Björn Waske. “Can I Trust My One-Class Classification?” In: *Remote Sensing* 6.9 (2014), pp. 8779–8802. DOI: 10.3390/rs6098779.

[MSS11] Emmanuel Müller, Matthias Schiffer, and Thomas Seidl. “Statistical Selection of Relevant Subspace Projections for Outlier Ranking”. In: *International Conference on Data Engineering (ICDE)*. IEEE. 2011, pp. 434–445. DOI: 10.1109/ICDE.2011.5767916.

[Mül+12] Emmanuel Müller, Fabian Keller, Sebastian Blanc, and Klemens Böhm. “OutRules: A Framework for Outlier Descriptions in Multiple Context Spaces”. In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 828–832. DOI: 10.1007/978-3-642-33486-3_57.

[Ngu+13] Hoang Vu Nguyen, Emmanuel Müller, Jilles Vreeken, Fabian Keller, and Klemens Böhm. “CMI: An Information-Theoretic Contrast Measure for Enhancing Subspace Cluster and Outlier Detection”. In: *International Conference on Data Mining (ICDM)*. SIAM. 2013, pp. 198–206. DOI: 10.1137/1.9781611972832.22.

[NMB13] Hoang Vu Nguyen, Emmanuel Müller, and Klemens Böhm. “4S: Scalable Subspace Search Scheme Overcoming Traditional Apriori Processing”. In: *2013 IEEE International Conference on Big Data*. IEEE. 2013, pp. 359–367. DOI: 10.1109/BigData.2013.6691596.

[NSV19] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. “Graph Kernels: A Survey”. In: *arXiv* (2019). arXiv: 1904.12218.

[ODM17] Jack O’Neill, Sarah Jane Delany, and Brian MacNamee. “Model-Free and Model-Based Active Learning for Regression”. In: *Advances in Computational Intelligence Systems*. Springer, 2017, pp. 375–386. DOI: 10.1007/978-3-319-46562-3_24.

[Ols09] Fredrik Olsson. *A literature Survey of Active Machine Learning in the Context of Natural Language Processing*. Tech. rep. Swedish Institute of Computer Science, 2009.

[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.

[PCF18] R. Phillips, K. H. Chang, and S. A. Friedler. "Interpretable Active Learning". In: *Conference on Fairness, Accountability and Transparency*. 2018.

[Pea10] Beth Pearsall. "Predictive Policing: The Future of Law Enforcement?" In: *National Institute of Justice Journal* 266.1 (2010), pp. 16–19. DOI: 10.1037/e596372010-007.

[PFB19] Daniel Popovic, Edouard Fouché, and Klemens Böhm. "Unsupervised Artificial Neural Networks for Outlier Detection in High-Dimensional Data". In: *European Conference on Advances in Databases and Information Systems*. Springer. 2019, pp. 3–19. DOI: 10.1007/978-3-030-28730-6_1.

[PKC17] Sergiy Peredriy, Deovrat Kakde, and Arin Chaudhuri. "Kernel Bandwidth Selection for SVDD: The Sampling Peak Criterion Method for Large Data". In: *International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 3540–3549. DOI: 10.1109/BigData.2017.8258344.

[PM15] Heiko Paulheim and Robert Meusel. "A Decomposition of the Outlier Detection Problem into a Set of Supervised Learning Problems". In: *Machine Learning* 100.2-3 (2015), pp. 509–531. DOI: 10.1007/s10994-015-5507-y.

[PS11] Karim Pichara and Alvaro Soto. "Active Learning and Subspace Clustering for Anomaly Detection". In: *Intelligent Data Analysis* 15.2 (2011), pp. 151–171. DOI: 10.3233/ida-2010-0461.

[Qi+18] Di Qi, Joshua Arfin, Mengxue Zhang, Tushar Mathew, Robert Pless, and Brendan Juba. "Anomaly Explanation Using Metadata". In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1916–1924. DOI: 10.1109/wacv.2018.00212.

[Ram+17] Maria E Ramirez-Loaiza, Manali Sharma, Geet Kumar, and Mustafa Bilgic. "Active Learning: An Empirical Study of Common Baselines". In: *Data mining and knowledge discovery* 31.2 (2017), pp. 287–313. DOI: 10.1007/s10618-016-0469-7.

[RAV18] Oscar Reyes, Abdulrahman H Altalhi, and Sebastián Ventura. "Statistical Comparisons of Active Learning Strategies over Multiple Datasets". In: *Knowledge-Based Systems* 145 (2018), pp. 274–288. DOI: 10.1016/j.knosys.2018.01.033.

[Rin+08] Eric K Ringger, Marc Carmen, Robbie Haertel, Kevin D Seppi, Deryle Lonsdale, Peter McClanahan, James L Carroll, and Noel Ellison. "Assessing the Costs of Machine-Assisted Corpus Annotation through a User Study". In: *International Conference on Language Resources and Evaluation (LREC).* Vol. 8. 2008, pp. 3318–3324.

[RMJ06] Hema Raghavan, Omid Madani, and Rosie Jones. "Active Learning with Feedback on Features and Instances". In: *Journal of Machine Learning Research* 7.Aug (2006), pp. 1655–1686.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *International Conference on Knowledge Discovery and Data Mining (KDD).* ACM. 2016, pp. 1135–1144. DOI: `10.18653/v1/n16-3020`.

[Sac+16] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C North, and Daniel A Keim. "Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis". In: *Visualization and Computer Graphics* 23.1 (2016), pp. 241–250. DOI: `10.1109/TVCG.2016.2598495`.

[SB17] Georg Steinbuss and Klemens Böhm. "Hiding Outliers in High-Dimensional Data Spaces". In: *International Journal of Data Science and Analytics* 4.3 (2017), pp. 173–189. DOI: `10.1007/s41060-017-0068-8`.

[SC00] Greg Schohn and David Cohn. "Less is More - Active Learning with Support Vector Machines". In: *International Conference on Machine Learning (ICML).* 2000.

[Sch+01] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. "Estimating the Support of a High-Dimensional Distribution". In: *Neural Computation* 13.7 (2001), pp. 1443–1471. DOI: `10.1162/089976601750264965`.

[Sch+19] Katrin Schulz, Stephan Kreis, Holger Trittenbach, and Klemens Böhm. "Data-driven Crack Assessment Based on Surface Measurements". In: *Engineering Fracture Mechanics* 218 (2019), p. 106552. DOI: `10.1016/j.engfracmech.2019.106552`.

[Sco15] David Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons, 2015.

[Set11] Burr Settles. "From Theories to Queries: Active Learning in Practice". In: *International Conference on Artificial Intelligence and Statistics (AISTATS) Workshops.* 2011, pp. 1–18.

[Set12] Burr Settles. "Active Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2012), pp. 1–114. 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: *Annual Meeting on Association for Computational Linguistics (ACL).* 2004, p. 589. DOI: 10.3115/1218955.1219030.

[Sid+18] Md Amran Siddiqui, Alan Fern, Thomas G Dietterich, Ryan Wright, Alec Theriault, and David W Archer. "Feedback-Guided Anomaly Discovery via Online Optimization". In: *International Conference on Knowledge Discovery and Data Mining (KDD).* ACM. 2018, pp. 2200–2209. DOI: 10.1145/3219819.3220083.

[Sil18] Bernard W Silverman. *Density Estimation for Statistics and Data Analysis.* Routledge, 2018. DOI: 10.1201/9781315140919.

[Soh+18] Fahad Sohrab, Jenni Raitoharju, Moncef Gabbouj, and Alexandros Iosifidis. "Subspace Support Vector Data Description". In: *International Conference on Pattern Recognition (ICPR).* IEEE. 2018, pp. 722–727. DOI: 10.1109/icpr.2018.8545819.

[SOS92] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. "Query by Committee". In: *Proceedings of the fifth annual workshop on Computational learning theory.* ACM. 1992, pp. 287–294. DOI: 10.1145/130385.130417.

[Spe+18] Fabian Sperrle, Jürgen Bernard, Michael Sedlmair, Daniel Keim, and Mennatallah El-Assady. "Speculative Execution for Guided Visual Analytics". In: *VIS Workshops.* 2018.

[Sta+18] Philipp Staudt, Yannick Träris, Benjamin Rausch, and Christof Weinhardt. "Predicting Redispatch in the German Electricity Market using Information Systems based on Machine Learning". In: *International Conference on Information Systems (ICIS).* 2018.

[Sto+08] Jack W Stokes, John C 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 and Computer Sciences* 24.1 (2016), pp. 298–312. DOI: 10.3906/elk-1307-137.

[SZ05] Xuehua Shen and ChengXiang Zhai. "Active Feedback in Ad Hoc Information Retrieval". In: *Conference on Research and Development in Information Retrieval (SIGIR).* ACM. 2005, pp. 59–66. DOI: 10.1145/1076034.1076047.

[TB19a] Holger Trittenbach and Klemens Böhm. "Dimension-Based Subspace Search for Outlier Detection". In: *International Journal of Data Science and Analytics* 7.2 (2019), pp. 87–101. DOI: 10.1007/s41060-018-0137-7.

[TB19b] 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`.

[TBA19] Holger Trittenbach, Klemens Böhm, and Ira Assent. “Active Learning of SVDD Hyperparameter Values”. In: *arXiv* (2019). arXiv: `1912.01927`.

[TBB18] Holger Trittenbach, Jakob Bach, and Klemens Böhm. “On the Tradeoff between Energy Data Aggregation and Clustering Quality”. In: *International Conference on Future Energy Systems (e-Energy)*. ACM. 2018, pp. 399–401. DOI: `10.1145/3208903.3212038`.

[TBB19] Holger Trittenbach, Jakob Bach, and Klemens Böhm. “Understanding the Effects of Temporal Energy-Data Aggregation on Clustering Quality”. In: *it-Information Technology* 61.2-3 (2019), pp. 111–123. DOI: `10.1515/itit-2019-0014`.

[TC10] Chih-Fong Tsai and Ming-Lun Chen. “Credit Rating by Hybrid Machine Learning Techniques”. In: *Applied soft computing* 10.2 (2010), pp. 374–380. DOI: `10.1016/j.asoc.2009.08.003`.

[TD01] David MJ Tax and Robert PW Duin. “Uniform Object Generation for Optimizing One-class Classifiers”. In: *Journal of Machine Learning Research (JMLR)* 2 (2001), pp. 155–173.

[TD04] David MJ Tax and Robert PW Duin. “Support Vector Data Description”. In: *Machine Learning* 54.1 (2004), pp. 45–66. DOI: `10.1023/B:MACH.0000008084.60811.49`.

[TD13] Andreas Theissler and Ian Dear. “Autonomously Determining the Parameters for SVDD with RBF Kernel from a One-Class Training Set”. In: *World Academy of Science, Engineering and Technology (WASET)* (2013), p. 732. DOI: `10.5281/zenodo.1087117`.

[TEB18] Holger Trittenbach, Adrian Englhardt, and Klemens Böhm. “An Overview and a Benchmark of Active Learning for Outlier Detection with One-Class Classifiers”. In: *arXiv* (2018). arXiv: `1808.04759`.

[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.

[Thi+15] Patrick Thiam, Markus Kächele, Friedhelm Schwenker, and Guenther Palm. “Ensembles of Support Vector Data Description for Active Learning Based Annotation of Affective Corpora”. In: *Symposium Series on Computational Intelligence*. IEEE. 2015, pp. 1801–1807. DOI: `10.1109/SSCI.2015.251`.

[TK18] Stefano Teso and Kristian Kersting. ““Why Should I Trust Interactive Learners?” Explaining Interactive Queries of Classifiers to Users”. In: *arXiv* (2018). arXiv: `1805.08578`.

[TK19] Stefano Teso and Kristian Kersting. "Explanatory Interactive Machine Learning". In: *Conference on Artificial Intelligence (AAAI).* 2019. DOI: 10.1145/3306618.3314293.

[TLD05] Quang-Anh Tran, Xing Li, and Haixin Duan. "Efficient Performance Estimate for One-Class Support Vector Machine". In: *Pattern Recognition Letters* 26.8 (2005), pp. 1174–1182. DOI: 10.1016/j.patrec.2004.11.001.

[TM04] David MJ Tax and K-R Muller. "A consistency-based model selection for one-class classification". In: *International Conference on Pattern Recognition (ICPR).* Vol. 3. IEEE. 2004, pp. 363–366. DOI: 10.1109/ICPR.2004.1334542.

[Tri+18] Holger Trittenbach, Martin Gauch, Klemens Böhm, and Katrin Schulz. "Towards Simulation-Data Science–A Case Study on Material Failures". In: *International Conference on Data Science and Advanced Analytics (DSAA).* IEEE. 2018, pp. 450–459. DOI: 10.1109/dsaa.2018.00058.

[Ver+18] Vincent Vercruyssen, Meert Wannes, Verbruggen Gust, Maes Koen, Bäumer Ruben, and Davis Jesse. "Semi-supervised Anomaly Detection with an Application to Water Analytics". In: *International Conference on Data Mining (ICDM).* IEEE. 2018. DOI: 10.1109/icdm.2018.00068.

[Vol+19] Michael Vollmer, Adrian Englhardt, Holger Trittenbach, Pawel Bielski, Shahab Karrari, and Klemens Böhm. "Energy Time-Series Features for Emerging Applications on the Basis of Human-Readable Machine Descriptions". In: *International Conference on Future Energy Systems (e-Energy).* ACM. 2019, pp. 474–481. DOI: 10.1145/3307772.3331022.

[Wan+13] Shijin Wang, Jianbo Yu, Edzel Lapira, and Jay Lee. "A Modified Support Vector Data Description Based Novelty Detection Approach for Machinery Components". In: *Applied Soft Computing* 13.2 (2013), pp. 1193–1205. DOI: 10.1016/j.asoc.2012.11.005.

[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.

[Wei+16] Gary M Weiss, Jessica L Timko, Catherine M Gallagher, Kenichi Yoneda, and Andrew J Schreiber. "Smartwatch-based Activity Recognition: A Machine Learning Approach". In: *International Conference on Biomedical and Health Informatics (BHI).* IEEE. 2016, pp. 426–429. DOI: 10.1109/bhi.2016.7455925.

[Wer+19] Dominik Werle, Daniel Warzel, Simon Bischof, Anne Koziolek, Holger Trittenbach, and Klemens Böhm. "The Effect of Temporal Aggregation on Battery Sizing for Peak Shaving". In: *International Conference on Future Energy Systems (e-Energy) Workshops.* ACM. 2019, pp. 482–485. DOI: 10.1145/3307772.3331023.

[Wil17] Leland Wilkinson. "Visualizing Big Data Outliers through Distributed Aggregation". In: *Visualization and Computer Graphics* 24.1 (2017), pp. 256–266. DOI: 10.1109/tvcg.2017.2744685.

[WZT15] Tinghua Wang, Dongyan Zhao, and Shengfeng Tian. "An Overview of Kernel Alignment and its Applications". In: *Artificial Intelligence Review* 43.2 (2015), pp. 179–192. DOI: 10.1007/s10462-012-9369-4.

[Xia+14] Yingchao Xiao, Huangang Wang, Lin Zhang, and Wenli Xu. "Two Methods of Selecting Gaussian Kernel Parameters for One-Class SVM and their Application to Fault Detection". In: *Knowledge-Based Systems* 59 (2014), pp. 75–84. DOI: 10.1016/j.knosys.2014.01.020.

[XWX14] Yingchao Xiao, Huangang Wang, and Wenli Xu. "Parameter Selection of Gaussian Kernel for One-Class SVM". In: *Transactions on Cybernetics* 45.5 (2014), pp. 941–953. DOI: 10.1109/TCYB.2014.2340433.

[YL18] Yazhou Yang and Marco Loog. "A Benchmark and Comparison of Active Learning for Logistic Regression". In: *Pattern Recognition* 83 (2018), pp. 401–415. DOI: 10.1109/TVCG.2017.2744685.

[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.

[ZCS14] Arthur Zimek, Ricardo JGB Campello, and Jörg Sander. "Ensembles for Unsupervised Outlier Detection: Challenges and Research Questions a Position Paper". In: *SIGKDD Explorations Newsletter* 15.1 (2014), pp. 11–22. DOI: 10.1145/2594473.2594476.

[ZGW06] Ji Zhang, Qigang Gao, and Hai Wang. "A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm". In: *International Conference on Data Mining (ICDM)*. IEEE. 2006, pp. 731–740. DOI: 10.1109/ICDM.2006.6.

[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.

[Zim19] Albrecht Zimmermann. "Method Evaluation, Parameterization, and Result Validation in Unsupervised Data Mining: A Critical Survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2019), e1330. DOI: 10.1002/widm.1330.

[ZS17] Arthur Zimek and Erich Schubert. "Outlier Detection". In: *Encyclopedia of Database Systems*. Springer, 2017, pp. 1–5. DOI: 10.1007/978-1-4899-7993-3_80719-1.

[ZSK12] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data". In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387. DOI: 10.1002/sam.11161.

[ZTH11] Jinfeng Zhuang, Ivor W Tsang, and Steven CH Hoi. "A Family of Simple Non-Parametric Kernel Learning Algorithms". In: *Journal of Machine Learning Research (JMLR)* 12.Apr (2011), pp. 1313–1347.