An Overview and a Benchmark of Active Learning for One-Class Classification

Holger Trittenbach
Karlsruhe Institute of Technology
holger.trittenbach@kit.edu

Adrian Englhardt
Karlsruhe Institute of Technology
adrian.englhardt@kit.edu

Klemens Böhm
Karlsruhe Institute of Technology
klemens.boehm@kit.edu

Abstract—Active learning stands for methods which increase classification quality by means of user feedback. An important subcategory is active learning for one-class classifiers, i.e., for imbalanced class distributions. While various methods in this category exist, selecting one for a given application scenario is difficult. This is because existing methods rely on different assumptions, have different objectives, and often are tailored to a specific use case. All this calls for a comprehensive comparison, the topic of this article.

This article starts with a categorization of the various methods. We then propose ways to evaluate active learning results. Next, we run extensive experiments to compare existing methods, for a broad variety of scenarios. One result is that the practicality and the performance of an active learning method strongly depend on its category and on the assumptions behind it. Another observation is that there only is a small subset of our experiments where existing approaches outperform random baselines. Finally, we show that a well-laid-out categorization and a rigorous specification of assumptions can facilitate the selection of a good method for one-class classification.

I. INTRODUCTION

Active learning methods involve users in machine learning tasks, by asking for ancillary information, such as class labels. Examples of active learning come from various domains, such as energy systems monitoring [1] or natural language processing [2], where ancillary information increases the result quality of learning algorithms. Naturally, providing such information requires time and intellectual effort of the users. With large volumes of data, one must allocate such resources efficiently. Research on active learning focuses on strategies to boost result quality with minimal user effort.

Important applications of active learning are settings with highly imbalanced class distributions. Examples are network security [3], [4] or fault monitoring [5] where observations of breaches or catastrophic failures are rare to non-existent. In this case, one can use one-class classifiers, which learn to identify whether objects belong to a majority class.

Class imbalance has important implications on active learning. Well-established concepts, like the margin between two classes or density estimation for the minority class, are no longer applicable. This has motivated research on active learning specific to one-class classification [6]–[11]. However, as we will show, these methods differ in their objectives and in the assumptions behind them. In addition, evaluation of active learning methods in that specific context may lack reliability and comparability [12]. 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 as well as to assess novel contributions in this field.

Difficulties. Summarizing the current state of active learning for one-class classification is difficult for two reasons. First, we are not aware of any 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 [13]. In general, active learning methods may perform differently with 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.

Knowing the range of objectives and assumptions is crucial to assess one-class active learning. Related work however may 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, it is unclear how to report active learning results on a large body of benchmark data. The reason is that “quality” can have several meanings with active learning.

Example 1. Figure 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 1 plots two active learning methods A and B from an initial state $t_{\text{init}}$ to the final iteration $t_{\text{end}}$. Both methods apparently have different strengths. A yields better quality at $t_{\text{init}}$, while B improves faster in the first few iterations. However, quality increases non-monotonically, because feedback can bias the classifier temporarily. At $t_{\text{end}}$, 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 method with higher quality at $t_{\text{end}}$. However, the choice of $t_{\text{end}}$ is arbitrary, and one can easily 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.

**Contributions.** In this article, we review, categorize and evaluate active learning methods for one-class classification, as follows. (1) We propose learning scenarios, i.e., combinations of a learning objective and an initial setup, to classify the various methods. (2) We review existing methods and introduce a common notation. One important insight we gain from our categorization is that some methods only are compatible with a limited number of learning scenarios. (3) We propose several complementary ways to summarize progress curves. We use these metrics to (4) conduct and evaluate a comprehensive benchmark with around 18,900 combinations of learning scenarios, classifiers, and active learning methods. We make the implementation of our benchmark suite and the raw results publicly available.1

One important takeaway from our study is that there is no universal set of rules to select good active-learning methods for one-class classifiers. Although establishing such rules is desirable, our benchmark suggests that this is not feasible for state-of-the-art methods, as we will explain in Section IV-B5. A consequence is that the selection of a good method is use-case specific. We therefore (5) report on a case study we have carried out to illustrate how to decide on a query strategy for a given scenario. The study shows that a rigorous specification of assumptions and the use of progress-curve summaries do facilitate a good selection.

A further insight from our benchmark is that the learning scenario has a significant impact on classification quality. Beyond that, we have been somewhat surprised that random baselines outperform more sophisticated methods in many cases. Therefore, for novel active-learning methods for one-class classifiers, we recommend to explicitly specify the learning scenario and to compare against random baselines.

**II. ONE-CLASS ACTIVE LEARNING**

Active learning (AL) for one-class classification is an interactive system to train and update classifiers. There are different ways to design one-class AL 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 [2], [14], [15] and on benchmarks for balanced [16] and multi-class classification [10].

In this section we discuss assumptions for one-class AL, structure the aspects in which one-class AL systems differ from each other and discuss implications of design choices on the AL system. We structure our discussion into three building blocks that constitute a one-class AL system. The first block is the *learning scenario*, which establishes assumptions regarding the training data and the process of gathering user feedback. A learning scenario specifies the initial setup of the system before the actual active learning starts. It also specifies how to evaluate the result. 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*. It is a method to select observations that a user is asked to provide feedback for.

We call observations that a query strategy selects *query objects*, the entity that provides the label an *oracle*, and the process of providing label information *feedback*. In a real scenario, the oracle is a user. For benchmarks, the oracle is simulated, based on a given gold standard.

Figure 2 graphs the building blocks. In the following, we explain the blocks and discuss dependencies between them.

**A. Learning Scenario**

Researchers make assumptions regarding the interaction between system and user as well as assumptions regarding the application scenario. Literature on one-class AL 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. We distinguish between two types: general and specific assumptions.

1) **General Assumptions:** General assumptions specify modalities of the feedback, independent of the application. They typically impose limits on how applicable AL is in real settings. These assumptions have been discussed for standard

1https://www.ipd.kit.edu/ocal
binary classification [14], and many of them are accepted in the literature. We highlight the ones important for one-class AL.

**Feedback Type:** Existing one-class AL methods assume that the feedback is a class label, i.e., the decision whether an observation belongs to a class or not. However, other types of feedback certainly are conceivable as well, such as feature importance [17], [18]. But to our knowledge, research on one-class AL has been limited to label feedback. Next, the most common mechanism in literature is sequential feedback, i.e., for one observation at a time. However, asking for feedback in batches might have certain advantages, e.g., increased efficiency of the labeling process. However, a shift from sequential to batch queries is not trivial and requires additional diversity criteria [19]. We focus on sequential feedback in this current article.

**Feedback Budget:** A primal motivation for active learning is that the amount of feedback a user can provide is bounded. For instance, he can have a fixed time or cost budget or a limited attention span to interact with the system. Assigning costs to feedback acquisition is difficult, and a reasonable budget restriction is likely to be application-specific. In some cases, feedback on observations from the minority class may be more costly. 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 he 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 [20], and it has only recently been studied for interactive learning [21], [22]. Concepts to support users with an explanation of outliers [23], [24] have not been studied in the context of active learning either. In any case, a thorough evaluation would require a user study. However, existing one-class AL systems bypass the difficulty of interpretation and assume a perfect oracle, i.e., an oracle which provides feedback with a predefined accuracy.

2) **Specific Assumptions:** Specific assumptions specify the learning objective and the data for a particular AL 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 are 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 [8], [9]. 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 our paper, and we stick to it in the following.

Under the first definition, one can further differentiate between characterizations of outliers. 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 common 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 distribution of several rare-class distributions. In this case, a probability density for each mixture component exists. In consequence, the probability density for the mixture as a whole exists as well. However, the estimation of this probability density is hard, because the sample for each component is tiny. The characteristics of the outlier distribution has implications on the separation of the data into train and test partitions, as we will explain in category **Split Strategy**.

**Learning Objective:** The learning objective is the benefit expected from an AL system. The most common learning objective is to improve the accuracy of a classifier. However, there are alternatives. For instance, users of one-class classification often have a specific interest in the minority class [13]. In this case, it is reasonable to assume that users prefer giving feedback on minority observations if they will examine them anyhow later on. Consequently, a good active learning method yields a high proportion of queries from the minority class. This may contradict the objective of accuracy improvement.

There are also 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 [7], [8]. A classifier benefits from extending the number of majority-class labels. Consequently, this learning objective favors active learning methods that select observations from the majority class.

**Initial Setup:** The initial setup is the label information available at the beginning of the AL 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 [19]. 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 [8]. In our article, we consider different initial setups.

- (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.
- (Pa) Pool attributes: As many labeled inliers as number of attributes.
The rationale behind $P_4$ is that the correlation matrix of labeled observations is singular if there are fewer labeled observations than attributes. With a singular correlation matrix, some of the query strategies are infeasible.

**Split Strategy:** To train binary classifiers, one typically partitions the data into a train and a test set, which ideally are identically distributed. However, the assumptions regarding the class distribution also determine which data splits are rational. If outliers do not follow a common 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. The resubstitution error is an optimistic estimate of classification quality. However, 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. If the learning objective is to reliably estimate the majority-class distribution, one may restrict the training set to inliers (cf. [7], [8]). 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.

In practice, the application determines which specific assumptions take effect. We call the combination of an initial setup and a split strategy a **learning scenario**. Figure 3 is an overview of the learning scenarios discussed in this section.

### B. Notation

Before we introduce the remaining two building blocks, we specify some notation. $\mathcal{X} \subseteq \mathbb{R}^m$ is a data space with $m$ attributes. $X$ is a sample from $\mathcal{X}$ of $n$ observations $\{x_1, x_2, \ldots, x_n\}$, where each observation $x_i$ is a vector of $m$ attribute values, i.e., $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$. In this article, each observation either belongs to the minority or to the majority class. For brevity, we call an observation from the minority class **outlier** and one from the majority class **inlier**, and we encode them with a categorical class label $y_i \in \{\text{in}, \text{out}\}$. Common synonyms for inlier are “target”, “positive observation” or “regular observation”, and for outlier “anomalous observation” or “exceptional observation”. $X$ can be partitioned into the unlabeled set of observations $U$, i.e., observations where $y_i$ is unknown, and the labeled set of observations $L$ where $y_i$ is known. We distinguish between the labeled inliers $L_{\text{in}}$ and the labeled outliers $L_{\text{out}}$.

### C. Base Learner

A base learner is a one-class classifier that discerns between inliers and outliers. It takes a set of observations and a set of class labels as input and returns a decision function, as follows.

**Definition 1 (Decision Function).** A **decision function** $f$ is a function of type $f: \mathcal{X} \rightarrow \mathbb{R}$. An observation is assigned to the minority class if $f(x) > 0$ and to the majority class otherwise.

One-class classifiers fall into two categories: support-vector methods and non-support-vector classifiers [25]. In our article, we focus on support-vector methods, the prevalent choice as base learners for one-class AL. However, the query strategy is independent from a specific instantiation, as long as the base learner returns a decision function.

One can further distinguish between semi-supervised and unsupervised one-class classifiers. Both of them have been used with one-class AL, but whether they are applicable depends on the learning scenario. A **semi-supervised** base learner uses both unlabeled data and labeled data with class labels for training. Labels can either come from both classes or only from the minority class [26]. An **unsupervised** base learner does not have any mechanism to use label information directly to train the decision function. Instead, one can manipulate the unsupervised base learner by changing the training data, e.g., by restricting it to labeled inliers only. This limits unsupervised base learners to scenarios that strive for a robust estimate of the majority distribution, and to split strategy Si. Figure 4 illustrates both scenarios.

In our evaluation, we use three one-class classifiers: unsupervised SVDD [26], semi-supervised SVDDneg [26] with labels from the minority class, and the semi-supervised SSAD [11] with labels from both classes. In the following, we briefly review their principles.

---

![Diagram](image-url)
1) Support Vector Data Description (SVDD): One of the most popular support-vector methods is Support Vector Data Description (SVDD) [26]. The core idea is to fit a sphere around the data that encompasses all or most observations. This can be expressed as a Minimum Enclosing Ball (MEB) optimization problem of the following form

\[ \begin{align*}
\text{minimize} & \quad R^2 + C \sum_{i=1}^{N} \xi_i \\
\text{subject to} & \quad \|\phi(x_i) - a\| \leq R + \xi_i, \ i = 1, \ldots, n. 
\end{align*} \] (1)

with the center of the ball \(a\), the radius \(R\), slack variables \(\xi_i\), a cost parameter \(C \in [0, 1]\), and a function \(\phi: \mathcal{X} \to \mathcal{F}\) which maps \(x\) into a reproducing kernel Hilbert space \(\mathcal{F}\). Solving the optimization problem gives a fixed \(a\) and \(R\) and the decision function

\[ f(x_i) = \|\phi(x_i) - a\| - R. \] (2)

The slack variables \(\xi_i\) relax the MEB, i.e., they introduce a trade-off to allow observations to fall outside the sphere at cost \(C\). If \(C\) is high, observations falling outside of the sphere are expensive. In other words, \(C\) controls the share of objects that are outside the decision boundary.

The optimization problem from Equation 1 can be solved in the dual space. In this case, the problem contains only inner products of the form \(\langle \phi(x), \phi(x') \rangle\). This allows to use the kernel trick, i.e., to replace the inner products with a kernel function \(k(x, x') \to \mathbb{R}, x, x' \in \mathcal{X}\). A common kernel is the Radial Basis Function (RBF) Kernel

\[ k_{\text{RBF}}(x, x') = e^{-\gamma \|x-x'\|}, \] (3)

with parameter \(\gamma\).

2) SVDD with Negative Examples (SVDDneg): SVDDneg [26] 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.

\[ \begin{align*}
\text{minimize} & \quad R^2 + C_1 \cdot \sum_{i : x_i \in \mathcal{U} \cup \mathcal{L}_{\text{in}}} \xi_i + C_2 \cdot \sum_{i : x_i \in \mathcal{L}_{\text{out}}} \xi_i \\
\text{subject to} & \quad \|\phi(x_i) - a\| \leq R^2 + \xi_i, \ i : x_i \in \mathcal{U} \cup \mathcal{L}_{\text{in}} \\
& \quad \|\phi(x_i) - a\| \geq R^2 - \xi_i, \ i : x_i \in \mathcal{L}_{\text{out}} \\
& \quad \xi_i \geq 0, \ i = 1, \ldots, n. 
\end{align*} \] (4)

3) Semi-Supervised Anomaly Detection (SSAD): SSAD [11] 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 \(\kappa\).

High values of \(\kappa\) increase the weight of \(\mathcal{L}\) on the solution, i.e., SSAD is more likely to overfit to instances in \(\mathcal{L}\).

\[ \begin{align*}
\text{minimize} & \quad a, R, \xi, \tau \quad R^2 - \kappa T + C_1 \cdot \sum_{i : x_i \in \mathcal{U}} \xi_i + C_2 \cdot \sum_{i : x_i \in \mathcal{L}_{\text{in}}} \xi_i \\
\text{subject to} & \quad \|\phi(x_i) - a\| \leq R^2 + \tau, \ i : x_i \in \mathcal{U} \\
& \quad \|\phi(x_i) - a\| \geq R^2 - \xi_i + \tau, \ i : x_i \in \mathcal{L}_{\text{out}} \\
& \quad \xi_i \geq 0, \ i = 1, \ldots, n. 
\end{align*} \] (5)

Under mild assumptions, SSAD can be reformulated as a convex problem [11].

Parameterizing the kernel function and cost parameters is difficult, because a good parameterization typically depends on the data characteristics and the application. There are several strategies to find a good parametrization that use data characteristics [27]–[29], artificial outliers [30]–[32], or SVDD-specific properties like the number of support vectors [33]. However, optimizing the parametrization is not a focus of this article, and we rely on widely used heuristics to select the kernel and the cost parameters, see Section IV.

D. Query Strategies

A query strategy is a method that selects observations for feedback. In this section, we elaborate on their basic principles, and we review existing query strategies for one-class classification. We have found that the varying notation in the literature would make such an overview difficult to follow. Thus, we rely on notation introduced earlier.

To decide on observations for feedback, query strategies rank unlabeled observations according to an informativeness measure.

Definition 2 (Informativeness). 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 \(x \in \mathcal{U} \cup \mathcal{L}\) to \(\mathbb{R}\).

For brevity, we only write \(\tau(x)\), as \(\mathcal{U}, \mathcal{L}, f\) are fix during one active learning iteration. \(\tau(x)\) quantifies how valuable feedback for observation \(x \in \mathcal{U}\) is for the classification model. This definition is general, and there are different ways to interpret valuable. Feedback can be valuable if the model is uncertain with the prediction of an observation, or if the classification error is expected to decrease. Some query strategies also balance between the representativeness of observations and the exploration of sparse regions. In this case, local density estimates affect the value of an observation.

A query strategy selects one or more observations based on their informativeness.

Definition 3 (Query Strategy). A query strategy \(QS\) is a function of type

\[ QS : \mathcal{U} \times \mathbb{R} \to \mathcal{Q} \]

with \(\mathcal{Q} \subseteq \mathcal{U}\).
The feedback on Q from an oracle results in an updated set of labeled \( L' = L \cup Q \) and unlabeled data \( U' = U \setminus Q \). To select Q, query strategies rank unlabeled observations according to an informativeness measure. In this current article, we only consider single queries (cf. Section II-A). Given this, we assume query strategies to always return the observation with the highest informativeness

\[
Q = \arg \max_{x \in U} \tau(x).
\]  

(6)

If necessary, we will adapt definitions from literature so that they align with Equation 6.

We now review existing query strategies from literature. To this end, we partition them into three categories. The first category is data-based query strategies. 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.

1) 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 a well-known method from binary classification and is referred to as measures of uncertainty [14]. If a classifier does not explicitly return posterior probabilities, one can use the Bayes rule to infer them. However, 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. This proportion 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 of data-based strategies deems observations informative if the classifier is uncertain about their class label. These are the observations with equal probability of being classified as inlier and as outlier. The following two strategies quantify informativeness in this way.

Minimum Margin [7]: This QS relies on the difference between posterior class probabilities

\[
\tau_{\text{MM}}(x) = -|p(\text{in}|x) - p(\text{out}|x)|
\]

\[
= -\frac{p(x|\text{in}) \cdot p(\text{in}) - p(x|\text{out}) \cdot p(\text{out})}{p(x)}
\]

\[
= -\frac{2 \cdot p(x|\text{in}) \cdot p(\text{in}) - p(x)}{p(x)}.
\]

(7a)

(7b)

(7c)

where Equation 7b and Equation 7c follow from the Bayes rule. If \( p(\text{in}) \) and \( p(\text{out}) \) are known, one can make direct use of Equation 7c. 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

\[
\tau_{\text{EMM}}(x) = \mathbb{E}_{p(\text{out})}(\tau_{\text{MM}}(x))
\]

\[
= \left(\frac{p(x|\text{in})}{p(x)} - 1\right) \cdot \text{sign} \left(0.5 - \frac{p(x|\text{in})}{p(x)}\right).
\]

(8)

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

Maximum-Entropy [7]: This QS selects observations where the distribution of the class probability has a high entropy

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

(9)

Applying the Bayes rule and taking the expected value similarly to Equation 8 gives

\[
\tau_{\text{EME}}(x) = \mathbb{E}_{p(\text{out})}(\tau_{\text{ME}}(x))
\]

\[
= \frac{\left(\frac{p(x|\text{in})}{p(x)}\right)^2 \cdot \log \left(\frac{p(x|\text{in})}{p(\text{in})}\right) + p(x|\text{in})}{2 \cdot \frac{p(x|\text{in})}{p(x)}}
\]

\[+ \frac{\left(\frac{p(x|\text{in})}{p(x)} - 1\right)^2 \cdot \log \left(1 - \frac{p(x|\text{in})}{p(\text{in})}\right)}{2 \cdot \frac{p(x|\text{in})}{p(x)}}.\]

(10)

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 5 visualizes \( \tau_{\text{MM}} \), \( \tau_{\text{EMM}} \) and \( \tau_{\text{EME}} \) for univariate data generated from two Gaussian distributions, with \( p(\text{out}) = 0.1 \). The authors of \( \tau_{\text{EME}} \) suggest to estimate the densities with kernel density estimation (KDE) [7]. However, entropy is defined on probabilities and is not applicable to densities. Moreover, \( \tau_{\text{EME}} \) is not defined for \( \frac{p(x|\text{in})}{p(x)} \geq 1 \). We set \( \tau_{\text{EME}} = 0 \) in this case. For \( \tau_{\text{MM}} \), we use Equation 7c 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** [8]: Under the minimum-loss strategy, observations have high informativeness if they are expected to improve the KDE 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 7c, this requires knowledge of the prior class probabilities.

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

Case 1: $x$ is inlier

$$\tau_{ML\text{-}in}(x) = \frac{1}{|\mathcal{L}_{in}^x|} \sum_{i : x_i \in \mathcal{L}_{in}^x} \hat{g}^{in,x}_{\mathcal{L}_{in}^x}(x_i) - \frac{1}{|\mathcal{L}_{out}|} \sum_{i : x_i \in \mathcal{L}_{out}} \hat{g}^{in,x}(x). \quad (11)$$

Case 2: $x$ is outlier

$$\tau_{ML\text{-}out}(x) = \frac{1}{|\mathcal{L}_{in}|} \sum_{i : x_i \in \mathcal{L}_{in}} \hat{g}^{in}_{\mathcal{L}_{in}}(x_i) - \frac{1}{|\mathcal{L}_{out}^x|} \sum_{i : x_i \in \mathcal{L}_{out}^x} \hat{g}^{in,x}(x). \quad (12)$$

The expected value over both cases is

$$\tau_{ML}(x) = p(in) \cdot \tau_{ML\text{-}in}(x) + (1 - p(in)) \cdot \tau_{ML\text{-}out}(x). \quad (13)$$

We illustrate Equation 11, Equation 12 and $\tau_{ML}$ in Figure 6. As expected, $\tau_{ML}$ yields high informativeness in regions of high inlier density. $\tau_{ML}$ gives an almost inverse pattern compared to the Minimum-Margin and the Maximum-Entropy strategies. This illustrates that existing query strategies are ambiguous. It is unclear how to decide between them solely based on theoretical considerations.

2) **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.

**High-Confidence** [6]: This QS selects observations that match the inlier class the least. For SVDD this is

$$\tau_{HC}(x) = f(x). \quad (14)$$

**Decision-Boundary**: This QS selects observation closest to the decision boundary

$$\tau_{DB}(x) = -|f(x)|. \quad (15)$$

3) **Hybrid Query Strategies**: Hybrid query strategies combine the approach of data-based and model-based strategies.

**Neighborhood-Based** [11]: This QS 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

$$\tilde{\tau}_{NB}(x) = - \left( 0.5 + \frac{1}{2k} \cdot |\{x' \in NN_k(x) : x' \in L_{in}\}| \right). \quad (16)$$

The strategy then combines this number with the distance to the decision boundary, i.e., $\tau_{NB} = \eta \cdot \tau_{DB} + (1 - \eta) \cdot \tilde{\tau}_{NB}$. Parameter $\eta \in [0, 1]$ controls the influence of the number of already labeled instances in the neighborhood on the decision. The authors do not recommend any specific parameter value, and we use $\eta = 0.5$ in our experiments.

**Boundary-Neighbor-Combination** [5]: The core of this query strategy is a linear combination of the normalized...
learning progress curves. We then review common quality numbers that result from a quality evaluation in each iteration. The reason is that the result of the evaluation of static methods. The reason is that the result of the evaluation of static methods. The reason is that the result of the evaluation of static methods.

In the remainder of our article, we only consider feasible building-block combinations. Table I is an overview of the query strategy feasibility. In addition to the strategies introduced so far, we use the following baselines.

Random: This QS draws each unlabeled observation with equal probability

\[ \tau_{\text{rand}}(x) = \frac{1}{|U|}. \]

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

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

E. Feasible Building-Block Combinations

In the previous subsections, we have introduced three building blocks: learning scenarios, consisting of an initial setup and a split strategy, base learners and query strategies (QS). However, one cannot combine these blocks arbitrarily, for the following reasons.

- A learning scenario with no labels (Pu) does not work with a split strategy that fits a model on inliers (Si).
- Pu rules out any data-based QS. This is because database QS require labeled observations for the density estimations.
- Kernel-density estimation requires the number of labeled observations to be at least equal to the number of attributes. A special case is inliers (Si).

which requires \( |L_{\text{outlier}}| \geq d \). As a remedy, one can omit the subtrahend in Equation 11 and Equation 12 in this case.

Table I is an overview of the query strategy feasibility. In the remainder of our article, we only consider feasible combinations.

III. Evaluation Methods

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

In the following, we discuss the 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. Finally, we discuss different ways to summarize active learning curves.

A. Progress Curves

The sequence of quality evaluations can be visualized as a progress curve, see Figure 1. We call the interval from \( t_{\text{init}} \) to \( t_{\text{end}} \) an active learning cycle. The x-axis of a progress curve quantifies the active learning progress. Literature tends to use the percentage or the absolute number of labeled observations to quantify progress. However, this percentage is 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. In general, there are two ways to evaluate this quality for imbalanced class distributions: either by computing a summary statistic on the binary confusion matrix, or by assessing the ranking induced by the decision function.

B. One-Class Evaluation Metrics

In this article, we use the Matthews Correlation Coefficient (MCC) and Cohen’s kappa to evaluate the binary output. Both metrics can be computed from the confusion matrix. MCC returns values in \([-1, +1]\), where high values indicate high classification quality 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 an agreement that is no 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. We use the area under the ROC curve (AUC) and its variant partial AUC (pAUC) to evaluate the ranking. 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. 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 [11].

C. Summary of the Active-Learning Curve

The visual comparison of active learning via progress plots does not scale well to a large number of experiments. For instance, our benchmark would require to compare 18,900 different learning curves. This is prohibitive. So we propose...
Influence of the last $k$ average QR between init and end LS as the ratio of the average QR in the last $k$ additional feedback does not improve the quality. We define hand, indicates that the classifier tends to be saturated, i.e., continuing the active learning cycle. A low LS, on the other hand, indicates that one can expect further improvement from the active learning cycle. A low LS, on the other hand, indicates that one can expect further improvement from the active learning cycle.

**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 $QR$($init, end$), 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 temporary quality decrease. The choice of $t_{end}$ often is arbitrary and can coincide with a temporary bias. Therefore, 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 improve 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.

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

**Average Quality Gain (AQG) and Average Quality Loss (AQL):** The Average Quality Gain and Loss indicate how much a short term bias affects the classifier. There is only a low or no short term bias for models with high AQG and low AQL. High AQG that co-occurs with high AQL indicates that changes to affect model quality significantly. We define the AQG as the average of strictly positive increases in the progress curve.

$$AQG = \frac{1}{|inc|} \sum_{i \in inc} QM(i) - QM(i - 1),$$

where $inc = \{i \in [init + 1, end] : QM(i) - QM(i - 1) > 0\}$. AQL is defined similarly for strictly negative decreases.

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

$$ROQ = \frac{|L_{end - out}|}{|L_{end}|}$$

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.

A further hypothesis is that these metrics may help to select different query strategies for different phases of the active learning cycle. For instance, one could start the active learning 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 current article.

### IV. Experiments and Results

The plethora of ways to design and to evaluate AL 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 all of them. This section therefore does provide some guidance how to navigate this overwhelming design space.

We begin by explaining our experiments that we have conducted on well-established benchmark data sets for outlier detection [34]. In total, we run experiments on over 18,900 configurations. Our analysis of the results consists of two parts. In the first part, we use the summary metrics introduced in Section III to arrive at general insights and trends regarding the three building blocks. A result from this analysis is a small subset of configurations where query strategies perform well.

### Table I

**Overview over the number of labels required by different query strategies:** $m$ is the number of attributes. Feasible: $\checkmark$, Feasible with modification: $(\checkmark)$, Not feasible: $\times$.

| Scenario | $\tau_{MM}$ | $\tau_{EME}$ | $\tau_{EME}$ | $\tau_{ML}$ | $\tau_{HC}$ | $\tau_{DB}$ | $\tau_{NB}$ | $\tau_{INC}$ | $\tau_{rand}$ | $\tau_{rand-out}$ |
|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|----------------|
| $|L_{in}| > m \land |L_{out}| = 0$ | $\times$ | $\times$ | $\times$ | $\times$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| $|L_{in}| \geq m \land |L_{out}| = 0$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| $|L_{in}| \geq m \land |L_{out}| \geq m$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
In the second part, we report on a case study to select a good configuration for a specific application.

A. Experimental Setup

Our experiments cover several instantiations of the building blocks. Table II lists the data sets, and Table III 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 \( n = 2000 \) because of experiment runtimes. In the following, we refer to the combination of a data set and a specific experimental setup as a setting.

**Parameters:** Parameter selection for base learners and query strategies is difficult, because of the large number of possible combinations. We use heuristics to select the kernel and cost parameters for the base-learners (cf. Section II-C). In preliminary experiments, we have observed that Scott’s rule of thumb [28] for the kernel parameter \( \gamma \) and the initialization strategy of Tax et al. [26] for the cost parameter \( C \) produce reasonable results. For SSAD, the authors suggest to set the trade-off parameter \( \kappa = 1 \) [11]. 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, selection of strategy-specific parameters is described in Section II-D. The data-based query strategies use the same \( \gamma \) for kernel density estimation as the base learner.

**Termination:** The number of queries in an active learning cycle depends on the application, and there is no general rule how to select a good number. In our experiments, we perform 50 iterations. In Section IV-B5, we increase the number of iterations to 100 to illustrate how the number of iterations can influence the selection of the query strategy.

**Implementation:** We have implemented the base learners, the query strategies and the benchmark setup in Julia [35]. Our implementation, the raw results of all settings and notebooks to reproduce experiments and evaluation are publicly available at https://www.ipd.kit.edu/ocal.

### Table II

**Overview on Data Sets.**

| Dataset     | Observations (n) | Attributes (m) |
|-------------|------------------|----------------|
| Annthyroid  | 2000             | 22             |
| Cardiotocography | 1734          | 22             |
| HeartDisease | 157             | 14             |
| Hepatitis   | 70               | 20             |
| PageBlocks  | 2000             | 11             |
| Parkinson   | 50               | 23             |
| Pima        | 526              | 9              |
| SpamBase    | 2000             | 58             |
| Stamps      | 325              | 10             |

### Table III

**Overview on Experimental Setup.**

| Experiment Dimension | Configuration |
|----------------------|---------------|
| Initial Pools        | \( P_u, P_p (p = 0.1), P_n (n = 20), P_a \) |
| Split Strategy       | Sh, Sf (80% train, 20% test), Si |
| Base Learner         | SVDD, SVDDneg, SSAD \( (\kappa = 1.0, 0.5, 0.1) \) |
| Query strategy       | \( \tau_{MM}, \tau_{MMM}, \tau_{EMM}, \tau_{HC}, \tau_{DB}, \tau_{NB}, \tau_{RNC}, \tau_{rand}, \tau_{rand-out} \) |

To conclude, we think that pAUC and AUC may be misleading in evaluating one-class classification. Therefore, we will use only MCC in the remainder of the evaluation.

1) **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). A comparison on all settings shows that metrics of the same type have a high correlation for AEQ. So we simplify the evaluation by selecting one metric of each type.

Further, there is an important difference between both types. Figure 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 highlighted part of the plot. 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. However, the classifier actually may not have fit a good decision boundary.

To conclude, we think that pAUC and AUC may be misleading in evaluating one-class classification. Therefore, we will use only MCC in the remainder of the evaluation.

2) **Split Strategies:** Split strategies have a significant influence on classification quality. Figure 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 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 AEQ close to 0 in Sh, e.g., for SVDDneg, rather suggests that outliers do not follow
a homogeneous distribution (cf. Section II-A). 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 classification quality is low because the training split only contains a small fraction of the inliers. Based on these observations, we question whether Si is suitable to evaluate outlier detection. Therefore, we exclude Si from the subsequent analysis.

Next, we compare the quality of the base learners. SVDD only reports a positive AEQ for Si. For Sh and Sf, SVDD fails because it is fully unsupervised, i.e., cannot benefit from feedback. For SSAD, the quality fluctuates with increasing parameter \( \kappa \). Finding an explanation for this is difficult. We hypothesize that this is because SSAD overfits to the feedback for high \( \kappa \) values. For Sf, \( \kappa = 0.1 \) is the best choice.

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

We also have investigated the boxplot outliers for Sf and Si. Two data sets, Hepatitis and Parkinson, yield unusually high AEQ. Our explanation is that both data sets are so small that most of the data is labeled after 50 iterations. In this case, the active learning strategy is irrelevant to the classification quality at the end of the active learning cycle. Thus, we exclude Hepatitis and Parkinson from now on.

3) Initial Pool Strategies: The initial pool strategy specifies the number of labeled observations at \( t_{\text{init}} \). Intuitively, increasing this number should affect the start quality positively, as more information about 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 experimental results confirm this intuition. Figure 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 Annthyroid and HeartDisease, see Table IV. For HeartDisease, Pp and Pn result in the same number of initial labels. For Annthyroid, the number of labels with Pp is larger than with Pn by an order of magnitude. The consequences are that SQ on Annthyroid is high for Pp, and that AEQ is only slightly higher than SQ. This means that active learning only has 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 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. So one must carefully explain how the initial sample is obtained. Otherwise, it is unclear whether high quality according to AEQ is due to the active learning strategy or to the initial pool.

4) Query Strategy: We have arrived at a subset of the experimental space where comparing different query strategies is reasonable. To compare the strategies, we fix the initial pool strategy to Pn. In this way, we can include the data-based QS which all require initial labels. We obtain the initial pool by uniform stratified sampling. We repeat each setting 5 times to reduce the bias of the initial sample.

Table V 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 three data sets, Annthyroid, Cardio and SpamBase, data-based query strategies fail because the initial pool is too small for KDE. For the remaining data sets, we make two observations. First, the QR achieved differs between data sets. Some of the data sets, like Annthyroid and Parkinson, seem to be more difficult and only result in a small QR. Second, the quality of a specific QS differs significantly between data sets. For instance, \( \tau_{DB} \) is the best strategy on Stamps, but does not improve the classification quality on PageBlocks. In four cases, \( \tau_{\text{rand-out}} \)
clearly outperforms the remaining strategies. There neither is a QS category nor a single QS that is superior on all data sets. This also holds for other metrics like RU and ROQ.

Query strategies differ more clearly with respect to data characteristics. First, data-based QS are only applicable when the pool of labeled data is sufficiently large to perform KDE. Next, runtimes for $\tau_{ML}$ are an order of magnitude larger than for all other strategies. For PageBlocks, the average runtime per query selection for $\tau_{ML}$ is 1.31 s, compared to 6 s for $\tau_{NB}$.

To summarize, there is no one-fits-all query strategy for one-class AL. However, the requirements for data-based query strategies may be difficult to meet in practice. If this is the case, all model-based and hybrid strategies we have evaluated except for $\tau_{BNC}$ may be a good choice. They result in significant increases over 50 iterations for most data sets and scale well with the number of observations. In any case, one must compare results to random baselines, as they are easy to implement and outperform the more complex strategies in many cases.

5) Case Study: A conclusion from the previous sections is that there is no universal set of rules to select an active-learning method for one-class classification. However, one can think of several extensions to our benchmark to facilitate the extraction of decision rules. We now present possible extensions and elaborate on them.

(i) One can complement the benchmark with additional real-world data sets. However, such data sets are only useful to validate whether rules that have already been identified are applicable to other data as well. So, in our case, i.e., without existing decision rules, we instead expect additional real-world data sets to just scatter the results.

(ii) One may strive towards a narrow set of 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. However, this is difficult for two reasons. First, it is unclear what interesting data characteristics would be in this case. Even if one can come up with some characteristics, it still is difficult to generate synthetic data with arbitrary characteristics. Second, reliable statements on selection rules would require a benchmark on a large number of data sets. This entails a huge number of experimental combinations and experiment runtimes that are prohibitive, not only for us.

(iii) One could strive for theoretical guarantees on query strategies. The strategies discussed in Section IV-B4 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 current article.

To conclude, deriving a set of rules based on our results is not within reach, and possible ways to do so go way beyond the scope of this current article. So one must select active learning methods for a use case individually. Our systematic approach presented in 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 comparison. We now illustrate how to do this for a specific application.

The scenario is as follows. The data set is Stamps, and we assume that there are no initial labels available, i.e., the initial pool strategy is $Pu$, and data-based QS are not applicable. Based on our earlier results, we have selected SVDDneg as the classifier and use $SF$ to evaluate the classification quality. We choose $\tau_{DB}$ and $\tau_{NB}$ as potential query strategies and terminate the active-learning cycle after 100 iterations.

Figure 10 graphs the progress curves for both query strategies. A first observation is that it depends on the progress which one is better. For example, $\tau_{DB}$ results in a better MCC after 20 iterations, while $\tau_{NB}$ is superior after 75 iterations. After 30 iterations, both $\tau_{DB}$ and $\tau_{NB}$ perform equally well. At this point, the learning stability (LS) is 0, which speaks for stopping. Indeed, although there is some increase after 50 iterations, it is small compared to the overall improvement.

For a more detailed comparison, we look at the summary statistics. If only the final classification quality is relevant,
i.e., the budget is fixed to 100 observations, $\tau_{NB}$ is preferred because of higher EQ and AEQ values. For a fast adaption, one should prefer $\tau_{DB}$ with RU(5) = 0.31, compared to a RU(5) = 0.10 for $\tau_{NB}$. Regarding the outlier ratio, both query strategies query similarly with 13% and 14%. Users can now weigh these criteria based on their preferences to decide on the most appropriate query strategy.

V. CONCLUSIONS

Active learning for one-class classifiers requires methods that focus on imbalanced class distributions. While there are several approaches available in literature, finding a suitable method for a particular use case is challenging.

In this article, we have approached this challenge, in two steps. First, we have provided a categorization that can serve as a reference point to specify assumptions and learning objectives. Second, we have proposed several ways to summarize active learning results with distinct interpretation, and 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. Moreover, existing active learning methods only outperform random baselines in some regions of the experiment space. Next, we have shown that our categorization and summary statistics facilitate the query-strategy selection for a specific use case. We also conclude that the learning scenario has a significant impact on the quality and the interpretation of the results. We strongly recommend an explicit specification of the learning scenario in any case.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation.

REFERENCES

[1] V. Malbasa, C. Zheng, P. C. Chen, T. Popovic, and M. Kezunovic, “Voltage stability prediction using active machine learning,” IEEE Trans. Smart Grid, pp. 3117–3124, Nov. 2017.
[2] F. Olsson, “A literature survey of active learning in the context of natural language processing,” Swedish Institute of Computer Science, Tech. Rep., 2009.
[3] N. Gornitz, M. Kloft, K. Rieck, and U. Brefeld, “Active learning for network intrusion detection,” in AI Sec Workshop. ACM, 2009.
[4] J. W. Stokes, J. C. Platt, J. Kravis, and M. Shilman, “Aladin: Active learning of anomalies to detect intrusions;” Microsoft Research, Tech. Rep., 2008.
[5] L. Yin, H. Wang, and W. Fan, “Active learning based support vector data description method for robust novelty detection;” Knowl-Based Syst., pp. 40–52, 2018.
[6] V. Barnabé-Lortie, C. Bellinger, and N. Jakpawicz, “Active learning for one-class classification,” in ICMLA. IEEE, 2015, pp. 390–395.
[7] A. Ghasemi, H. R. Rabiee, M. Fadaee, M. T. Manzuri, and M. H. Rohban, “Active learning from positive and unlabeled data;” in ICDM Workshop, 2011, pp. 244–250.
[8] A. Ghasemi, M. T. Manzuri, H. R. Rabiee, M. H. Rohban, and S. Haghiri, “Active one-class learning by kernel density estimation,” in MLSP Workshop, 2011, pp. 1–6.
[9] P. Juszczak and R. P. Duin, “Selective sampling methods in one-class classification problems,” in ICANN/ICONIP. Springer, 2003, pp. 140–148.
[10] P. Juszczak and R. P. W. Duin, “Uncertainty sampling methods for one-class classifiers,” in ICML Workshop, 2003, pp. 81–88.
[11] N. Gornitz, M. Kloft, K. Rieck, and U. Brefeld, “Toward supervised anomaly detection,” JAIR, pp. 235–262, 2013.
[12] D. Kortke, A. Calma, D. Husejic, G. Krenpl, and B. Sick, “Challenges of reliable, realistic and comparable active learning evaluation,” in ECML PKDD Workshop, 2017, pp. 2–14.
[13] S. Das, W. Wong, T. Dietterich, A. Fern, and A. Emmott, “Incorporating expert feedback into active anomaly discovery,” in ICDM, 2016, pp. 853–858.
[14] B. Settles, “Active learning,” Synthesis Lectures on Artificial Intelligence and Machine Learning, pp. 1–114, 2012.
[15] C. Beyer, G. Krenpl, and V. 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.
[16] J. Bernard, M. Zeppelzauer, M. Lehmann, M. Müller, and M. Sedlmair, “Towards user-centered active learning algorithms,” in Comput Graph Forum, 2018, pp. 121–132.
[17] H. Raghavan, O. Madani, and R. Jones, “Active learning with feedback on features and instances,” JMLR, pp. 1655–1686, 2006.
[18] G. Druck, B. Settles, and A. McCallum, “Active learning by labeling features,” in EMNLP, 2009, pp. 81–90.
[19] P. Juszczak, “Learning to recognize; A study on one-class classification and active learning,” Ph.D. dissertation, TU Delft, Delft University of Technology, 2006.
[20] Z. C. Lipton, “The mythos of model interpretability,” in ICML Workshop, 2016.
[21] R. Phillips, K. H. Chang, and S. A. Friedler, “Interpretable active learning,” in Conference on Fairness, Accountability and Transparency. PMLR, 2018.
[22] S. Teso and K. Kersting, “Why should I trust interactive learners? Explaining interactive queries of classifiers to users,” arXiv preprint arXiv:1805.08578, 2018.
[23] B. Micenková, R. T. Ng, X. Dang, and I. Assent, “Explaining outliers by subpace separability,” in ICDM, 2013, pp. 518–527.
[24] J. Kauffmann, K.-R. Müller, and G. Montavon, “Towards explaining anomalies: A deep taylor decomposition of one-class models,” arXiv preprint arXiv:1805.06230, 2018.
[25] S. S. Khan and M. G. Madden, “One-class classification: taxonomy of study and review of techniques,” The Knowledge Engineering Review, p. 345–374, 2014.
[26] D. M. Tax and R. P. Duin, “Support vector data description,” Machine Learning, pp. 45–66, 2004.
[27] B. W. Silverman, Density estimation for statistics and data analysis. Routledge, 2018.
[28] D. W. Scott, Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons, 2015.
[29] Y. Xiao, H. Wang, L. Zhang, and W. Xu, “Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection,” Knowl-Based Syst., pp. 75–84, 2014.
[30] D. M. J. Tax and R. W. P. Duin, “Uniform object generation for optimizing one-class classifiers,” JMLR, pp. 155–173, 2002.
[31] A. Bánhalmi, A. Koosor, and R. Buxa-Fekete, “Counter-example generation-based one-class classification,” in ECML-Springer, 2007, pp. 543–550.
[32] S. Wang, Q. Liu, E. Zhu, F. Porikli, and J. Yin, “Hyperparameter selection of one-class support vector machine by self-adaptive data shifting,” Pattern Recognition, pp. 198–211, 2018.
[33] S. Wang, J. Yu, E. Lapira, and J. Lee, “A modified support vector data description based novelty detection approach for machinery components,” Appl Soft Comput, pp. 1193–1205, 2013.
[34] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Mincenková, E. Schubert, I. Assent, and M. E. Houle, “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study,” Data Min Knowl Disc, pp. 891–927, 2016.
[35] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” SIAM Review, pp. 65–98, 2017.