University of Potsdam
Institute of Mathematics

Sequential Learning For Optimal Detection of Rare Events

Master's Thesis

Pouya Abbasi

Supervisors:
Prof. Dr. Alexandra Carpentier
Dr. Tomáš Kocák

Potsdam, November 2024

Statement of Originality

I hereby declare that I wrote this work independently and did not use any tools or sources other than those I specified. Parts of the sources used that are taken literally or in terms of content are marked as such.

Furthermore, I certify that neither I nor anyone else has published this work or submitted it in a modified form as proof of performance in another event.

Name: Pouya Abbasi
Matriculation number: 806307
E-Mail: abbasi1@uni-potsdam.de
Date of submission: November 2024

Abstract

Rare event detection presents a significant challenge across various domains, including finance, healthcare, and autonomous systems. Despite their low probability of occurrence, such events often carry substantial consequences, necessitating the development of effective and efficient detection mechanisms. This thesis investigates sequential decision-making frameworks designed to address the challenges of data imbalance and the high costs of obtaining labeled data inherent in rare event detection. By examining both realizable and noisy scenarios, the study evaluates the performance of active learning algorithms in optimizing label complexity and improving detection accuracy.

The proposed methodologies employ disagreement-based active learning strategies and dyadic grid representations to facilitate robust detection under diverse conditions. Experimental results exhibit variability depending on dataset configurations. While the frameworks validated theoretical expectations under certain conditions, particularly in realizable cases, their performance was inconsistent in noisy scenarios, especially with datasets characterized by severe class imbalance and highly intertwined classes. These findings underscore the necessity for further refinement of the noisy scenario framework to enhance its robustness and efficacy in practical applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The prompt and accurate detection of rare events is a critical challenge across various real-world domains due to the potentially severe consequences of such events. Examples include financial market crashes, power grid failures, cybersecurity breaches, early detection of rare diseases, and issues encountered by autonomous systems. Rare events are defined by their low probability of occurrence; however, when they do occur, their impacts can be substantial. In the financial sector, for instance, early detection of market anomalies can avert significant economic losses. Similarly, timely identification of early symptoms of rare diseases in healthcare can be life-saving. In autonomous systems, accurately differentiating between a real pedestrian and a reflection on a surface is paramount. The rarity and high-stakes nature of these events necessitate the development of detection methods that maximize detection probabilities while minimizing false positives and associated costs.

Detecting rare events poses significant challenges, primarily due to their infrequency, the inherent imbalance in the data, and the difficulty in distinguishing between normal and anomalous patterns. Traditional detection methods often rely on static models, which passively analyze data and utilize fixed thresholds to identify anomalies. These approaches are frequently inadequate for rare event detection, as they are unable to adapt to new information in real time and often require extensive labeled datasets—acquisition of which is both costly and impractical.

To address these limitations, this thesis proposes a framework that combines active learning with sequential decision-making, referred to as Sequential Learning for Optimal Detection of Rare Events. This approach integrates active learning strategies with dynamic decision-making to optimize resource allocation and improve detection performance. Sequential learning frameworks are inherently adaptive, enabling continuous model updates based on newly acquired data while allowing for the strategic selection of the most informative data points for labeling. Active learning functions as a feedback-driven system, wherein the algorithm iteratively requests and incorporates new information to refine its model. This adaptive approach is particularly well-suited for rare event detection, where labeled data is sparse and expensive to obtain.

The feedback loop intrinsic to active learning frameworks also facilitates the reconstruction of decision boundaries and dynamic adjustment of instance labels as new information becomes available. This dynamic adaptability is especially valuable in scenarios where patterns evolve or where unanticipated rare events emerge [Hin20].

Throughout this thesis, the terms active learning and sequential learning are used interchangeably, underscoring the iterative and adaptive nature of the methodology in addressing the complexities of rare event detection.

1.1 Motivation

The primary focus of this thesis is the optimal detection of rare events using active learning strategies. In this context, it is essential to precisely define the terms *Active Learning*, *Optimal Detection*, and *Rare Events*, both conceptually and mathematically, to establish a rigorous foundation for the methodologies employed.

Active Learning: This research emphasizes active learning, where the algorithm selectively labels the most informative data points rather than passively processing large volumes of potentially irrelevant data. This selective approach is particularly advantageous for rare event detection, as it reduces the cost and effort associated with labeling while prioritizing data points that contribute most to model improvement. Unlike passive learning, which assumes all data points are equally informative and typically selects them randomly, active learning strategically identifies instances with the potential to enhance model accuracy, especially for rare events that might otherwise be overlooked.

Optimal Detection: The objective of optimal detection in this context is to devise a strategy that maximizes detection accuracy while minimizing detection delays and false positives. This requires balancing exploration (acquiring new information) and exploitation (leveraging existing knowledge to detect rare events) to efficiently navigate the data space. Given the infrequency of these events, the learning algorithm must explore the data space effectively to identify rare occurrences with minimal resource expenditure, including computational resources and labeling costs.

Rare Events: Rare events are instances that, due to their low probability and infrequent occurrence, constitute a minority within the dataset. Despite their rarity, their potential severity necessitates careful monitoring and accurate detection to prevent significant consequences. Examples from domains such as finance, healthcare, and autonomous systems underscore the critical nature of these events. The definition of a rare event is inherently context-dependent, determined by the distribution characteristics and frequency within the dataset. Furthermore, the nature of rare events may evolve; an event initially classified as rare could become more common, necessitating the model's adaptability in detection strategies.

In many machine learning tasks, obtaining large quantities of unlabeled data is relatively straightforward and cost-effective. However, acquiring labeled data remains a significant challenge, often requiring substantial investments of time, effort, or specialized expertise. This is largely due to the need for manual annotation or expert input to produce meaningful labels, making the process labor-intensive and resource-demanding. Consequently, a pronounced disparity exists between the abundance of unlabeled data and the difficulty of acquiring labeled data, which is a common bottleneck in the development and training of machine learning models [H$^+$14].

For example, training models for self-driving cars necessitates millions of hours of street traffic videos. A mere ten-second video can comprise approximately 300 frames, each requiring the labeling of multiple objects, which is an extremely time-consuming process. Specifically, one second of video consists of around 30 frames, and if each frame contains 20 cars that need to be boxed (labeled), annotating just

10 seconds of video could take nearly 100 minutes, as illustrated in Figure 1.1. This exemplifies a critical bottleneck in machine learning: while the acquisition of data is often straightforward, the labeling process is both arduous and resource-intensive.

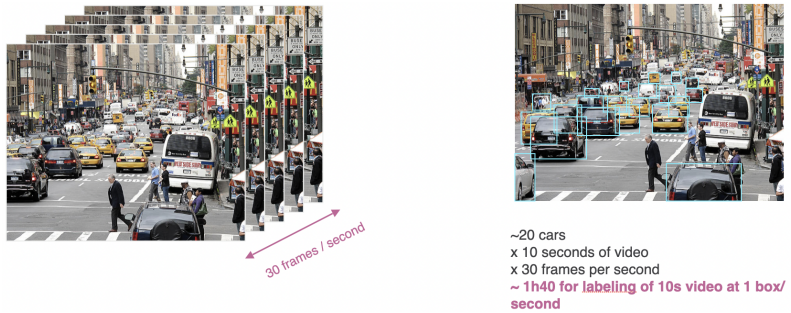


Figure 1.1: Labeling objects. in a ten-second video (Images from [Pre])

To address this challenge, Active/Sequential Learning has emerged as a highly effective paradigm. This approach enables models to focus on the most informative instances, thereby reducing the volume of labeled data required and significantly enhancing efficiency. Unlike passive learning, which relies on randomly selected labeled datasets and lacks feedback between the model and the data selection process, active learning strategically prioritizes data points near decision boundaries. These points are the most likely to improve the model's performance, particularly in complex scenarios.

This thesis aims to develop an active learning framework capable of identifying the most informative data points for labeling, particularly in the context of rare and high-impact events. By addressing this challenge, the thesis contributes to the development of cost-effective and accurate detection strategies that can adapt to real-world data. Furthermore, it emphasizes the comparative analysis of realizable and noisy cases to enhance robustness in practical applications.

The realizable and noisy cases will be discussed in greater technical detail in Chapters 2 and 3, respectively. As a precursor to this discussion, it is valuable to highlight their general properties, as well as the key similarities and differences between these cases.

1.2 Realizable and Noisy Cases

The purpose of this work is to compare the performance of realizable and noisy cases in the detection of rare events. The realizable case is employed as a benchmark, as it assumes ideal conditions that shield the model from noise and distractions. While such assumptions rarely align with real-world scenarios, they provide a controlled setting to evaluate and measure the efficiency and behavior of alternative methods, such as those designed for noisy cases.

The realizable case offers a foundational framework, enabling the exploration of noisy cases, which are more representative of real-world conditions. In noisy cases, data collection, labeling, and model fitting are often influenced by numerous unpredictable factors, creating a more challenging and realistic environment. The following sections provide a non-technical overview of these two scenarios.

1.2.1 Realizable Case

Most active learning approaches are based on the assumption that data are perfectly separable, the hypothesis space contains a function capable of perfectly classifying both training and testing data, and there is no error or noise in the labeling process [Kää06]. However, these assumptions are often unrealistic in real-world scenarios.

The first widely recognized general-purpose approach for realizable cases was introduced by Cohn, Atlas, and Ladner [CAL94], commonly referred to as the CAL algorithm, named after its creators. The CAL strategy operates as follows: the algorithm sequentially evaluates each instance within the unlabeled data pool. If two classifiers exist in the version space (a subset of classifiers/hypothesis class consistent with all previously observed labels) that disagree on the label of the current instance, the algorithm requests the oracle to provide the label. Otherwise, no label is requested. Meanwhile, inconsistent classifiers are removed from the version space.

Algorithms inspired by CAL are categorized as disagreement-based methods. These methods are sometimes referred to as "mellow" active learning because they adopt a minimalistic approach: the algorithm only queries labels when it cannot infer the label from the available information, without actively seeking out the most informative examples. In this framework, informativeness is treated as a binary characteristic—an instance is either deemed informative or not, with no further differentiation based on degrees of informativeness [Han12]. Notably, the algorithm does not account for the extent of disagreement among classifiers. For example, it does not differentiate between an instance that is inconsistent with a single classifier and one for which the majority of classifiers disagree. Additionally, the sequential querying process and the removal of incompatible classifiers significantly influence the data selection for labeling.

1.2.2 Noisy Case

A realistic scenario arises when the assumptions of the realizable case are relaxed. Significant progress has been made in addressing the challenges posed by imperfect annotators, underspecified feature spaces, and model specification errors [Han12]. In general, noisy cases—whether in passive or active learning—are considerably more challenging than realizable cases.

This domain of active learning is commonly referred to as *agnostic active learning*, which has its roots in the agnostic PAC (Probably Approximately Correct) model [KSS92]. In this setting, a wide variety of classifiers, such as linear models, neural networks, and decision trees, can be employed. However, no flawless classifier exists, and in some instances, labels may also be noisy. The primary objective in such cases is to identify a classifier that minimizes the discrepancy with the best possible hypothesis.

A notable advancement in agnostic active learning was achieved with the introduction of the A^2 algorithm by Balcan, Beygelzimer, and Langford [BBL06, BBL09]. Designed to be resilient to noise, this algorithm utilizes disagreement-based strategies and is applicable to various classifiers under diverse noise conditions [Han12]. The core assumption underpinning their work is that the A^2 algorithm operates with access to a continuous supply of unlabeled data points sampled independently and identically from a stationary distribution (i.i.d.). However, this assumption does not hold in many real-world scenarios. For instance, data drifting is a common occurrence,

particularly when sensors are repaired or replaced, leading to non-stationary data distributions.

Active learning inherently seeks to select samples based on their informativeness rather than randomly. Thus, while the overall unlabeled dataset may adhere to the i.i.d. assumption, the specific data points chosen for labeling—limited by a budget—often deviate from this assumption.

Furthermore, the output of their work is restricted to a narrow class of classifiers and is, in some sense, parametric. An alternative line of research seeks to alleviate this restriction by adopting a *nonparametric* setting, which aims to expand the relationship between parametric distributions and learning rates [LCK17]. Minsker [Min12], along with Locatelli, Carpentier, and Kpotufe [LCK17], have addressed this issue under the conventional nonparametric assumptions for regression functions. Specifically, the regression function is assumed to belong to a specified Hölder class $\Sigma(\beta, K, [0,1]^k)$ and to satisfy the low-noise condition.

This work introduces a novel approach based on *plug-in classifiers*. This method demonstrates superior performance compared to passive algorithms, particularly under the *Tsybakov low-noise* condition. This condition is characterized by the existence of positive constants B and $\lambda > 0$ such that:

$$\forall t > 0, \mathrm{P}_X(x : |\eta(x)| \leq t) \leq Bt^{\lambda}, \tag{1.1}$$

where $\mathcal{X} \in \mathbb{R}^k$ and $\mathcal{Y} \in \{0,1\}$ are couple from unknown $\mathrm{P}_{X,Y}$ distribution, with P_X as the marginal distribution of $\mathcal{X}$. And $\eta(x) = \mathbb{E}(Y|X = x)$, a regression function from Hölder class $\Sigma(\beta, K, [0,1]^k)$, where $(X, Y) \sim \mathrm{P}_{X,Y}$.

The related algorithm achieves exponential convergence in *label complexity*, a significant improvement over the typically polynomial rates of passive learning. The plug-in classifier discussed in this context refers to a classification method based on the nonparametric estimation of the regression function, $\eta(x)$, which belongs to the Hölder class $\Sigma(\beta, K, [0,1]^k)$. The central idea of a plug-in classifier is to estimate this regression function and utilize the sign of the estimate to make predictions. The plug-in classifier introduced in [Min12] operates as follows:

In a binary classification problem where the labels Y are either 0 or 1, the regression function $\eta(x)$ provides the expected value of Y given a feature vector $X = x$. Specifically, $\eta(x)$ represents the probability that a label is 1, with higher values indicating that Y is more likely to be 1. The classifier's objective is to predict Y using X, which is achieved by determining whether $\eta(x)$ is greater than or less than zero.

Since the true regression function $\eta(x)$ is unknown, it is estimated from the data using a nonparametric method, such as local polynomial regression. Once $\eta(x)$ is estimated (denoted as $\hat{\eta}(x)$), the plug-in classifier is defined as $f(x) = \mathrm{sign}(\hat{\eta}(x))$. This function $f(x)$ classifies a data point x based on the sign of the estimated regression function. The term "plug-in" classifier arises from the fact that the decision boundary is determined by substituting the estimate $\hat{\eta}(x)$ into the decision rule $\mathrm{sign}(\hat{\eta}(x))$.

The classifier employs a piecewise-constant estimator for $\mathrm{sign}(\hat{\eta}(x))$. The feature space is partitioned into small regions (dyadic cubes), within which the regression function is assumed to be constant. For each region R_i in the feature space, the estimator $\mathrm{sign}(\hat{\eta}(x))$ assigns the average of the observed labels for all data points within that region.

The algorithm presented in [LCK17], which will be explored in this work, builds upon the approach in [Min12] with some notable differences. It operates as follows:

The algorithm begins with a hierarchical dyadic partitioning of the feature space $[0,1]^k$. The space is divided into dyadic cells, which are cubes of progressively smaller sizes as the algorithm refines the grid. These cells collectively cover the entire domain of interest. The purpose of this partitioning is to focus the classification task on regions of interest, particularly near the decision boundary where classification uncertainty is highest. As the algorithm progresses, the diameter of the cells decreases, enabling increasingly fine local estimates of $\eta(x)$.

For each cell in the partition, the algorithm computes a local estimate of $\eta(x)$, which represents the probability that the label is 1. This estimate is typically based on the labels of points within the cell or in neighboring cells. The key idea is to prioritize regions where $\eta(x)$ is close to 0.5, as these regions indicate uncertainty in classification—whether to label x as 0 or 1. These areas are critical for labeling since small changes in $\eta(x)$ can significantly impact classification decisions.

This step is central to active learning as it enables the algorithm to focus on the most informative points, thereby reducing label complexity. Once the regions of uncertainty are identified, the algorithm requests labels for points within these regions. The strategy is to concentrate sampling efforts in areas where classification is most challenging, allowing the algorithm to reduce uncertainty and refine the classifier effectively.

The algorithm iteratively refines the partition and requests labels until a predefined stopping condition is met. This stopping condition may depend on the labeling budget, the degree of uncertainty reduction achieved, or a specified confidence level in the classifier's performance.

1.2.3 Version Space

In Active Learning, a key concept is the *version space*, which plays a critical role in guiding the selection of informative samples. The version space is defined as the subset of the hypothesis space that contains all classifiers consistent with the labeled data observed so far. This concept is particularly prominent in realizable active learning, where the hypothesis class is assumed to be fixed and finite. In this context, the version space progressively narrows as the algorithm receives more labeled instances, enabling the learner to strategically query data points that are most likely to reduce the version space. The objective is to eliminate as many inconsistent classifiers as possible, thereby converging to the optimal classifier with minimal labeling effort.

In contrast, noisy active learning does not assume a predefined hypothesis class. Instead, the version space starts undefined, and the algorithm incrementally learns a model from the data without assuming a fixed parameterization. Rather than explicitly managing a version space, noisy active learning represents classifiers through data-driven approaches such as sampling-based methods or kernel-based estimators. Techniques like nearest neighbors, decision trees, or Gaussian processes are commonly employed. As labeled data accumulates, the algorithm implicitly updates its classifier representation, dynamically adjusting the decision boundary without explicitly maintaining a well-defined version space.

The distinction between realizable and noisy cases also influences how uncertainty is measured and how the algorithm selects subsequent queries. In realizable settings, uncertainty is often derived from disagreements among the classifiers within the ver-

sion space. Conversely, in noisy settings, uncertainty typically stems from data density or model variance and is addressed using methods like uncertainty sampling or expected model change strategies. While both frameworks aim to minimize labeling effort while ensuring high accuracy, the underlying mechanisms for representing and updating classifiers differ fundamentally between realizable and noisy cases.

1.2.4 Label Complexity

Label complexity reflects the number of labeled data points required for a learning algorithm to achieve a specified level of accuracy. In both realizable and noisy cases, label complexity is influenced by various factors, though the key parameters differ between the two.

According to [H+14], in the *realizable case*, label complexity is determined by several critical factors. First, it depends on the *VC dimension* (d) of the hypothesis class, which quantifies its complexity. A higher VC dimension typically increases label complexity, as more labeled data is needed to narrow down the set of consistent hypotheses. Another essential factor is the *target error* (ε); achieving higher accuracy (lower ε) necessitates more labels, with label complexity growing logarithmically as $1/\varepsilon$. Additionally, the *disagreement coefficient* (θ) plays a pivotal role by quantifying the likelihood that a new data point lies in the region of disagreement among consistent hypotheses. A smaller disagreement coefficient reduces label complexity, as it indicates faster shrinkage of the version space with each query.

In contrast, as described in [LCK17], the *noisy case* introduces additional complexities. Label complexity in this setting is influenced by the *noise margin* (β), which measures the level of uncertainty near the decision boundary. A larger β corresponds to fewer ambiguous points, thereby lowering the label complexity. Another critical parameter is the *smoothness* of the decision boundary, represented by α. A smoother decision boundary (larger α) reduces label complexity, while a rougher boundary (smaller α) increases it. The *dimensionality* (k) of the feature space also impacts label complexity, as higher dimensions generally require more labeled data to adequately explore the data space. Similar to the realizable case, the target error ε remains a factor; however, its relationship with label complexity is more intricate under noisy conditions.

This comparison highlights the distinct ways in which label complexity is influenced by the structure of the hypothesis space, noise, and data characteristics in realizable and noisy cases. While the realizable case benefits from the simplicity of assuming a perfect classifier, the noisy case introduces additional challenges due to the necessity of managing uncertainty and adapting to noise.

Chapter 2

Realizable Cases

This section will investigate the active learning framework under the *realizable* case. As mentioned before the realizable case contains assumptions that do not align with real-world scenarios. Due to numerous uncontrollable factors, it is almost impossible to get a clean data set, flawless oracle, or perfect model. However, to have a benchmark to evaluate other methods against the realizable states, studying and characterizing them is substantial.

In the following first the initial setting and fundamental definitions will be provided. Then the main algorithm and related supporting theorems will be explored.

2.1 Setting

In this thesis, the following formal framework which is according to [H+14] applies. Let $\mathcal{X} \in [0,1]^k$ be the instance space, equipped with a σ-algebra $\mathcal{B}_{\mathcal{X}}$. For simplicity, assume $(\mathcal{X}, \mathcal{B}_{\mathcal{X}})$ is a standard Borel space. The label space is $\mathcal{Y} = \{0,1\}$, and the product space $\mathcal{X} \times \mathcal{Y}$ is equipped with the product σ-algebra $\mathcal{B} = \mathcal{B}_{\mathcal{X}} \bigotimes 2^{\mathcal{Y}}$. Let $\mathrm{P}_{X,Y}$ denote a probability measure on $\mathcal{X} \times \mathcal{Y}$, referred to as the target distribution, and let P_X represent the marginal distribution of $\mathrm{P}_{X,Y}$ over $\mathcal{X}$. For each $x \in \mathcal{X}$, define

$$\eta(x) = \mathrm{P}_{X,Y}(Y = +1 | X = x),$$

where $(X,Y) \sim \mathrm{P}_{X,Y}$. A function $h : \mathcal{X} \to \mathcal{Y}$ is called a classifier, and the *error rate* of any classifier h is given by:

$$er(h) = \mathrm{P}_{X,Y}((x,y) : h(x) \neq y),$$

which represents the probability that h makes a misclassification for a random point $(X,Y) \sim \mathrm{P}_{X,Y}$.

In this context, our focus is on learning from data, to find a classifier h that minimizes $er(h)$ using samples drawn from $\mathrm{P}_{X,Y}$. Let $\mathcal{Z} = \{(X_i, Y_i)\}_{i=1}^{\infty}$ be a sequence of independent random variables distributed according to $\mathrm{P}_{X,Y}$, referred to as the labeled data sequence. For $m \in \mathbb{N}$, let $\mathcal{Z}_m = \{(X_i, Y_i)\}_{i=1}^{m}$ denote the first m labeled data points, and define $\mathcal{Z}_{\mathbf{X}} = \{X_i\}_{i=1}^{\infty}$ as the unlabeled data sequence. In practice, the number of unlabeled data points is large but finite; however, for analysis purposes, we assume access to the entire sequence $\mathcal{Z}_{\mathbf{X}}$. The number of unlabeled samples used by the algorithm will be addressed in the respective analyses.

In the active learning protocol, the algorithm is provided with a budget n and access to the unlabeled sequence $\mathcal{Z}_{\mathbf{X}}$. For $t = \{0, 1, ..., n\}$, the index $i_t \in \mathbb{N}$ is related to the t-th requested label. The algorithm can select an index i_1 and request the label Y_{i_1}. After receiving Y_{i_1}, it can request another label Y_{i_2}, and so on. This continues until the budget of n labels is exhausted, at which point the algorithm outputs a classifier $\hat{h}$. Formally, this protocol involves a family of estimators that map the labeled data to a classifier $\hat{h}$, with the condition that $\hat{h}$ is conditionally independent of the labeled sequence, given the unlabeled data and the sequence $(i_1, Y_{i_1}), \ldots, (i_n, Y_{i_n})$.

2.2 Definitions

In the core of active learning, there is a concept called *label complexity*. Following is its formal definition:

Definition 2.1 ([H$^+$14])**.** *For any active learning algorithm $\mathcal{A}$, we say $\mathcal{A}$ attains a label complexity of Λ if, for any $\varepsilon \geq 0$ and $\delta \in [0, 1]$, any distribution $\mathrm{P}_{X,Y}$ over $\mathcal{X} \times \mathcal{Y}$, and for all integer $n \geq \Lambda(\varepsilon, \delta, \mathrm{P}_{X,Y})$, the classifier h returned by $\mathcal{A}$ after using a budget of n will, with probability at least $1 - \delta$, satisfy the condition that the error $er(h)$ is at most ε.*

In active learning, the *label complexity* is a crucial metric. This is a concept based on that, it is possible to monitor the performance of the algorithm and classifiers. It represents the number of labeled examples that an active learning algorithm needs to achieve a certain level of accuracy or performance. This metric is crucial because active learning aims to minimize the number of labeled samples required, thereby reducing labeling costs.

The label complexity of a machine learning task is influenced by several factors, including the underlying data distribution and the complexity of the hypothesis space. Highly imbalanced datasets or those with clusters of similar instances may require fewer labels to accurately model the target function. Conversely, a complex hypothesis space, characterized by a large number of potential hypotheses, may necessitate more labels to effectively generalize from the training data. Additionally, noise in the data can increase label complexity, as the learner may need to acquire additional information to distinguish true patterns from random fluctuations.

Generally speaking, for a given $\mathrm{P}_{X,Y}$ distribution and maximum labeling budget n, we want a classier $\hat{h}$ that with the probability of $1-\delta$, is not more than ε far from the perfect classifier. Simply, the minimum number of labels required to achieve a desired level of accuracy. As [H$^+$14] elucidates, we will focus on how many labels an active learning algorithm needs to achieve a low error rate compared to the best possible error rate within a specific hypothesis class $\mathcal{C}$. Specifically, having $\nu = inf_{h \in \mathcal{C}} er(h)$, known as the *noise rate*, we are concerned about the value of $\Lambda(\nu + \varepsilon, \delta, \mathrm{P}_{X,Y})$, which is a function of ν, ε, δ, and $\mathrm{P}_{X,Y}$.

In this work, in Chapter 2 we will investigate cases in which the value of ν will be equal to zero. Having $\nu = 0$ implies that the perfect classifier $f^* \in \mathcal{C}$ is not contaminated by noises, namely $er(f^*) = 0$.

Based on the seminal work of Vapnik and Chervonenkis [VC15], we define the notion of shattering. For a *set of classifiers* $\mathcal{H}$, or interchangeably *hypostasises class of* $\mathcal{H}$, a sequence of points $(x_1, ..., x_m) \in \mathcal{X}^m$ is said to be shattered by $\mathcal{H}$ if, for

any possible labeling $(y_1, ..., y_m) \in \mathcal{Y}^m$, there exists a classifier $h \in \mathcal{H}$ that correctly classifies all points in the sequence. In essence, shattering implies that $\mathcal{H}$ can realize all 2^m distinct labelings of $(x_1, ..., x_m)$. This means that the hypothesis class has enough flexibility to fit any combination of labels for that specific set of points.

The Vapnik-Chervonenkis (VC) dimension of a non-empty set $\mathcal{H}$, represented as $\text{vc}(\mathcal{H})$, is defined as the maximum integer m for which there exists a set $S \in \mathcal{X}^m$ that can be shattered by $\mathcal{H}$ into every possible labeling. If no such integer exists, the VC dimension is considered infinite. We denote $d = \text{vc}(\mathcal{C})$ and suppose d finite throughout the thesis. The VC dimension tells us the "shattering capacity" of $\mathcal{H}$— i.e., how many points it can consistently label in every possible way. The VC dimension helps estimate a model's generalization capacity. If a model can shatter a large number of points (high VC dimension), it may overfit, meaning it might fit the training data well but perform poorly on new data. Conversely, a model with a low VC dimension may underfit, meaning it might not have enough complexity to capture the patterns in the data.

We also have the following notations; For any set A, let $\mathbf{1}_A$ denote the indicator function for A. This function assigns a value of 1 if $x \in A$ and 0 otherwise, represented by $\mathbf{1}_A(x) = 1$ if $x \in A$, and $\mathbf{1}_A(x) = 0$ if $x \notin A$. Similarly, for any logical condition L, we may use the notation $\mathbf{1}[L]$, where $\mathbf{1}[L] = 1$ when L is true, and $\mathbf{1}[L] = 0$ when L is false.

For a classifier h and a set of labeled data points $\mathcal{L} \in \bigcup_{m\in\mathbb{N}}(\mathcal{X} \times \mathcal{Y})^m$, we define the *empirical error rate* of h on $\mathcal{L}$ as

$$\text{er}_{\mathcal{L}}(h) = \frac{1}{|\mathcal{L}|} \sum_{(x,y)\in\mathcal{L}} \mathbf{1}[h(x) \neq y].$$

This metric represents the proportion of misclassified points in $\mathcal{L}$. For completeness, we set $\text{er}_{\emptyset}(h) = 0$. When $\mathcal{L} = \mathcal{Z}_m$, where $\mathcal{Z}_m$ represents the first m labeled data points,we abbreviate:

$$\text{er}_m(h) = \text{er}_{\mathbb{Z}_m}(h). \tag{2.1}$$

We also define the *version space*,

$$V_m^* = \{h \in \mathcal{C} : \forall i \leq m, h(X_i) = f^*(X_i)\},$$

which includes all classifiers consistent with the first m data points $\{X_1, \ldots, X_m\}$.

For a given set of classifiers $\mathcal{H}$ and any $\varepsilon \in [0, 1]$, define the $\varepsilon-minimal$ set as:

$$\mathcal{H}(\varepsilon) = \{h \in \mathcal{H} : er(h) \leq \varepsilon\}.$$

Additionally, for any classifier h, the $\varepsilon-ball$ centered at h is defined as:

$$\mathcal{B}_{\mathcal{H},\text{P}_X}(h, \varepsilon) = \{g \in \mathcal{H} : \text{P}_X(x : g(x) \neq h(x)) \leq \varepsilon\}.$$

When $\mathcal{H}$ is the hypothesis class $\mathcal{C}$, we abbreviate $\mathcal{B}_{\text{P}_X}(h, \varepsilon)$ as $\mathcal{B}_{\mathcal{C},\text{P}_X}(h, \varepsilon)$, and if P_X is implicit, we write $\mathcal{B}_{\mathcal{H}}(h, \varepsilon)$ for $\mathcal{B}_{\mathcal{H},\text{P}_X}(h, \varepsilon)$. Moreover, the radius of the set $\mathcal{H}$, denoted radius($\mathcal{H}$), is the smallest ε such that $\mathcal{H} = \mathcal{B}_{\mathcal{H}}(f^*, \varepsilon)$, where f^* is the optimal classifier (i.e. the perfect classifier). Lastly, define the region of disagreement of $\mathcal{H}$ as:

$$\mathrm{DIS}(\mathcal{H}) = \{x \in \mathcal{X} : \exists h, g \in \mathcal{H} \text{ such that } h(x) \neq g(x)\},$$

the set of points where different classifiers within the set $\mathcal{H}$ disagree about the correct label, as in [H$^+$14].

The particular algorithm for the realizable cases that will be investigated in this thesis belongs to a specific group called "*disagreement-based*" active learning. As [H$^+$14] puts it, they have a mechanism that step by step update a set, V, which contains candidate classifiers—one of which will ultimately be chosen. Unlabeled samples are processed sequentially, and the algorithm requests labels for those samples, Y_i, only when there is a lack of consensus in V about the predicted label for sample X_i. The set V is updated periodically by eliminating classifiers that perform poorly based on the labels obtained [H$^+$14].

In other words, as the author has explained in [H$^+$14], the label complexity of this approach involves analyzing the characteristics of the disagreement regions, $\mathrm{DIS}(V)$, that are formed during execution. Specifically, since labels are only requested for samples in $\mathrm{DIS}(V)$, it is essential to assess the likelihood, $\mathrm{P}_X(\mathrm{DIS}(V))$, that a randomly chosen sample falls into this disagreement region. As we will demonstrate, it is often practical to set a simple bound on the radius of V for the sets encountered in these algorithms. Consequently, to derive a straightforward bound on label complexity, it can be useful to approximate $\mathrm{P}_X(\mathrm{DIS}(V))$ by a linear function of the radius of V. The disagreement coefficient is formally defined as follows:

Definition 2.2 ([H$^+$14])**.** *For any $r_0 \geq 0$ and a classifier h, the disagreement coefficient of h with respect to a hypothesis class $\mathcal{C}$ and distribution P_X is given by:*

$$\theta_h(r_0) = \sup_{r > r_0} \frac{\mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(h, r)))}{r} \vee 1,$$

where $\mathrm{DIS}(\mathrm{B}(h, r))$ represents the region where classifiers in $\mathcal{C}$ disagree on labels within radius r of h. When $h = f^*$, this is simplified as $\theta(r_0) = \theta_{f^*}(r_0)$, referred to as the disagreement coefficient for the class $\mathcal{C}$ under the distribution P_X.

The disagreement coefficient quantifies the rate at which the region of disagreement among classifiers diminishes as their uncertainty decreases. A low disagreement coefficient signifies that the classifiers' predictions converge rapidly with reduced error, suggesting a lower requirement for label requests. Conversely, a high disagreement coefficient indicates a slower convergence, necessitating more label queries.

The coefficient is equal to zero when there is no disagreement among classifiers, even with the largest possible radius of the $\varepsilon-$*ball*. This implies that the classifiers are perfectly consistent in their predictions. Conversely, it becomes one when there is always disagreement among classifiers, regardless of the radius of the $\varepsilon-$*ball*. This indicates that the classifiers are highly diverse and have fundamentally different ways of making predictions.

We also write small-o notation to express asymptotic dependences. So, the statements like:

$$\lim_{\varepsilon \to 0} \frac{u(\varepsilon)}{v(\varepsilon)} = 0,$$

could be written as:

$$u(\varepsilon) = o(v(\varepsilon)). \tag{2.2}$$

The equation 2.2 means that $u(\varepsilon)$ should be smaller than $v(\varepsilon)$. See Figure2.1 please.

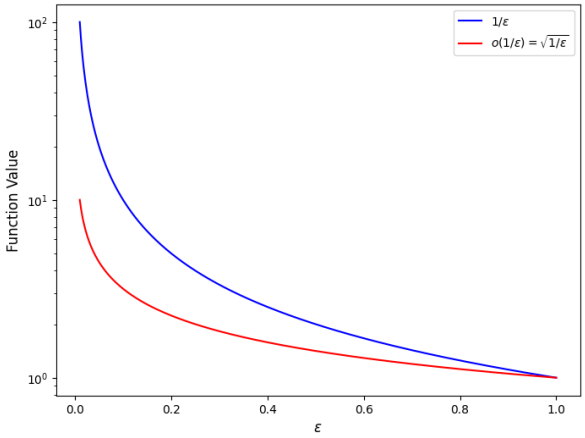


Figure 2.1: Function $v(\varepsilon) = \frac{1}{\varepsilon}$ is in blue, and function $u(\varepsilon) = \sqrt{\frac{1}{\varepsilon}}$ or equivalently $o(\frac{1}{\varepsilon}) = \sqrt{\frac{1}{\varepsilon}}$ is in red.

We will see the application of a function $o(v(\varepsilon))$ later during determining n, the labeling budget.

We use the notation $u(\varepsilon, \delta) \lesssim v(\varepsilon, \delta)$, means that there exists a universal constant $c \in (0, \infty)$ such that $u(\varepsilon, \delta) \leq cv(\varepsilon, \delta)$ for all $\varepsilon, \delta \in (0, 1)$. Similarly, $u(\varepsilon, \delta) \gtrsim v(\varepsilon, \delta)$ indicates that $u(\varepsilon, \delta) \geq cv(\varepsilon, \delta)$ for all $\varepsilon, \delta \in (0, 1)$, where $c \in (0, \infty)$ is an implicit universal constant. In this context (later we will state Theorem 2.4 regarding the properties that lead to choosing $o(v(\varepsilon))$), we define λ^k as the Lebesgue measure on $\mathbb{R}^k$. For a probability measure P_X on $\mathbb{R}^k$ to have a density $p : \mathbb{R}^k \to [0, \infty]$ (relative to λ^k), p must be a measurable function. For any measurable subset $\mathrm{A} \subseteq \mathbb{R}^k$, the probability $\mathrm{P}_X(\mathrm{A})$ is then given by

$$\mathrm{P}_X(\mathrm{A}) = \int \mathbf{1}_{\mathrm{A}}(x) p(x)\, \lambda^k(dx).$$

According to the Radon-Nikodym theorem, P possesses a density if and only if every measurable subset $\mathrm{A} \subseteq \mathbb{R}^k$ with $\lambda^k(A) = 0$ also satisfies $\mathrm{P}_X(\mathrm{A}) = 0$.

The following notations also apply. The x represents a specific data point or an individual instance from the instance space X. It is a generic variable used to denote any arbitrary data point. X_i, however, refers to a specific instance within a sequence of unlabeled data points. Each X_i is one element in a sequence $\{X_1, X_2, ...\}$, where i typically indexes individual samples in this sequence. Additionally, we use the notation $\mathrm{Log}(x) = \max\{\ln(x), 1\}$ for $x \geq 0$.

In this work for realizable cases, we will use linear classifiers, namely:

$$\mathcal{C} = \{h : \mathbf{w} \in [0, 1]^k, \mathbf{b} \in [0, 1]\} \tag{2.3}$$

where,

$$h(x) = \mathbf{w}^T \cdot x + \mathbf{b}. \tag{2.4}$$

Furthermore, by defining the sign function as follows:

$$\text{sign}(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & otherwise, \end{cases}$$

and plugging in 2.4 in sign function we get:

$$\text{sign}(h(x)) = \begin{cases} 1 & \text{if } \mathbf{w}^T \cdot x + \mathbf{b} > 0 \\ 0 & otherwise. \end{cases}$$

Now, we are ready to dig into the algorithm and supporting theorems in detail.

2.3 Algorithm and Theorems

One of the well-known works that has studied the realizable case is the paper written by [CAL94]. This work is famous for its algorithm so-called CAL named after the authors. There are several equivalent ways that this algorithm has been described. Here, as in [H$^+$14], to simplify and keep it more coherent the version of pseudo-code that is based on version space will be used. In the following, we will explain how each step of the procedure works in this algorithm. After that, a precise mathematical analysis for characterizing it will be provided.

Algorithm 1 CAL(n)

1: $m \leftarrow 0$, $t \leftarrow 0$, $V \leftarrow \mathcal{C}$
2: **while** $t < n$ and $m < 2^n$ **do**
3: $\quad m \leftarrow m + 1$
4: $\quad$ **if** $X_m \in \text{DIS}(V)$ **then**
5: $\quad\quad$ Request label Y_m; let $V \leftarrow \{h \in V : h(X_m) = Y_m\}$, $t \leftarrow t + 1$
6: $\quad$ **end if**
7: **end while**
8: **return** any $\hat{h} \in V$

In the first line three parameters $m, t, and\ V$, corresponding to respectively, the number of iterations, the number of queried labels, and the version space is initialized. Then it begins the loop and proceeds next steps as long as both of $t\ and\ m$ simultaneously meet the criteria that are being less than query budget, n, for t, and similarly being less than the iteration threshold, 2^n, for m. We can see that the iteration threshold is the function of the query budget. Afterward, inside the *while* loop, m will be incremented by one. Then in the *line 4*, the condition of the disagreement region is checked. That is, whether the sample X_m is a member of this set or not. If this constraint is not satisfied by X_m, it means that so far there were not any classifiers that label X_m differently. Namely,

$$\nexists\ h, h' \in H\ s.t.\ h(X_m) \neq h'(X_m).$$

On the other hand, if the sample X_m violates this condition, the cascade of the subsequent updates will occur:

- algorithm will request label Y_m from the oracle.
- V, the version space n q will be updated such that:

$$\forall\ h, h' \in H:\ h(X_m) = h'(X_m) = Y_m.$$

- t, the counter of query budget will be incremented to $t + 1$

The moment that the logical condition of *while* loop has been violated by either exceeding the query budget or iteration threshold, the algorithm will go out of the loop and implement the last step that is returning the $\hat{h} \in V$.

In the following the related theorems for characterizing the mechanism of algorithm 1 will be expressed.

Theorem 2.1 ([H$^+$14])**.** *CAL achieves a label complexity* Λ *such that, for* $\mathrm{P}_{X,Y}$ *in the realizable case,* $\forall \varepsilon, \delta \in (0,1)$*,*

$$\Lambda(\varepsilon, \delta, \mathrm{P}_{X,Y}) \lesssim \theta(\varepsilon)\left(d\mathrm{Log}(\theta(\varepsilon)) + \mathrm{Log}\left(\frac{\mathrm{Log}(1/\varepsilon)}{\delta}\right)\right)\mathrm{Log}(1/\varepsilon).$$

This theorem provides an upper bound on the label complexity of the CAL algorithm. Here, the label complexity Λ measures the number of labeled examples needed by the active learning algorithm to achieve a certain level of accuracy within ε with high probability (at least $1 - \delta$).

If we note to the Algorithm 1, we can see that to run CAL we need n, the labeling budget, and the initial hypotheses class $\mathcal{C}$. So before running CAL, utilizing Theorem 2.1 we can find the minimum labeling budget. The VC dimension d and the disagreement coefficient $\theta(\varepsilon)$ are also two key components to fulfill mapping ε, δ, and $\mathrm{P}_{X,Y}$ to the labeling budget n. As mentioned in the previous section, we are going to use a class of linear hypotheses. In the context of linear classifiers, the determination of the Vapnik-Chervonenkis (VC) dimension (Theorem 2.5) and the disagreement coefficient (Theorem 2.4) is grounded in established theoretical frameworks and we will state those without proof. Later during implementation, we will use these theorems in practice.

Proof of Theorem 2.1. [H$^+$14] The proof includes five steps as following:

Step 1: Setting Parameters
First we set parameters by selecting ε and δ such that both lay within the interval $(0,1)$, then execute the CAL algorithm with a label budget $n \in \mathbb{N}$, chosen to be sufficiently large:

$$n \geq \log_2(2/\delta) + 8ec'\theta(\varepsilon)\left(d\log(\theta(\varepsilon)) + 2\log\left(\frac{2\log_2(4/\varepsilon)}{\delta}\right)\right)\log_2(2/\varepsilon).$$

This ensures that the number of label requests is large enough to meet the desired error and confidence levels.

Step 2: Version Space Definition
Now we define key concepts. Set $M \subseteq \{0, \ldots, 2^n\}$ describe the range of values m can get while CAL runs. For each $m \in M$, let V_m as the version space at that point, containing all hypotheses consistent with the labels seen so far. Assume an even E, of probability 1, such that for every $m \in \mathbb{N}$ has $Y_m = f^*(X_m)$. By induction, the version space V_m at each step m contains all classifiers that agree with the observed labels, i.e., $V_m = V_m^*$. Formally:

$$\forall m \in M, f^* \in V_m = V_m^* = \{h \in \mathcal{C} : er_m(h) = 0\}.$$

Please note that to create the above V_m^*, which is equivalent to check the $er_m(h) = 0$ criteria, mean to check the $h(x) = y$. Hence, in reality, we are unaware of y. Thus, the practical alternative would be exploiting the fact that the optimal hypothesis, f^*, is always in version space and comparing the prediction of other hypotheses with it.

Step 3: Setting the Fundamental Error Constraints
To proceed, we incorporate Lemma 2.1 to establish the fundamental bounds necessary for our proof:

Lemma 2.1 ([H$^+$14])**.** *There exists a universal constant $c \in (1, \infty)$ such that, for any $\gamma \in (0, 1)$, $a \in [1, \infty)$, $\alpha \in [0, 1]$, and any $m \in \mathbb{N}$, with $\varepsilon_m = \left(\frac{ad}{m}\right)^{\frac{1}{2-\alpha}}$, and*

$$U(m, \gamma) = c \ \min \begin{cases} \left(\frac{a(d\mathrm{Log}(\theta(a\varepsilon_m^{\alpha})) + \mathrm{Log}(1/\gamma))}{m}\right)^{\frac{1}{2-\alpha}} \\ \frac{d\mathrm{Log}(\theta(d/m)) + \mathrm{Log}(1/\gamma)}{m} + \sqrt{\frac{\nu(d\mathrm{Log}(\theta(\nu)) + \mathrm{Log}(1/\gamma))}{m}} \end{cases}$$

with probability at least $1 - \gamma$, $\forall h \in \mathcal{C}$, the following inequalities hold:

$$er(h) - er(f^*) \leq \max\left\{2(er_m(h) - er_m(f^*)), U(m, \gamma)\right\},$$

$$er_m(h) - \min_{g \in \mathcal{C}} er_m(g) \leq \max\left\{2(er(h) - er(f^*)), U(m, \gamma)\right\},$$

where $er_m(h)$ is defined as in 2.1, and $d = \mathrm{vc}(\mathcal{C})$.

This lemma provides a bound, $U(m, \lambda)$ on the maximum discrepancy between empirical error $er_m(h)$ and true error $er(h)$, allowing us to understand how close our model is to the true error rate after observing a certain number of labeled samples. It ensures that any hypothesis selected from the hypothesis class maintains a small error compared to an optimal or true hypothesis with a high probability $1 - \lambda$. Given a desired error threshold ε, Lemma 2.1 helps us solve for the smallest m such that $U(m, \lambda) \leq \varepsilon$, ensuring the error bound is met with high probability. So, by Lemma 2.1 we can establish a mechanism between the minimum number of samples in the worst-case scenario to guarantee with a high probability, here $1 - \lambda$, that the error is under control during executing the Algorithm 1.

It is worth mentioning that the noise control parameters, namely, α, ν, and a should be chosen such that the setting meets the realizable case environment. That is to say, these parameters would be $\alpha = 1$, $a = 1$, and $\nu = 0$. By having some computation we can find the minimum m such that the $U(m, \lambda)$ is guaranteed to be

less than a specific error threshold. Namely, for some global fixed $c' \in [1, \infty)$, for any $m \in \mathbb{N}$, and $\varepsilon, \lambda \in (0, 1)$, we have:

$$m \geq c' \left(\frac{\nu + \varepsilon}{\varepsilon^2} \right) (d\text{Log}(\theta(\nu + \varepsilon) + \text{Log}(1/\lambda)) \Longrightarrow U(m, \lambda) \leq \varepsilon, \tag{2.5}$$

and by plugging in $\alpha = 1, a = 1$, and $\nu = 0$, 2.5 becomes:

$$m \geq c' \left(\frac{1}{\varepsilon} \right) (d\text{Log}(\theta(\varepsilon) + \text{Log}(1/\lambda)) \Longrightarrow U(m, \lambda) \leq \varepsilon. \tag{2.6}$$

Step 4: Error Bound Across Iterations
Now define $i_\varepsilon = \lceil \log_2(1/\varepsilon) \rceil$, as the number of iterations to achieve the error ε and set $I = \{0, \ldots, i_\varepsilon\}$. At the end of each iteration, the version space is updated. For $i \in I$, let $\varepsilon_i = 2^{-i}$; this represents the decreasing error thresholds for each step. furthermore, let $m_0 = 0$, and for c' as in 2.6, for each $i \in I \setminus \{0\}$, define

$$m_i = \left\lceil c' \left(\frac{1}{\varepsilon_i} \right) \left(d\text{Log}(\theta(\varepsilon_i)) + \text{Log} \left(\frac{2(2 + i_\varepsilon - i)^2}{\delta} \right) \right) \right\rceil.$$

Step 5: Bounding Error Probability
By Lemma 2.1, and 2.6, and involving a union bound we show that, with high probability, the maximum error of any classifier in the version space is below a certain threshold for all iterations. Let's break it down in detail. The objective is to show that, with high probability, for every iteration i in the set I, the supremum (maximum) error rate of hypotheses in the version space V_m^* is at most ε_i. In other words, regarding the error probability for individual iteration, namely, $i \in I$, we want to bound the probability that the error rate $\sup_{h \in V_{m_i}^*} er(h) \leq \varepsilon_i$ exceeds ε_i. From Lemma 2.1 we have:

$$\text{P}_{X,Y} \left(\sup_{h \in V_{m_i}^*} er(h) > \varepsilon_i \right) \leq \frac{\delta}{2(2 + i_\varepsilon - i)^2}.$$

This inequality states that the probability of having an error rate greater than ε_i for any hypothesis in $V_{m_i}^*$ is bounded by a decreasing function of i. Then by the union bound, we combine the probabilities of multiple events. It states that the probability of the union of several events is at most the sum of the probabilities of those events. Applying the union bound here allows us to combine the error probabilities for all iterations $i \in I/\{0\}$. We sum the probabilities from Lemma 2.1 across all iterations:

$$\text{P}_{X,Y} \left(\bigcup_{i=1}^{i_\varepsilon} \sup_{h \in V_{m_i}^*} er(h) \geq \varepsilon_i \right) \leq \sum_{i=1}^{i_\varepsilon} \frac{\delta}{2(2 + i_\varepsilon - i)^2}. \tag{2.7}$$

This sum captures the total probability that any of the iterations i will have an error rate exceeding ε_i. Based on the fact that the series $\sum_{i=1}^{i_\varepsilon} \frac{1}{(2+i_\varepsilon - i)^2}$ converges to a value less than 1, we can bound the left-hand side of the inequality 2.7:

$$\sum_{i=1}^{i_\varepsilon} \frac{\delta}{2(2 + i_\varepsilon - i)^2} \leq \frac{\delta}{2}. \tag{2.8}$$

This ensures that the combined probability across all iterations is still small and manageable. By combining (2.7) and (2.8) we can see:

$$\mathrm{P}_{X,Y}\left(\bigcup_{i=1}^{i_\varepsilon} \sup_{h\in V^*_{m_i}} er(h) \geq \varepsilon_i\right) \leq \frac{\delta}{2}. \tag{2.9}$$

Now we can define an event over the intersection of classifiers in $V^*_{m_i}$. Let's call this even E_δ. Please note that E_δ is the completeness of the inequality in (2.7). Namely, it can be written as follows:

$$\begin{aligned}
\mathrm{P}_{X,Y}\left(\bigcap_{i=1}^{i_\varepsilon} \sup_{h\in V^*_{m_i}} er(h) \leq \varepsilon_i\right) &= 1 - \mathrm{P}_{X,Y}\left(\bigcup_{i=1}^{i_\varepsilon} \sup_{h\in V^*_{m_i}} er(h) \geq \varepsilon_i\right) \\
&= 1 - \sum_{i=1}^{i_\varepsilon} \mathrm{P}_{X,Y}\left(\sup_{h\in V^*_{m_i}} er(h) \geq \varepsilon_i\right) \\
&\geq 1 - \sum_{i=1}^{i_\varepsilon} \frac{\delta}{2(2+i_\varepsilon - i)^2} \\
&\geq 1 - \frac{\delta}{2},
\end{aligned} \tag{2.10}$$

this guarantees that with probability at least $1 - \frac{\delta}{2}$ the error rates stay within the desired bounds for each step i.

It is worth mentioning that the CAL will process every new point to check whether it falls into the most recent corresponding version space's disagreement area. For event E, the total number of these requested labels up to the m_{i_ε} visited point is:

$$\sum_{m=1}^{\min\{m_{i_\varepsilon}, \max M\}} \mathbf{1}_{\mathrm{DIS}(V_{m-1})}(X_m) = \sum_{m=1}^{\min\{m_{i_\varepsilon}, \max M\}} \mathbf{1}_{\mathrm{DIS}(V^*_{m-1})}(X_m), \tag{2.11}$$

The summation on the left-hand side of the (2.11) also can be brake down to iteration, namely:

$$\sum_{m=1}^{m_{i_\varepsilon}} \mathbf{1}_{\mathrm{DIS}(V^*_{m-1})}(X_m) = \sum_{i\in I/\{0\}} \sum_{m=m_{i-1}+1}^{m_i} \mathbf{1}_{\mathrm{DIS}(V^*_{m-1})}(X_m), \tag{2.12}$$

where, $\mathbf{1}_{\mathrm{DIS}(V_{m-1})}(X_m)$, means if X_m is in the disagreement set.

The iteration i, corresponds to a particular threshold ε_i, that decreases exponentially. During each iteration i, the CAL algorithm requests labels for instances within a specific segment of indices, from $m_{i-1}+1$ to m_i. This range can be express like $m \in \{m_{i-1}+1, ..., m_i\}$. These are actually processed points between two consecutive labels requested by CAL. The endpoints m_{i-1} and m_i are chosen such that the number of label requests in each iteration is sufficient to meet the error threshold ε_i. In addition, the version space V^*_m is monotonically decreasing, meaning it gets smaller as more labels are requested, that is $V^*_{m_i} \subseteq V^*_{m_{i-1}}$. This monotonicity plays a crucial role in bounding the number of label requests, as it implies that once a point leaves the disagreement set, it never re-enters it. Please also observe that the disagreement

area of the specific version space for points processed during iteration i stays the same, for instance, $m \in \{m_{i-1}+1, ..., m_i\}$ have the same related disagreement area of the version space which particularly it is $V^*_{m_i}$. From the monotonicity of V^*_m, every m and i have $\mathrm{DIS}(V^*_{m-1}) \subseteq \mathrm{DIS}(V^*_{m_{i-1}})$. Thus, (2.10) conveys that on event E_δ, $V^*_{m_{i-1}} \subseteq \mathrm{B}(f^*, \varepsilon_{i-1})$, and accordingly $\mathrm{DIS}(V^*_{m_{i-1}}) \subseteq \mathrm{DIS}(\mathrm{B}(f^*, \varepsilon_{i-1}))$.

Hence, (2.12) is at most

$$\sum_{i \in I \setminus \{0\}} \sum_{m=m_{i-1}+1}^{m_i} \mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*, \varepsilon_{i-1}))}(X_m). \tag{2.13}$$

To bound the $\sum_{m=1}^{m_{i_\varepsilon}} \mathbf{1}_{\mathrm{DIS}(V^*_{m-1})}(X_m)$, we use the fact that the indicators are Bernoulli random variables representing whether a point X_m is in the disagreement area of version space at step $m-1$. The expected value of each indicator $\mathbf{1}_{\mathrm{DIS}(V^*_{m-1})}(X_m)$, equals the probability that X_m lies within the disagreement set. This probability can be upper-bounded using the size of the version space and the properties of the disagreement set. Thus, the sum of m_{i_ϵ} independent Bernoulli random variables, have the following expected value:

$$\sum_{i \in I \setminus \{0\}} (m_i - m_{i-1}) \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, \varepsilon_{i-1}))).$$

Consequently, by applying the *Chernoff bound*, we show that with high probability, on an event E'_δ of probability at least $1 - \delta/2$, the term in (2.13) is at most:

$$\log_2(2/\delta) + 2e \sum_{i \in I \setminus \{0\}} (m_i - m_{i-1}) \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, \varepsilon_{i-1}))). \tag{2.14}$$

Step 6: Concluding the Proof

According to the definition of m_i, as i increases, $\varepsilon_i = 2^{-i}$ decreases exponentially. The expression inside the ceiling function is positive and increases as i increases because the term $1/\varepsilon_i$ increases exponentially as i increases. Although $\mathrm{Log}(\theta(\varepsilon_i))$ and $\mathrm{Log}(\frac{2(2+i_\varepsilon - i)^2}{\theta})$) are growing more slowly and logarithmically, they are positive and contribute to the overall increase. Since c' is a positive constant the overall expression inside the ceiling function increases. By definition of the disagreement coefficient we know that, $\mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, \varepsilon_{i-1}))) \leq \theta(\varepsilon_{i-1})\varepsilon_{i-1}$. Then by integrating this with the fact that $\forall i, m_{i-1} \leq m_i$, for $i \in I \setminus \{0\}$, $m_i \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, \varepsilon_{i-1})))$ is at most

$$4c'\theta(\varepsilon_{i-1}) \left(d\mathrm{Log}(\theta(\varepsilon_i)) + \mathrm{Log}\left(\frac{2(2 + i_\varepsilon - i)^2}{\delta} \right) \right). \tag{2.15}$$

Consequently, in the meantime $\theta(\varepsilon_{i-1}) \leq \theta(\varepsilon_i) \leq \theta(\varepsilon)$ for $i \in I \setminus \{0\}$, 2.14 is less than

$$\log_2(2/\delta) + 8ec'\theta(\varepsilon) \left(d\mathrm{Log}(\theta(\varepsilon)) + 2\mathrm{Log}\left(\frac{2\log_2(4/\varepsilon)}{\delta} \right) \right) \log_2(2/\varepsilon) \leq n.$$

Specifically, we have demonstrated that during the event $E \cap E_\delta \cap E'_\delta$, $\sup_{h \in V^*_{m_{i_\varepsilon}}} er(h) \leq \varepsilon_{i_\varepsilon} \leq \varepsilon$, and the cumulative quantity of label requests initiated by the CAL while $m \leq m_{i_\varepsilon}$ is less than n; since $2^n > m_{i_\varepsilon}$, This crucial factor implies that we absolutely

require $\max M \geq m_{i_\varepsilon}$, so that $\hat{h} \in V^*_{m_{i_\varepsilon}}$, and thus $er(\hat{h}) \leq \varepsilon$. Taking into account that $E \cap E_\delta \cap E'_\delta$ has probability at least $1-\delta$ (by means of union bound), and that

$$\begin{aligned}\log_2(2/\delta) + 8ec'\theta(\varepsilon)\left(d\mathrm{Log}(\theta(\varepsilon)) + 2\mathrm{Log}\left(\frac{2\log_2(4/\varepsilon)}{\delta}\right)\right)\log_2(2/\varepsilon)\\ \lesssim \theta(\varepsilon)\left(d\mathrm{Log}(\theta(\varepsilon)) + \mathrm{Log}(\mathrm{Log}(1/\varepsilon)/\delta)\right)\mathrm{Log}(1/\varepsilon),\end{aligned}$$

demonstrates the validity of the proof. □

In the realizable case, there are two other important measures worth inspecting. The first one is the number of labels CAL would request among the first m unlabeled data points, namely, it is

$$for\ n, m \in \mathbb{N},\ N(m) = \sum_{i=1}^{m} \mathbf{1}_{\mathrm{DIS}(V^*_{i-1})}(X_i),$$

and, the other one is the number of unlabeled data points CAL would investigate up to its n-th label request, namely:

$$M(n) = min\{k \in \mathbb{N} : N(k) = n\} \cup \{\infty\}.$$

The Theorems (2.2, 2.3) will formally characterize these properties. For extended proofs please refer to the [Han12].

Theorem 2.2 ([H$^+$14]). *For any $m \in \mathbb{N} \cup \{0\}$ and $r \in (0,1)$,*

$$\mathbb{E}[\mathrm{P}_X(\mathrm{DIS}(V^*_m))] \geq (1-r)^m \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r))).$$

This also implies that for any $\varepsilon \in (0,1)$,

$$\mathbb{E}[N(\lceil 1/\varepsilon \rceil)] \geq \theta(\varepsilon)/2.$$

According to this theorem, we can assess the relative performance of the algorithm CAL using specific criteria. Requesting more labels when m is low is generally advantageous. Early in the process, the version space V^*_m is large because the algorithm has observed fewer labeled samples, encompassing a wide range of classifiers consistent with the limited data. Consequently, the disagreement region is extensive since the algorithm has not yet refined the classifier subset.

Frequent label requests at low m indicate that the algorithm is encountering numerous samples within this disagreement region. This is beneficial as it reduces uncertainty early, rapidly shrinking the version space by eliminating classifiers inconsistent with the observed labels in high-uncertainty areas. This accelerates convergence toward an optimal classifier, enabling the algorithm to make more informed decisions as it progresses.

Proof of Theorem 2.2. [Han12] The proof consists of following steps:

Step 1: Setting Definition
Let's put $D_m = \mathrm{DIS}(V^*_m \cap \mathrm{B}(f^*, r))$; This is a set of points that are in the disagreement area where classifiers belong to both *version space* and $r-ball$ centered at f^*. On the other hand, we know that $V^*_m \subseteq \mathrm{B}(f^*, r)$. Thus, $D_m = \mathrm{DIS}(V^*_m \cap \mathrm{B}(f^*, r)) = \mathrm{DIS}(V^*_m)$. Please note that none of the m previously observed points belong to the D_m, namely,

for all m previously processed points $\mathrm{P}_X(\mathrm{DIS}(V_m^*)) = 0$. Therefore, the probability of occurrence of $\mathrm{P}_X(\mathrm{DIS}(V_m^*))$ means to investigate the area where the rest of the unobserved points will happen concerning V_m^* and $\mathrm{B}(f^*, r)$.

Step 2: Formulating Expected Value
Now let us put:

$$\begin{aligned}\mathbb{E}\left[\sum_{m=1}^{\lceil 1/r\rceil} \mathbf{1}_{D_{m-1}}(X_m)\right] &= \sum_{m=1}^{\lceil 1/r\rceil} \mathbb{E}\left[X_m \in D_{m-1} | V_{m-1}^*\right] \\ &= \sum_{m=1}^{\lceil 1/r\rceil} \mathbb{E}[\mathrm{P}_X(D_{m-1})], \end{aligned} \tag{2.16}$$

and make lower bound for $\mathbb{E}[\mathrm{P}_X(D_{m-1})]$ for $m \in \mathbb{N} \cup \{0\}$.

Step 3: Lower Bound for Expected Probability
Please observe that in general if any $x \in \mathrm{B}(f^*, r)$, by definition there could be one or several $h_x \in \mathrm{B}(f^*, r)$ such that the $\mathrm{P}_X(x : h_x(x) \neq f^*(x)) \leq r$ and in case that the $h_x \in V_m^*$, this leads to the fact that $x \in D_m$. So, we have:

$$\forall x, \mathbf{1}_{D_m}(x) \geq \mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*,r))}(x).\mathbf{1}_{V_m^*}(h_x).$$

To convert the $\mathbf{1}_{V_m^*}(h_x)$ as a function of x, we need to interpret the meaning of it in terms of the disagreement area. Hence, if $h_x \in V_m^*$ by definition it means that for all $i \leq m$, X_i's should stay in area where $h_x(X_i) = f^*(X_i)$. This implies that every $i \leq m$, X_i's should belong to **complement** area of where $h_x(X_i) \neq f^*(X_i)$, or equivalently, $\prod_{i=1}^m \mathbf{1}_{\mathrm{DIS}(\{h_x, f^*\})^c}(X_i)$. Accordingly, we have:

$$\forall x, \mathbf{1}_{D_m}(x) \geq \mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*,r))}(x).\mathbf{1}_{V_m^*}(h_x) = \mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*,r))}(x).\prod_{i=1}^m \mathbf{1}_{\mathrm{DIS}(\{h_x,f^*\})^c}(X_i).$$

Now, again according to [Han12]:

$$\begin{aligned}\mathbb{E}(\mathrm{P}_X(D_m)) = \mathrm{P}_X(X_{m+1} \in (D_m)) &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{1}_{D_m}(X_{m+1})\middle| X_{m+1}\right]\right] \\ &\geq \mathbb{E}\left[\mathbb{E}\left[\mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*,r))}(X_{m+1}).\prod_{i=1}^m \mathbf{1}_{\mathrm{DIS}(\{h_x,f^*\})^c}(X_i)\middle| X_{m+1}\right]\right] \\ &= \mathbb{E}\left[\prod_{i=1}^m \mathrm{P}_X\left(h_{X_{m+1}}(X_i) = f^*(X_i)\middle| X_{m+1}\right)\mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*,r))}(X_{m+1})\right] \\ &\geq \mathbb{E}\left[(1-r)^m.\mathbf{1}_{\mathrm{DIS}(\mathrm{B}(f^*,r))}(X_{m+1})\right] \\ &= \left(1-r\right)^m \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*,r))). \end{aligned} \tag{2.17}$$

Step 4: Conclusion
We know the following about geometric series :

$$\sum_{m=1}^{\lceil 1/r\rceil} (1-r)^{m-1} = \frac{1-(1-r)^{\lceil 1/r\rceil}}{r},$$

and also we have $\big(1-r\big)^{\lceil 1/r \rceil} \geq \frac{1}{e}$ for $r \in (0, \epsilon)$ for small $\varepsilon > 0$. Finally, by plugging in the last term in 2.17 into the 2.16 we will have:

$$\begin{aligned}\sum_{m=1}^{\lceil 1/r \rceil} (1-r)^{m-1} \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r))) &= \Big(1-(1-r)^{\lceil 1/r \rceil}\Big) \frac{\mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r)))}{r} \\ &\geq \left(1-\frac{1}{e}\right) \frac{\mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r)))}{r} \\ &\geq \frac{\mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r)))}{2r} = \frac{\theta(r)}{2}, \end{aligned} \tag{2.18}$$

which completes the proof. □

A natural intention and desire is to reduce the probability of disagreement area of version space. It is reasonable to expect some relationship between the number of requested labels and this reduced misclassification yielded by classifiers in version space. On the other hand, having a low expected probability of this set of classifiers implies the performance of chosen classifiers. Theorem 2.3, characterizes this property. The following proof is inspired by the exact proof that could be found in [Han12].

Theorem 2.3 ([H$^+$14]). *For any $n \in \mathbb{N}$ and $r \in (0,1)$,*

$$\mathbb{E}\Big[\mathrm{P}_X\Big(\mathrm{DIS}\Big(V^*_{M(n)}\Big)\Big)\Big] \geq \mathrm{P}_X\Big(\mathrm{DIS}\Big(\mathrm{B}(f^*, r)\Big)\Big) - nr.$$

This also implies that for any $n \in \mathbb{N}$ and $\varepsilon \in (0,1)$,

$$n \leq \theta(\varepsilon)/2 \Longrightarrow \mathbb{E}\Big[\mathrm{P}_X\Big(\mathrm{DIS}\Big(V^*_{M(n)}\Big)\Big)\Big] \geq \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r)))/2.$$

The theorem quantifies the anticipated impact of each label request on reducing the uncertainty associated with the region of disagreement, specifically bounding the expected decrease in probability mass within this region [Han12]. This theorem also measures the performance of the algorithm CAL. Investigating fewer unlabeled data points before each n-th label request is advantageous, as it enhances algorithmic efficiency by rapidly identifying regions of high disagreement among classifiers within the version space $V^*_{M(n)}$. By concentrating on a smaller, more informative subset of data, the algorithm strategically samples areas where label information is most impactful, optimizing label requests while reducing computational and labeling costs.

In sequential learning, the objective is to achieve high accuracy with minimal labeled data, thereby minimizing label complexity. Examining fewer unlabeled points before each label request accelerates the refinement of the version space $V^*_{M(n)}$ and hastens convergence toward an optimal classifier by effectively shrinking the disagreement region and tightening classifier boundaries.

Conversely, exploring too many unlabeled points can lead the algorithm into regions with minimal disagreement, yielding less informative data and expending unnecessary computational effort. Prompt label requests after investigating fewer unlabeled points ensure that each label significantly contributes to reducing classifier uncertainty and prevents wasting resources on less relevant regions.

Proof of Theorem 2.3. [Han12]The proof proceeds as follows:

Step 1: Define Disagreement Region
For every $m \in \mathbb{N} \cup \{0\}$, let $D_m = \text{DIS}(\text{B}(f, r) \cap V_m^*)$. For simplicity, let $M(0) = 0$. Explicitly since $\mathcal{C} = V_0^*$, thus, we have $\mathbb{E}\big[\text{P}_X\big(D_{M(0)}\big)\big] = \mathbb{E}\big[\text{P}_X\big(D_0\big)\big] = \text{P}_X\big(\text{DIS}(\text{B}(f, r)\big)$, which during the proof we will use it as base case.

Step 2: Base Case Analysis
Now, for any fixed $n \in \mathbb{N}$ suppose the following term as the inductive hypothesis:

$$\mathbb{E}\big[\text{P}_X\big(D_{M(n-1)}\big)\big] \geq \text{P}_X\big(\text{DIS}(\text{B}(f, r)\big) - (n-1)r.$$

Step 3: Establish Probability in Disagreement Region
For any $x \in D_{M(n-1)}$, according to the definition of disagreement area, there exist $h_x \in \text{B}(f, r) \cap V_{M(n-1)}^*$ such that $h_x(x) \neq f(x)$. Please note that the h_x is a random variable that strictly belongs to $V_{M(n-1)}^*$. Now if this h_x is also a member of $V_{M(n)}^*$, then we can conclude that $x \in D_{M(n)}$. Formally,

$$\forall x, \mathbf{1}_{D_{M(n)}}(x) \geq \mathbf{1}_{D_{M(n-1)}}(x).\mathbf{1}_{V_{M(n)}^*}(h_x) \tag{2.19}$$

$$= \mathbf{1}_{D_{M(n-1)}}(x).\mathbf{1}_{\text{DIS}(\{h_x, f\})^c}(X_{M(n)}). \tag{2.20}$$

To have $x \in D_{M(n)}$, two conditions should be satisfied. Namely, this x should be in the disagreement area of $V_{M(n-1)}^*$ and the related classifier, h_x also be in the set of the next version space, $V_{M(n)}^*$. Obviously, when $x \in D_{M(n)}$, the index of X should be greater or equal to $M(n)+1$. Because by definition of $V_{M(n)}^*$, for all classifiers in this set, we have $\forall i \leq M(n), h_x(X_i) = f(X_i)$. On the other hand, investigating points whether they are in $D_{M(n)}$ or not, naturally implies that the $D_{M(n-1)}$ (and clearly $V_{M(n-1)}^*$) and $V_{M(n)}^*$ are available. This also means $M(n)$ points have already been processed. Therefore, for deciding about the state of any new point, investigation of current and previous version space, namely, $V_{M(n-1)}^*$ and $V_{M(n)}^*$ is enough. Please note that in the (2.19) we have the right-hand side equal to or greater than the left-hand side. This inequality can be interpreted as follows:

$$(X_{M(n)+1} \in D_{M(n-1)} \;\; \& \;\; h_{X_{M(n)+1}} \in V_{M(n)}^*) \Longrightarrow X_{M(n)+1} \in D_{M(n)}. \tag{2.21}$$

It is important to note that in inequality (2.19), we are restricted only to infer the left-hand side of the inequality from the right-hand side of it. To be more precise, it says that if the right-hand side is one, for sure the left-hand side should be one; otherwise it would be a contradiction. The reason why the other way around is not used is because of the uncertainties and contradictions that come up. Suppose the left-hand side is one then the following states could happen:

- $1 \geq 1 \times 1$
- $1 \geq 1 \times 0$
- $1 \geq 0 \times 1$.

However, for induction having three possibilities is not going to help. Also, logically based on the established setting, the third case, namely, $1 \geq 0 \times 1$, contextually contradicts the setting. It means that there would be a $x \in D_{M(n)}$ and thereof a $h_x \in V_{M(n)}^*$, but $x \notin D_{M(n-1)}$. As mentioned before, contextually this is not correct. That is, there could be situations where the left-hand side is one but the right-hand

side is not. For instance, one equivalent state is when x belongs to both $D_{M(n)}$ and $D_{M(n-1)}$, but the classier h_x that because of it x leis in $D_{M(n-1)}$ is not a member of $V^*_{M(n)}$. In this case, x has been lied in $D_{M(n)}$ due to a different classifier than h_x. Thus, we can say that both logically and setting-wise this inequality is correct. However, the direction and outcome of this inequality are not practical for induction.

Step 4: Calculate Expected Values of Disagreement
Then we will proceed by applying expectations of both sides the (2.20):

$$\begin{aligned}\mathbb{E}\left[\mathrm{P}_X(D_{M(n)})\right] &= \mathbb{E}\left[\mathbf{1}_{D_{M(n)}}(X)\right] \geq \mathbb{E}\left[\mathbf{1}_{D_{M(n-1)}}(X).\mathbf{1}_{DIS(\{h_x,f\})^c}(X_{M(n)})\right] \\ &= \mathbb{E}\left[\mathbf{1}_{D_{M(n-1)}}(X)\mathrm{P}_X\left(h_x(X_{M(n)}) = f(X_{M(n)})\middle|X, V^*_{M(n-1)}\right)\right]. \end{aligned} \tag{2.22}$$

The conditional distribution of $X_{M(n)}$ given $V^*_{M(n-1)}$ is the original distribution P but restricted to the set defined by $V^*_{M(n-1)}$. This means that while the distribution P might be defined over a larger space, we are only interested in the portion of that space where the elements of $V^*_{M(n-1)}$ disagree.

Since $V^*_{M(n-1)}$ represents a smaller subset (the disagreement region) where the hypotheses disagree, the distribution needs to be renormalized. Renormalization adjusts the probabilities so that they sum to one within this restricted region, effectively creating a new probability measure conditioned on being in the disagreement region, namely, $\mathrm{P}_X(.\big|V^*_{M(n-1)})$. Furthermore, since for any $x \in D_{M(n-1)}$ the $\mathrm{DIS}\{h_x, f\} \subseteq \mathrm{DIS}(V^*_{M(n-1)})$ is valid, we can say:

$$\mathrm{P}_X\left(h_x(X_{M(n)}) \neq f(X_{M(n)})\middle|V^*_{M(n-1)}\right) = \frac{\mathrm{P}_X(\mathrm{DIS}(\{h_x, f\}))}{\mathrm{P}_X\left(\mathrm{DIS}(V^*_{M(n-1)})\right)} \leq \frac{r}{\mathrm{P}_X(D_{M(n-1)})}. \tag{2.23}$$

The inequality arises from the fact that $\mathrm{P}_X(\mathrm{DIS}(\{h_x, f\}))$ represents the probability that h_x and f disagree on a randomly chosen data point. Since $h_x \in \mathrm{B}(f, r)$, the maximum probability of disagreement (over all possible h_x in this ball) is at most r. This is a property of the ball $\mathrm{B}(f, r)$; no hypothesis inside this ball can disagree with f more frequently than r. In the following, by plugging in the *complement* of (2.23) into (2.22), its would be at least:

$$\begin{aligned}&\mathbb{E}\left[\mathbf{1}_{D_{M(n-1)}}(X)\left(1 - \frac{r}{\mathrm{P}_X(D_{M(n-1)})}\right)\right] \\ &\quad= \mathbb{E}\left[\mathrm{P}_X\left(X \in D_{M(n-1)}\middle|D_{M(n-1)}\right)\left(1 - \frac{r}{\mathrm{P}_X(D_{M(n-1)})}\right)\right] \\ &\quad= \mathbb{E}\left[\mathrm{P}_X\left(D_{M(n-1)}\right)\left(1 - \frac{r}{\mathrm{P}_X(D_{M(n-1)})}\right)\right] = \mathbb{E}\left[\mathrm{P}_X\left(D_{M(n-1)}\right)\right] - r.\end{aligned}$$

Step 5: Conclusion
Finally, the expansion of the induction proves it:

$$\begin{aligned}\mathbb{E}\left[\mathrm{P}_X(D_{M(n)})\right] &\geq \mathbb{E}\left[\mathrm{P}_X\left(D_{M(n-1)}\right)\right] - r \\ &\geq \mathbb{E}\left[\mathrm{P}_X\left(D_{M(n-2)}\right)\right] - 2r \\ &... \\ &... \\ &... \\ &\geq \mathbb{E}\left[\mathrm{P}_X\left(D_{M(0)}\right)\right] - nr \\ &= \mathrm{P}_X\left(\mathrm{DIS}(\mathrm{B}(f, r))\right) - nr. \end{aligned} \tag{2.24}$$

By maximizing over $r \geq \varepsilon$ and plugging in $n \leq \theta(\varepsilon)/2$ into the 2.24, we will get the implication:

$$\mathbb{E}\Big[\mathrm{P}_X\Big(\mathrm{DIS}\Big(V^*_{M(n)}\Big)\Big)\Big] \geq \mathrm{P}_X(\mathrm{DIS}(\mathrm{B}(f^*, r)))/2. \tag{2.25}$$

□

Following are two theorems that based on them the d, VC dimension, and disagreement coefficient, $\theta(\varepsilon)$ in Theorem 2.1 are derived.

Theorem 2.4 ([H$^+$14]). *If $\mathcal{C}$ is the class of k-dimensional linear separators, and P_X has density (with respect to λ^k), then:*

$$\forall h \in \mathcal{C}, \theta_h(\varepsilon) = o(\frac{1}{\varepsilon}).$$

According to this theorem, the $\theta_h(\varepsilon)$ should be an asymptotic function of $\frac{1}{\varepsilon}$ as in equation 2.2. Figure 2.2 shows some candidates for such a function. As we can see in the figure, as ε approaches zero all of the realizations of $o(\frac{1}{\varepsilon})$ grow significantly asymptotically slower than $\frac{1}{\varepsilon}$.

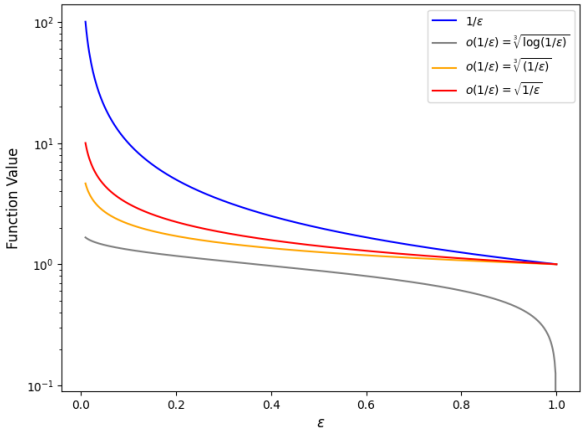


Figure 2.2: Function $v(\varepsilon) = \frac{1}{\varepsilon}$ is in blue, and three samples of asymptotic functions of it.

The last theorem is about the VC dimension of linear classifiers.

Theorem 2.5 ([SSBD14]). *The VC-dimension of general linear classifiers in $\mathbb{R}^k$ is $d = k + 1$.*

This theorem highlights the relationship between the dimensionality of the input space and the capacity of linear classifiers to shatter points. Since as in 2.4, we are using linear classifiers with a domain in $[0, 1]^k$, thus:

$$d = \mathrm{vc}(\mathcal{C}) = k + 1. \tag{2.26}$$

Chapter 3

Noisy Cases

Unlike the realizable case that in its setting, the space of classifiers contains a group of specific and fixed members that at the end, after reducing the non-consistent hypothesis/classifiers, at least one of them would be a desired classifier, in this section we are going to investigate the noisy case where it is based on the relaxation of particular assumptions in realizable cases by injecting noise and distortion into instance space, hypothesis class, and labeling. Furthermore, the output is an estimation of the classifier. Namely, there are no predetermined members among classifiers. This does not mean there would be no frame for the hidden classifier. Some assumptions are taken and would be imposed, but in contrast to realizable scenarios, this method is significantly flexible. Dissimilar to the realizable cases that in some sense impose their classifiers upon the instance space, this noisy framework tries to offer the best representation of the natural configuration of instance space by taking samples from the most uncertain points. In other words, in the end, the algorithm will estimate the optimal classifiers. This is more aligned with real-world problems, where it is less probable that we can fit or even guess the factual classifier(s).

In the following, the setting and definitions are described and then the algorithm and related theorems are investigated.

3.1 Setting

The setting is according to the setting authors have used in [LCK17].

Consider a feature-label pair (X, Y) that follows the joint distribution $\mathrm{P}_{X,Y}$. The marginal distribution of X, denoted by P_X, is supported on the unit cube $[0,1]^k$, while the label Y is binary and takes values in $\{0,1\}$. The conditional distribution of Y given $X = x$, denoted as $\mathrm{P}_{Y|X=x}$, is described by the regression function:

$$\eta(x) := \mathbb{E}[Y|X = x], \quad \forall x \in [0,1]^k.$$

The function $\eta(x)$ represents the probability that $Y = 1$ given that $X = x$. Thus, $\eta(x)$ serves as the core function that defines the relationship between X and Y over the feature space $[0,1]^k$. Additionally, we use P whenever we want to show sampling.

We extend η to a function $\eta : \mathbb{R}^k \to [0,1]$, although our main interest is within the unit cube $[0,1]^k$. The goal of a learning algorithm is to find a classifier $f : [0,1]^k \to$

$\{0, 1\}$ that minimizes the probability of misclassification. The Bayes optimal classifier f^* with a classifier f that has a minimal excess risk $\mathcal{E}(f)$. Where,

$$\mathcal{E}(f) := \mathcal{E}_{\mathrm{P}_{X,Y}}(f) := R(f) - R(f^*).$$

The excess risk $\mathcal{E}(f)$ quantifies the additional error incurred by f relative to the optimal Bayes classifier f^*. It can also be expressed as:

$$\mathcal{E}(f) = \int_{x\in[0,1]^k} |1 - 2\eta(x)| \, \mathrm{dP}_X(x), \tag{3.1}$$

however, it is clear that over $\{x \in [0, 1]^k : f(x) = f^*(x)\}$, the $R(f) - R(f^*) = 0$. Thus, the excess risk is the result of the region where $\{x \in [0, 1]^k : f(x) \neq f^*(x)\}$. That is:

$$\mathcal{E}(f) = \int_{x\in[0,1]^k : f(x)\neq f^*(x)} |1 - 2\eta(x)| \, \mathrm{dP}_X(x). \tag{3.2}$$

This integral indicates that the excess risk depends on the regions where f and f^* disagree, weighted by $1 - 2\eta(x)$. The factor $1 - 2\eta(x)$ reflects the uncertainty associated with each x; points, where $\eta(x)$ is close to 0.5, contribute less to the excess risk since the classification decision is less certain.

Furthermore, we will use the symbol $\wedge$, to show the minimum between two numbers. For example:

$$a \wedge b = \min(a, b),$$

and the symbol $\vee$, as the logical OR operation. For instance:

$$a \vee b = a \text{ OR } b.$$

Additionally, For any set A, let $\mathbf{1}_A$ denote the indicator function for A. This function assigns a value of 1 if $x \in A$ and 0 otherwise, represented by $\mathbf{1}_A(x) = 1$ if $x \in A$, and $\mathbf{1}_A(x) = 0$ if $x \notin A$. Similarly, for any logical condition L, we may use the notation $\mathbf{1}[L]$, where $\mathbf{1}[L] = 1$ when L is true, and $\mathbf{1}[L] = 0$ when L is false.

The learner can interactively query the label Y for any $x \in [0, 1]^k$. The query process follows a Bernoulli trial with a success probability $\eta(x)$, meaning the learner receives $Y = 1$, with probability $\eta(x)$ and $Y = 0$, with probability $1 - \eta(x)$. This setup is equivalent to drawing samples from the conditional distribution $\mathrm{P}_{Y|X=x}$ for $x \in [0, 1]^k$ [LCK17].

Given a fixed budget of n samples, $n \in \mathbb{N}$, the learner's objective is to construct a classifier $\hat{f}_n$ that minimizes the excess risk $\mathcal{E}(\hat{f}_n)$. The learner must decide which points x to sample based on the available budget to best approximate f^* with high confidence.

The challenge lies in designing a sampling strategy that chooses the most informative points for querying, such that the resulting classifier $\hat{f}_n$ achieves low excess risk $\mathcal{E}(\hat{f}_n)$ with high probability. This involves selecting points x where the label Y is uncertain, as these points are likely to yield the most informative updates to the classifier.

3.2 Definitions

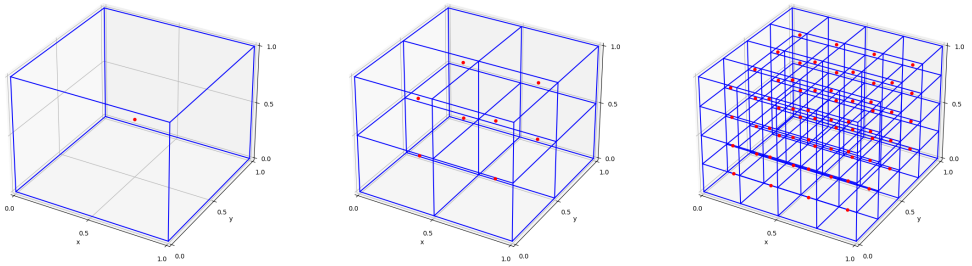
In this section, we will have the required definitions.

Definition 3.1. [LCK17] *[Dyadic grid G_l, cells C, center x_C, and diameter r_l] A dyadic grid, denoted by G_l, is a regular grid that subdivides the unit cube into 2^{lk} smaller, identical cubes called cells. Each cell has a side length of 2^{-l} and a volume of 2^{lk} . These cells fill the unit cube without overlapping, namely, $[0,1]^k = \bigcup_{C\in G_l} C$ and $C \cap C' = \emptyset$ if $C \neq C'$, with $C, C' \in G_l$. The center of a cell C, denoted by x_C, is its geometric center. The diameter of the cell C is:*

$$r_l \doteq \max_{x,y\in C} |x-y|_2 = \sqrt{k}2^{-l}, \tag{3.3}$$

where $|.|_2$ is Euclidean norm.

The Figure(3.1) shows an example of dyadic grid G_l with $k = 3$ for different levels, $l = [0, 1, 2]$.



(a) l= 0 (b) l= 1 (c) l= 2

Figure 3.1: Dyadic grid G_l for different depths l=[0, 1, 2] with k=3; Center of each cell, x_C, is in red.

Assumption 3.1. [LCK17] (**Strong density**) *There is a definite positive constant c_1 such that for any level $l \geq 0$ and any cell C in the grid G_l with a non-zero probability $\mathrm{P}_X(C_l)$, it holds:*

$$\mathrm{P}_X(C_l) \geq c_1 k^{k/2} 2^{-lk}.$$

This assumption ensures that the probability of any cell in the grid is not too small. It allows us to establish a minimum value for the probability of a cell. This assumption holds, for example, when the probability distribution P_X is uniform or nearly uniform [LCK17]. This is why it is an *inequality* and not simply an *equation*.

Definition 3.2. [LCK17] (**Hölder smoothness**) *For functions $g : \mathbb{R}^k \to [0,1]$, from Hölder class $\Sigma(\alpha, \lambda)$ with $\alpha > 0$ and $\lambda > 0$, that are $\lfloor\alpha\rfloor$ times continuously differentiable, for any $j \in \mathbb{N}, j \leq \alpha$ it holds:*

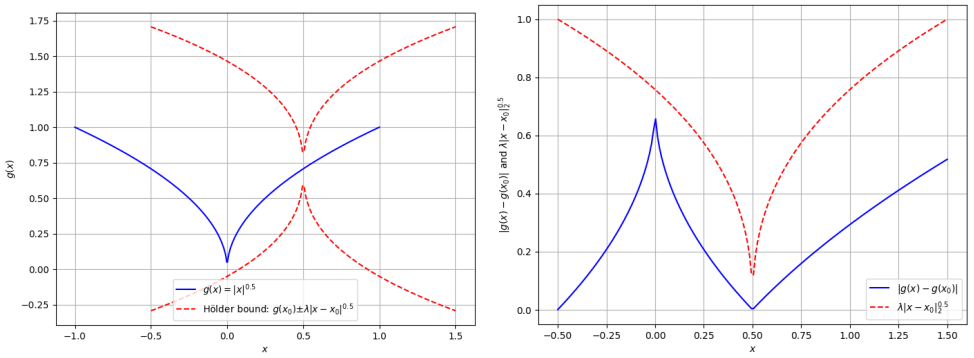
$$\sup_{x\in\mathbb{R}^k} \sum_{|s|=j} |D^s g(x)| \leq \lambda, \quad \textit{and,} \quad \sup_{x,y\in\mathbb{R}^k, |x-y|\leq 1} \sum_{|s|=\lfloor\alpha\rfloor} \frac{|D^s g(x) - D^s g(y)|}{|x-y|_2^{\alpha-\lfloor\alpha\rfloor}} \leq \lambda,$$

where $D^s f$ denotes the mixed partial derivative with parameter s. Note that for $\alpha \leq 1$ and $\lambda \geq 1$, the condition is reduced to:

$$\sup_{x,y \in \mathbb{R}^k} \frac{|g(y) - g(x)|}{|y - x|_2^\alpha} \leq \lambda.$$

α-Hölder functions possess smoothness properties, facilitating approximation by polynomials of degree $\lfloor \alpha \rfloor$ or other approximation schemes, including kernel methods.

Assumption 3.2. [LCK17] (**Hölder smoothness** of η) *Assume η is a member of the class $\Sigma(\alpha, \lambda)$, defined for positive α and non-negative λ.*



(a) (b)

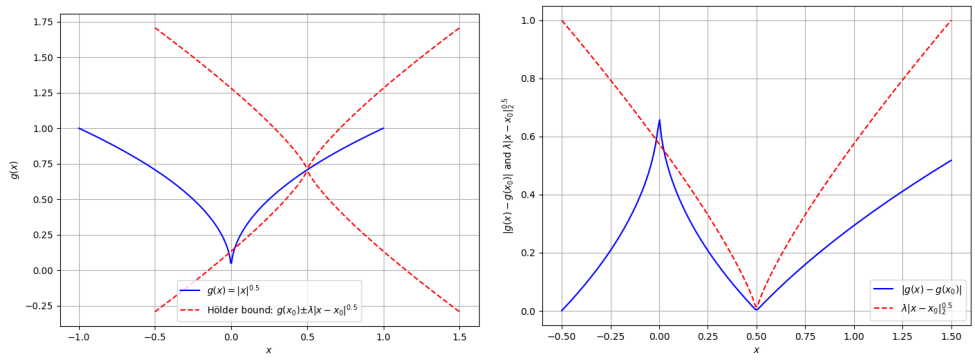
Figure 3.2: Function $g(x) = \sqrt{|x|}$ is α-Hölder for class of $\Sigma(\alpha = 0.4, \lambda = 1)$

Figure(3.2) illustrates function $g(x) = \sqrt{|x|}$ that holds the Hölder smoothness condition for the class $\Sigma(\alpha, \lambda)$ with $\alpha = 0.4$ and $\lambda = 1$. Figures (3.2b) demonstrate the Hölder smoothness in the form, $|g(x) - g(x_0)| \leq \lambda |x - x_0|_2^\alpha$, while Figure(3.2a) is the extended realization of absolute value:

$$g(x) \leq g(x_0) + \lambda |x - x_0|_2^\alpha,$$

$$g(x) \geq g(x_0) - \lambda |x - x_0|_2^\alpha.$$

On the other hand, just by changing α and putting it equal to 0.8, the function $g(x)$ violates the Hölder smoothness conditions at class $\Sigma(0.8, 1)$; Figure(3.3). The intersection of the red and blue lines visualizes the violation of Hölder smoothness's condition. Therefore, we can see that it could happen for some combinations of α and λ, the function $g(x) = \sqrt{|x|}$ will not be according to the definition of the Hölder smoothness.



(a) (b)

Figure 3.3: Function $g(x) = \sqrt{|x|}$ is ***not*** α-Hölder for class of $\Sigma(\alpha = 0.8, \lambda = 1)$.

Assumption 3.3. [LCK17] (**Margin condition**) *There is a non-negative* c_3, Δ_0, *and* β *for which* $\forall \Delta > 0$*:*

$$\mathrm{P}_X(|\eta(X) - 1/2| < \Delta_0) = 0, \text{ and, } \mathrm{P}_X(|\eta(X) - 1/2| \leq \Delta_0 + \Delta) \leq c_3 \Delta^{\beta}.$$

This assumption essentially describes the behavior of the function $\eta(X)$ near the decision boundary, where $\eta(X)$ is close to $1/2$. The parameters Δ_0, Δ, and β control how fast the probability of being close to the decision boundary decreases as we move away from it. Since $\Delta_0 \geq 0$, it is obvious that for $\Delta_0 = 0$ the $\mathrm{P}_X(|\eta(X) - 1/2| < \Delta_0)$ would be zero (i.e. distance cannot be negative). Additionally, for $\Delta_0 > 0$ having $\mathrm{P}_X(|\eta(X) - 1/2| < \Delta_0)$ equal to zero implies probabilistically it is impossible that $\eta(X)$ stays in a distance smaller than Δ_0 from $1/2$. Moreover, by increasing the distance from Δ_0 to $\Delta_0 + \Delta$, the probability changes. This change will establish an upper bound on the probability. However, consistent with the provided explication about the Δ_0, the probability that the absolute difference between $\eta(X)$ and one-half is less than or equal to the sum of Δ_0 and Δ is bounded above by *only* c_3,Δ and β; to be more precise, $c_3\Delta^{\beta}$. It is worth mentioning two well-established instances in literature including *Tsybakov's noise condition* ($\Delta_0 = 0, \beta > 0$) and *Massart's margin condition* ($\Delta_0 = 0, \beta > 0$) that are commonly studied in this field.

Definition 3.3. [LCK17] *We define* $\mathcal{P}(\alpha, \beta, \Delta_0) := \mathcal{P}(\alpha, \beta, \Delta_0; \lambda, c_3)$ *as the set of classification problems* $\mathrm{P}_{X,Y}$ *characterized by* $(\eta; \mathrm{P}_X)$*, where Assumptions 3.2 and 3.3 hold with parameters* $\alpha > 0$, $\beta \geq 0$, *and* $\Delta_0 \geq 0$, *for some fixed* $\lambda \geq 1$ *and* $c_3 > 0$*. Additionally, we denote* $\mathcal{P}^*(\alpha, \beta, \Delta_0)$ *as the subset of* $\mathcal{P}(\alpha, \beta, \Delta_0)$ *where* P_X *also satisfies Assumption 3.1.*

Please note that the constant values of $c_3 > 0$ and $\lambda \geq 1$ are for all subsequent analyses.

Definition 3.4. [LCK17] ((δ, Δ, n)-**correct algorithm**) *Consider a process designed to generate two distinct, measurable subsets, denoted as* S^0, S^1*, within the k-dimensional unit cube,* $[0, 1]^k$*. Let* $0 < \delta < 1$*, and* $\Delta \geq 0$*. We refer to such a procedure as* ***weakly*** (δ, Δ, n)-***correct*** *for a classification problem* $\mathrm{P}_{X,Y}$ *(characterized by* (η, P_X)*) provided*

that, with a probability exceeding $1 - 8\delta$ *across a maximum of* n *label inquiries:*

$$\left\{x \in [0,1]^k : \eta(x) - 1/2 > \Delta\right\} \subset S^1, \text{ and } \left\{x \in [0,1]^k : 1/2 - \eta(x) > \Delta\right\} \subset S^0.$$

Furthermore, assuming the occurrence of the same probabilistic event across a maximum of n *label requests, we observe that:*

$$S^1 \subset \left\{x \in [0,1]^k : \eta(x) - 1/2 > 0\right\}, \text{ and } S^0 \subset \left\{x \in [0,1]^k : \eta(x) - 1/2 < 0\right\}.$$

Consequently, such a procedure is conventionally termed (δ, Δ, n)-***correct*** *for* $\mathrm{P}_{X,Y}$.

In the next section, an algorithm and the theory behind it will be discussed.

3.3 Algorithm and Theorems

We start with the following algorithm which is a modification of the original version in [LCK17]. Since, in this work, we are supposed to focus on the case where $\alpha \leq 1$, the algorithm has been modified accordingly. Thus, wherever in the original work at [LCK17], there was $\alpha \wedge 1$ or $\alpha \vee 1$, we would replace them accordingly by α and 1.

Algorithm 2

Input: $n, \delta, \alpha, \lambda$
Initialisation: $t = 2^k t_{1,\alpha}$, $l = 1$, $\mathcal{A}_1 = G_1$(active space), $\forall l' > 1$, $\mathcal{A}_{l'} = \emptyset$, $S^0 = S^1 = \emptyset$
while $t \leq n$ **do**
  for each active cell $C \in A_l$ **do**
    Request $t_{l,\alpha}$ samples $(\tilde{Y}_{C,i})_{i \leq t_{l,\alpha}}$ at the center x_C of C
    if $|\hat{\eta}(x_C) - 1/2| \leq B_{l,\alpha}$ **then**
      $\mathcal{A}_{l+1} = \mathcal{A}_{l+1} \cup \{C' \in G_{l+1} : C' \subset C\}$ ▷ keep all children C' of C active
    else
      Let $y = 1\{\hat{\eta}(x_C) \geq 1/2\}$
      $S^y = S^y \cup C$ ▷ label the cell as class y
    end if
  end for
  Increase depth to $l = l + 1$, and set $t = t + |\mathcal{A}_l| \cdot t_{l,\alpha}$
end while
Output: S^y for $y \in \{0, 1\}$, and $\hat{f}_{n,\alpha} = \mathbf{1}\{S^1\}$

The algorithm presented is designed given the λ and α, to focus its exploration of the data space specifically on regions where classifying the problem is most challenging. This difficulty arises particularly in areas where the function η, hovers near the critical threshold of 1/2, making it hard to classify. The algorithm achieves this targeted exploration by incrementally refining a partition of the space. This partitioning is structured based on a dyadic tree, which is for recursively splitting the space into smaller and smaller sections.

At each level of depth l within this partitioning tree, the algorithm selects a specific point, x_C, which is the center of an active cell C within the partition $\mathcal{A}_l$. The algorithm then samples this point x_C a predetermined number of times, denoted as $t_{l,\alpha}$ where:

$$t_{l,\alpha} = \frac{\log_2(1/\delta_{l,\alpha})}{2b_{l,\alpha}^2}, \quad \text{if } \alpha \leq 1. \tag{3.4}$$

Specifically, $b_{l,\alpha}$ is given by $\lambda k^{(\alpha)/2} 2^{-l\alpha}$, while $\delta_{l,\alpha}$ scales the $\delta 2^{-l(k+1)}$, where δ is a predefined parameter that controls the confidence level of the classification.

The core of the algorithm's decision-making process lies in estimating the function $\eta(x_C)$ at the point x_C using the collected samples. This estimate, $\hat{\eta}(x_C)$, , is computed as the average of the sample values $\tilde{Y}_{C,i}$ at x_C normalized by the number of samples $t_{l,1}$, that is:

$$\hat{\eta}(x_C) = t_{l,\alpha}^{-1} \sum_{i=1}^{t_{l,\alpha}} \tilde{Y}_{C,i}.$$

The next step in the algorithm involves comparing the absolute difference between this estimate $\hat{\eta}(x_C)$ and the critical threshold 1/2. If this difference, $|\hat{\eta}(x_C) - 1/2|$, is sufficiently large, relative to a total error comprising bias and deviation bound:

$$B_{l,\alpha} = 2\left[\sqrt{\frac{\log_2(1/\delta_{l,\alpha})}{2t_{l,\alpha}}} + b_{l,\alpha}\right], \tag{3.5}$$

then the algorithm can confidently label the cell C as belonging to the class S^0 or S^1. This labeling process is crucial for the algorithm to decide the class of cells based on the empirical evidence:

$$\mathbf{1}\big\{\hat{\eta}(x_C) \geq 1/2\big\}.$$

However, if the difference $|\hat{\eta}(x_C) - 1/2|$ is too small, indicating that the algorithm is not confident enough in the classification, the partition is further refined. The cell C is divided into smaller subcells, and these new, finer cells become the active cells at the next depth level $l + 1$.

The process of refining and labeling continues until the algorithm reaches a point where the total used budget—measured in terms of the number of samples taken—that is $t + t_{l,\alpha}.|\mathcal{A}_l|$ exceeds the allowed limit n. When this happens, the algorithm stops, and the final depth reached is denoted as L.

The Equation 3.5, which makes a confidence bound around $\eta(x)$, consists of two main components. The deviation of the unknown optimal classifier, $\sqrt{\frac{\log(1/\delta_{l,\alpha})}{2t_{l,\alpha}}}$, plus the deviation of its estimation, $b_{l,\alpha}$. Adding them up and multiplying them by two will establish $B_{l,\alpha}$ a significantly conservative bound to decide about the cell's destiny.

Theorem 3.1 ([LCK17]). *Algorithm 2 run on a problem $\mathcal{P}^*(\lambda, \alpha, \beta, \Delta_0)$ with input parameters $n, \delta, \alpha, \lambda$ is $(\delta, \Delta^*_{n,\delta,\alpha,\lambda}, n)$-correct, with:*

$$\Delta^*_{n,\delta,\alpha,\lambda} = 12\sqrt{k}\left(\frac{c_7\lambda^{(\frac{k}{\alpha}\vee\beta)}\log\big(\frac{2d\lambda^2 n}{\delta}\big)}{(2\alpha + [k - \alpha\beta]_+)\alpha\ n}\right)^{\frac{\alpha}{2\alpha+[k-\alpha\beta]_+}}, for\ \alpha \leq 1,$$

with $c_7 = 2(k + 1)c_5$ and $c_5 = 2^\beta \max\big(\frac{c_3}{c_1}8^\beta, 1\big)$, where c_1 and c_3 are the constants involved in Assumption 3.1 and 3.3 respectively.

The proof of Theorem 3.1, depends on several lemmas and consists of six steps. The lemmas will be stated step by step. The proof of lemmas will be declared separately at the end after the proof of the Theorem 3.1. This helps avoid confusion.

Proof of Theorem 3.1. [LCK17] We will use the following abbreviation to make life easier:

$$t_l = t_{l,1},\ b_l = b_{l,\alpha},\ \delta_l = \delta_{l,1},\ B_l = B_{l,1},\ \text{and}\ N_l = |\mathcal{A}_l|.$$

Step 1: A favorable event.
Let C be a cell at depth l. The following event is defined:

$$\xi_{C,l} = \left\{\left|t_l^{-1}\sum_{u=1}^{t_l}\mathbf{1}(\tilde{Y}_{C,i} = 1) - \eta(x_C)\right| \leq \sqrt{\frac{\log(1/\delta_l)}{2t_l}}\right\}, \tag{3.6}$$

such that the $(\tilde{Y}_{C,i})_{i\leq t_l}$ are samples obtained within cell C at point x_C if the algorithm chooses to sample in cell C. Recall that:

$$\hat{\eta}(x_C) = t_l^{-1} \sum_{i=1}^{t_l} \mathbf{1}(\tilde{Y}_{C,i} = 1).$$

We define the event ξ as follows:

$$\xi = \left\{ \bigcap_{l\in\mathbb{N}^*, C\in G_l} \xi_{C,l} \right\}.$$

The left-hand side of inequality in the above set (3.6) comes from Hoeffding's inequality for estimating *confidence intervals*. We know that $\eta(x_C)$ is a function between zero and one, so:

$$\mathrm{P}\left(\hat{\eta}(x_C) \notin [\eta(x_C) - \varepsilon, \eta(x_C) + \varepsilon]\right) \geq 2\mathrm{exp}(-2\varepsilon^2 t_l),$$

which is equivalent to:

$$\mathrm{P}\left(|\hat{\eta}(x_C) - \eta(x_C)| > \varepsilon\right) \geq 2\mathrm{exp}(-2\varepsilon^2 t_l). \tag{3.7}$$

Now, let's put $\delta_l = \exp(-2\varepsilon^2 t_l)$. We will have ε as following:

$$\frac{1}{\delta_l} = \exp(2\varepsilon^2 t_l), \tag{3.8}$$
$$\varepsilon = \sqrt{\frac{\ln(\frac{1}{\delta_l})}{2t_l}}.$$

Since for the given $0 < \delta_l < 1$, it always holds that $\ln(\frac{1}{\delta_l}) \geq \log((\frac{1}{\delta_l}))$, thus:

$$\sqrt{\frac{\log(\frac{1}{\delta_l})}{2t_l}} \leq \varepsilon. \tag{3.9}$$

Finally by plugging in (3.8) and (3.9) into (3.10), we get the following inequality that later will be used for proving other lemmas:

$$\mathrm{P}\left(|\hat{\eta}(x_C) - \eta(x_C)| > \sqrt{\frac{\log(\frac{1}{\delta_l})}{2t_l}}\right) \geq 2\delta_l. \tag{3.10}$$

Lemma 3.1. [LCK17] *It follows that*

$$\mathrm{P}(\xi) \geq 1 - 4\delta.$$

Furthermore, on the event ξ, we have

$$|\widehat{\eta}(x_C) - \eta(x_C)| \leq b_l. \tag{3.11}$$

Step 2: No mistakes on labeled cells.
For any integer $l \in \mathbb{N}^*$, let C be an element of G_l. We can express:

$$\widehat{k}_C^* = \mathbf{1}\{\widehat{\eta}(x_C) \geq 1/2\} \quad \text{additionally, let us define} \quad k_C^* \doteq \mathbf{1}\{\eta(x_C) \geq 1/2\}.$$

Lemma 3.2. [LCK17] *We have that on ξ,*

$$\forall y \in \{0,1\}, \forall C \in S^y, \forall x \in C, \quad \mathbf{1}\{\eta(x) \geq 1/2\} = y. \tag{3.12}$$

This implies that:

$$S^1 \subset \{x : \eta(x) - 1/2 > 0\} \quad \textit{and,} \quad S^0 \subset \{x : \eta(x) - 1/2 < 0\}.$$

Step 3: Maximum gap with respect to 1/2 for all active cells
When cell C is split and incorporated into $\mathcal{A}_{l+1}$ at depth $l \in \mathbb{N}^*$, the Algorithm (2) and the definition of ξ, as outlined in Equation (3.11), and by the means of properties of triangle inequalities ensure that:

$$\begin{aligned} |\eta(x_C) - 1/2| - |\widehat{\eta}(x_C) - 1/2| &\leq |\eta(x_C) - \widehat{\eta}(x_C)| \\ &\leq |\eta(x_C) - 1/2| + |\widehat{\eta}(x_C) - 1/2| \leq 2b_l, \end{aligned}$$

which simply gives us the upper bound for the left-hand side of it as:

$$|\eta(x_C) - 1/2| - |\widehat{\eta}(x_C) - 1/2| \leq 2b_l, \tag{3.13}$$

and similarly:

$$\begin{aligned} |\widehat{\eta}(x_C) - 1/2| - |\eta(x_C) - 1/2| &\leq |\eta(x_C) - \widehat{\eta}(x_C)| \\ &\leq |\eta(x_C) - 1/2| + |\widehat{\eta}(x_C) - 1/2| \leq 2b_l, \end{aligned} \tag{3.14}$$

yields following upper bound too:

$$\begin{aligned} |\widehat{\eta}(x_C) - 1/2| &\leq |\eta(x_C) - \widehat{\eta}(x_C)| + |\eta(x_C) - 1/2| \\ &\leq 2|\eta(x_C) - 1/2| + |\widehat{\eta}(x_C) - 1/2| \leq 3b_l, \end{aligned} \tag{3.15}$$

or simply:

$$|\widehat{\eta}(x_C) - 1/2| \leq 3b_l. \tag{3.16}$$

Finally combining (3.13) and (3.16) results in:

$$|\eta(x_C) - 1/2| - b_l \leq |\widehat{\eta}(x_C) - 1/2| \leq 4b_l, \tag{3.17}$$

which suggests $|\eta(x_C) - 1/2| \leq 5b_l$. Based on Equation (3.23), we can conclude that for any cell C scheduled to be split and added to $\mathcal{A}_{l+1}$ and for any element $x \in C$, the following condition holds on ξ:

$$|\eta(x) - 1/2| - b_l \leq |\eta(y) - 1/2| \leq 5b_l,$$

plainly:

$$|\eta(x) - 1/2| \leq 6b_l \doteq \Delta_l. \tag{3.18}$$

Step 4: Bound on the number of active cells
For any $\Delta \geq 0$, we define the set:

$$\Omega_\Delta = \{x \in [0,1]^k : |\eta(x) - 1/2| \leq \Delta\},$$

and any positive integer l, let $N_l(\Delta)$ denote the count of cells C belonging to the set G_l that are completely contained within the region Ω_Δ, namely, $C \subset \Omega_\Delta$.

Lemma 3.3. [LCK17] *On the event ξ, we have:*

$$\begin{aligned} N_{l+1} &\leq \frac{c_3}{c_1}[\Delta_l - \Delta_0]_+^\beta r_{l+1}^{-k} \\ &\leq c_5 \lambda^\beta r_{l+1}^{-[k-(1)\beta]_+} \mathbf{1}\{\Delta_l > \Delta_0\}. \end{aligned} \tag{3.19}$$

This lemma is significant in bounding the complexity of the active learning process. By controlling the number of active cells, one can manage the sample complexity and ensure the learning process remains efficient. The lemma essentially says that as the resolution increases (as l increases), the number of cells that are still "active" and entirely within the uncertain region Ω can be bounded in terms of the size of the cells and the noise parameter Δ.

Step 5: A minimum depth.

Lemma 3.4. [LCK17] *On the event ξ, we obtain the following results regarding L: since $\alpha \leq 1$ we have the following*

$$L \geq \frac{1}{2\alpha + [k - \alpha\beta]_+} \log_2 \Big(\frac{(2\alpha + [k - \alpha\beta]_+)2\alpha n}{c_7 \lambda^{\beta-2} \log(\frac{2k\lambda^2 n}{\delta})} \Big) - 1, \tag{3.20}$$

with $c_7 = 2c_5(k+1)$. Alternatively, the algorithm may terminate before reaching depth L, resulting in $\mathcal{E}(\hat{f}_n) = 0$.

Step 6: Conclusion.
From this point on, we write S^0, S^1 for the sets that Algorithm (2) outputs at the end (so the sets at the end of the algorithm).

We present the following lemma.

Lemma 3.5. [LCK17] *If S^1 and S^0 are disjoint sets, $S^1 \cap S^0 = \emptyset$, and if there exists a non-negative value $\Delta_L \geq 0$, such that on some event ξ', we have*

$$\{x \in [0,1]^k : \eta(x) - \Delta_L \geq 1/2\} \subset S^1, \text{ and } \{x \in [0,1]^k : \eta(x) + \Delta_L \leq 1/2\} \subset S^0,$$

Then, on the event ξ', it follows that

$$\sup_{x \in [0,1]^k : \hat{f}_{n,a} \neq f^*(x)} |\eta(x) - 1/2| \leq \Delta_L, \text{ and } \mathrm{P}_X(\hat{f}_{n,a} \neq f^*) \leq c_3 \Delta_L^\beta \mathbf{1}\{\Delta_L \geq \Delta_0\},$$

and

$$\mathcal{E}(\hat{f}_{n,a}) \leq c_3 \Delta_L^{1+\beta} \mathbf{1}\{\Delta_L \geq \Delta_0\}.$$

Finally, the above Lemmas gives us the requirements to prove the Theorem. Observe that by definition of the algorithm, $S^1 \cap S^0 = \emptyset$, which means they are disjoint sets. According to Equations (3.12), (3.18) we know that on the event ξ (and consequently with a probability exceeding $1 - 4\delta$):

$$\{x \in [0,1]^k : \eta(x) - \Delta_L \geq 1/2\} \subset S^1, \text{ and } \{x \in [0,1]^k : \eta(x) + \Delta_L \leq 1/2\} \subset S^0, \quad (3.21)$$

where

$$\begin{aligned} \Delta_L &\leq 6 b_{l,\alpha} \\ &= 6\lambda k^{\frac{\alpha}{2}} 2^{-l\alpha}. \end{aligned} \quad (3.22)$$

thus by manipulating Equations 3.20

$$\begin{aligned} \Delta_L &\leq 6\lambda k^{\alpha/2} 2^{\alpha} \Big(\frac{c_7 \lambda^{\beta-2} \log\left(\frac{2k\lambda^2 n}{\delta}\right)}{(2\alpha + [k-\alpha\beta]_+) \cdot 2\alpha n} \Big)^{\alpha/(2\alpha+[k-\alpha\beta]_+)} \\ &\leq 12\lambda k^{\alpha/2} \Big(\frac{c_7 \lambda^{\beta-2} \log\left(\frac{2d\lambda^2 n}{\delta}\right)}{(2\alpha + [k-\alpha\beta]_+) \cdot 2\alpha n} \Big)^{\alpha/(2\alpha+[k-\alpha\beta]_+)} \\ &\leq 12\sqrt{k} \Big(\frac{c_7 \lambda^{\beta-2} \log\left(\frac{2k\lambda^2 n}{\delta}\right)}{(2\alpha + [k-\alpha\beta]_+) \cdot 2\alpha n} \Big)^{\alpha/(2\alpha+[k-\alpha\beta]_+)}. \end{aligned}$$

According to Equation (3.20), this implies the validity of Theorem 3.1 for $\alpha \leq 1$. Consequently, by Lemma (5), we have on the event ξ (and therefore with a probability exceeding $1 - 4\delta$):

$$\begin{aligned} \sup_{x \in [0,1]^k : \hat{f}_{n,a} \neq f^*(x)} |\eta(x) - 1/2| &\leq \Delta_L \\ &\leq 12\sqrt{k} \Big(\frac{c_7 \lambda^{\beta-2} \log\left(\frac{2k\lambda^2 n}{\delta}\right)}{(2\alpha + [k-\alpha\beta]_+) \cdot 2\alpha n} \Big)^{\alpha/(2\alpha+[k-\alpha\beta]_+)}. \end{aligned}$$

Furthermore,

$$\begin{aligned} \mathrm{P}_X(\hat{f}_{n,a} \neq f^*(x)) &\leq c_3 \Delta_L^{\beta} \mathbf{1}\{\Delta_L \geq \Delta_0\} \\ &\leq c_3 12^{\beta} \sqrt{k} \Big(\frac{c_7 \lambda^{(\frac{k}{\alpha} \vee \beta)} \log\left(\frac{2k\lambda^2 n}{\delta}\right)}{(2\alpha + [k-\alpha\beta]_+) \cdot 2\alpha n} \Big)^{\alpha\beta/(2\alpha+[k-\alpha\beta]_+)} \end{aligned}$$

and

$$\begin{aligned} \mathcal{E}(\hat{f}_{n,a}) &\leq c_3 \Delta_L^{\beta+1} \mathbf{1}\{\Delta_L \geq \Delta_0\} \\ &\leq c_3 12^{\beta+1} \sqrt{k} \Big(\frac{c_7 \lambda^{(\frac{k}{\alpha} \vee \beta)} \log\left(\frac{2k\lambda^2 n}{\delta}\right)}{(2\alpha + [k-\alpha\beta]_+) \cdot 2\alpha n} \Big)^{\alpha(\beta+1)/(2\alpha+[k-\alpha\beta]_+)}. \end{aligned}$$

□

In this section, the proofs of the Lemmas are given. They are somehow similar to proofs in [LCK17]. However, more details and explanations are given wherever it could help for more clarification and understanding.

Proof of Lemma 3.1. [LCK17] According to Hoeffding's inequality, and definition of $\xi_{C,l}$ in (3.6) we know that:

$$\mathrm{P}(\xi_{C,l}) = \mathrm{P}\left(\left|t_l^{-1}\sum_{u=1}^{t_l}\mathbf{1}(\tilde{Y}_{C,i}=1) - \eta(x_C)\right| \leq \sqrt{\frac{\log(1/\delta_l)}{2t_l}}\right) \geq 1-2\delta_l.$$

And if for its complement, $\xi_{C,l}^c$:

$$\mathrm{P}(\xi_{C,l}^c) = 1 - \mathrm{P}(\xi_{C,l}) \leq 2\delta_l,$$

which is equivalent to:

$$\mathrm{P}(\xi_{C,l}^c) = 1 - \mathrm{P}(\xi_{C,l}) \leq 2\delta_l.$$

Let us now turn our attention to,

$$\xi = \left\{\bigcap_{l\in\mathbb{N}^*, C\in G_l} \xi_{C,l}\right\},$$

We examine the intersection of events defined by the condition that, for every depth l and every cell $C \in G_l$, the preceding event occurs. Given that there are 2^{ld} such events at depth l, a straightforward application of the union bound yields:

$$\begin{aligned}
\mathrm{P}(\xi) = \mathrm{P}\left(\bigcap_{l\in\mathbb{N}^*, C\in G_l} \xi_{C,l}\right) &= 1 - \mathrm{P}\left(\bigcup_{l\in\mathbb{N}^*, C\in G_l} \xi_{C,l}^c\right) \\
&\geq 1 - \sum_l 2\delta_l 2^{lk} \\
&= 1 - \sum_l 2\delta 2^{-l(k+1)} 2^{lk} \\
&= 1 - 2\delta\sum_l 2^{-l} \\
&= 1 - 2\delta\left(\frac{1}{1-\frac{1}{2}}\right) \\
&= 1 - 4\delta,
\end{aligned}$$

or simply, $\mathrm{P}(\xi) \geq 1 - \sum_{l=1} 2^{lk}\delta_i \geq 1 - 4\delta$ as we have set $\delta_l = \delta 2^{-l(k+1)}$. We define $b_l = \lambda k^{1/2} 2^{-l)}$ for any $l \in \mathbb{N}^*$. Due to Defination (3.2) and Assumption (3.2), given any $x, y \in C$, where $C \in G_l$, it follows that:

$$|\eta(x) - \eta(y)| \leq \lambda |x-y|_2^{\alpha} \leq \lambda k^{\frac{1}{2}} 2^{-l} = b_l, \tag{3.23}$$

this is because the farthest distance between two points in the cell C is at most equal to the diameter of the cell, namely:

$$r_l \doteq \max_{x,y \in C} |x - y|_2 = k^{\frac{1}{2}} 2^{-l}.$$

Given event ξ, for all $l \in \mathbb{N}^*$, as we have defined $t_l = \log(1/\delta_l)$, substituting this value into the bound yields the following inequality at time t_l for every cell $C \in G_l$:

$$|\widehat{\eta}(x_C) - \eta(x_C)| \leq b_l.$$

□

Proof of Lemma 3.2. [LCK17] Using Equations 3.11 and 3.23, we derive:

$$4b_l < \widehat{\eta}_{\widehat{k}^*_C}(x_C) - \frac{1}{2} < \eta_{\widehat{k}^*_C}(x_C) + b_l - \frac{1}{2},$$

which indicates that $\eta_{\widehat{k}^*_C}(x_C) - \frac{1}{2} > 3b_l > 0$. Therefore, according to the definition of k^*_C, it follows that $\widehat{k}^*_C = k^*_C$.

Additionally, under the smoothness condition, for any $x \in C$, we have:

$$|\eta(x) - \eta(x_C)| \leq \lambda k^{1/2} |x - x_C| \leq b_l.$$

Assuming without loss of generality that $\widehat{k}^*_C = 1$, we obtain from the previous result that $\widehat{k}^*_C = k^*_C = 1$ and $\eta(x_C) - \frac{1}{2} > 2b_l$. Thus, for $x \in C$, we have $\eta_{k^*_C}(x) - \frac{1}{2} > 0$, confirming that k^*_C is indeed the optimal class within the cell C. The labeling $\hat{k}^*_C = k^*_C$ aligns with the Bayes classifier across the entire cell, ensuring no additional risk within C.

Summarizing, on ξ, we have:

$$\forall y \in \{0, 1\}, \forall C \in S^y, \forall x \in C, \quad 1\{\eta(x) \geq \frac{1}{2}\} = y.$$

Consequently, this implies that:

$$S^1 \subset \{x : \eta(x) - \frac{1}{2} > 0\} \quad \text{and} \quad S^0 \subset \{x : \eta(x) - \frac{1}{2} < 0\}.$$

□

Proof of Lemma 3.3. [LCK17] By Assumption (3.3), $\mathrm{P}_X(\Omega) \leq c_3 [\Delta - \Delta_0]_+^\beta \mathbf{1}\{\Delta \geq \Delta_0\}$ exists. Thus, we can infer from Assumption (3.1) that:

$$N_l(\Delta) \leq \frac{c_3}{c_1} [\Delta - \Delta_0]_+^\beta r_l^{-k} \mathbf{1}\{\Delta \geq \Delta_0\}. \tag{3.24}$$

On the other hand from Definition (3.2), Hölder Smoothness, where $\alpha \leq 1$ we know that:

$$[\Delta_l - \Delta_0]_+^\beta \leq \lambda^\beta r_{l+1}^\beta. \tag{3.25}$$

Plugging in (3.25) into (3.24) will result in:

$$\begin{aligned} N_{l+1}(\Delta_l) &\leq \frac{c_3}{c_1}[\Delta_l - \Delta_0]_+^{\beta} r_l^{-k} \mathbf{1}\{\Delta_l \geq \Delta_0\} \\ &\leq \frac{c_3}{c_1}\lambda^{\beta} r_{l+1}^{\beta-k} \mathbf{1}\{\Delta_l \geq \Delta_0\} \\ &\leq c_5 \lambda^{\beta} r_{l+1}^{-[k-\beta]} \mathbf{1}\{\Delta_l \geq \Delta_0\}, \end{aligned} \tag{3.26}$$

where N_{l+1} represents the count of active cells at the commencement of the depth $(l+1)$ round, and $[.]_+ = max(., 0)$ and $c_5 = max(\frac{c_3}{c_1}, 1)$. This formula holds true for $L - 1 \geq l \geq 0$.

□

Proof of Lemma 3.4. [LCK17] For each depth level $1 \leq l \leq L$, we sample these active cells a total of $t_l = \frac{\log(1/\delta_l)}{2b_l^2}$ times. Initially, let's examine the case where $\Delta_0 = 0$. To establish an upper bound on the total number of samples needed by the algorithm to attain depth L, we can refer to Equation (3.26), which states that on the event ξ:

$$\begin{aligned} \sum_{l=1}^{L} N_l t_l + N_L t_L &\leq 2\left(\sum_{i=1}^{L}\left(c_5 \lambda^{\beta} r^{-[k-\alpha\beta]_+}\right)\frac{\log(1/\delta_l)}{2\lambda^2 r_l^{2\alpha}}\right) \\ &\leq 2c_5\lambda^{\beta-2}\log(1/\delta_L)\sum_{l=1}^{L} r_l^{-(2\alpha+[k-\alpha\beta]_+)} \\ &\leq 2c_5 k^{-(2\alpha+k-\alpha\beta)/2}\lambda^{\beta-2}\log(1/\delta_L)\frac{2^{L(2\alpha+[k-\alpha\beta]_+)}}{2^{(2\alpha+[k-\alpha\beta]_+)}-1} \\ &\leq \frac{4c_5}{k^{(2\alpha+k-\alpha\beta)/2}}\lambda^{\beta-2}\log(1/\delta_L)\frac{2^{L(2\alpha+[k-\alpha\beta]_+)}}{2\alpha+[k-\alpha\beta]_+}. \end{aligned}$$

Since $2a - 1 \geq a/2$ for any positive real number, $a \in \mathbb{R}^+$, we can conclude that on the event ξ:

$$\sum_{l=1}^{L} N_l t_l + N_L t_L \leq 4c_5\lambda^{\beta-2}\log(1/\delta_L)\frac{2^{L(2\alpha+[k-\alpha\beta]_+)}}{2\alpha+[k-\alpha\beta]_+}. \tag{3.27}$$

To establish an upper bound for L, we'll employ a straightforward approach. Since t_L, the number of samples at depth L must be less than n (if there were only one active cell, which represents the minimum possible number of active cells, the allocated budget would be insufficient), we can deduce the following:

$$\frac{\log(1/\delta_L)}{2\lambda^2 r_L^{2\alpha}} \leq n,$$

this directly implies, given that $\delta_L < \delta < e^{-1}$, that:

$$L \leq \frac{1}{2\alpha}\log_2(2k\lambda^2 n).$$

We can now determine an upper bound for $\log(1/\delta_L)$:

$$\begin{aligned}\log(1/\delta_L) = \log(2^{L(k+1)}/\delta) &\leq \frac{k+1}{2\alpha}\log_2(2k\lambda^2 n) + \log(1/\delta) \\ &\leq \frac{k+1}{2\alpha}\log_2(\frac{2k\lambda^2 n}{\delta}).\end{aligned} \tag{3.28}$$

By combining Equations (3.28) and (3.27), we can infer that on the event ξ, the allocated budget is sufficient to achieve the target depth:

$$L \geq \left\lfloor \frac{1}{2\alpha + [k-\alpha\beta]_+} \log_2 \left(\frac{(2\alpha + [k-\alpha\beta]_+)2\alpha n}{c_7 \lambda^{\beta-2} \log_2\left(\frac{2k\lambda^2 n}{\delta}\right)} \right) \right\rfloor,$$

where $c_7 = 2c_5(k+1)$. Alternatively, the algorithm may terminate before reaching depth L, with $S^1 \cup S^0 = [0,1]^k$, leading to a zero extra risk.

□

Proof of Lemma 3.5. [LCK17] The lemma's assumption directly implies the first conclusion. The second conclusion is a direct corollary of the lemma's hypothesis and Assumption (3.3). In other words, we know that:

$$\mathrm{P}_X(\hat{f}_{n,a} \neq f^*) = \int_{x\in[0,1]^k} \mathbf{1}\{\hat{f}(x) \neq f^*(x)\}\, \mathrm{dP}_X(x). \tag{3.29}$$

However, instead of calculating this difficult integral, we can find an upper bound for it. This can happen through the Assumption (3.3):

For the third conclusion, we have the following integral that again we prefer to find an upper bound for it with the worst-case scenario:

$$\begin{aligned}\mathcal{E}(\hat{f}_{n,a}) &= \int_{x\in[0,1]^k : \hat{f}_{n,a} \neq f^*(x)} |1-2\eta(x)|\ \mathrm{dP}_X(x) \\ &\leq \mathrm{P}_X(\hat{f}_{n,a} \neq f^*) \sup_{x\in[0,1]^k} |\hat{f}_{n,a}(x) - f^*(x)|.\end{aligned} \tag{3.30}$$

The right-hand side term in 3.30, is yielded as follows. we know that for $x \in [0,1]^k$ whenever the $f_{n,a} \neq f^*(x)$, the value of $|1-2\eta(x)|$ is in $[0,1]$. On the other hand, we know that the $|1-2\eta(x)| \leq \sup_{x\in[0,1]^k} |\hat{f}_{n,a}(x) - f^*(x)|$. Thus, by multiplying $\sup_{x\in[0,1]^k} |\hat{f}_{n,a}(x) - f^*(x)|$ and $\mathrm{P}_X(\hat{f}_{n,a} \neq f^*)$, we achieve the upper bound for $\mathcal{E}(\hat{f}_{n,a})$. □

In the next chapter, we will conduct an empirical comparison of the performance of the realizable and noisy cases algorithm in detecting rare events.

Chapter 4

Implementation

In this section, we will empirically examine the performance of sequential learning for rare event detection in two scenarios: realizable and noisy cases. For this purpose, we will implement Algorithms 1 and 2.

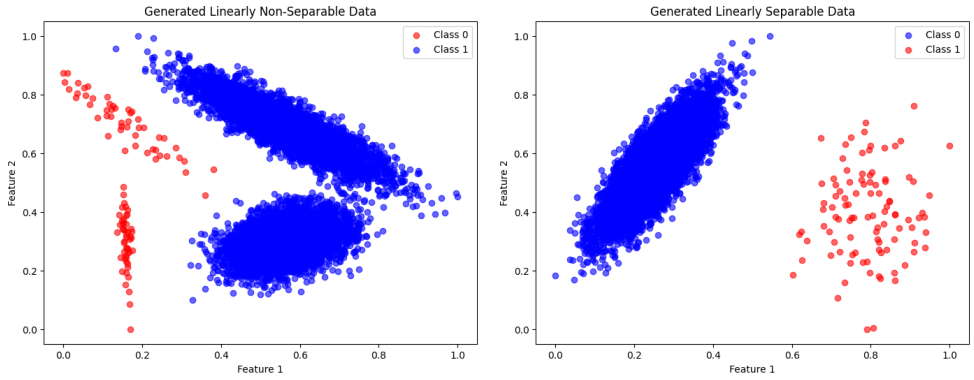
Realizable cases are expected to exhibit optimal performance under well-defined, idealized assumptions, free from external disturbances. In contrast, noisy cases are designed to function in comparatively disordered environments. Therefore, the realizable case will serve as our benchmark. Experiments will be conducted on synthetic data to evaluate the performance of both algorithms under controlled conditions.

4.1 Data Generation

To evaluate the proposed methods, we will first generate a synthetic dataset that is linearly separable, where rare events and dominant data can be perfectly classified using a linear classifier in the realizable case. The labeling procedure will be oracle-based and error-free. For simplicity and to ensure accurate labeling, a function acting as an oracle will automatically label the queried points without errors.

In the noisy case, noise will be introduced into the dataset to blur the boundaries between classes, simulating the complexity of real-world scenarios where perfect classification is unattainable. Additionally, the labeling procedure will include an arbitrary amount of error to mimic human-made labeling mistakes. Both datasets will consist of two classes in a two-dimensional feature space. One class will represent rare events, constituting a small percentage, $p \in [1, 5]$, of the total data.

The following figures illustrate these two scenarios. Figure 4.1b depicts the linearly separable dataset, where an optimal linear classifier can pass through the point [0, 0]. In contrast, Figure 4.1a shows a linearly non-separable dataset, where no linear classifier can effectively separate the two classes.



(a) (b)

Figure 4.1: Two examples of linearly separable 4.1b and linearly non-separable 4.1a data. The red points belong to the rare category.

Cross-Validation: To enhance the reliability of the results, we will perform k-fold cross-validation with $k = 5$ on each dataset. This technique involves dividing the data into k subsets (folds), using $k - 1$ folds for training and the remaining fold for testing in each iteration. This process is repeated k times, ensuring that each subset is used for testing exactly once.

Cross-validation enables the computation of average performance metrics (e.g., detection accuracy, false positive rate, and false negative rate) across different data splits, providing a robust evaluation of each algorithm's performance and generalization ability. Specifically, we will employ *Stratified k-fold cross-validation*, a variation of k-fold cross-validation that ensures each fold maintains approximately the same class proportions as the original dataset. This is particularly beneficial for imbalanced datasets, where one class may be significantly underrepresented.

Evaluation Metrics: The performance of each algorithm will be assessed using metrics such as detection accuracy, recall, and precision. These metrics collectively offer a comprehensive evaluation of the algorithms' ability to detect rare events with minimal error.

4.2 Realizable Case

We will begin by implementing Algorithm 1. As mentioned earlier, a linearly separable dataset consisting of two classes with a two-dimensional feature space as in Figure 4.1a, would be used. The algorithm's input are $\varepsilon = 0.05$, $\delta = 0.1$, and n. The labeling budget n, would be calculated according to Theorem 2.1. Namely:

$$n = \log_2(2/\delta) + 8ec'\theta(\varepsilon)\left(d\log_2(\theta(\varepsilon)) + 2\log_2\left(\frac{2\log_2(4/\varepsilon)}{\delta}\right)\right)\log_2(2/\varepsilon).$$

The disagreement coefficient $\theta(\varepsilon)$ and VC-dimension d are chosen according to respectively Theorem 2.5 and Theorem 2.4. That is $\theta(\varepsilon) = \frac{1}{\varepsilon^{\frac{1}{100}}}$ and $d = k + 1$. We could use any valid function that belongs to $o(\frac{1}{\varepsilon})$. However, due to the very fast convergence of the disagreement area, the algorithm stops significantly fast with a

small number of usage of query budget. Thus, in our case, there is no meaningful difference in getting the best function as a member of $o(\frac{1}{\varepsilon})$. In our setting given the above parameters and functions and with $c' = 1$, the query budget n would be almost 2068.

Initializing n, ε, and δ, the algorithm will create a set of linear classifiers (i.e. hypotheses class). For simplicity, the intercepts of all of these hypotheses are set to zero. Namely, they path through the point $[0, 0]$. To monitor the performance of the algorithm and have a correct result for evaluation, k-fold cross-validation (with $k = 5$) will be applied and the dataset will be split into train and test sets.

During each fold, *CALRealizable* asks the oracle (i.e. here is an automatic labeling function) to label the most critical point. This is a point that there is an inconsistency regarding its label among the current classifiers in version space at the current iteration. After labeling that point by oracle, the algorithm will remove all inconsistent classifiers and update the version space, and respectively the disagreement area will be shrunk. This procedure continues until no inconsistency exists for any processed point. Finally algorithm will return the last version space that contains the valid classifier or classifiers. This version space includes the optimal classifier(s). Figure 4.2 depicts a clear illustration of the iterative process where algorithm queries are made to the oracle, leading to adjustments in the version space for the initial fold of the dataset.

4.2.1 Model Performance

We evaluated the *CALRealizable* model using 5-fold cross-validation. As summarized in Table 4.1, the model achieved perfect accuracy (1.0) across all folds. This uniform performance, reflected by a standard deviation of (0.0), emphasizes the model's reliability and robustness within the realizable setting. These results highlight the model's capacity to effectively exploit the dataset's inherent structure, minimizing the disagreement region and attaining high accuracy with minimal labeled queries.

Notably, despite *Class 1* representing only 1% of the dataset, the model maintained flawless performance metrics, including precision, recall, and F1-score. This result illustrates the model's ability to manage highly imbalanced datasets effectively in realizable scenarios. The "support" denotes the number of true instances of a specific class within the test dataset. It represents the sample size used to compute the precision, recall, and F1-score for that particular class, providing insight into the reliability of these metrics.

In Figure 4.3, the performance and efficiency of the *CALRealizable* algorithm are analyzed across several dimensions. Image 4.3a visualizes the relationship between the number of requested labels and the number of unlabeled data points processed by the algorithm. It provides empirical evidence supporting Theorems (2.2) and (2.3), demonstrating the model's ability to efficiently request a minimal number of labels while processing relatively few data points.

From Image 4.3b, it is evident that the algorithm achieves a significant accuracy of over 99% by the fourth labeling request. Accuracy here is calculated as the proportion of correctly classified points, aggregated across all classifiers in the version space. This rapid increase in accuracy with minimal labeling underscores the algorithm's effectiveness in leveraging a small number of labels to achieve near-perfect classification performance.

Furthermore, Image 4.3c highlights the contraction of the disagreement region as

Model	**CALRealizable**			
Mean Accuracy	1.0			
Accuracy Std Dev	0.0			
Aggregated Report	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	1.0	1.0	1.0	1980.0
Class 1	1.0	0.99	0.99	20.0
Accuracy	1.0	1.0	1.0	1.0
Macro Avg	1.0	1.0	1.0	2000.0
Weighted Avg	1.0	1.0	1.0	2000.0

Table 4.1: Performance metrics for *CALRealizable*

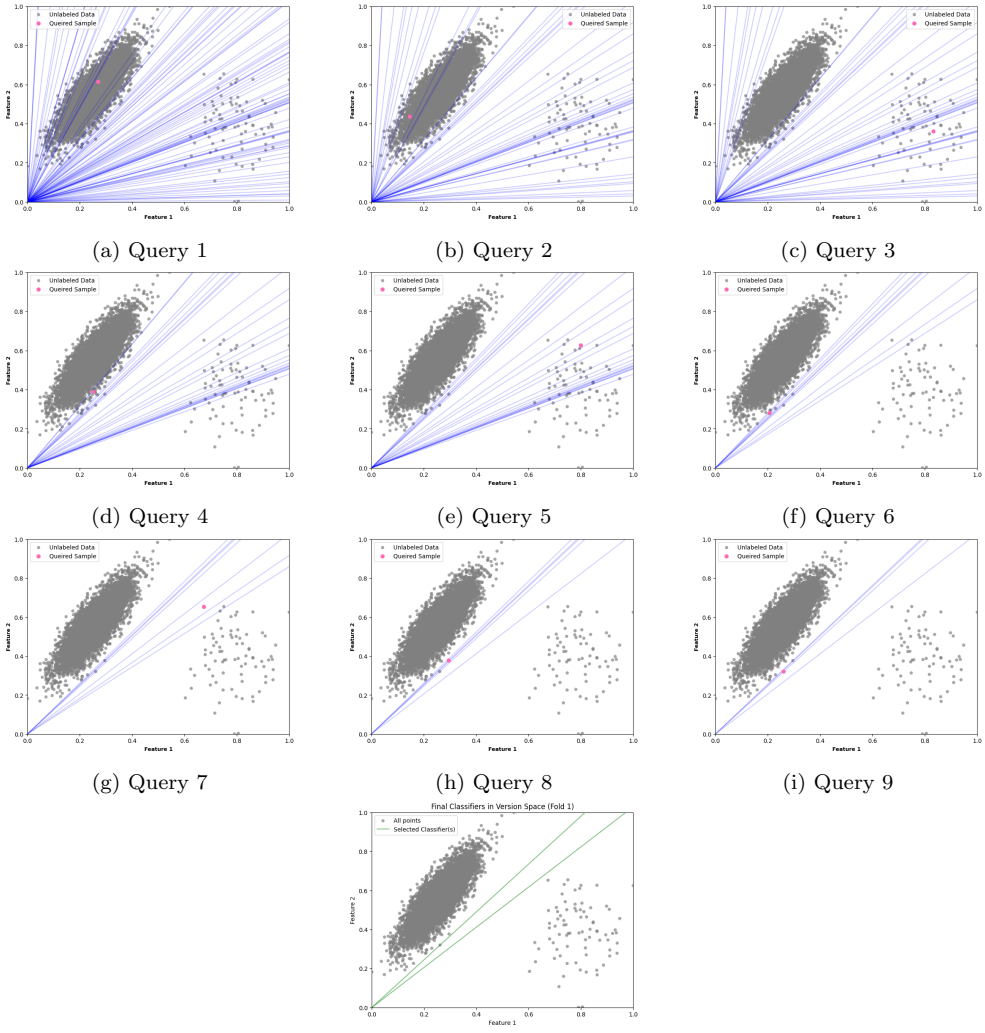
the number of label requests increases. By the fourth label request, the disagreement region is nearly eliminated, showcasing the algorithm's efficiency in reducing uncertainty within the feature space. This reduction aligns with theoretical expectations and illustrates the method's ability to narrow down the version space rapidly.

The figures collectively demonstrate that *CALRealizable* achieves high accuracy with minimal computational and labeling effort. By the time the fourth labeling request is made, fewer than 100 data points have been processed, emphasizing the algorithm's practical applicability and efficiency on this dataset.

A plausible explanation for achieving such high accuracy can be attributed to the class distribution within the dataset. Since most data points belong to a dominant class (i.e., *Class 0*), the probability of requesting a label from this class is significantly higher than the minority class (i.e., *Class 1*), which is both sparse and less populated. This behavior is evident in Figure 4.2, particularly in Image 4.2d. Starting from the sixth label request (see Image 4.2f), the remaining classifiers primarily focus on the decision boundary between classes, which is sparsely populated.

As the algorithm explores these less populated regions near the class boundaries, the probability of encountering points in these areas diminishes significantly. Consequently, a larger number of data points must be processed to identify informative queries for label requests. This explains the sharp increase in the number of processed samples required as the algorithm progresses. For example, in the third fold, over 7500 points were processed between the fifth and tenth label requests. Despite this extensive exploration, these additional samples resulted in less than a 1% improvement in both accuracy and the reduction of the disagreement area.

This observation underscores the impact of the relative central distance between classes and the spatial distribution of data points around their respective centers. In particular, the algorithm's efficiency decreases in regions where classes are more closely intertwined, and informative samples are harder to identify due to the sparsity of data in these areas. In other words, all results support the fact that the algorithm struggles when it comes to finding more informative points around the borders. This means if the aim is to find "rare points", despite its high accuracy, the algorithm will have a hard time detecting them.



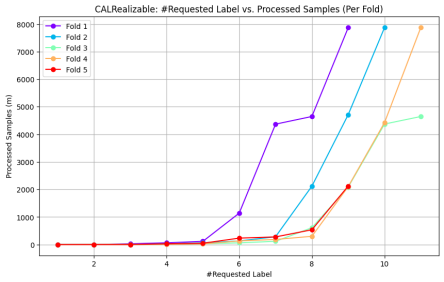
(a) Query 1 (b) Query 2 (c) Query 3

(d) Query 4 (e) Query 5 (f) Query 6

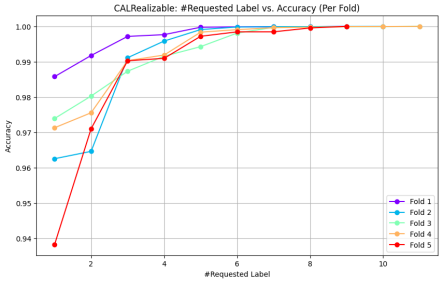
(g) Query 7 (h) Query 8 (i) Query 9

(j) Final Classifier(s)

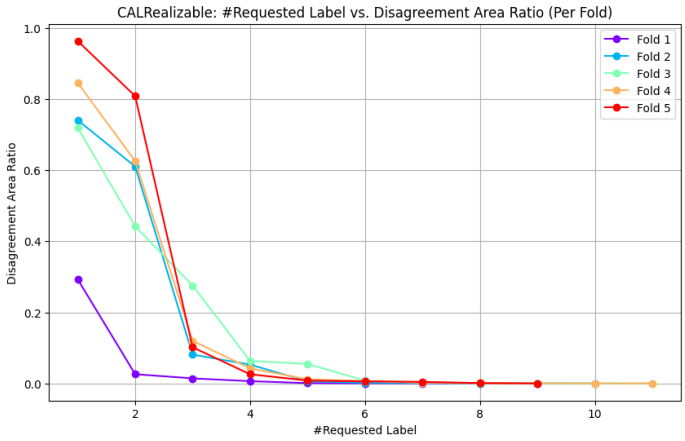
Figure 4.2: Labeling the queried point (in pink color) by oracle and shrinking both disagreement area and version space.



(a) #Processed points vs. #Requested Label



(b) Accuracy vs. #Requested Label



(c) Disagreement Shrink vs. #Requested Label

Figure 4.3: Accuracy, Disagreement Area Shrink Rate and Number of Processed Points Change Over Number of Requested Labels.

4.3 Noisy Case

In this section, we are going to investigate the performance of the Algorithm 2 on both separable and non-separable datasets as in Figure 4.1. Let's call the model trained by this algorithm '*LCKNoisy*'. First, we evaluate the robustness and adaptability of the *LCKNoisy* on separable data and then on non-separable one that contains noise. This noise will happen by oracle during labeling queried points.

The Algorithm 2 gets n, δ, α, and λ as inputs and returns S^0, S^1, and $\hat{f}_{n,\alpha}$. All of the S^0, S^1, and $\hat{f}_{n,\alpha}$ are sets of cells that are classified as either one or zero. We will train model *LCKNoisy* on different values of these parameters to compare their impact on the result. Similar to *CALRealizable* to monitor the performance of the algorithm and have a correct result for evaluation, k-fold cross-validation (with $k = 5$) will be applied and the dataset will be split into train and test sets. The confidence threshold $B_{l,\alpha}$ also has a profoundly drastic effect on classification. As in the logical condition of the inequality 4.1 in line six of Algorithm 2, its value rules whether a cell can be classified or will be passed as an active cell to the next level:

$$|\hat{\eta}(x_C) - 1/2| \leq B_{l,\alpha}. \tag{4.1}$$

As in 4.2, $B_{l,\alpha}$ is a function of $t_{l,\alpha}, \delta_{l,\alpha}$, and $b_{l,\alpha}$:

$$B_{l,\alpha} = 2\left[\sqrt{\frac{\log_2(1/\delta_{l,\alpha})}{2t_{l,\alpha}}} + b_{l,\alpha}\right]. \tag{4.2}$$

These functions also deep down depend on α, δ, and λ. Since δ is the confidence level of classification, we want it decently high. Thus, for a given feature space $\mathcal{X} \in [0,1]^k$, with k-dimension (here, $k = 2$), we fix it at $\delta = 0.05$, and for different depths of zooming in cells $l \in \{1, 2, 3\}$, will loop over $\alpha \in (0, 1]$ and $\lambda \in (0, 5]$ (i.e. as in smoothness class $\Sigma(\lambda, \alpha)$). Figure 4.4 demonstrates the behavior of the $B_{l,\alpha}$.

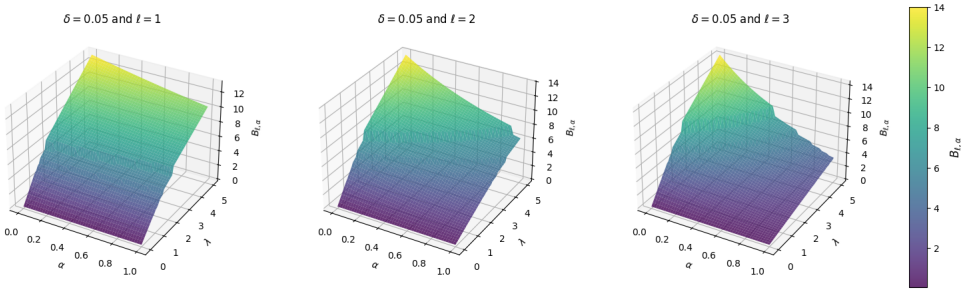


Figure 4.4: The behavior of $B_{l,\alpha}$ against α and λ for $\delta = 0.05$ and $\ell \in \{1, 2, 3\}$

As depicted in Figure 4.4, $B_{l,\alpha}$ increases as α tends to zero and λ grows. Practically, a larger $B_{l,\alpha}$ with sufficient sampling budget n leads to the classification of nearly all cells at the next level as active. This cascading effect may result in overfitting at the final level L, where all cells are classified as active, necessitating arbitrary decisions. Additionally, it is observed that for higher levels (l), the value of $B_{l,\alpha}$ diminishes as both α and λ tend towards one.

Now, if want to address this issue we need to choose smaller values for λ and push α close to one. This will a yield considerably big value of t, the initial requested number of samples in each level.

$$t = 2^k t_{l,\alpha},$$

where,

$$t_{l,\alpha} = \frac{\log_2(1/\delta_{l,\alpha})}{2b_{l,\alpha}^2}.$$

Figure 4.5 depicts the behavior of t across various values of α and λ. A notable observation is the inverse correlation between t and both α and λ. Specifically, as α and λ tend towards one to address underfitting issues associated with high $B_{l,\alpha}$ values, the required sampling number t increases dramatically. This poses a significant constraint, as the initial condition of the *While* loop, $t \leq n$, necessitates a substantially larger query budget n. This requirement directly contradicts the fundamental goal of sequential learning, which prioritizes efficient learning with a minimal query budget.

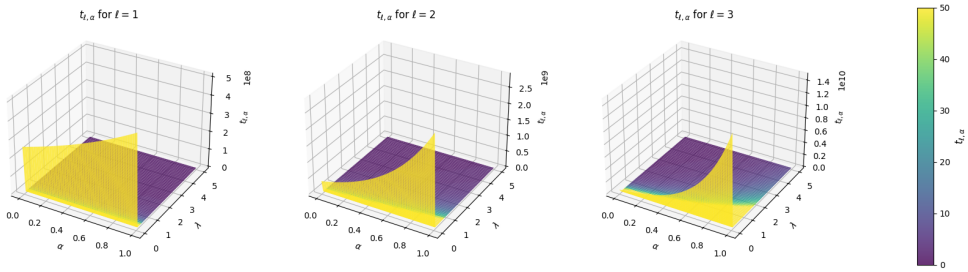


Figure 4.5: The behavior of t against α and λ for $\ell \in \{1, 2, 3\}$

4.3.1 Model Performance

In this section, we will evaluate the performance of the *LCKNoisy* model across various configurations of hyperparameters α, δ, λ and n, while maintaining a fixed confidence level $\delta = 0.05$. Due to computational constraints, the sample size is limited to 10,000. To ensure at least one iteration of the While loop (i.e., $l = 1$), we must select values for α, λ that results in a corresponding t value less than or equal to 10,000.

Table 4.2 presents the minimum required number of samples t for different combinations of α, λ to initiate the first iteration. Numbers in the header are different values of α and the first column shows different values of λ. For any combination of α and λ there is a number that is t, the requested number of samples at level $l = 1$. The query budget n must therefore be at least as large as t. While optimizing n is an important consideration, it is not the primary focus of this study. Instead, we will select combinations of α, δ, λ, and n that allow for a fair comparison with our benchmark model, *CALRealizable*.

$\lambda \setminus \alpha$	0.001	0.112	0.223	0.334	0.445	0.556	0.667	0.778	0.889	1
0.0001	1015738576	1096973856	1184706048	1279454764	1381781152	1492291256	1611639576	1740532968	1879734804	2030069528
0.0100	101,576	109,700	118,472	127,948	138,180	149,232	161,164	174,056	187,976	203,008
0.0500	4,064	4,388	4,740	5,120	5,528	5,972	6,448	6,964	7,520	8,124
0.1000	1,016	1,100	1,188	1,280	1,384	1,496	1,612	1,744	1,880	2,032
0.3000	116	124	132	144	156	168	180	196	212	228
0.5000	44	44	48	52	56	60	68	72	76	84
0.6000	32	32	36	36	40	44	48	52	56	60
1.1500	8	12	12	12	12	12	16	16	16	16
1.7000	4	4	8	8	8	8	8	8	8	8
2.2500	4	4	4	4	4	4	4	4	4	8
2.8000	4	4	4	4	4	4	4	4	4	4
3.3500	4	4	4	4	4	4	4	4	4	4
3.9000	4	4	4	4	4	4	4	4	4	4
4.4500	4	4	4	4	4	4	4	4	4	4
5.0000	4	4	4	4	4	4	4	4	4	4

Table 4.2: The requested sampling number t for $l = 1$ and the corresponding λ and α values.

Figure 4.6 illustrates the growth of the required sample size per cell, $t_{l,\alpha}$, across different levels. Notably, $t_{l,\alpha}$ exhibits significant sensitivity to λ, especially as λ approaches zero.

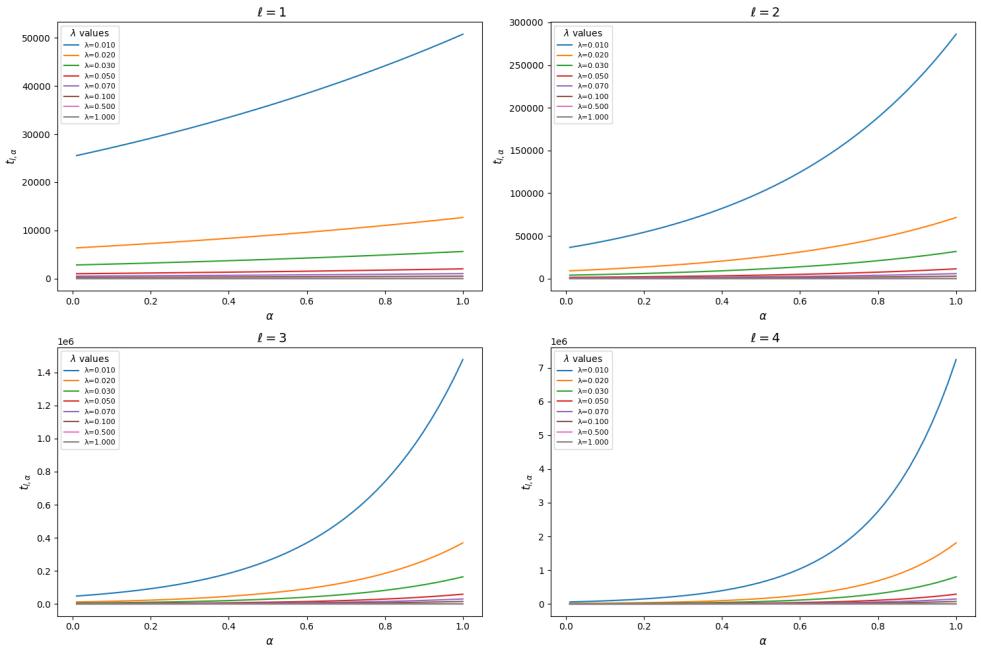


Figure 4.6: The behavior of number of requested samples per cells $t_{l,\alpha}$ against α and λ for $\ell \in \{1, 2, 3, 4\}$

Figure 4.7 illustrates the escalating per-cell sample requirement, $t_{l,\alpha}$, as we move up the hierarchical levels for fixed hyperparameters α and λ. This trend has significant implications, particularly for higher levels (lower l) where cells encompass larger regions of the feature space. In scenarios with imbalanced data distributions, fewer samples may be insufficient to accurately represent the underlying data distribution within these larger cells, potentially leading to biased classification outcomes.

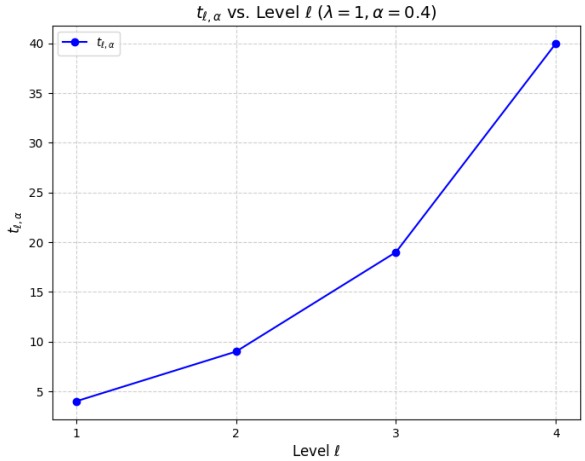


Figure 4.7: The increasing trend of per-cell sample requirement $t_{l,\alpha}$ against fixed $\alpha = 0.4$ and $\lambda = 1$ for $\ell \in \{1, 2, 3, 4\}$

4.3.2 Linearly Separable Case

We evaluate the performance of the *LCKNoisy* model on a linearly separable dataset, as shown in Figure 4.1b, under varying query budget constraints $n \in \{8, 20, 200, 2000\}$. The hyperparameters were fixed at $\alpha = 0.4$ and $\lambda = 1.7$, chosen to achieve a balance between the cell activation threshold $B_{l,\alpha}$ and the required sample size t, as illustrated in Figures 4.4 and 4.5. To simulate noise in the labeling process, a 5% error rate was introduced in the oracle's responses. Furthermore, the confidence threshold defined in Equation 4.1 was applied to evaluate its impact on classification decisions. The results of these experiments are summarized in Table 4.3.

The metrics shown in Table 4.3 are the mean of 5-fold cross-validation. The main reason for such a poor performance is the effect of $B_{l,\alpha}$ in letting cells be classified or not. In Figure 4.4 the scale of it has been visualized. In practice, it always is dramatically high and all cells for all levels are being passed to active cells set where at the end they will be classified randomly. Thus, the metrics in Table 4.3 are nothing but random performance.

Table 4.3: Model *LCKNoisy* performance metrics with dynamic $B_{l,\alpha}$ for different query budgets

Query Budget (n)	Mean Accuracy	Accuracy Std Dev	Precision		Recall		F1-Score	
			Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
8	0.63	0.27	0.99	0.04	0.63	0.57	0.73	0.07
20	0.44	0.30	0.82	0.01	0.44	0.22	0.55	0.02
200	0.54	0.23	0.99	0.05	0.54	0.60	0.68	0.07
2000	0.47	0.10	0.99	0.01	0.47	0.53	0.63	0.02

Subsequently, to address the problem of randomness we kept the settings for all other parameters unchanged and conducted an additional experiment, where the original threshold $B_{l,\alpha}$ in the algorithm was replaced with a constant value $c_B = \frac{1}{2}$. This value was chosen based on the observation that, in practice, $|\hat{\eta}(x_C) - 1/2| \leq \frac{1}{2}$,

providing an intuitive justification for the selection of c_B. Results are shown in Table 4.4. Our results indicate that the *LCKNoisy* model exhibits diminished performance for smaller query budgets ($n \in \{8, 20\}$) compared to larger ones. This is likely due to the model's increased difficulty in learning accurate representations with limited data. Additionally, the lower performance metrics for *Class 1* can be attributed to its relative underrepresentation in the dataset compared to *Class 0*.

Two important considerations are the model's robustness to noisy labels and its computational efficiency. The *LCKNoisy* model is robust to noisy oracle labels, achieving impressive performance despite these imperfections. Second, the model's computational efficiency is superior to *CALRealizable*, as it processes exactly the number of points specified by the query budget. Further optimizations can be realized through careful tuning of hyperparameters α and λ.

Figure 4.8 shows the partitioned dyadic grid based on the returned S^0 and S^1. Practically the depth of level l that model will go depends on query budget n. Cells in red or blue color have been classified based on the logical condition $|\hat{\eta}(x_C) - 1/2| \leq c_B$. The bigger the cell means, the lower l. Partitions with a ($\times$) sign in them are those cells that have been left as active cells and have been classified arbitrarily.

The assumption that the sampling center of a cell x_c is always practical is not universally valid. This assumption presumes that irrespective of the level, data is consistently available for sampling and that the sample size meets or exceeds the threshold $t_{l,\alpha}$. However, due to the natural configuration of data, certain levels can contain empty or sparsely populated cells. Addressing this issue is critical.

For empty cells, this work adopts an arbitrary classification approach. These cells are represented in Figure 4.8d as the empty blue cells without a ($\times$) marker. For cells where the number of points is less than $t_{l,\alpha}$, two strategies are conceivable:

1. Querying an oracle to label all available points within the cell and estimating $\hat{\eta}$ based on the observed points, regardless of their insufficient size.

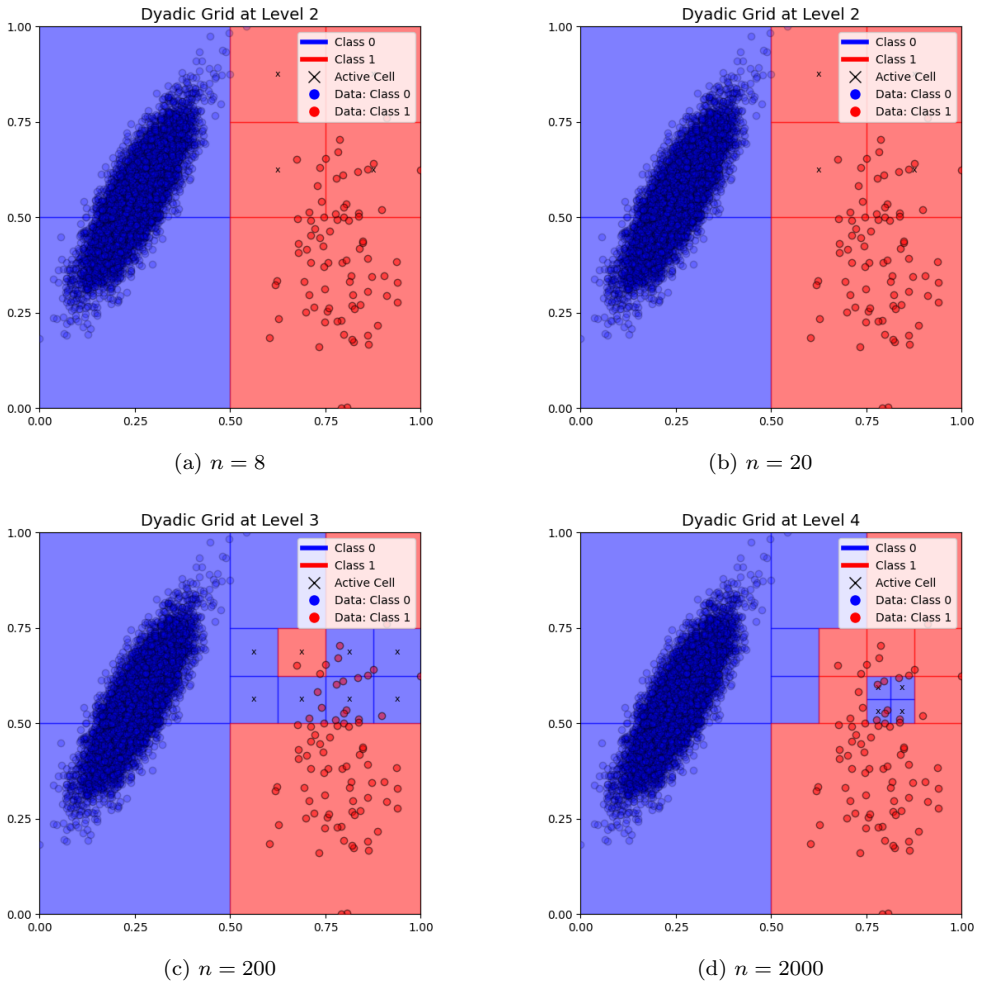
2. Combining the labeling of existing points with arbitrary labeling of additional points until the total size equals $t_{l,\alpha}$.

Each method has distinct advantages and disadvantages, which could form the basis of future research. In this work, we limit our focus to the first approach.

Table 4.4: Model *LCKNoisy* performance metrics with constant $c_B = 0.5$ for different query budgets

Query Budget (n)	Mean Accuracy	Accuracy Std Dev	Precision		Recall		F1-Score	
			Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
8	0.999	0.000	1.000	0.990	0.999	1.000	0.999	0.995
20	0.999	0.000	1.000	0.990	0.999	1.000	0.999	0.995
200	0.999	0.001	0.999	0.990	0.999	0.970	0.999	0.979
2000	0.999	0.000	0.999	0.990	0.999	0.990	0.999	0.990

Interestingly, we observe that the *LCKNoisy* model (evaluated under a controlled random state) demonstrates slightly superior performance for smaller query budgets, specifically $n = \{8, 20\}$, compared to larger query budgets $n = \{200, 2000\}$. However, the performance metrics for *Class 1* remain consistently lower across all query budget levels in comparison to *Class 0*. This behavior aligns with the response of the



(a) $n = 8$ (b) $n = 20$

(c) $n = 200$ (d) $n = 2000$

Figure 4.8: Classified dyadic grid of model *LCKNoisy* for different query budget n.

CALRealizable model in the context of rare event detection. Notably, this outcome aligns with the expectation that detecting rare points presents inherent challenges.

In the subsequent section, we will examine the performance of the *LCKNoisy* model on linearly non-separable data.

4.3.3 Linearly Non-separable Case

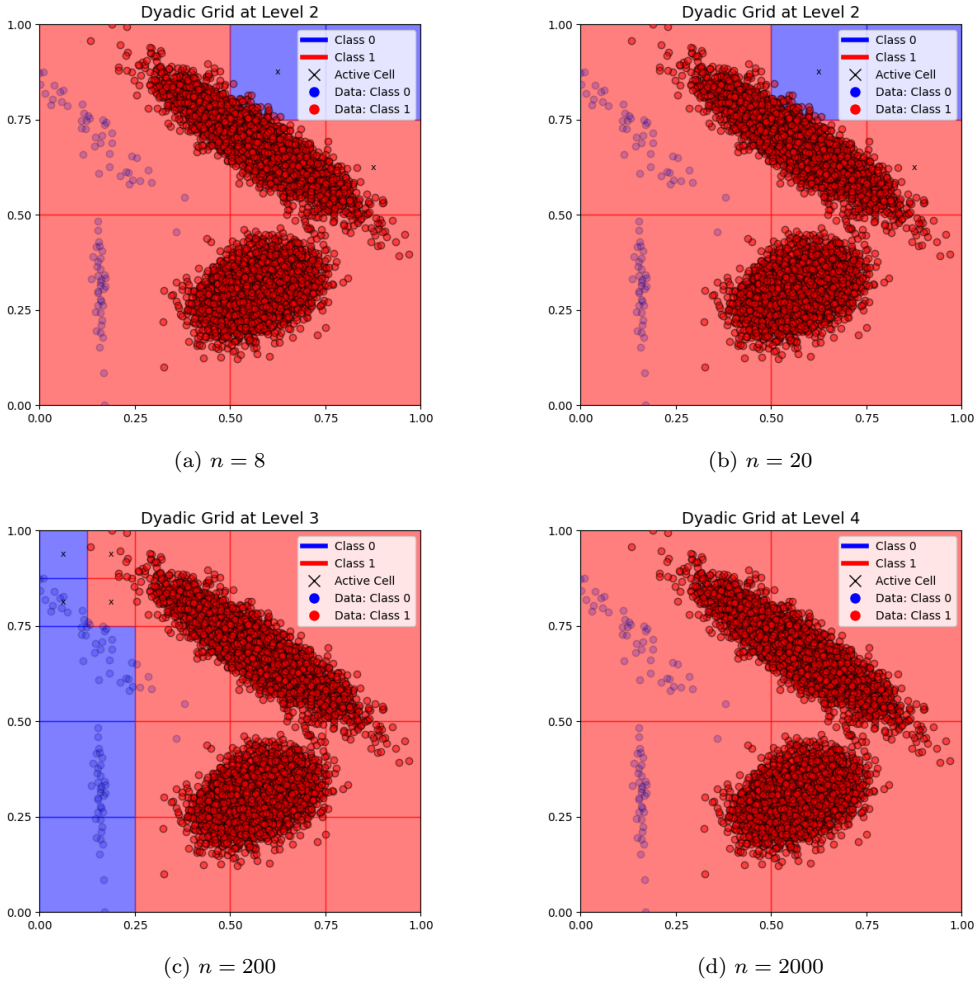
we have tried *LCKNoisy* model on linearly non-separable data as in Figure 4.1a. In this dataset *Class 0* is in minority resemblance of rare events. All of the hyperparameters particularly $c_B = 0.5$ (i.e. instead of $B_{l,\alpha}$) are kept as in the previous section. Table 4.5 and Figure 4.9 demonstrate the final results. As we can see the model is unable to detect *Class 0*. All metrics of this class are almost zero.

For instance, in Figure 4.9d all partitions have been classified as *Class 1*. However, there have been cases the model has classified points significantly well as in Figure 4.9c. One reason probably is due to the imbalance of classes. Since the model randomly

selects points in cells as a requested sample for labeling by the oracle, points in more frequent classes are more probable to be chosen. Consequently, this leads to a biased classification toward the more populated class.

Table 4.5: Model *LCKNoisy* performance metrics with constant $c_B = 0.5$ for different query budgets on linearly non-separable data

Query Budget (n)	Mean Accuracy	Accuracy Std Dev	Precision		Recall		F1-Score	
			Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
8	0.759	0.291	0.017	0.983	0.080	0.766	0.028	0.824
20	0.759	0.291	0.017	0.983	0.080	0.766	0.028	0.824
200	0.958	0.054	0.406	0.993	0.350	0.964	0.324	0.978
2000	0.955	0.071	0.200	0.990	0.080	0.963	0.114	0.975



(a) $n = 8$ (b) $n = 20$

(c) $n = 200$ (d) $n = 2000$

Figure 4.9: Classified dyadic grid of model *LCKNoisy* for different query budget n on non-separable data.

Potential modifications that could enhance the performance of the *LCKNoisy* model involve adjustments to the initial level of partitioning, l, and the size of the requested sample set, $t_{l,\alpha}$. The current implementation begins the dyadic grid partitioning at $l = 1$. For datasets characterized by class imbalance and intertwined distributions, initializing the dyadic grid at $l = 2$ may improve performance by effectively "zooming in," enabling more precise sampling from critical regions of the feature space.

Moreover, the size of the requested sample set for labeling should be dynamically adjusted based on the distribution of data points across partitions. Specifically, the sample size should have a direct relationship with the density of points within a partition and an inverse relationship with the level l. Higher levels (lower l), which encompass a larger number of points, are more susceptible to introducing classification bias. Consequently, it is advisable to allocate a larger portion of the sampling budget to these early levels to mitigate this bias and improve overall classification performance.

Chapter 5

Conclusions

This thesis contributes to the field of rare event detection by analyzing sequential learning frameworks tailored for scenarios with sparse and imbalanced data. The distinction between realizable and noisy cases provides valuable insights into how different data conditions influence algorithm performance. In the realizable case, the framework demonstrates efficient label utilization and rapid convergence to optimal classifiers, benefiting from the absence of noise. Conversely, the noisy case highlights the importance of adaptive techniques that address labeling errors and model imperfections.

The use of dyadic grids and plug-in classifiers proved to be effective in managing uncertainty and focusing resources on high-value data points, particularly near decision boundaries. Theoretical analyses, supported by experimental results, confirm that the proposed method achieves significant improvements in label efficiency and detection accuracy over realizable cases, exemplified by approaches like CAL. However, for noisy scenarios, challenges such as class imbalance and intertwined distributions were evident, suggesting potential areas for refinement.

To improve the performance of the *LCKNoisy* model, adjustments to the initial partitioning level l and the requested sample size $t_{l,\alpha}$ are recommended. Instead of starting the dyadic grid at $l = 1$, initializing at $l = 2$ for datasets with imbalanced and intertwined classes may enhance sampling precision by focusing on smaller, more critical regions of the feature space. Moreover, the requested sample size should be adapted dynamically based on the distribution of data points across partitions. Allocating more samples to higher levels (lower l)—where data density is greater—can help mitigate classification biases and improve the overall performance of the model.

Future work can extend this framework to handle dynamic and non-stationary environments, incorporating advanced techniques for noise-robust classification and real-time adaptation. The findings presented herein lay a strong foundation for developing scalable, cost-efficient solutions for detecting rare yet critical events in complex systems.

Bibliography

[BBL06] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 65–72, 2006.

[BBL09] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89, 2009.

[CAL94] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15:201–221, 1994.

[H+14] Steve Hanneke et al. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309, 2014.

[Han12] Steve Hanneke. Activized learning: Transforming passive to active with improved label complexity. *The Journal of Machine Learning Research*, 13(1):1469–1587, 2012.

[Hin20] Hideitsu Hino. Active learning: Problem settings and recent developments. *arXiv preprint arXiv:2012.04225*, 2020.

[Kää06] Matti Kääriäinen. Active learning in the non-realizable case. In *Algorithmic Learning Theory: 17th International Conference, ALT 2006, Barcelona, Spain, October 7-10, 2006. Proceedings 17*, pages 63–77. Springer, 2006.

[KSS92] Michael J Kearns, Robert E Schapire, and Linda M Sellie. Toward efficient agnostic learning. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 341–352, 1992.

[LCK17] Andrea Locatelli, Alexandra Carpentier, and Samory Kpotufe. Adaptivity to noise parameters in nonparametric active learning. In *Proceedings of the 2017 Conference on Learning Theory, PMLR*, 2017.

[Min12] Stanislav Minsker. Plug-in approach to active learning. *Journal of Machine Learning Research*, 13(1), 2012.

[Pre] Jennifer Prendki. Active learning: Why smart labeling is the future of data annotation. `https://shorturl.at/PG4ql`. Presentation slides.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[VC15] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.