Minimizing Queries for Active Labeling with Sequential Analysis

Jack Paparian

CMU-CS-16-130
October 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christopher James Langmead, Chair
Carl Kingsford, Faculty

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Keywords: Cluster-based Active Learning, Sequential Probability Ratio Test, Strong and Weak Oracles, Binary Space-partitioning Trees

For my family

Abstract

When building datasets for supervised machine learning problems, data is often labelled manually by human annotators. In domains like medical imaging, acquiring labels can be prohibitively expensive. Both active learning and crowdsourcing have emerged as ways to frugally label datasets. In active learning, there has been recent interest in algorithms that exploit the data's structure to direct querying. When learning from crowds, one must balance the accuracy and cost of different oracles when gathering labels; weak oracles are assumed to be most accurate when labeling samples from label-homogeneous regions of space.

In this thesis, we explore how the data's structure can be leveraged for both of these techniques. The sequential probability ratio test (SPRT) provides the backbone for our contributions. Using the SPRT, we provide a cluster-based active learning algorithm to find a small, homogeneous partitioning of the data. We also use the SPRT to measure the confidence of a weak oracle's label by analyzing its estimates on neighboring labels. The optimality of the SPRT allows the algorithms to inherently minimize the average number of queries required before their termination.

Acknowledgments

Firstly, this thesis would not have been possible without the guidance of my advisor, Christopher James Langmead. I had the privilege of working with him for the past two and a half years on a variety of projects. In all of those ventures, he encouraged my curiosity and helped shape my nebulous ideas into tangible approaches. I'm grateful for all the great discussions we had in front of the whiteboard in his office. I would also like to thank Carl Kingsford for serving on my thesis committee. Special thanks goes to Rachel Jue for helping me with the diagrams in this document. Finally, I would like to thank all my friends at Carnegie Mellon for everything that they have taught me over the past five years. It's been a blast!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

With the vast amount of content on the Internet, it is now easier than ever to compile datasets for various machine learning tasks. Sites hosting images, videos, and documents can be crawled, and data can be stored inexpensively due to great technological developments in storage systems. The increase in readily available data makes it easier to train sophisticated machine learning models. However, most machine learning learning tasks are *supervised*, meaning that each sample requires a label to distinguish it from other samples. Ultimately, the challenge in building useful datasets for supervised machine learning tasks becomes assigning labels to a number of samples sufficient to train the desired model.

In most cases, labels are acquired by human annotators. This makes labeling every unlabeled sample an enormous use of time. One strategy to avoid labeling all unlabeled samples is *active learning*. Active learning is an interactive framework for iteratively training a machine learning algorithm in which the algorithm decides whether a sample will be useful enough to label and incorporate into the training set [13]. A budget is set to place a limit on the amount of samples that the algorithm can afford to label. This encourages the algorithm to label only the samples that will be most useful for training the data.

A burgeoning area of active learning research is *cluster-based active learning*, which leverages the spatial positions of queried samples to infer the labels of unlabeled samples. These algorithms try to divide the data into label homogeneous regions in order to propagate the majority label in each partition to the unlabeled points. The latest approaches in these algorithms have been the use of cluster trees or binary space-partitioning trees to represent partitions of the data and descend down the tree to divide each region of space into smaller sections.

Another area of active learning research is the generalization of the oracle. Though active learning algorithms are traditionally defined as having access to a single, infallible oracle, this assumption is often unrealistic in real life [6]. An extension to active learning is to consider the existence of multiple oracles that the algorithm can query. It is realistic to assume that the oracles have various levels of experience with working with a domain of data. This means that some oracles will label samples more accurately than others, and it is common for the oracle with more experience to cost more for each label than the less experienced oracle. For example, consider the task of labeling medical images with their corresponding diagnosis. A medical expert is more likely to accurately label these images than medical students, but a medical expert is likely to be much more expensive and less available than medical students. In these situations, algorithms should try to leverage all oracles, only querying the strongest oracles in situations where the weak oracle is not confident.

1.2 Contributions

In this work, we describe two novel algorithms for cluster-based active learning and active learning with strong and weak oracles, named SeqPLAL and SeqUBS respectively. Both algorithms are based on sequential hypothesis testing, a framework for testing a null hypothesis against an alternative hypothesis after analyzing each sample in a stream [21]. At each step, the algorithm decides whether to reject the null hypothesis, accept it, or look at another sample to gain more information. It is commonly used on binary labeled data to determine whether a population has more or less than a certain threshold of samples with the same label.

In Chapter 2 we describe the assumptions we make about structure of data in cluster-based active learning, explain the assumptions we make about the strong and weak oracles we use, and define the sequential probability ratio test, the backbone of the algorithms in this thesis. Chapter 3 defines the the SeqPLAL algorithm for augmenting the PLAL cluster-based active learning algorithm [18] with the sequential probability ratio test to minimize the number of queries. In Chapter 4, we define the SeqUBS algorithm to minimize the number of queries to the weak oracle and bound the calls to the strong oracle when estimating the confidence in the correctness of labels assigned by the weak oracle. Finally, in Chapter 5 we summarize our results and provide future directions for continuing this work.

1.3 Notation

Table 1.1 defines the notation that we use in this thesis.

Symbol	Definition
d	sample dimension
$\mathcal{X}$	set of samples from domain $[0, 1]^d$
P	distribution over $\mathcal{X} \times \{0, 1\}$
$P_{\mathcal{X}}$	marginal distribution of P over $\mathcal{X}$
l	labeling function with domain $\mathcal{X}$ and target $\{0, 1\}$

Table 1.1: Notation used in this thesis

Chapter 2

Background

In this section we explain the fundamentals of the algorithms presented in the following chapters. In Section 2.1 we discuss some previous work on active learning with multiple, imperfect oracles. Then in Section 2.2 we explain background work on cluster-based active learning. Next we explain the notion of Probabilistic Lipschitzness of data in Section 2.3. Finally we discuss in detail the sequential probability ratio test in Section 2.4.

2.1 Active Learning with Multiple Oracles

In its simplest form, active learning consists of a single, perfectly accurate oracle that labels samples when queried. In order to make this more realistic, recent work has generalized the active learning framework to interact with a set of oracles who may have different levels of accuracy or different costs. With Internet tools such as Amazon's Mechanical Turk[1], it has become possible for multiple oracles to provide labels to points, though these labels may be less accurate than those a domain expert would provide. The extensions to active learning's labeling model fall into two main camps: proactive learning and crowdsourcing. We give a brief overview of their ideas in the following sections.

2.1.1 Proactive Learning

Proactive learning describes pool-based active learning with two oracles that have different distinguishing properties [6]. Generally, proactive learning describes situations where one of the oracles is more available or accurate than the other; however, this oracle tends to be more expensive. When labeling points, a decision must be made about which oracle to query to minimize labeling cost without acquiring too many incorrect labels.

At each labeling step, proactive learning chooses a tuple of a point and oracle, denoted (x^*, k^*), by maximizing a utility function $U(x, k)$, which defines the tradeoff between labeling value and cost. Formally, this is defined as

[1] `https://www.mturk.com/`

$$(x^*, k^*) = \arg\max_{x \in X, k \in K} U(x, k) \tag{2.1}$$

A common choice of utility function is

$$U(x, k) = \frac{P(ans \mid x, k) * V(x)}{C_k} \tag{2.2}$$

where $P(ans \mid x, k)$ is the probability that oracle k will correctly label sample x, $V(x)$ is the "value" of labeling point x, and C_k is the cost of using oracle k to label a point. The cost of using an oracle is typically known upfront, and the value of labeling a sample x is usually estimated with a heuristic such as uncertainty or information density [14]. The challenge then becomes estimating $P(ans \mid x, k)$. One approach is to cluster the data as an initial step and test whether the weak oracle is available or accurate at labeling the cluster exemplars. Then, availability or accuracy can be propagated to the other points in the clusters by decreasing confidence as a function of distance from the centroid [6]. Another approach is to select a small batch of points to have both oracles label and store $P(ans \mid x, k)$ as a count of correct predictions on the batch [10].

2.1.2 Active Learning from Crowdsourcing

With the help of services like Amazon's Mechanical Turk, it is now possible to hire annotators to label samples in datasets. However, some annotators may be more or less accurate than others, and there is no *a priori* knowledge about which annotators will be less accurate. The challenge then becomes identifying the most accurate annotators from the labels.

Some approaches assume that annotators can be assumed to have a fixed rate of error on any samples they are asked to label. Under these assumptions, researchers have developed algorithms for identifying annotators that are too inaccurate to continue querying [7, 11]. Other approaches have been to learn the parameters than govern an annotator's probability of labeling a given sample correctly [23, 24].

2.2 Related Work on Cluster-based Active Learning

While the majority of active learning research has been focused on searching for candidate classifiers in a hypothesis space, an emerging area of research aims to directly use the structure of the data to make inferences about the labels or unlabeled points. These methods operate under the assumption that samples close together in space are likely to have the same label. This paradigm is known as *cluster-based active learning* [2, 4].

One of the first efforts to use the structure of data was made by Zhu, Lafferty, and Ghahramani [26]. They imposed a neighborhood graph on the data and propagate labels from labeled points to nearby points. At each step of the active learning model, they find the unlabeled points which when labeled would maximally reduce the estimated risk of the Bayes classifier. However, this method does not correct for the sampling bias that will occur by selecting points only based on estimated risk.

Combining the ideas of label propagation and avoiding sampling bias, Dasgupta and Hsu developed an hierarchical sampling algorithm that finds the homogeneous clusters while bounding label error [5]. Their algorithm, known as the DH algorithm, takes a cluster tree of unlabeled points as input and maintains a set of active clusters called a pruning from which to sample points. At each step of the algorithm, a cluster is chosen from the pruning either randomly or actively based on a heuristic, an unlabeled point is labeled, and a decision is made to split the cluster if it contains a sufficient number of different labels. Since the cluster tree is fixed before any labels are assigned, labels from parent clusters can be used in analysis of child clusters without introducing sampling bias [2]. Since the bound of the number of queries depends on the depth of the cluster tree, the performance of the algorithm will be dependent on the size of the cluster tree's pruning with label homogeneous clusters. We will work with a variation of this idea developed by Urner et al. [18] that bounds the labeling error without having the depth of the tree as a factor.

2.3 Probabilistic Lipschitzness

2.3.1 Definition

When designing hierarchical active learning algorithms, one assumes that a hierarchical partitioning of the data will split the data into near-homogeneous regions. The DH algorithm relies on the assumption that there exists an agglomerative clustering of the data that creates a homogeneous pruning of the data. Instead of conditioning on the hierarchical clustering provided to the algorithm, we will be making assumptions about the data itself.

This clustering assumption we make is called Probabilistic Lipschitzness, and it is a relaxation of the standard Lipschitz condition of labeling function. Probabilistic Lipschitzness bounds the probability that any two points with opposite labels have less than λ distance separating them. This formalizes a cluster assumption that points which are closer together are likely to have the same label by bounding the amount of points that violate the Lipschitz condition. A visualization of this assumption is shown in Figure 2.1.

We formally define Probabilistic Lipschitz as follows:

Definition Probabilistic Lipschitzness Let $\phi\colon \mathbb{R} \to [0,1]$. We call $f\colon \mathcal{X} \to \mathbb{R}$ a *ϕ-Probabilistic Lipschitz* function with respect to a $P_{\mathcal{X}}$ over $\mathcal{X}$ if $\forall\ \lambda > 0$:

$$\Pr_{x\sim P_{\mathcal{X}}}\left[\Pr_{y\sim P_{\mathcal{X}}}\Big[|f(x)-f(y)| > (1/\lambda)\|x-y\|\Big] > 0\right] \le \phi(\lambda) \tag{2.3}$$

Probabilistic Lipschitzness [16] allows us to bound the heterogeneity of a given region of space. For a data distribution $P_{\mathcal{X}}$ with labeling function l that is ϕ-Lipschitz, the weight of points that have opposite label in a λ-ball around any point x is bounded by $\phi(\lambda)$, where ϕ is a non-decreasing function. Examples of ϕ analyzed in [16] are $\phi(\lambda) = \lambda^n$ and $\phi(\lambda) = e^{-1/\lambda}$. We later use $\phi(\lambda)$ to bound the data-diameter of our dataset.

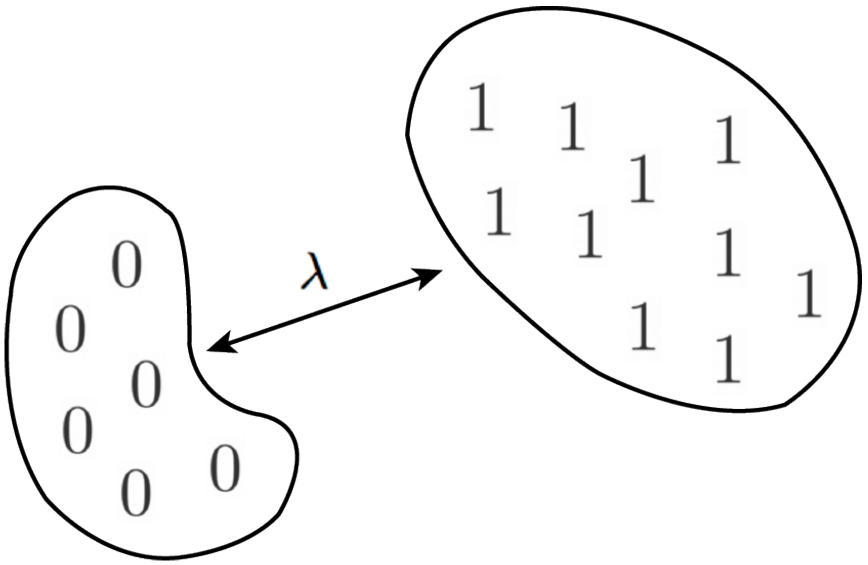


Figure 2.1: Two clusters satisfying the Lipschitz condition with Lipschitz constant $1/\lambda$. Probabilistic Lipschitzness bounds the amount of points that violate this condition.

2.4 Sequential Probability Ratio Test

2.4.1 Definition

The sequential probability ratio test (SPRT) is one of the most basic and fundamental tools in sequential analysis. It allows for the testing of hypotheses by analyzing one sample at a time, instead of observing a pre-defined number of samples at once [21].

In its most basic form, the sequential probability ratio tests two hypotheses: a null hypothesis H_0 and an alternative hypothesis H_1. In our example we will test the mean of the binomial distribution representing the distribution over the labels of the points where H_0 denotes $p = p_0$ and H_1 denotes $p = p_1$.

At the each stage of the test we observe the label of a new sample. For each step m of the test we calculate:

$$\frac{p_{1_m}}{p_{0_m}} = \prod_{i=1}^{m} \frac{\Pr(X_i = x_i \mid p = p_1)}{\Pr(X_i = x_i \mid p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}} \tag{2.4}$$

where d_m is the number of 1s seen. Analyzing the ratio of the two likelihood functions allows us to compute which value of p is more likely given the observations. We choose two positive constants A and B, where $B < A$, as boundaries for stopping the test. Hypothesis H_0 is accepted if

$$\frac{p_{1_m}}{p_{0_m}} \leq B \tag{2.5}$$

and hypothesis H_1 is accepted if

$$\frac{p_{1_m}}{p_{0_m}} \geq A \,. \tag{2.6}$$

In practice, we compute $\log\left(\frac{p_{1_m}}{p_{0_m}}\right)$ and test if it is $\leq \log(B)$ or $\geq \log(A)$ after each observed sample.

We choose the boundaries A and B as functions of the false positive rate α and the false negative rate β. The tuple (α, β) is called the *strength* of the test. In order to construct a test with strength (α, β), we set

$$A = \frac{1-\beta}{\alpha} \tag{2.7}$$

and

$$B = \frac{\beta}{1-\alpha} \,. \tag{2.8}$$

Let us interpret the sequential probability ratio test geometrically. Let f_m represent the log-likelihood function after m samples. The test accepts H_0 if $f_m \leq \log(\frac{\beta}{1-\alpha})$ and accepts H_1 if $f_m \geq \log(\frac{1-\beta}{\alpha})$. We can draw the regions of acceptance of H_0 and H_1 by expressing the termination criteria as $d_m \geq h_0 + ms$ and $d_m \leq h_1 + ms$ respectively, where h_0, h_1, and s are defined as:

$$h_0 = \frac{\log \frac{\beta}{1-\alpha}}{\log \frac{p_1(1-p_0)}{p_0(1-p_1)}} \qquad h_1 = \frac{\log \frac{1-\beta}{\alpha}}{\log \frac{p_1(1-p_0)}{p_0(1-p_1)}} \qquad s = \frac{\log \frac{1-p_0}{1-p_1}}{\log \frac{p_1(1-p_0)}{p_0(1-p_1)}} \tag{2.9}$$

The line $d_m = h_0 + ms$ is the H_0 acceptance line, and $d_m = h_1 + ms$ is the H_1 acceptance line. The different acceptance regions are shown in Figure 2.2, where the black curve is an example trajectory where enough 1s have been seen to accept H_1.

Despite its simplicity, the sequential probability ratio test is optimal in the sense that it requires the average fewest observations of all tests with the same power [22]. This property allows the sequential probability ratio test to be the foundation of our results in the following chapters.

2.4.2 Average Sample Number

In the context of sequential analysis, the SPRT's expected sample size $E_p(n)$ is called the *average sample number*. An approximate formula for the ASN in the binomial case is:

$$\widetilde{E}_p(n) = \frac{L(p)\log\frac{\beta}{1-\alpha} + (1-L(p))\log\frac{1-\beta}{\alpha}}{p\log\frac{p_1}{p_0} + (1-p)\log\frac{1-p_1}{1-p_0}} \tag{2.10}$$

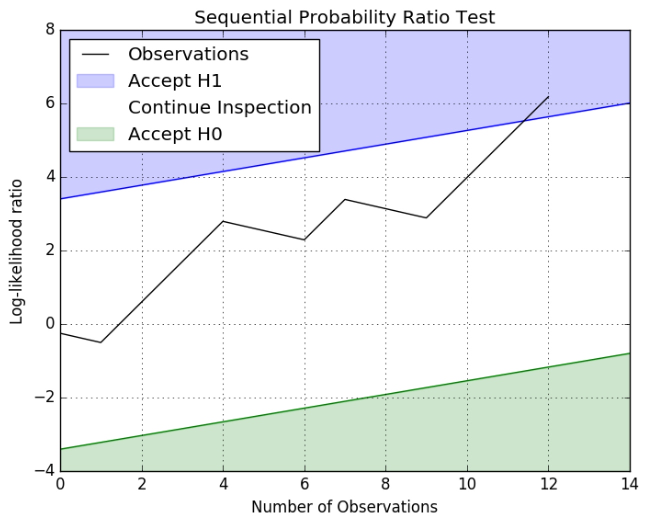


Figure 2.2: A sequential probability ratio test example where $p_0 = 0.1, p_1 = 0.3, \alpha = 0.01$, and $\beta = 0.01$

where $L(p)$ is the operating characteristic (OC) curve (the probability that the null hypothesis will be accepted). $L(p)$ is approximated as follows:

$$\widetilde{L}(p) = \frac{\left(\dfrac{1-\beta}{\alpha}\right)^h - 1}{\left(\dfrac{1-\beta}{\alpha}\right)^h - \left(\dfrac{\beta}{1-\alpha}\right)^h} \tag{2.11}$$

where h is determined by:

$$p = \frac{1 - \left(\dfrac{1-p_1}{1-p_0}\right)^h}{\left(\dfrac{p_1}{p_0}\right)^h - \left(\dfrac{1-p_1}{1-p_0}\right)^h} \quad . \tag{2.12}$$

This formulation of $\widetilde{L}(p)$ makes the computation of $\widetilde{E}_p(n)$ nontrivial for arbitrary values of p. However, we can approximate $\widetilde{E}_p(n)$ by computing its value at five specific values of p and fitting a function to those points. Those five points are listed in Table 2.1.

2.4.3 Lower bounding the probability that the SPRT will terminate in $\leq k$ steps

Though the sequential probability ratio test is guaranteed to eventually terminate, there is no guarantee it will terminate after k steps for some $k \in \mathbb{N}$. However, we can easily bound the

p	$\widetilde{L}(p)$	$\widetilde{E}_p(n)$
0	1	$\dfrac{\log\dfrac{\beta}{1-\alpha}}{\log\dfrac{1-p_1}{1-p_0}}$
p_0	$1-\alpha$	$\dfrac{(1-\alpha)\log\dfrac{\beta}{1-\alpha}+\alpha\log\dfrac{1-\beta}{\alpha}}{p_0\log\dfrac{p_1}{p_0}+(1-p_0)\log\dfrac{1-p_1}{1-p_0}}$
s	$\dfrac{\log\dfrac{1-\beta}{\alpha}}{\log\dfrac{1-\beta}{\alpha}-\log\dfrac{\beta}{1-\alpha}}$	$\dfrac{-\log\dfrac{\beta}{1-\alpha}\log\dfrac{1-\beta}{\alpha}}{\log\dfrac{p_1}{p_0}\log\dfrac{1-p_0}{1-p_1}}$
p_1	β	$\dfrac{\beta\log\dfrac{\beta}{1-\alpha}+(1-\beta)\log\dfrac{1-\beta}{\alpha}}{p_1\log\dfrac{p_1}{p_0}+(1-p_1)\log\dfrac{1-p_1}{1-p_0}}$
1	0	$\dfrac{\log\dfrac{1-\beta}{\alpha}}{\log\dfrac{p_1}{p_0}}$

Table 2.1: Five values for $\widetilde{E}_n(n)$ used to approximate the ASN curve

probability that an SPRT will terminate in $\leq k$ steps. We walk through the proof provided in [21].

Let n be the random variable for the number of samples required by the sequential ratio probability test. For any positive integer k, let $P_0(n \leq k)$ be the probability that $n \leq k$ when H_0 is true, and let let $P_1(n \leq k)$ be the probability that $n \leq k$ when H_1 is true.

Let us define $z = \log \dfrac{f(x;\theta_1)}{f(x;\theta_0)}$, and let $z_i = \log \dfrac{f(x_i;\theta_1)}{f(x_i;\theta_0)}$ for the ith sample x_i.
If $\displaystyle\sum_{i=1}^{k} z_i \geq \log A$, then $n \leq k$. Likewise, If $\displaystyle\sum_{i=1}^{k} z_i \leq \log B$, then $n \leq k$. It follows that

$$P_1(\sum_{i=1}^{k} z_i \geq A) \leq P_1(n \leq k) \tag{2.13}$$

and

$$P_0(\sum_{i=1}^{k} z_i \leq B) \leq P_0(n \leq k) \tag{2.14}$$

With a simple application of the Central Limit Theorem, we find that

$$P_1(\sum_{i=1}^{k} z_i \geq A) = 1 - \Phi\left(\frac{\log A - kE_1(z)}{\sqrt{k}\sigma_1(z)}\right) \tag{2.15}$$

where $\sigma_1(z)$ is the standard deviation of z when H_1 is true and Φ is the cumulative distribution function of the standard Normal distribution. From Equation 2.13 and Equation 2.15 we arrive at the lower bound

$$1 - \Phi\left(\frac{\log A - kE_1(z)}{\sqrt{k}\sigma_1(z)}\right) \leq P_1(n \leq k) \tag{2.16}$$

Similarly, we find that

$$P_0(\sum_{i=1}^{k} z_i \leq B) = \Phi\left(\frac{\log B - kE_0(z)}{\sqrt{k}\sigma_0(z)}\right) \tag{2.17}$$

where $\sigma_0(z)$ is the standard deviation of z when H_0 is true. From Equation 2.14 and Equation 2.17 we arrive at the lower bound

$$\Phi\left(\frac{\log A - kE_0(z)}{\sqrt{k}\sigma_0(z)}\right) \leq P_0(n \leq k) \tag{2.18}$$

These two inequalities provide tight lower bounds for the probability that the SPRT will terminate after k steps.

For direct calculation, we note that

$$\begin{aligned} E_\theta(z) &= E_\theta\left(\log\frac{f(x;\theta_1)}{f(x;\theta_0)}\right) \\ &= \theta\log\frac{\theta_1}{\theta_0} + (1-\theta)\log\frac{1-\theta_1}{1-\theta_0} \end{aligned} \tag{2.19}$$

and

$$\begin{aligned} \sigma_\theta(z) &= \sqrt{Var_\theta(z)} \\ &= \sqrt{E_\theta(z^2) - E_\theta^2(z)} \\ &= \sqrt{\theta\log^2\frac{\theta_1}{\theta_0} + (1-\theta)\log^2\frac{1-\theta_1}{1-\theta_0} - \left(\theta\log\frac{\theta_1}{\theta_0} + (1-\theta)\log\frac{1-\theta_1}{1-\theta_0}\right)^2} \\ &= \sqrt{\theta(1-\theta)\left(\log^2\frac{\theta_1}{\theta_2} + \log^2\frac{1-\theta_1}{1-\theta_0} - 2\log\frac{\theta_1}{\theta_0}\log\frac{1-\theta_1}{1-\theta_0}\right)} \end{aligned} \tag{2.20}$$

Chapter 3

Sequential PLAL

In this chapter we introduce the PLAL algorithm proposed by Urner et al. [18] and modify it with the sequential probability ratio test to query fewer labels while achieving the same error bound. In Section 3.1 we describe the PLAL algorithm and its properties. Then in Section 3.2 we introduce the SeqPLAL algorithm and explain how the modification of PLAL changes its expected number of queries. We compare PLAL with SeqPLAL with experiments in Section 3.3 that demonstrate the savings achieved with SeqPLAL while meeting the same PAC bounds.

3.1 PLAL Algorithm

The PLAL algorithm is similar in spirit to the DH algorithm in that it descends down a binary space-partitioning tree to identify a homogeneous partitioning of the data. The algorithm starts at the root of the tree (which represents the entire dataset) and proceeds down to the leaves of the tree (individual samples). At each node of the tree, the algorithm queries randomly selected nodes and asks an oracle for their labels. If all the labels are the same, then that node is declared homogeneous, and all unlabeled points in that node are assigned the majority label. If the randomly selected points have a mixture of labels, then the node is declared heterogeneous, and the algorithm will split that node and analyze its children. PLAL pseudocode is provided in Algorithm 1, and the algorithm is illustrated in Figure 3.1.

PLAL takes in user-defined ϵ and δ parameters to specify the amount of error they are willing to incur. By choosing smaller values for ϵ and δ, more points are queried by the oracle. Since ϵ and δ control the amount of queries made, the parameters define the budget for the algorithm.

For data drawn from any data-generating distribution with m samples, PLAL labels at least $(1-\epsilon)m$ samples correctly with probability at least $(1-\delta)$. The bound on the number of queries assumes that the data satisfies a Probabilistic Lipschitz condition and requires an upper bound on the data-diameter of the points at each tree level to bound the number of samples. The authors prove more specific query bounds for various data-diameter bounds on a dyadic spatial tree in [18].

In contrast to the DH algorithm, which descends down a hierarchical clustering of the data, PLAL uses a spatial tree to descend down different regions of the input space. Though the authors focused on dyadic trees, the PLAL algorithm can use any spatial tree, including those that are

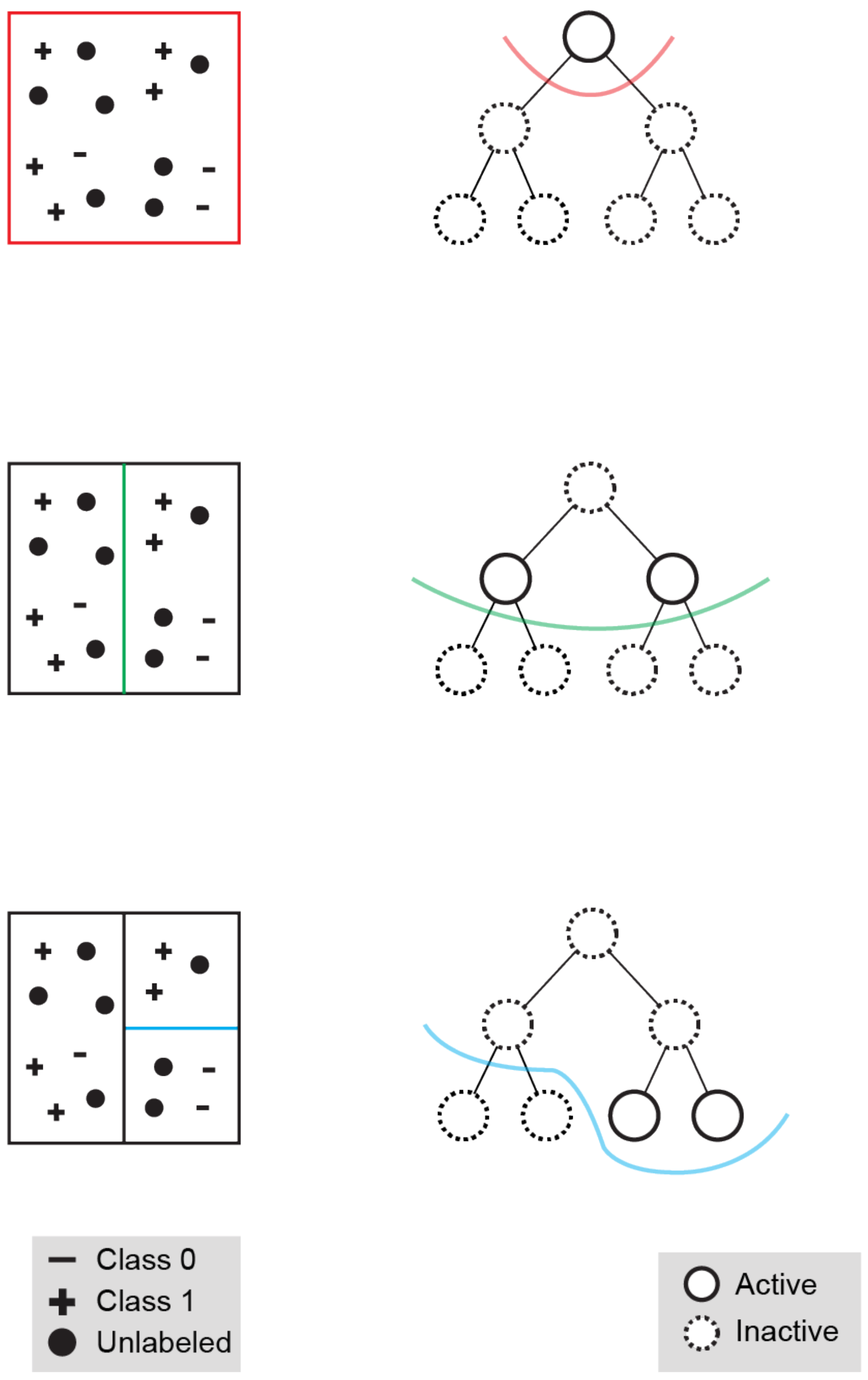


Figure 3.1: PLAL descending down three levels of the binary space-partitioning tree.

adaptive to intrinsic dimension such as a random projection tree or a PCA-tree [20].

Algorithm 1 PLAL

```
**Input:** unlabeled sample $S_{\mathcal{X}} = (x_1, \dots, x_m)$, spatial tree $T$, parameters $\epsilon, \delta$
$level = 0$
$active\_cells[0].append(Root(T))$
**while** $active\_cells[level]$ not empty **do**
    $q\_level = \frac{level \cdot 2 \cdot \ln(2) + \ln(1/\delta)}{\epsilon}$
    **for all** $C$ **in** $active\_cells[level]$ **do**
        $C.query(m)$
        **if** all labels in $C$ are the same **then**
            label all points in $C \cap S$ with that label and mark $C$ inactive
        **else**
            **if** there are unqueried points in $C \cap S$ **then**
                $active\_cells[level+1].append(Left(C), Right(C))$
            **end if**
        **end if**
    **end for**
    $level = level + 1$
**end while**
```

3.2 Sequential PLAL

PLAL's appeal comes from its bounds on error and queries. However, PLAL frequently queries large percentages of the data in order to estimate whether or not a region of space is homogeneous. By introducing sequential hypothesis testing, PLAL can be altered to query less points while providing the same bounds on error and better average bounds on number of queries. We demonstrate this melding of ideas in our SeqPLAL algorithm.

3.2.1 Sequential PLAL Algorithm

The Sequential PLAL algorithm, shortened as SeqPLAL, follows the same framework as PLAL. We formulate the problem of homogeneity testing as one of hypothesis testing. We provide pseudocode for the SeqPLAL algorithm in Algorithm 2 and its querying procedure in Algorithm 3.

At each run of the algorithm, we run two sequential probability ratio tests in parallel: one tests whether the cell is homogeneous 0s or has more than an ϵ-fraction of 1s, and the other tests whether the cell is homogeneous 1s or has more than an ϵ-fraction of 0s. Each SPRTs test the mean of the binomial distribution, where p represents the probability of a point having label 1.

For the first SPRT we set $H_0 : p = 0$ and $H_1 : p = \epsilon$. This setup establishes an *indifference region*, a range of values for p for which we cannot make a decision, at $(0, \epsilon)$. We terminate the SPRT the first time a 1 is seen, as this is not possible under H_0, or when a sufficient number of 0s have been seen. Similarly, for the second SPRT we set $H_2 : p = 1 - \epsilon$ and $H_3 : p = 1$. This

SPRT terminates the first time a 0 is seen, or when a sufficient number of 1s have been seen. Let us define

$$test_0 = \log \frac{f(x;p_1)}{f(x;p_0)} \tag{3.1}$$

and

$$test_1 = \log \frac{f(x;p_3)}{f(x;p_2)} \tag{3.2}$$

where p_i corresponds to the value of p for hypothesis H_i.

The user defines α and β parameters that determine the stopping boundaries A and B, which we use to stop both SPRTs. There are four ways for the querying procedure to terminate.

1. We declare the cell inactive and label all the unlabeled points in it with 0s if

$$test_0 \leq \log B \wedge test_1 \leq \log B. \tag{3.3}$$

2. We declare the cell inactive and label all the unlabeled points in it with 1s if

$$test_0 \geq \log A \wedge test_1 \geq \log A. \tag{3.4}$$

3. We declare the cell heterogeneous and split it if

$$test_0 \geq \log A \wedge test_1 \leq \log B. \tag{3.5}$$

4. We have queried up to q_level points, at which point declare the cell heterogeneous and split it.

It is common in sequential analysis to enforce a limit on the number of samples to observe before making a decision, as we do in option 4. This is because there is no guarantee that the sequential probability ratio test will terminate after k samples, even though it is guaranteed to terminate eventually. Unlike traditional truncated SPRTs, we do not incur any additional error by forcing an early decision. This is because no error is incurred by splitting a cell and querying points from its children [21].

This parallel SPRT model is similar to the hierarchical hypothesis testing proposed in [25]. We test whether the cell has some 0s or some 1s, and after making that decision we test whether the cell is homogeneous or heterogeneous. However, this parallel model allows us to test all hypotheses simultaneously. This scheme is also similar to the Armitage method for sequentially testing multiple hypotheses [1], except we avoid making unnecessary comparisons between hypotheses.

Algorithm 2 SeqPLAL

```
procedure SeqPlal(T, $\alpha, \beta$)
    Input: unlabeled sample $S_{\mathcal{X}} = (x_1, \ldots, x_m)$, spatial tree $T$, parameters $\epsilon, \alpha, \beta$
    $level = 0$
    $active\_cells[0].append(Root(T))$
    while $active\_cells[level]$ not empty do
        $q\_level = \frac{level \cdot 2 \cdot \ln(2) + \ln(1/\delta)}{\epsilon}$
        for all $C$ in $active\_cells[level]$ do
            $(isHomogeneous, label) = \text{QUERY}(C, m)$
            if $isHomogeneous$ then
                label all points in $C \cap S$ with $label$ and mark $C$ inactive
            else
                if there are unqueried points in $C \cap S$ then
                    $active\_cells[level + 1].append(Left(C), Right(C))$
                end if
            end if
        end for
        $level = level + 1$
    end while
end procedure
```

3.2.2 Error Bound

Our proof for the bound on the error of SeqPLAL follows the format of the proof of the PLAL error bound from [18].

Theorem 3.2.1. *With probability at least* $1 - \delta$*, where* $\delta = \alpha + \beta$*, SeqPLAL labels at least* $(1 - \epsilon)m$ *many points from* $S_{\mathcal{X}}$ *correctly.*

Proof Consider a cell C in the spatial tree that is marked inactive by SeqPLAL. C was either marked inactive because all points in the cell had been labeled by the oracle, in which case all the labels assigned are correct, or SeqPLAL marked C homogeneous when the hypothesis testing procedure terminated. In the latter case, we show that Sequential PLAL mislabeled at most an ϵ-fraction of the points in C. Let us assume that $\min\{\Pr[l = 1 \mid C], \Pr[l = 0 \mid C]\} \geq \epsilon$, meaning that a label is assigned with probability at most $1 - \epsilon$. Let us consider the sequential querying procedure for C. In the previous section we considered the four possible ways that the $Query$ function could terminate.

The first termination possibility is when C is declared homogeneous with 0 labels. Since the indifference region of $test_0$ is $(0, \epsilon)$, we accept the mislabeling of at most ϵ-fraction of points in C when we choose H_0. By the definition of the SPRT, H_0 will be accepted when H_1 is true with probability at most β. Since $test_0$ will make an ϵ-bad decision with probability at most β, it will make an ϵ-good decision with probability at least $1 - \beta$.

Similarly, the second termination possibility is when C is declared homogeneous with 1 labels. Since the indifference region of $test_1$ is $(1 - \epsilon, 1)$, we accept the mislabeling of at most ϵ-fraction of points in C when we choose H_3. With probability α, H_3 will be accepted when H_2

Algorithm 3 SeqPLAL

```
procedure QUERY($C, numPoints, \alpha, \beta, \epsilon$)
    $A = \log((1-\beta)/\alpha), B = \log(\beta/(1-\alpha))$
    $Z_1 = 0, Z_2 = 0$
    $p_0 = 0, p_1 = \epsilon, p_2 = 1-\epsilon, p_3 = 1$
    $labels = [\,]$
    $points = randomSample(C, numPoints)$
    for point $p \in points$ do
        $l = queryLabel(p)$
        $labels.append(l)$
        if $l == 1$ then
            $Z_1 = Z_1 + \log(p_1/p_0)$
            $Z_2 = Z_2 + \log(p_3/p_2)$
        else
            $Z_1 = Z_1 + \log((1-p_1)/(1-p_0))$
            $Z_2 = Z_2 + \log((1-p_3)/(1-p_2))$
        end if
        if $Z_1 \leq B \wedge Z_2 \leq B$ then return $(True, 0)$
        else if $Z_1 \geq A \wedge Z_2 \geq A$ then return $(True, 1)$
        else if $Z_1 \geq A \wedge Z_2 \leq B$ then return $(False, None)$
        end if
    end for
    return $(False, None)$
end procedure
```

is true. Just as in the previous case, $test_1$ will make an ϵ-bad decision with probability at most α, so it will make a good decision with probability $1-\alpha$.

The last two ways resulted in C being declared heterogeneous and C's children are passed to the next level of the algorithm. Since the majority label is not assigned to the unlabeled points in C, no error is incurred when C is declared heterogeneous.

We can convert the strength of the tests (α, β) to PAC bounds as long as $\alpha+\beta=\delta$ (due to the union bound), since this will bound the probability that we accept the result of a test that makes an ϵ-bad decision. For situations where we have no prior knowledge of the distribution of 0s and 1s in the data, it makes the most sense to set $\alpha=\beta=\delta/2$. The connection between the strength of an SPRT to PAC bounds has previously used to reduce the number of samples observed before returning a hypothesis [12].

Thus, with probability at least $1-\delta$ (and test strength $(\delta/2, \delta/2)$), the SeqPLAL will label at least $(1-\epsilon)m$ points correctly. ∎

3.2.3 Query Bound

Our proof of the query bound follows that format of the PLAL query bound proof in [18]. For our bound on the number of queries, we define spread of the samples points at level k of the spatial tree as the *data diameter*. We denote λ_k^S as the maximum data-diameter in a cell at the kth level of the tree. Formally, $\lambda_k^S = \max\{\text{diam}(C,S) : C \text{ is a cell at level } k\}$, where $\text{diam}(C,S) = \max\limits_{x,y\in C\cap S} \|x-y\|$.

Theorem 3.2.2. *Let $\mathcal{X} = [0,1]^d$ be the domain, $P_{\mathcal{X}}$ a distribution over $\mathcal{X}$, $l\colon \mathcal{X} \to \{0,1\}$ a labeling function that is ϕ-Lipschitz for some function ϕ, let $q_i = \frac{i*2*\ln(2)+\ln(1/\delta)}{\epsilon}$ denote the query number of SeqPLAL for level i and let $(\lambda_i)_{i\in\mathbb{N}}$ be a decreasing sequence with $\lambda_i \in [0,\sqrt{d}]$.*

Then the expected maximum number of queries that SeqPLAL makes on an unlabeled i.i.d sample S from $P_{\mathcal{X}}$ of size m, given that the data diameter of S at level k satisfies $\lambda_k^S \leq \lambda_k$ $\forall k$, is bounded by $\min\limits_{k\in\mathbb{N}}(q_k 2^k + \phi(\lambda_k)\cdot m)$.

Proof The Probabilistic Lipschitzness property of the data $P_{\mathcal{X}}$ allows us to bound the number of points that lie in heterogeneous cells at any level k of the spatial tree T. First, let's consider the number of unlabeled points at the beginning of round $k+1$. For any sample point x that lies in a label heterogeneous cluster at level k, there is a sample point y in the cluster such that the labeling function l on x and y violates the standard Lipschitz condition for Lipschitz constant $1/\lambda_{k^S}$. Since $\lambda_k^S \leq \lambda_k$, the labeling function l on x and y violates the standard Lipschitz condition for Lipschitz constant $1/\lambda_k$ as well. By the Probabilistic Lipschitz assumption of the data, we can bound the total weight of points that violate the standard Lipschitz condition by $\phi(\lambda_k)$. Since λ_k was fixed before drawing the sample, the expected number of sample points that lie in heterogeneous clusters at level k is bounded by $\phi(\lambda_k)\cdot m$. This means that the expected number of points that are unlabeled at the beginning of round $k+1$ is bounded by $\phi(\lambda_k)\cdot m$.

Now we will bound the number of queries made up to level k. Let's consider the partition of the space that SeqPLAL returned at the start of round k, which consists of cells declared homogeneous in earlier rounds and cells that are active in round k. The sequential probability ratio test terminates after a maximum of q_k steps, so q_k is an upper bound on the number of label-queries the algorithm has made so far on each cell in the partition. Since the sequence $(q_i)_{i \in \mathbb{N}}$ is non-decreasing and there are at most 2^k cells in the partition, it follows that $q_k 2^k$ is an upper bound on the number of queries made up to level k.

Combining these two bounds, it follows that the number of queries made by SeqPLAL is bounded by $q_k + \phi(\lambda_k) \cdot m$ for any k. ∎

In practice, the amount of queries that SeqPLAL makes for each cell will usually be less than q_k. Similar to the role of the ϵ and δ parameters in PLAL, α, β, and ϵ control the amount of queries made in SeqPLAL. In order to show that the number of queries made by SeqPLAL will usually be less than q_k, let us bound the number of samples in the two general termination conditions for SPRTs: when H_0 is accepted and when H_1 is accepted. The average samples numbers of the sequential probability ratio tests are compared with the value of m for different values of k in Figure 3.2. Though the average sample number may be larger than m at the top levels of the tree, the value of m increased as the algorithm descends down the tree while the average sample number of the SPRT stays constant. This means that in practice SeqPLAL should query less points than PLAL while guaranteed to never query more points than PLAL.

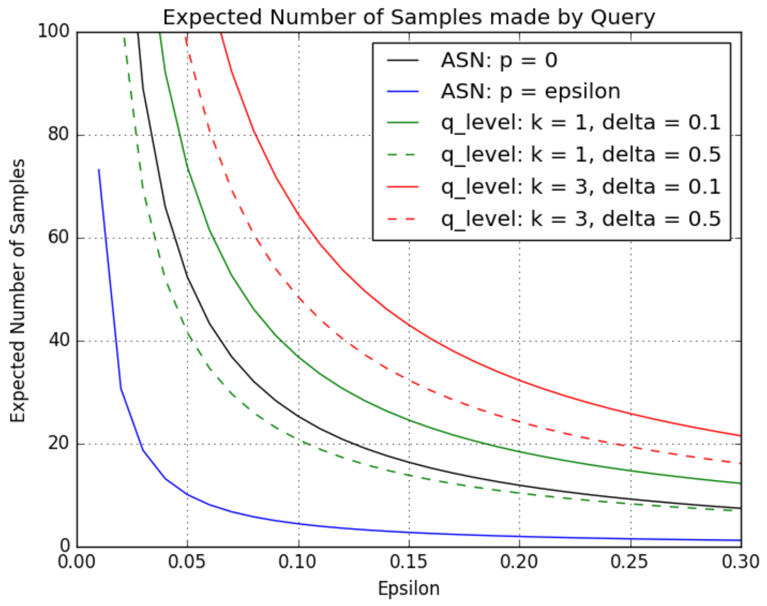
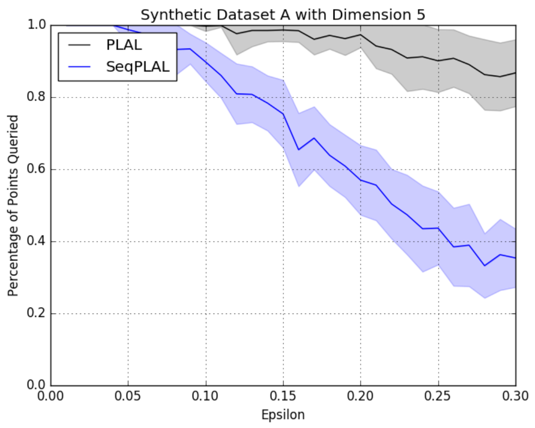


Figure 3.2: Number of samples made during the Query functions of PLAL and SeqPLAL. For SeqPLAL $\alpha = \beta = 0.05$.

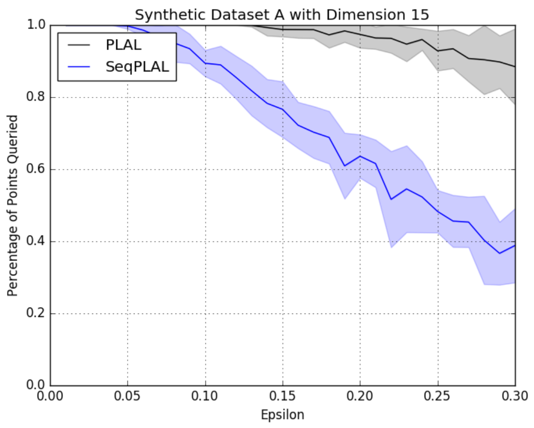
3.3 Experiments

To compare the performance of SeqPLAL with PLAL, we first run the two algorithms on synthetic datasets. Following the examples in [18], we create three datasets by sampling 2000 points from a Gaussian Mixture Model.

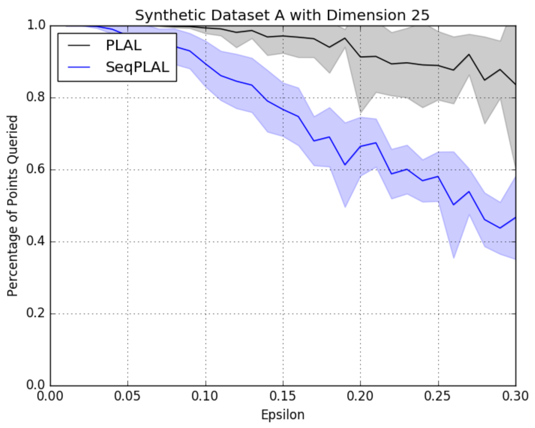
Each data distribution consisted of four dense Gaussians and four sparse Gaussians, all of which had diagonal covariance matrices. 80% of the points were sampled from the dense Gaussians, and 20% were sampled from the sparse Gaussians. The clusters are assigned binary labels. Each of the three datasets has a different variance assigned to the dense and sparse Gaussians. Dataset A has dense clusters with variance 0.1 and sparse clusters with unit variance. Dataset B has dense clusters with variance 0.01 and sparse clusters with variance 0.1. Dataset C has dense clusters with variance 0.001 and sparse clusters with variance 0.1. This makes dataset C the easiest to cluster and dataset A the hardest to cluster.



(a) Synthetic data A with 5 dimensions averaged over 30 trials.



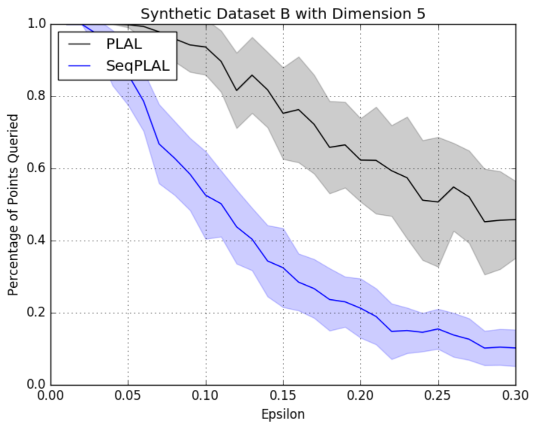
(b) Synthetic data A with 15 dimensions averaged over 30 trials.



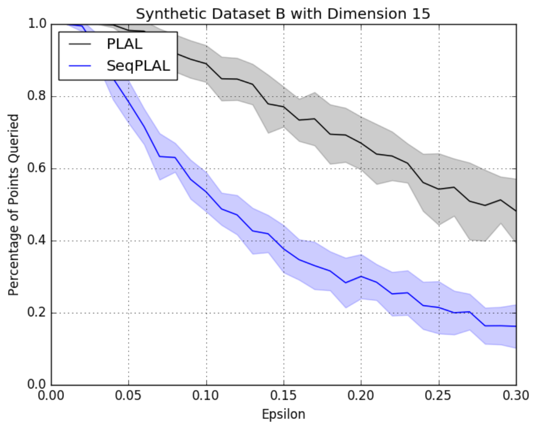
(c) Synthetic data A with 25 dimensions averaged over 30 trials.

Figure 3.3: Percentage of points queried across different dimensions of dataset A for different values of ϵ.

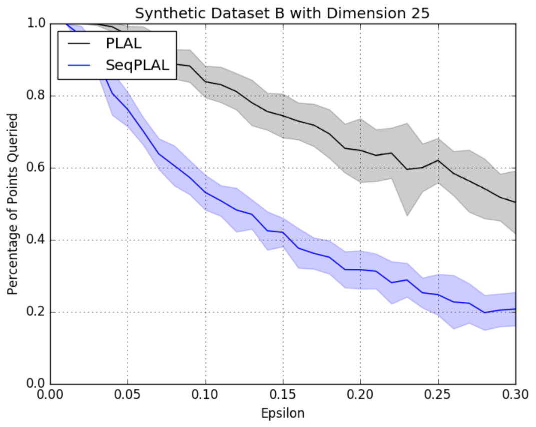
The number of queries made on the synthetic datasets is shown in Figures 3.3, 3.4, and 3.5. Across synthetic datasets A, B, and C, SeqPLAL consistently queries less points than PLAL across values of ϵ ranging from 0.01 to 0.30. The number of queries made did not vary as the



(a) Synthetic data B with 5 dimensions averaged over 30 trials.



(b) Synthetic data B with 15 dimensions averaged over 30 trials.



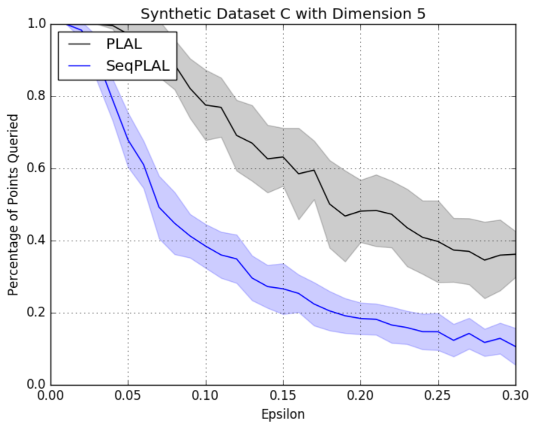
(c) Synthetic data B with 25 dimensions averaged over 30 trials.

Figure 3.4: Percentage of points queried across different dimensions of dataset B for different values of ϵ.

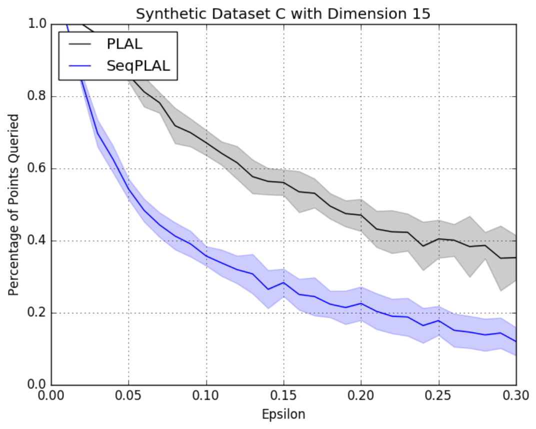
dimension of each dataset increased from 5 to 15 to 25 dimensions.

After testing SeqPLAL on synthetic data, we tested it on real world datasets. Figure 3.9 compares SeqPLAL and PLAL performance on the Wisconsin Diagnostic Breast Cancer dataset [15]. The number of queries on the data was similar between the two algorithms for small values of ϵ. As the value of ϵ increased, SeqPLAL queried less points while incurred about the same error. On the UCI mushroom dataset SeqPLAL outperformed PLAL for all values of ϵ, while incurring slightly more error than PLAL. The performance of the two algorithms on the mushroom dataset is shown in Figure 3.10.

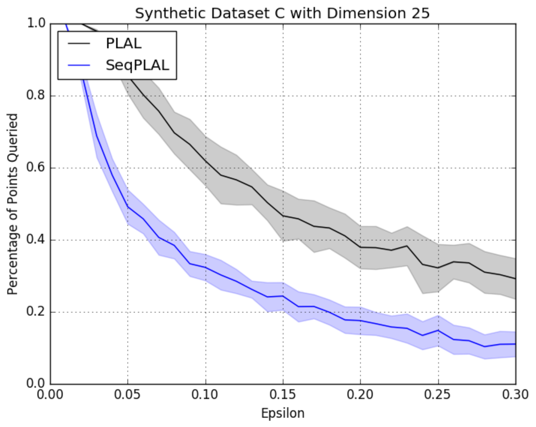
While the error and query bounds hold for all datasets, both SeqPLAL and PLAL will have the best performance on datasets that are Probabilistic Lipschitz for a sufficiently small Lipschitz constant. We next discuss two datasets on which both SeqPLAL and PLAL have difficulty:



(a) Synthetic data C with 5 dimensions averaged over 30 trials.



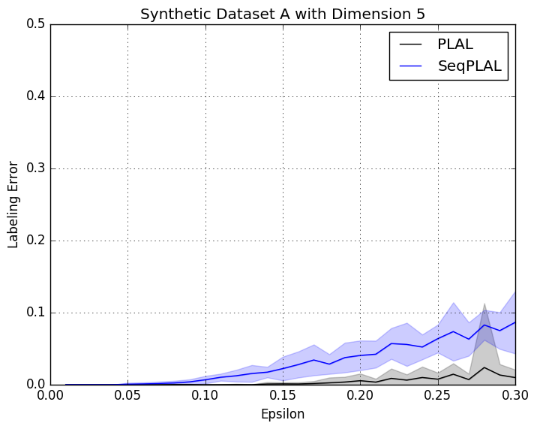
(b) Synthetic data C with 15 dimensions averaged over 30 trials.



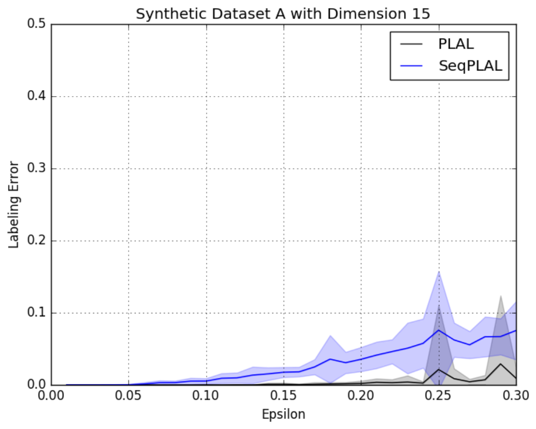
(c) Synthetic data C with 25 dimensions averaged over 30 trials.

Figure 3.5: Percentage of points queried across different synthetic datasets for different values of ϵ.

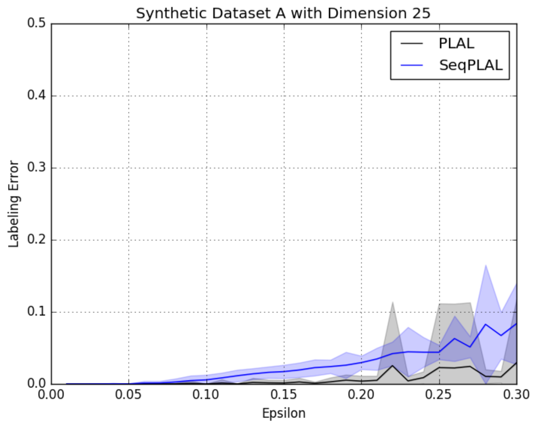
Ringnorm and 20 Newsgroups. The Ringnorm dataset [3] was designed as an adversarial dataset to test decision tree algorithms. The 20 Newsgroups dataset is a collection of news articles spanning 20 different categories [9]; for our experiments we compared the religion and atheism categories and constructed feature vectors using bag-of-words from the top 200 words. Since these datasets are not as well clusterable as the previous ones, neither PLAL nor SeqPLAL were able to significantly reduce the number of queries made when labeling the dataset. However, for values of ϵ greater than 0.15, SeqPLAL tended to query less points than PLAL without significantly increasing the empirical error. This shows even on datasets for which cluster-based active learning may not work well, SeqPLAL will still have practical advantages over PLAL while guaranteeing the same PAC bounds.



(a) Synthetic data A with 5 dimensions averaged over 30 trials.



(b) Synthetic data A with 15 dimensions averaged over 30 trials.

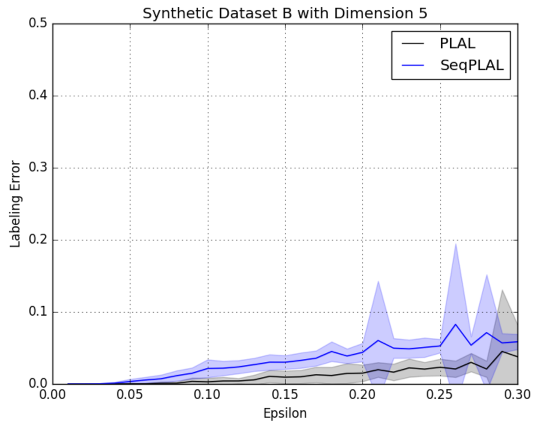


(c) Synthetic data A with 25 dimensions averaged over 30 trials.

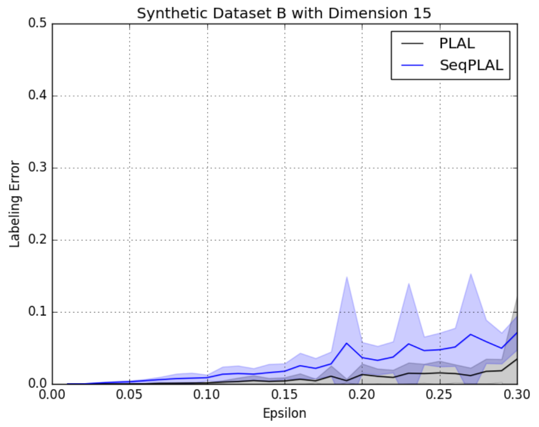
Figure 3.6: Labeling error across different dimensions of dataset A for different values of ϵ.

3.4 Conclusion

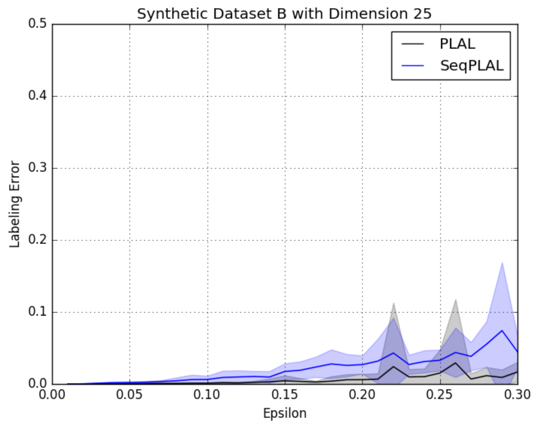
In this section we explained how the sequential probability ratio test can augment the PLAL algorithm that query less points while achieving the same PAC bounds on error. Our experiments validate our theoretical results, as SeqPLAL never queries more than PLAL and frequently queries less than PLAL on datasets from a variety of domains.



(a) Synthetic data B with 5 dimensions averaged over 30 trials.

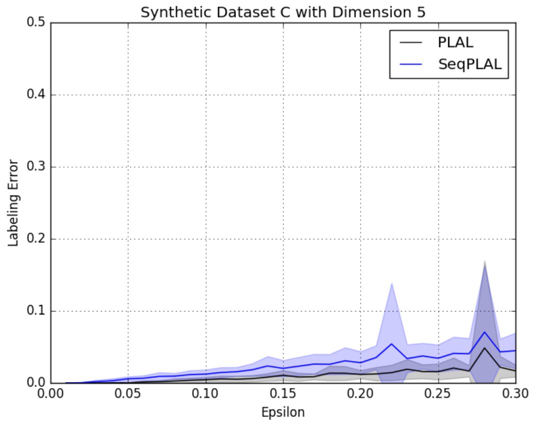


(b) Synthetic data B with 15 dimensions averaged over 30 trials.

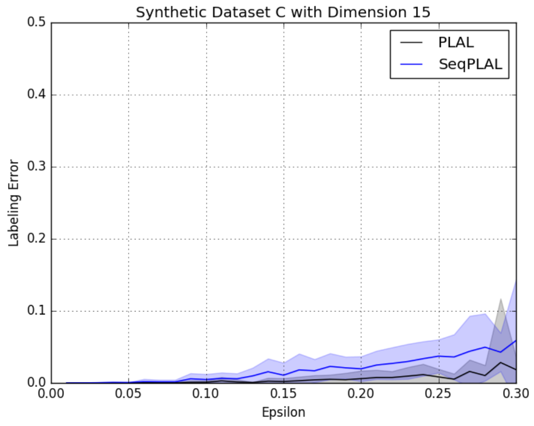


(c) Synthetic data B with 25 dimensions averaged over 30 trials.

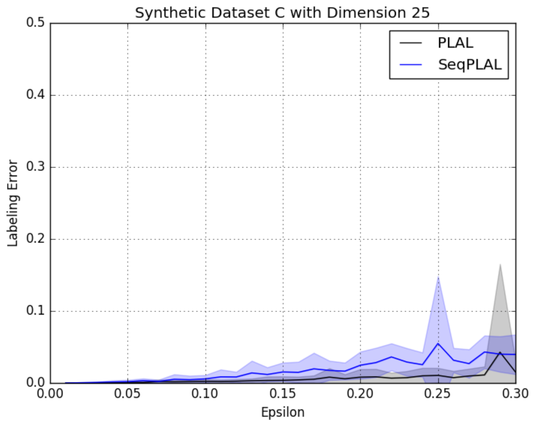
Figure 3.7: Labeling error across different dimensions of dataset B for different values of ϵ.



(a) Synthetic data C with 5 dimensions averaged over 30 trials.

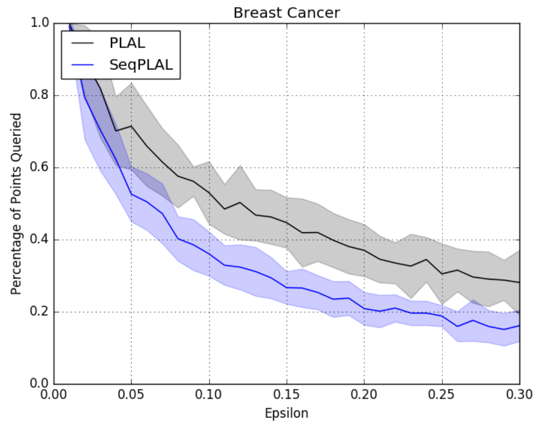


(b) Synthetic data B with 15 dimensions averaged over 30 trials.

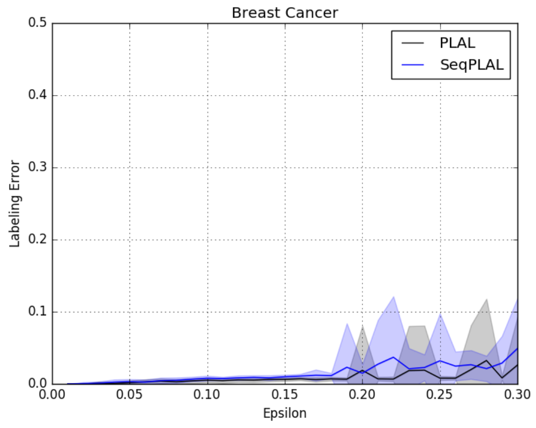


(c) Synthetic data C with 25 dimensions averaged over 30 trials.

Figure 3.8: Labeling error across different dimensions of dataset C for different values of ϵ.

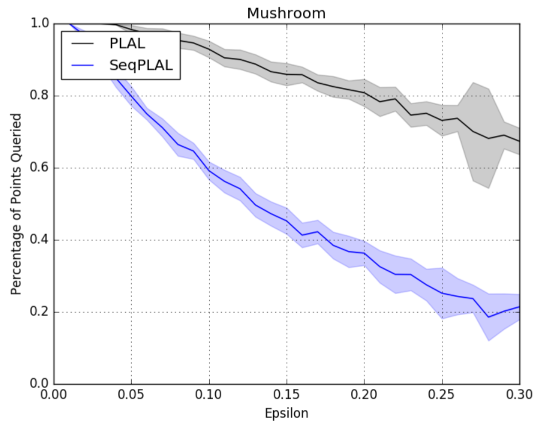


(a) Percent of data queried on breast cancer dataset.

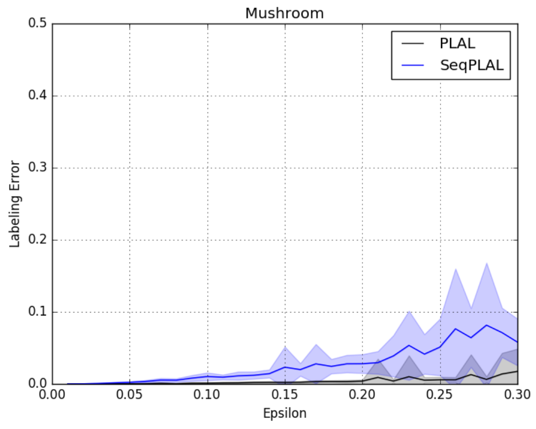


(b) Labeling error on breast cancer dataset.

Figure 3.9: Wisconsin Diagnostic Breast Cancer Dataset

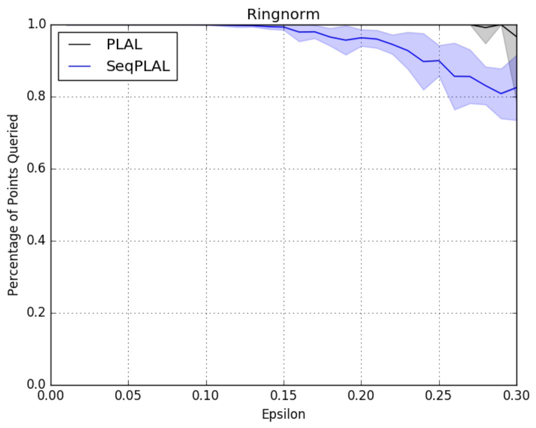


(a) Percent of data queried on mushroom dataset.

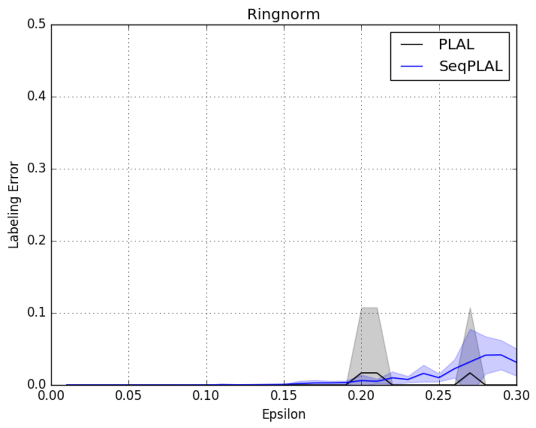


(b) Labeling error on mushroom dataset.

Figure 3.10: UCI Mushroom Dataset

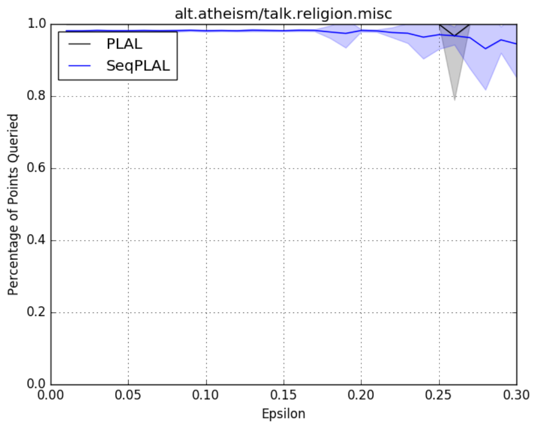


(a) Percentage of queries made on Ringnorm dataset.

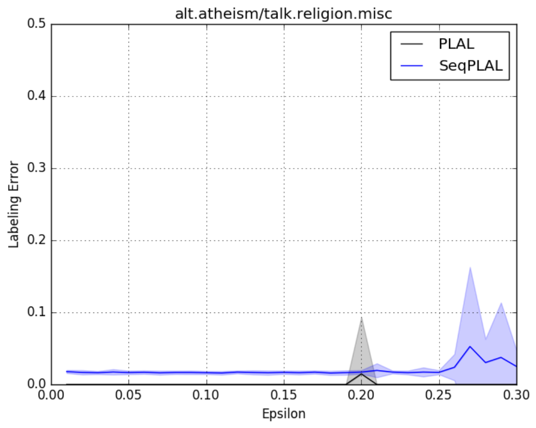


(b) Labeled error on Ringnorm dataset.

Figure 3.11: Ringnorm Dataset



(a) Percentage of queries made on 20 Newsgroups dataset.



(b) Labeled error on 20 Newsgroups dataset.

Figure 3.12: 20 Newgroups dataset with the atheism and religion categories.

Chapter 4

Sequential Learning from Strong and Weak Oracles

In this chapter we explain different approaches to labeling samples when both a strong and weak oracle are available. The weak oracle is assumed to be significantly cheaper to query than the strong oracle. In Section 4.1 we discuss the UBS algorithm introduced in [17] and discuss why it is not feasible to implement directly. We modify the UBS algorithm with the sequential probability ratio test in a more practical algorithm we call SeqUBS. SeqUBS is defined is Section 4.2. We explore simulations of SeqUBS on different datasets in Section 4.3. We then combine SeqPLAL and SeqUBS into a proactive learning algorithm and compare its performance to that of SeqPLAL.

4.1 UBS Algorithm

As discussed in the overview in Chapter 2, there have been many approaches to working with multiple oracles or crowd workers. However, the errors made are typically modeled as random noise. In reality, the types of errors that a oracle makes are unlikely to be random. Urner et al. [17] instead models weak oracles as making correct labels when points lie in homogeneous regions and making errors when points lie in heterogeneous regions. This assumption is more realistic than the assumption that a weak oracle will make random mistakes. The authors also assume that the labeling function l is Lipschitz with a constant $1/L$.

In their paper, Urner et al. define an *s-smoothed labeling function* $\pi_s(x)$ as a measure of neighborhood heterogeneity around a point x, where s is a similarity function such as Euclidean distance. They denote $w : \mathcal{X} \to [0, 1]$ as the probability that the weak oracle will assign the label 1 to a $x \in \mathcal{X}$, and they assume that the $w(x)$ should closely match $\pi_s(x)$. In their algorithm, they try to estimate the probability that a weak oracle' estimates are correct based on the homogeneity of the neighborhood around points. We denote this algorithm as UBS, after the surnames of the authors.

UBS assumes that the algorithm has access to a pool of samples $P_{\mathcal{X}}$ that can be labeled by a weak or strong oracle. It also assumes the labels are binary. From the set of unlabeled points, we wish to label a subset of points $T \subset P_X$ to pass to an agnostic learner. The UBS algorithm pseudocode is shown in Algorithm 4.

Algorithm 4 UBS Algorithm

```
Input: unlabeled sample $\mathcal{X}$, weak oracle $w$, strong oracle $s$, ball radius $\beta$, threshold $\eta$
$(S, T) = drawRandomSamples(\mathcal{X})$
$S_{xy} = \{\}$
for $s \in S$ do
    $y = w.label(s)$
    $S_{xy} = S_{xy} \cup (s, y)$
end for
$T_{xy} = \{\}$
for $t \in T$ do
    $B = S_{xy}.getPointsInBall(t, \beta)$
    $w = mean(labels(B \cap S))$
    if $w < \eta$ then
        $L = L \cup (t, 0)$
    else
        if $w > 1 - \eta$ then
            $L = L \cup (t, 1)$
        else
            $y = s.label(t)$
            $L = L \cup (t, y)$
        end if
    end if
end for
```

First, two random samples of domain points, S and T, are sampled i.i.d. from the marginal distribution $P_{\mathcal{X}}$. All the points in S are labeled by the weak oracle. For each point $x \in T$, we estimate the confidence in the correctness of the label that the weak oracle would be assign to x. This is calculated by centering a ball of radius β at each $x \in T$. We then calculate the average label of all the points in S contained in the ball:

$$w_{S,\beta}(x) = \frac{1}{|S|} \sum_{(z,y) \in B_\beta(x) \cap S} y \tag{4.1}$$

where $B_\beta(x) = \{y \in \mathcal{X} \mid d(x, y) \leq \beta\}$ for some distance function d.

The user sets a confidence threshold η to determine when to query the strong oracle. If $\min\{w_{S,\beta}(x), 1 - w_{S,\beta}(x)\}$ then the algorithm calls the strong oracle. Otherwise the algorithm labels the point as 1 if $\min\{w_{S,\beta}(x), 1 - w_{S,\beta}(x)\} > 1/2$ and 0 if $\min\{w_{S,\beta}(x), 1 - w_{S,\beta}(x)\} < 1/2$.

This algorithm only works if there are a sufficient number of points in $B_\beta(x) \cap S$; without a sufficiently large sample, the confidence estimate this provides will be weak. The authors prove that there will exist a $|S|$ such that the difference between $\pi_s(x)$ and $w_{S,\beta}(x)$ is small. However, their proof is nonconstructive in that they do not provide a way to calculate a sufficient size for S. Additionally, their proof sets the value of the ball's radius β to ϵ/L, where ϵ is the error between $\pi_s(x)$ and $w_{S,\beta}(x)$ and L. Generally, it is not known what the Lipschitz constant of the labeling function will be, since the labels are unknown. Alas, though this algorithm is simple, bounds the error made in the confidence estimates, and bounds the calls to the strong oracle, it does not explain how to choose the proper variable values to make the algorithm work in real life. In the next section, we discuss our modifications to UBS to make their algorithm practical.

4.2 Sequential UBS

Thought the UBS algorithm is simple and intuitive, it requires an unknown number of points to be labeled by the weak oracle. Additionally, it is not known how to practically set the radius β since the Lipschitz constant of the labeling function is not known. Since there is no general way to find satisfactory values, we propose the following alterations to the UBS algorithm.

Though we cannot use the same β value used in the UBS algorithm, it is still desirable to estimate the labels for each $x \in T$ from the points within the neighborhoods of x. In their task of finding the size of neighborhoods that each weak oracle can accurately label, the authors of [19] estimate the radius by finding the smallest distance that minimizes the disagreement that different weak oracles make when labeling the same points. Since we assume access to only one weak oracle, we cannot estimate the "range" of the weak oracle by optimizing over its disagreement with other weak oracles.

Since we do not know the appropriate size of S to set to bound the error that $w_{S,\beta}(x)$ makes, we instead propose randomly choosing a set of points T to pass to the agnostic learner. For each $x \in T$, we get the nearest neighboring points to x, and we randomly select neighbors for the weak oracle to label. Using the same joint sequential probability ratio test defined in Section 3.2, we can decide whether the points inside the neighborhood are η-homogeneous or heterogeneous. If the sample is η-homogeneous, then we assign the majority label to the point; otherwise, we query the strong oracle to label the point. This method uses the same idea of labeling T by sampling points in their neighborhood, but it has the advantage that the sequential probability ratio test will require a minimal number of samples before making a decision. The algorithm pseudocode is provided in Algorithms 5 and 6.

One challenge is that we need to choose a value for k, the number of neighbors. Our solution is to choose k based on the maximum cost a user is willing to incur for a single point. If all of the neighbors have been labeled by the weak oracle and the sequential probability ratio test has not terminated, then we will label x with the strong oracle. In this case, for each $x \in T$ that maximal cost incurred will be $k \cdot c_w + c_s$, where c_w is the cost of the weak oracle and c_s is the cost of the strong oracle. For our experiments, we set $k = c_s/c_w$ to bound the maximal cost for a given point by $2c_s$.

For any k that we choose, we can lower bound the probability that the sequential probability ratio test will terminate after k samples using the bound from Section 2.4.3. This bound will provide a conservative estimate of whether the strong oracle will be queried.

Algorithm 5 SeqUBS Algorithm

```
**Input:** unlabeled sample set $\mathcal{X}$, weak oracle $wt$, strong oracle $st$, sample size $m$, threshold $\eta$,
number of neighbors $k$
$(S, T) = drawRandomSamples(\mathcal{X})$
$L = \{\}$
for $t \in T$ do
    $N = S.getNeighbors(t, k)$
    $(isHomogeneous, label) =$ GETCONFIDENCE$(N, \alpha, \beta, \eta, wt)$
    if $isHomogeneous$ then
        $L = L \cup (t, label)$
    else
        $y = st.label(t)$
        $L = L \cup (t, y)$
    end if
end for
return $L$
```

Algorithm 6 getConfidence Function

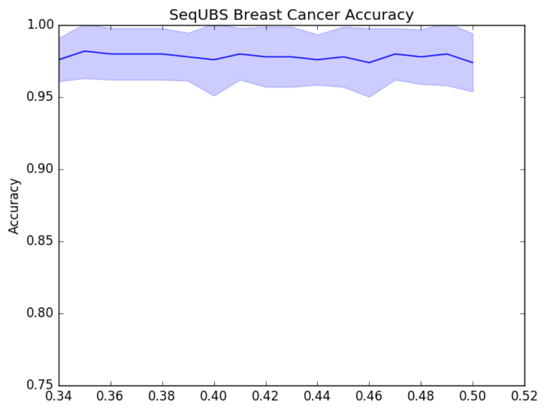
```
procedure GETCONFIDENCE$(N, \alpha, \beta, \eta)$
    $A = \log((1-\beta)/\alpha), B = \log(\beta/(1-\alpha))$
    $Z_1 = 0, Z_2 = 0$
    $p_0 = 0, p_1 = \eta, p_2 = 1-\eta, p_3 = 1$
    $labels =$ [ ]
    $neighbors = randomShuffle(N)$
    for point $p \in neighbors$ do
        $l = wt.label(p)$
        $labels.append(l)$
        if $l == 1$ then
            $Z_1 = Z_1 + \log(p_1/p_0)$
            $Z_2 = Z_2 + \log(p_3/p_2)$
        else
            $Z_1 = Z_1 + \log((1-p_1)/(1-p_0))$
            $Z_2 = Z_2 + \log((1-p_3)/(1-p_2))$
        end if
        if $Z_1 \leq B \wedge Z_2 \leq B$ then return $(True, 0)$
        else if $Z_1 \geq A \wedge Z_2 \geq A$ then return $(True, 1)$
        else if $Z_1 \geq A \wedge Z_2 \leq B$ then return $(False, None)$
        end if
    end for
    return $(False, None)$
end procedure
```

4.3 SeqUBS Experiments

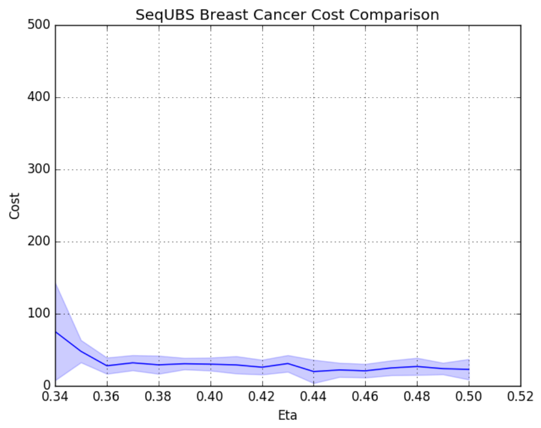
We simulated SeqUBS in order to assess its feasibility. Since UBS was presented as a mathematical proof of concept, the authors of [17] did not provide any simulation on any dataset on which to test. Even though we cannot compare SeqUBS, we present our simulated SeqUBS below.

Following the definition of the weak oracle in [17], we simulate a weak oracle by using a k-nearest neighbor classifier trained on points from dense regions of the dataset. To find these points, a kernel density estimator with Gaussian kernel is fitted to the data, and a small number of points are sampled from the model. This allows us to pick points that are representative of the dataset, which a weak oracle would be likely to know. For our experiments, we set $k = 3$ so the weak oracle's estimate would be locally restricted.

In our experiments we set the cost of the strong oracle to 10 and the cost of the weak oracle to 1. For each experiment trial we selected 50 random samples to form T, and using SeqUBS we labeled the points in T. In Figure 4.1 we show SeqUBS's accuracy and cost on the breast cancer dataset. Figure 4.1a shows that the accuracy is high across all thresholds η, and Figure 4.1b shows that for values of $\eta \geq 0.36$ the cost of labeling drops to less than 50. With only the strong oracle available for labeling, the cost would have been 500, so SeqUBS's leveraging of the weak oracle shows clear savings. Figure 4.2 demonstrates similar results on the UCI mushroom dataset. In Figure 4.3, we see that SeqUBS always calls the strong oracle for small values of η; when η increases, SeqUBS relies more on the estimates from the weak oracle, and cost drops significantly while accuracy drops modestly.



(a) SeqUBS accuracy on breast cancer dataset.

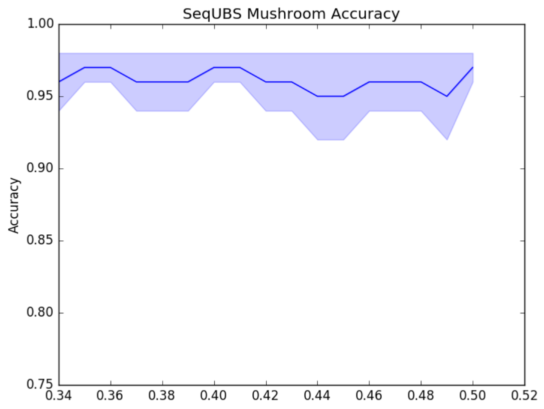


(b) SeqUBS cost on breast cancer dataset.

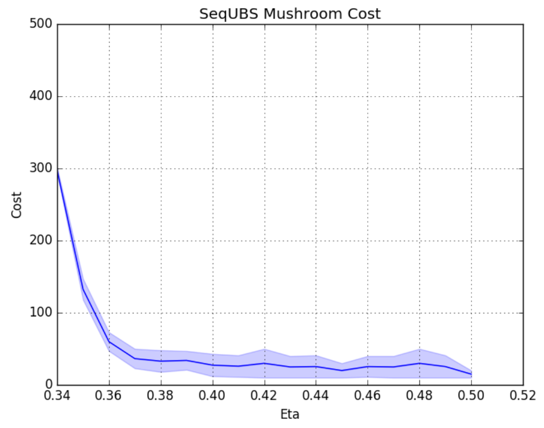
Figure 4.1: SeqUBS results on Wisconsin Diagnostic Breast Cancer dataset. $\alpha = \beta = 0.1$.

4.4 Proactive SeqPLAL with SeqUBS

In addition to being used as a practical algorithm for the same problem setting as UBS, SeqUBS can be used as a proactive subroutine for cluster-based active learning algorithms when both a strong and a weak oracle can be queried. We can easily combine SeqPLAL and SeqUBS by

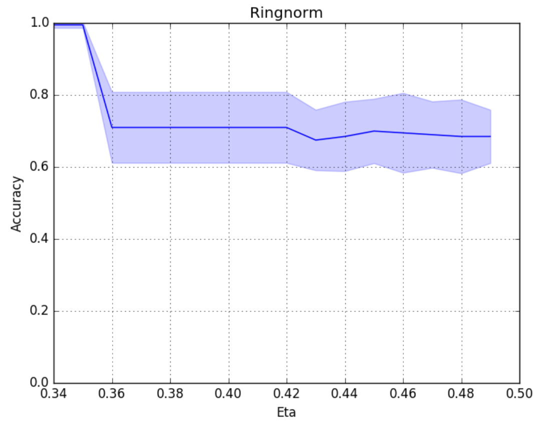


(a) SeqUBS accuracy on mushroom dataset.

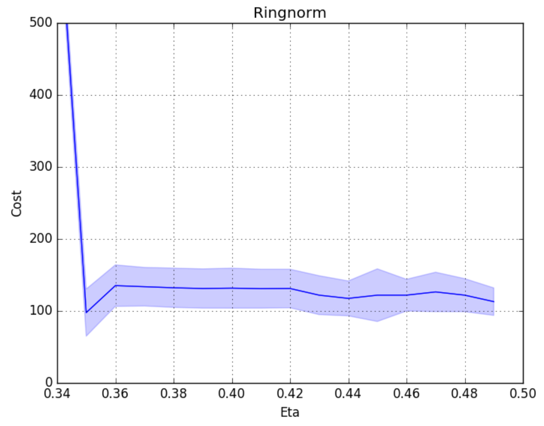


(b) SeqUBS cost on mushroom dataset.

Figure 4.2: SeqUBS results on UCI mushroom dataset. $\alpha = \beta = 0.1$.



(a) SeqUBS accuracy on Ringnorm dataset.

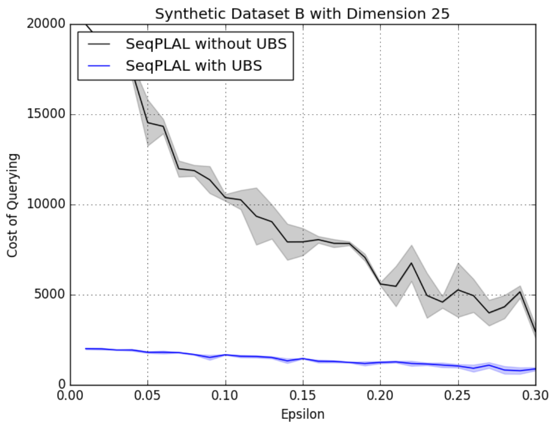


(b) SeqUBS cost on Ringnorm Dataset.

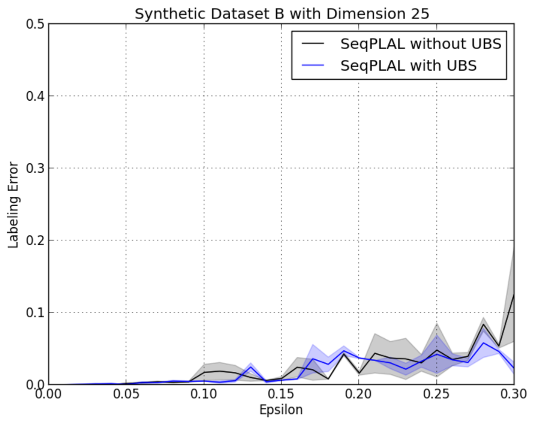
Figure 4.3: SeqUBS results on Ringnorm Dataset. $\alpha = \beta = 0.1$.

calling the $getConfidence$ subroutine from Algorithm 6 in place of the $queryLabel$ function in Algorithm 3. We call this algorithm Proactive SeqPLAL.

We compare the performance of SeqPLAL without UBS and Proactive SeqPLAL on dataset B with dimension 25 from Section 3.3 and the breast cancer data. For our experiments we set the cost of the strong oracle as 10 and the cost of the weak oracle as 1. We compare the respective costs and labeling errors of both algorithms on dataset B with dimension 25 in Figure 4.4. Access to the weak oracle clearly allows Proactive SeqPLAL to incur less cost, and calling the weak oracle doesn't increase its error beyond SeqPLAL's error. In Figure 4.5 we observe a similar decrease in cost while maintaining the same level of error on the breast cancer dataset.

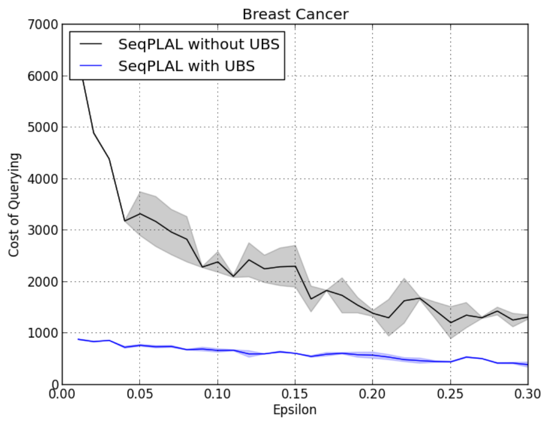


(a) Cost of labeling Synthetic Dataset B with 25 dimensions.

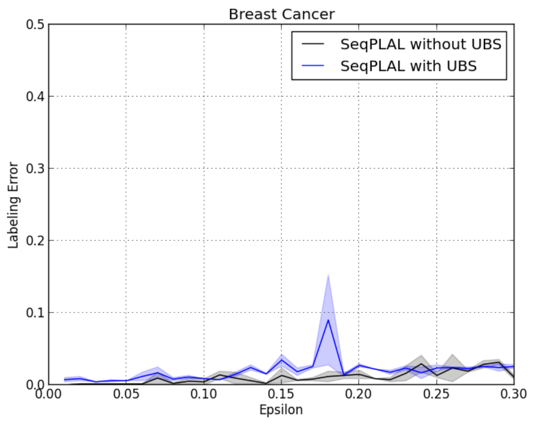


(b) Labeling error on Synthetic Dataset B with 25 dimensions.

Figure 4.4: SeqPLAL with and without SeqUBS on Synthetic Dataset B with 25 dimensions.



(a) Cost of labeling breast cancer dataset.



(b) Labeling error on breast cancer dataset.

Figure 4.5: SeqPLAL with and without SeqUBS on Wisconsin Diagnostic Breast Cancer dataset.

4.5 Conclusion

SeqUBS is a practical modification to the UBS algorithm. It captures the same idea as UBS that the confidence of a weak oracle's label can be estimated by the consistency of the labels of neighboring points. Using the sequential probability ratio test, SeqUBS minimizes the number of points that need to be seen before making a decision about inferring the weak label or calling the strong oracle. The probability that the SPRT will terminate in a given number of steps can be lower bounded, and the maximum number of neighboring points that SeqUBS observes can be set by the user, who can condition on the ratio between the cost of the strong oracle and weak oracle. Additionally, SeqUBS can be used as a proactive subroutine to SeqPLAL when a weak oracle is available in addition to the strong oracle. Using SeqUBS, Proactive SeqPLAL incurs significantly less cost than SeqPLAL without increasing the amount of labeling error.

Chapter 5

Conclusion

5.1 Discussion

We have introduced two algorithms, SeqPLAL and SeqUBS, that can save cost when labeling samples in datasets. SeqPLAL finds a small, label-homogeneous partitioning of the data and infers the labels of unlabeled points based on the majority label of each cell. SeqUBS measures the confidence of a weak oracle's labels by assessing whether the point is in a majority region; the strong oracle is only called when the sample is from a heterogeneous region. Both of these algorithms are built upon the concept of region homogeneity, and the sequential probability ratio test is an efficient way to test whether a region is sufficiently homogeneous or heterogeneous. The SPRT is truncated to enforce a limit on the maximal cost of each step.

In our experiments in Chapter 3, we show that SeqPLAL labels much fewer samples than the original PLAL algorithm both empirically and theoretically. The sequential probability ratio test parameters are set so SeqPLAL achieves the same PAC bounds on error as PLAL. Since UBS is a mathematical explanation of the validity behind the idea of measuring weak oracle labeling confidence, we do not provide a side-by-side comparison of UBS with SeqUBS in Chapter 4. Instead we provide simulations of the improvements that SeqUBS achieves over the baseline of access to a single strong oracle. Finally, we show that SeqUBS can be used as a subroutine to SeqPLAL when the algorithm has access to both a strong and weak oracle in an algorithm we call Proactive SeqPLAL. Access to a weak oracle allows Proactive SeqPLAL to incur less cost than SeqPLAL without losing accuracy.

5.2 Future Direction

We hope our work inspires interest in the use of sequential analysis for efficient region homogeneity testing for problems in active learning and crowdsourcing. There are a few areas we see for immediate future work.

An improvement to the SeqUBS algorithm could be to observe the samples in order of increasing distance away from the point being estimated. As in nearest neighbors methods, we assume the labels of the neighbors points closest to any given point will be most indicative of its label. However, observing samples in an order of increasing distance from a point would not be observing them independently, so the standard guarantees of the sequential probability ratio test may no longer apply. Previous work has shown that for certain probability mass functions of a label value observation, the average sample number of the sequential probability ratio test can be upper and lower bounded even when the samples are not observed independently [8]. Though this is valid for statistical models in item response theory, it remains unclear whether these results can be generalized to our conditions.

Another improvement could be an evaluation of asymmetric choices of values for the α and β parameters in the sequential probability ratio test. In some domains there may be prior knowledge that the classes are imbalanced, in which case it may be much more critical to accurately discover points from one class more than another. Since α and β control the false positive and false negative rates respectively, their tuning could allow the sequential probability ratio test to be more sensitive to a given class.

Bibliography

[1] P Armitage. Sequential analysis with more than two alternative hypotheses, and its relation to discriminant function analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 12(1):137–144, 1950. 3.2.1

[2] Maria-Florina Balcan and Ruth Urner. Active learning–modern learning theory. *Encyclopedia of Algorithms*, pages 8–13, 2016. 2.2

[3] Leo Breiman. Bias, variance, and arcing classifiers. 1996. 3.3

[4] Sanjoy Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19): 1767–1781, 2011. 2.2

[5] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM, 2008. 2.2

[6] Pinar Donmez and Jaime G Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 619–628. ACM, 2008. 1.1, 2.1.1, 2.1.1

[7] Pinar Donmez, Jaime G Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268. ACM, 2009. 2.1.2

[8] Yuan-chin Ivan Chang. Application of sequential probability ratio test to computerized criterion-referenced testing. *Sequential Analysis*, 23(1), 2004. 5.2

[9] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339, 1995. 3.3

[10] Seungwhan Moon and Jaime G. Carbonell. Proactive learning with multiple class-sensitive labelers. In *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, pages 32–38, Oct 2014. doi: 10.1109/DSAA.2014.7058048. 2.1.1

[11] Vikas C Raykar and Shipeng Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13(Feb):491–518, 2012. 2.1.2

[12] Dale Schuurmans and Russell Greiner. Sequential pac learning. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 377–384. ACM, 1995. 3.2.2

[13] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine*

Learning, 6(1):1–114, 2012. 1.1

[14] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008. 2.1.1

[15] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*, pages 861–870. International Society for Optics and Photonics, 1993. 3.3

[16] Ruth Urner and Shai Ben-David. Probabilistic lipschitzness a niceness assumption for deterministic labels. In *Learning Faster from Easy Data-Workshop@ NIPS*, 2013. 2.3.1

[17] Ruth Urner, Shai Ben-David, and Ohad Shamir. Learning from weak teachers. In *AISTATS*, 2012. 4, 4.1, 4.3

[18] Ruth Urner, Sharon Wulff, and Shai Ben-David. Plal: Cluster-based active learning. In *COLT*, pages 376–397, 2013. 1.2, 2.2, 3, 3.1, 3.2.2, 3.2.3, 3.3

[19] Shankar Vembu and Sandra Zilles. Interactive learning from multiple noisy labels. *CoRR*, abs/1607.06988, 2016. 4.2

[20] Nakul Verma, Samory Kpotufe, and Sanjoy Dasgupta. Which spatial partition trees are adaptive to intrinsic dimension? In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 565–574. AUAI Press, 2009. 3.1

[21] Abraham Wald. *Sequential Analysis*. Wiley, New York, 1947. 1.2, 2.4.1, 2.4.3, 3.2.1

[22] Abraham Wald and Jacob Wolfowitz. Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, 1948. 2.4.1

[23] Yan Yan, Glenn M Fung, Rómer Rosales, and Jennifer G Dy. Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1161–1168, 2011. 2.1.2

[24] Yan Yan, Rómer Rosales, Glenn Fung, Faisal Farooq, Bharat Rao, Jennifer G Dy, and PA Malvern. Active learning from multiple knowledge sources. In *AISTATS*, volume 2, page 6, 2012. 2.1.2

[25] Daniel Yekutieli. Hierarchical false discovery rate–controlling methodology. *Journal of the American Statistical Association*, 103(481):309–316, 2008. 3.2.1

[26] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, volume 3, 2003. 2.2