University of Alberta

NOVEL MACHINE LEARNING ALGORITHMS

by

Alireza Farhangfar

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Spring 2013
Edmonton, Alberta

To my parents: Hamideh and Houshang

and to my wife: Golnaz

Abstract

Many machine learning algorithms learn from the data by capturing certain interesting characteristics. Decision trees are used in many classification tasks. In some circumstances, we only want to consider *fixed-depth* trees. Unfortunately, finding the optimal depth-d decision tree can require time exponential in d. In the first part of this dissertation, we present OPTNBDT algorithm, which is a fast way to produce a near optimal fixed-depth decision tree under the Naïve Bayes assumption. We apply this technique to real-world datasets and find that our model improves the computational costs significantly while yielding relatively high accuracy.

In the second part of this dissertation, we present two novel algorithms for active learning. There are scenarios where we have access to many unlabeled data; however, obtaining the labels for the data is difficult. An active learning process tries to address this issue by selectively requesting the label of few unlabeled instances, with the goal of using these newly labeled instances to produce an effective classifier. First, we focus on active learning for image segmentation, which requires producing a label for each pixel in an image. We provide an active learner (LMU) that first selects the image whose expected label will reduce the uncertainty of other unlabeled images the most, and then after greedily selects the most informative image. The results of our experiments, over real-world datasets show that training on very few informative images can produce a segmenter that is as good as training on the entire dataset.

Finally, we present the importance sampling algorithm (*ISAL*) for actively learning in the standard classification framework, which we demonstrate its sample and label efficiency. In particular, on each iteration, *ISAL* identifies a distribution that puts large weight on instances whose labels are uncertain, then requests the label of an instance drawn from that distribution. We prove that *ISAL* can be more sample and label efficient than passive learning with an exponential convergence rate to the Bayes classifier on noise-free data. We also provide empirical studies that show *ISAL* is more effective than many other active learning algorithms.

Acknowledgements

I would like to express my thanks to Russ Greiner, for exposing me to various ideas and problems, his continuous support, vast knowledge and experience, and brilliant comments throughout my studies. I also would like to thank Csaba Szepesvari for his deep knowledge, support and the constructive feedbacks.

I would also like to thank the members of my defense committee, Dale Schuurmans, Jeniffer Neville, Nilanjan Ray for reviewing my dissertation. Thanks to all my friends and colleagues at University of Alberta.

This research would not have been possible without the generous support from NSERC and Alberta Innovates.

I also would like to thank my family for their unwavering support, encouragement and endless love.

Finally, my sincere gratitude goes to my beloved wife and best friend, Golnaz, for her support, invaluable discussions, constructive feedbacks, encouragement and love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this dissertation we focus on *supervised learning for the classification task*, where each data instance belongs to one of a small set of pre-defined categories (*e.g.*, "Spam" versus "non-Spam"; or "cancer" versus "healthy"; etc) and the goal of learning is to produce a classifier that can correctly predict the label given the features of a new instance.

1.1 Near-Optimal Fixed Depth Decision Trees

Many machine learning tasks involve producing a classification label for an instance. There is sometimes a fixed cost that the classifier can spend per instance, before returning a value; consider for example, per-patient capitation for medical diagnosis. If the tests can be performed sequentially, then the classifier may want to follow a *policy* [1] — *e.g.*, first perform a blood test, then if its outcome is positive, perform a liver test; otherwise perform an eye exam. This process continues until the funds are exhausted, at which point the classifier stops running tests and returns an outcome; either "healthy" or "sick".

Such policies correspond naturally to decision trees. There are many algorithms for learning a decision tree from a data sample; most systems [2], [3] use some heuristics that greedily seek the decision tree that best fits the sample, before running a post-processor to reduce overfitting. There is sometimes a fixed budget – which here translates to a hard bound on the depth of the tree. The first part of this dissertation focuses on the challenge motivated above: of finding the best "fixed-cost policy", which corresponds to finding the fixed-depth decision tree that best matches the data sample. There are several algorithms for this task, which typically use dynamic programming to find the "optimal" depth-d decision tree, *i.e.*, the tree that minimizes the 0/1-loss over the training data. These algorithms are invariably exponential in the depth d, as they spend almost all of

their time determining which features to test at final level, d [4]. If one is willing to accept the "Naïve Bayes" assumption [5] — that features are independent of each other given the class — then there is an efficient way to compute the final layer of depth-d features. In particular, under this assumption, we prove that the optimal depth-d feature essentially depends only on the *posterior probability of the class label given the tests previously performed*, but not the outcomes of the individual tests. We can therefore use a fast pre-processing step to create a so-called *opt-feature list*, OFL, that identifies which feature to use as a function of the posterior distribution, then use this list to quickly determine the last level of the tree. This technique results in a speedup of $O(n_f / \log n_f)$, where n_f is the number of features, and effectively means we can compute the optimal depth-d tree in the time typically required to compute the optimal depth-$(d-1)$ tree.

1.2 Active Learning

Most supervised learning methods require a set of labeled examples as the training data. In many applications there are plenty of unlabeled data available, however obtaining labels for those data is costly. An example here is the task of segmenting Magnetic Resonance (MR) images of the brain. Given an MR image of the brain, it is important to segment any tumor region present. Many effective segmentation systems involve a number of parameters that have to be adjusted; some of these systems therefore include a *learning* component that can learn effective parameters from a set of labeled (that is, segmented) images. In general, these systems require a large number of such labeled images to produce an effective segmenter. Fortunately, there are often a large number of available images — perhaps on the web, or in clinical databases. Unfortunately, most such images are unlabeled, and worse, it can be expensive to obtain the labels (as this may require paying a medical doctor to label each image, which is costly in terms of both time and money). This often limits the amount of training data available, which can lead to an inferior segmentation system. An *active learning process* tries to address this problem by identifying a small number of unlabeled instances (images in our example) that should be labeled, to produce an effective classifier.

Each application of active learning can be categorized under one of the following three scenarios.

- **Pool-based active learning**: many active learning applications assume that the learner has access to a large pool of unlabeled data. The task of active learning

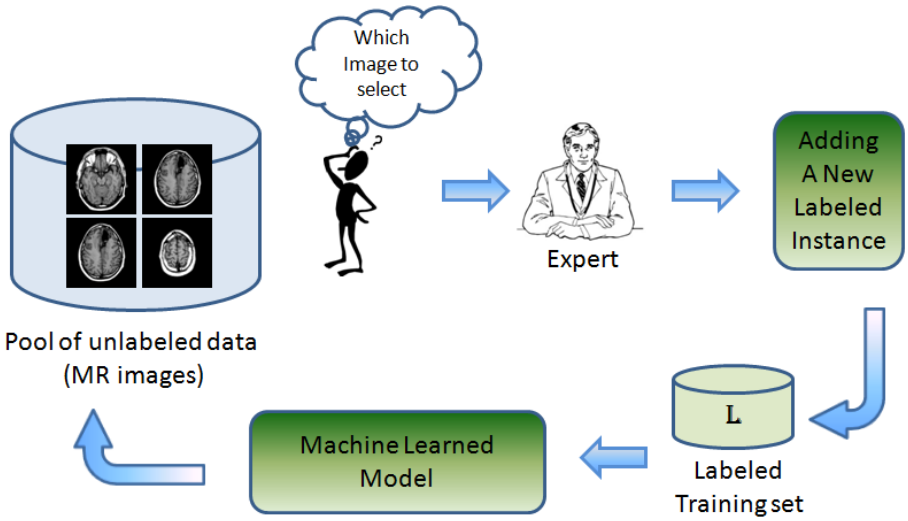

Figure 1.1: Pool-based active learning on MRI of human brain, where the active learner selects an unlabeled instance, obtains the label from the expert, adds it to the training set and a machine learned model is obtained from the training set. This process is then repeated for a certain number of times.

here is to evaluate all the unlabeled data and select the most informative ones for the learning task. Figure 1.1 shows a pool-based active learning scenario. Typically, the instances are chosen in a greedy way based on a scoring function. After each query[1], the scores are updated (based on the returned label for that query) and the next unlabeled instance is selected from the pool. The pool-based active learning is the most common scenario, which is studied in applications such as image classification [6], [7], image segmentation [8], object recognition [9], text classification [10]–[13], information extraction [14], [15], speech recognition [16] and many other domains.

- **Membership query based active learning**: In this scenario, new instances can be generated based on active learner's request, rather than selected from a pool of unlabeled instances. This setting is particularly useful in experimental design applications where the experimenter collects data to analyze the effect of a process on an object. For example, King *et al.* [17], [18] used membership query based active learning in biological experiments with the goal of discovering metabolic pathways in yeast.

- **Sequential selective sampling**: In this scenario, active learner has access to a

[1]Here, a "query" involves identifying a specific unlabeled instance, which is then labeled.

stream of unlabeled instances and it decides whether to obtain the label for each instance. This scenario differs from the membership query based scenario in that here, samples are drawn from the underlying distribution. Sequential selective sampling has been applied to the real world applications such as spam detection [19], handwritten digit recognition [20], sensor scheduling [21], learning ranking functions for information retrieval [22], and part-of-speech tagging [23].

Our LMU algorithm is designed for pool-based active learning, and it is one of the first algorithms to actively learn the parameters of an image segmentor. In contrast, *ISAL* is designed for membership query based active learning since in this setting, *ISAL* can be more sample efficient than other active learning methods (see Section 1.4). In addition, we have developed a variant of *ISAL* for pool-based data too (see Section 5.4.5).

1.3 Objectives and Contributions

This dissertation consists of two parts. In the first part, we present a fast way to produce a fixed-depth decision tree that is optimal under the Naïve Bayes assumption. We prove that the optimal depth-d feature essentially depends only on the posterior probability of the class label given the tests previously performed, but not on either the identity nor the outcomes of these tests. We can therefore precompute, in a fast pre-processing step, which features to use at the final layer. This results in a speedup of $O(n_f / \log n_f)$, where n_f is the number of features. We apply this technique to learning fixed-depth decision trees from standard datasets from the UCI repository [24], and find this model improves the computational cost significantly. Surprisingly, this approach still yields relatively high classification accuracy, despite the Naïve Bayes assumption.

In the second part of this dissertation, we present our work in the active learning domain. First, we address the task of actively learning a segmentation system, *i.e.*, given a pool of unsegmented images, and access to an oracle that can segment a given image, decide which images to provide, to quickly produce a segmenter that is accurate over this distribution of images. Most of the existing active learning algorithms have been used primarily to learn a classifier that maps each instance to a simple label; even most of the imaging works have focused on mapping each image to one of small set of labels. However, the image segmentation task requires producing a more complicated label where an image of $n_r \times n_c$ pixels is mapped to $n_r \times n_c$ individual (correlated) pixel-labels; if each pixel-label is binary, this means that the output label $Y = \{+1, -1\}^{n_r \times n_c}$.

We illustrate the capability of active learning on image segmentation tasks, and show that our novel algorithm (LMU) produces an effective segmenter using very few segmented images.

Our second contribution in active learning is to make active learning very sample-efficient, by drawing instances from some appropriate distribution. Here we consider that instances are not restricted to be in a prior pool of instances. There are some cases where existing active learning algorithms do not work effectively. For example, some active learning algorithms [25]–[27] only consider classifiers within the version space (set of all classifiers consistent with current labeled set) throughout the learning; this carries the risk of eliminating the best hypothesis in class, especially if the data is noisy. In addition, some active learning algorithms [20], [28], [29] sample from the underlying distribution, then use rejection sampling to accept only the instances matching the given criteria. This rejection sampling approach can force the learner to consider a large number of instances, in order to find an instance that matches the given criteria. For example, a medical research task may require learning how much (of a certain) pathogenic bacteria a rat can tolerate before it gets sick. One can find this threshold by sequentially exposing various rats to differing amounts of this bacteria and observing their symptoms. Here, "sampling an instance" requires exposing a rat to certain amount of pathogenic bacteria, which is a costly and time consuming task. The naïve approach would just sample the space of "bacteria quantity" uniformly. But if the data is noise-free, we can perform fewer experiments by cleverly specifying the amount of bacteria to the i-th rat, based on the observed response by the previous $i - 1$ rats, to effectively estimate the maximum amount of bacteria that a rat's body can tolerate without causing a disease [30].

We present the importance sampling active learning (*ISAL*) algorithm, which addresses all these issues by (1) not selecting only classifiers that are consistent with all of the labeled instances, and (2) directly sampling from some appropriate distributions, which can allow *ISAL* to request the label of relatively few instances. As described in Section 5.2, on each iteration, *ISAL* determines an appropriate sampling distribution q_i, draws an instance x_i from q_i and requests x_i's label. *ISAL* uses importance weights to reduce the error estimation bias caused by this importance sampling, while controlling the variance of error estimation.

1.4 Thesis Statements

This dissertation proposes machine learning algorithms and analysis to support the following theses:

i. In producing a fixed-depth decision tree, the OPTNBDT algorithm produces a decision tree that is optimal given the Naïve Bayes assumption and is provably much more efficient than the standard optimal decision tree learner that must exhaustively consider all the features in the last level of the tree.

ii. In the image segmentation task, LMU active learner is likely to produce an accurate segmentor using fewer labeled images than random sampling, by considering the relevance or representativeness of query images, assuming that there exist images in the dataset that are noticeably more informative than others about the relation between labels and image features.

iii. In an active learning task where new instances can be generated based on the specification of the active learner, on data that is perfectly separable by a linear classifier the (expected) sample and label complexity of importance sampling active learning algorithm (*ISAL*) is provably more efficient than those of passive learning, and that of the state-of-the-art active learners by Beygelzimer *et al.* [20] and Balcan *et al.* [28].

1.5 Outline

In Chapter 2, we present our near optimal fixed-depth decision tree algorithm, OPTNBDT. Section 2.1 surveys the relevant literature. Section 2.2 summarizing our OPTNBDT algorithm and proves the relevant theories. Section 2.3 presents empirical results that validates our approach, by applying it to the standard datasets from the UCI repository [24] and elsewhere. We find that our approach significantly improves the computational cost of finding a fixed-depth tree, surprising at little or no loss in accuracy.

In Chapter 3, we study general approaches to active learning and describe the strategies that are commonly used to query new instances.

Chapter 4 then presents our first active learning algorithm, LMU. It describes how the underlying segmentation system — here Conditional Random Field (CRF) — segments the images. Section 4.3 shows the results of the experiments using LMU active learner on two real-world datasets: Finding the sky in the Geometric Context dataset [31] and

segmenting tumors within MR images of human brains [32]. In particular, we compare the segmentation performance (on hold-out images) of a segmenter trained on all labeled images, versus one trained using only the first k images selected by our active learner, for various values of k. We also consider segmenters learned from k randomly-selected images. We find that the segmenter based on only $k = 2$ well-selected images is as good as the one based on *all* 85 images in the pool. But only if the images are well-selected; the segmenter based on $k = 2$ *random* images is typically considerably inferior, as is one based on a larger number of random images.

Chapter 5 presents importance sampling active learning algorithm (*ISAL*). We describe the *ISAL* algorithm in Section 5.2 and examine its convergence rate for (1) perfectly separable one-dimensional data (Section 5.2.1), (2) perfectly separable, uniformly distributed multi-dimensional data (Section 5.2.2), and (3) uniformly distributed non-realizable data (Section 5.2.4). In Section 5.3, we prove the consistency of *ISAL* using sequential Rademacher complexity [33]. In addition, in Section 5.3, we obtain an upper bound for variance of error estimation by *ISAL*. Experimental results for realizable and non-realizable settings are reported in Section 5.4. In addition, we discuss the extension of *ISAL* to the pool-based active learning where *ISAL* only queries instances from a pool of unlabeled data.

Chapter 2

Near Optimal Fixed-Depth Decision Tree

In this chapter, we introduce the OPTNBDT [34] algorithm. [1] Section 2.1 surveys the relevant literature. Section 2.2 summarizing our OPTNBDT algorithm and proves the relevant theories. Section 2.3 evaluates the performance of the algorithm on some real-world datasets.

2.1 Literature Review

We study the "standard" learning model [35] where the input to the learner is a set of labeled training instances and the output is an accurate classifier. Many projects use decision trees due to their simplicity, and interpretability [36]. There are many algorithms for learning decision trees. Many algorithms, including C4.5 [2] and CART [3], begin with a greedy method that incrementally identifies an appropriate feature to test at each point in the tree, based on some heuristic score. These algorithms grow the tree from the root down to a set of leaves, then perform a post-processing step to reduce overfitting. In our context of "shallow" decision trees, overfitting is not as big a concern, which explains why many of the algorithms including OPTNBDT that seek "fixed-depth" decision trees simply return the tree that best fits the data [4], [37], [38]. These algorithms can easily be extended to allow different tests to have different costs [39], [40], where the total cost of each path from root to leaf is bounded. Our OPTNBDT algorithm focuses on finding the best fixed-cost policy, by producing the fixed-depth decision tree that best matches the data sample.

[1] A version of this chapter has been published in the proceedings of the ISAIM2008 conference.

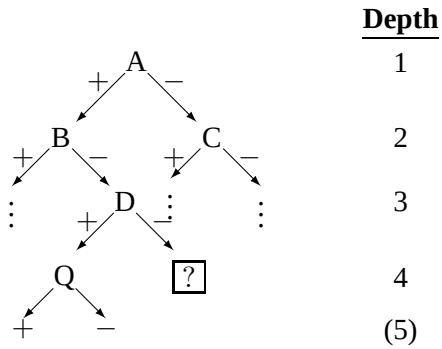

Figure 2.1: Decision Tree T_1, as it is being built.

There are some algorithms that consider developing a decision tree given a cost constraint on each path from the root to the leaf. In one of the earliest works, Turney [39] discusses the general challenge of learning the best classifier subject to some explicit cost constraint. Here, our "max depth" requirement corresponds to a constraint on the cost that the *classifier* must pay to see the features at *performance time*. There are some algorithms [41]–[43] that use greedy approaches to tackle this problem. However, these greedy approaches may not yield a decision tree with the optimal classification accuracy. On the other hand, the OPTNBDT algorithm is a fast way to produce the optimal fixed-depth decision tree given the Naïve Bayes assumption. (See our empirical results in Section 2.3)

Optimal decision tree algorithms in general require $O((n_f)^d)$ time to find the best depth-d decision tree over n_f variables. While this is a polynomial-time complexity for fixed d, in practice it is not effective except for small n_f and tiny d. The OPTNBDT algorithm addresses this problem by producing the optimal depth d decision tree in time that is essentially exponential in $d-1$, if the Naïve Bayes assumption [5] holds— *i.e.*, this Naïve Bayes assumption leads to a more efficient process.

There is, of course, a large body of work on building classifiers based on Naïve Bayes systems [44], [45], and on analyzing and characterizing their classification performance [46]. Those results, however, differ significantly from our task, which explicitly involves learning a *decision tree*; we use only the Naïve Bayes *assumption* in modeling the underlying distribution over instances. (But see the empirical comparisons in Section 2.3.)

2.2 OPTNBDT Algorithm

2.2.1 Foundations

Assume there are n_f features $\mathcal{F}_s = \{F^{(1)}, \ldots, F^{(n)}\}$ where each feature $F^{(j)}$ ranges over the $r^{(j)} \leq r$ values $\mathcal{V}_{F^{(j)}} = \{v^{(j)}_{f_1}, \ldots, v^{(j)}_{f_{r^{(j)}}}\}$, and there are two classes $\mathcal{Y} = \{+, -\}$.

A "decision tree" T is a directed tree structure, where each internal node ν is labeled with a feature $F(\nu) \in \mathcal{F}_s$, each leaf $l \in \mathcal{L}(T)$ is labeled with one of the classes $y(l) \in \mathcal{Y}$, and each arc descending from a node labeled F is labeled with a value $v_f \in \mathcal{V}_F$. We can evaluate such a tree T on a specific instance $\mathbf{f} = \{F^{(j)} = v^{(j)}_{f_i}\}_j$, to produce a value $\textsc{Val}(T, \mathbf{f}) \in \mathcal{Y}$ as follows: If the tree is a leaf $T = l$, return its label $y(l) \in \mathcal{Y}$; that is, $\textsc{Val}(T, \mathbf{f}) = y(l)$. Otherwise, the root of the tree is labeled with a feature $F \in \mathcal{F}_s$. We then find the associated value within the $\mathbf{f}$ (say $F = v_f$), and follow the v_f-labeled edge from the root to a new subtree; then recur. The value of $\textsc{Val}(T, \mathbf{f})$ will be the value of that subtree. Hence, given the instance $\mathbf{f} = \{A = +,\ B = -,\ \ldots\}$ and the tree T_1 in Figure 2.1, the value of $\textsc{Val}(T_1, \mathbf{f})$ will be the value of the subtree rooted in the B-labeled node on this instance (as we followed the $A = +$ arc from the root), which in turn will be the value of the subsubtree rooted in the D-labeled node (corresponding to the subsequent $B = -$ arc), etc. We say a decision tree T is "correct" for a labeled instance $\langle\, \mathbf{f}, y \,\rangle$ if $\textsc{Val}(T, \mathbf{f}) = y$. A labeled dataset is a set of labeled instances $S = \{\langle\, \mathbf{f}^{(j)}, y^{(j)} \,\rangle\}_j$. Our goal is to find the depth-d decision tree with the maximum expected accuracy, given our posterior beliefs, which are constructed given a Naïve Bayes prior and the dataset S. To define expected accuracy, we must first define the notion of a path $\pi(\nu)$ to any node ν in the tree, which is the sequence of feature-value pairs leading from the root to that node; hence, the path to the $\boxed{?}$ node in Figure 2.1 is $\pi(\boxed{?}) = \langle\, \langle\, A, + \,\rangle, \langle\, B, - \,\rangle, \langle\, D, - \,\rangle \,\rangle$. We can use this notion to define the probability[2] of reaching a node, which here is $P($ "reaching $\boxed{?}$" $) = P(\, \pi(\boxed{?})\,) = P(\, A = +,\, B = -,\, D = -\,)$.

In general, the accuracy of any tree T is

$$\text{Acc}(\, T\,) = \sum_{l \in \mathcal{L}(T)} P(\, \pi(l)\,) \times \text{Acc}(\, \pi(l)\,) \tag{2.1}$$

over the set of leaf nodes $\mathcal{L}(T)$. Note that this accuracy has been factored into a sum

[2] All probability values, as well as accuracy scores, are based on a posterior generated from a Naïve Bayes prior and the labeled data S. Note also that we will use various forms of $Acc(\cdot)$, for trees, paths and features appended to paths; here, the meaning should be clear from context.

of the accuracies associated with each leaf: it is the probability $P(\,\pi(l)\,)$ of reaching each leaf node $l \in \mathcal{L}(T)$ times the (conditional) accuracy associated with that node $\text{Acc}_{\pi(l)}(\,l\,) = P(\,Y = y(l)\,|\,\pi(l)\,)$. This factoring tells us immediately that we can decide on the appropriate class label for each leaf node, $y^*(l|\pi(l)\,) = \text{argmax}_y\, P(\,y\,|\,\pi(l)\,)$, with an associated accuracy of

$$\text{Acc}^*(\,\pi(l)\,) = \max_y P(\,y\,|\,\pi(l)\,) \tag{2.2}$$

based only on the single path; *i.e.*, it is independent of the rest of the tree.

We are searching for the "best" depth-d tree, $\arg\max_{T \in \text{DT}(d)} \text{Acc}(\,T,\,S)$, where $\text{DT}(d)$ is the set of decision trees of depth at most d — *i.e.*, each path from the root to any leaf involves at most d variables. An earlier system [47] precomputed the accuracy associated with each possible sequence of d tests (requiring $O(\binom{n_f}{d} r^d)$ time), and then constructed the best tree given these values, which required $O((rn_f)^d)$ time. Here we present a different technique that requires less time by precomputing a data structure that quickly provides the optimal feature given the Naïve Bayes assumption to use for each position in the bottom row.

To understand our approach, consider determining the feature $F(\nu)$ to use at the "final internal node" along a path $\pi(\nu)$ — *e.g.*, determine which feature to test at $\boxed{?}$ in Figure 2.1. As an obvious extension of the above argument, this decision will depend only on the path $\pi(\nu)$ to this node. Then for the $\boxed{?}$ node, it will depend on $\pi(\boxed{?}) = \langle\,\langle\,A, +\,\rangle, \langle\,B, -\,\rangle, \langle\,D, -\,\rangle\,\rangle$. Given our Naïve Bayes assumption, for any feature $F \in \mathcal{F}_s$ (except the features in π — *i.e.*, except for A, B, and D), the component of accuracy associated with the path that begins with π then performs F (and then descends to the leaf nodes immediately under this F), is

$$\begin{aligned}
\text{Acc}(\,\pi \circ F\,) &= \sum_{v_f \in \mathcal{V}_F} P(\,\pi,\, F = v_f\,) \times \text{Acc}^*(\,\pi \circ F{=}v_f\,) \\
&= \sum_{v_f \in \mathcal{V}_F} P(\,\pi,\, F{=}v_f\,) \times \max_y P(\,Y{=}y\,|\,\pi,\, F{=}v_f\,) \\
&= \sum_{v_f \in \mathcal{V}_F} \max_y P(\,Y = y,\, \pi,\, F = v_f\,) \\
&= \sum_{v_f \in \mathcal{V}_F} \max_y P(\,Y = y\,)\, P(\,\pi\,|\,Y = y\,) P(\,F = v_f\,|\,Y = y\,) \qquad (2.3) \\
&= P(\,\pi\,) \sum_{v_f \in \mathcal{V}_F} \max_y P(\,Y{=}y\,|\,\pi\,) P(\,F{=}v_f\,|\,Y{=}y\,)
\end{aligned}$$

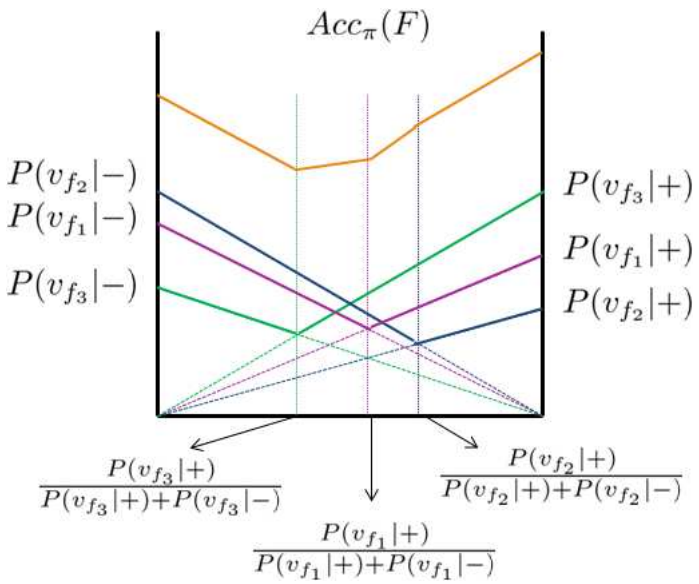

Figure 2.2: Computing $\text{Acc}_\pi(F)$ (Equation 2.4).

where Equation 2.3 is based on the Naïve Bayes assumption. Note that the feature F that optimizes $\text{Acc}(\pi \circ F)$ will also optimize

$$\begin{aligned}\text{Acc}_\pi(F) &= \frac{1}{P(\pi)}\text{Acc}(\pi \circ F) \\ &= \sum_{v_f \in \mathcal{V}_F} \max_y P(Y=y \mid \pi) P(F=v_f \mid Y=y).\end{aligned}$$

While our analysis will work for classes that range over any (finite) number of values, this section will focus on the binary case. Letting $x_{\pi,+} = P(Y=+ \mid \pi)$ and abbreviating "$F=v_f$" as "v_f", we have

$$\begin{aligned}\text{Acc}_\pi(F) &= \sum_{v_f \in \mathcal{V}_F} \max \left\{ \begin{array}{c} x_{\pi,+}\, P(v_f \mid +) \\ (1-x_{\pi,+})\, P(v_f \mid -) \end{array} \right\} \\ &= \sum_{v_f \in \mathcal{V}_F} \left\{ \begin{array}{ll} x_{\pi,+} P(v_f \mid +) & \text{if } x_{\pi,+} \leq \frac{P(v_f \mid -)}{P(v_f \mid +)+P(v_f \mid -)} \\ (1-x_{\pi,+}) P(v_f \mid -) & \text{otherwise.} \end{array} \right.\end{aligned} \tag{2.4}$$

For fixed values of $P(F=v_f \mid Y=y)$, *this* $\text{Acc}_\pi(F)$ *value does not depend on the features in* π *nor their values, but only on* $x_{\pi,+}$; we can therefore express $\text{Acc}_\pi(F)$ as $\text{Acc}(F, x_{\pi,+})$ to make this dependency explicit. For any value of $x_{\pi,+} \in [0,1]$, each summand in Equation 2.4, corresponding to a single $F = v_f$, is the maximum of two lines. Over the set of r_F values, this function is therefore a sequence of at most $1 + r_F$ linear segments; see Figure 2.2.

Before beginning to build the actual decision tree, our OPTNBDT algorithm will first compute these $\text{Acc}(F^{(j)}, x)$ functions for each $F^{(j)}$. It then uses these functions to

OPTNBDT(d: int; $P(\cdot)$: distribution over $\{Y\} \cup \mathcal{F}_s$)
  Compute *opt-feature list*
    OFL = $\{F_i^*(x) \mid x \in [0,1],\ i = 1, \ldots, d\}$
  Build optimal depth-$d-1$ tree using Dynamic Programming
    At each length-$d-1$ path π,
      with associated probability $x_{\pi,+} = P(\, Y = + \,|\, \pi\,)$
      Let $\vec{\mathcal{F}} = \langle\, F_1^*(x_{+,\pi}), \ldots, F_{d-1}^*(x_{+,\pi})\,\rangle$.
      Let $i^* = \min_i\{F_i^*(x_{+,\pi}) \notin \pi\}$ be the first in $\vec{\mathcal{F}}$ *not* in π
      Use feature $F_{i^*}^*(x)$ at level d, after π

Figure 2.3: OPTNBDT algorithm.

compute, for each $x \in [0,1]$, the optimal feature:

$$F_1^*(x) \;=\; \operatorname*{argmax}_F\{\text{Acc}(\,F,\,x)\}. \tag{2.5}$$

It also computes the *2nd-best* feature $F_2^*(x)$ for each x value, as well as $F_i^*(x)$ for $i = 3, 4, \ldots, d$. We refer to this $\{F_i^*(x) \mid x \in [0,1],\ i = 1, \ldots, d\}$ dataset as the OFL ("opt-feature list").

The OPTNBDT algorithm then uses a dynamic program to build the tree, but only to depth $d-1$. Whenever it reaches a depth $d-1$ node, at the end of the path $\pi = \langle\, \langle\, F_{\pi(1)}, v_{f_{\pi(1)}}\,\rangle, \langle\, F_{\pi(2)}, v_{f_{\pi(2)}}\,\rangle\, , \ldots, \langle\, F_{\pi(d-1)}, v_{f_{\pi(d-1)}}\,\rangle\,\rangle$, with associated conditional probability $x_{\pi,+}$, it then indexes this $x_{\pi,+}$ into the OFL, which returns an ordered list of d features, $\vec{\mathcal{F}}(x_{+,\pi}) = \langle\, F_1^*(x_{+,\pi}), \ldots, F_d^*(x_{+,\pi})\,\rangle$. OPTNBDT then returns the first F_i that is *not* in the $\vec{\mathcal{F}}(x_{+,\pi})$ list. This algorithm is shown in Figure 2.3.

To make this more concrete, imagine that in Figure 2.1 the list associated with $\pi(\boxed{?}) = \langle\, \langle\, A, + \,\rangle, \langle\, B, - \,\rangle, \langle\, D, - \,\rangle\,\rangle$ and $x_{\pi,+} = 0.29$ was $\vec{\mathcal{F}}(0.29) = \langle\, B, A, E, D\,\rangle$. While we would like to use the first value $F_1^*(0.29) = B$ as it appears to be the most accurate, we cannot use it as this feature has already been appeared in this π path and therefore Equation 2.4 does not apply. OPTNBDT would then consider the second feature, which here is $F_2^*(0.29) = A$. Unfortunately, as that appears in π as well, OPTNBDT have to consider the third feature, $F_3^*(0.29) = E$. Since that does not appear, OPTNBDT labels the $\boxed{?}$ node with this "E" feature.

2.2.2 Implementation of OPTNBDT

This section provides the details of our implementation, which requires recursively computing $x = x_{\pi,+}$, and computing and using the opt-feature list, OFL.

Computing $x_{\pi,+}$: It is easy to obtain $x_{\pi,+}$ as we grow the path π in the decision tree. Here, we maintain two quantities, $p^+_\pi = P(\,\pi, Y = +\,)$ and $p^-_\pi = P(\,\pi, Y = -\,)$, then for any path π, set $x_{\pi,+} = \frac{p^+_\pi}{p^+_\pi + p^-_\pi}$. For $\pi_0 = \{\}$, $p^y_{\pi_0} = P(\,Y = y\,)$ for $y \in \{+, -\}$. Now consider adding one more feature-value pair $\langle\, F, v_f \,\rangle$ to π_t, to form $\pi_{t+1} = \pi_t \circ \langle\, F, v_f \,\rangle$. Then thanks to our Naïve Bayes assumption, $p^y_{\pi_{t+1}} = p^y_{\pi_t} \times P(\,F = v_f \,|\, Y = y\,)$.

Computing and using the OFL**:** As noted above, OFL corresponds to a set of *piecewise linear functions* (see Figure 2.2), each of which is the union of a finite number of linear functions. Formally, a piecewise linear function $f : [0, 1] \to \mathbb{R}$ (with k pieces) can be described by a sequence of real number triples $\langle\, (a_1, m_1, b_1), \ldots, (a_k, m_k, b_k) \,\rangle$ where the a_is are endpoints such that $0 = a_0 < a_1 < \ldots < a_k = 1$, and for all $i \in \{1, \ldots, k\}$ we have $f(x) = m_i x + b_i$ for all $x \in [a_{i-1}, a_i]$.

A linear function is a piecewise linear function with one piece. The sum of two piecewise linear functions with k_1 and k_2 pieces is a piecewise linear function with no more than $k_1 + k_2 - 1$ pieces. We can compute this sum in $O(k_1 + k_2)$ time, as each component of the sum $f = f_1 + f_2$ is just the sum of the relevant m and b from both f_1 and f_2. Similarly, the maximum of two piecewise linear functions with k_1 and k_2 pieces is a piecewise linear function with no more than $(k_1 + k_2)$ pieces. This computation is slightly more involved, but can be done in a similar way.

For each feature F and each value $x_\pi \in [0, 1]$, we need to compute the sum over all $|\mathcal{V}_F|$ values of F using the Equation 2.4. We compute this total by adding the associated $|\mathcal{V}_F| \leq r$ piecewise linear functions, $\{\text{Acc}_\pi(\,F = v_f\,)\}_{v_f \in \mathcal{V}_F}$. We can do this recursively: Letting $r = |\mathcal{V}_F|$ and $acc_i = \text{Acc}_\pi(\,F = f_i\,)$ be the i^{th} function, we first define $acc_{1,2} = acc_1 + acc_2$ as the sum of acc_1 and acc_2, $acc_{3,4} = acc_3 + acc_4$, and so forth until $acc_{r-1,r} = acc_{r-1} + acc_r$; we next let $acc_{1,4} = acc_{1,2} + acc_{3,4}$, $acc_{5,8} = acc_{5,6} + acc_{7,8}$, etc.; continuing until computing $acc_{1,r}$ which is the sum over all r functions. By recursion, one can prove that this resulting piecewise linear function has no more than $r + 1$ pieces, and that the time complexity is $O(r \log r)$, due to $\log r$ levels of recursion, each of which involves a total of $O(2^i \times r/2^i) = O(r)$ time.

Equation 2.5 is the maximum of all of the piecewise linear functions $\{\text{Acc}_\pi(\,F\,)\}_{F \in \mathcal{F}_s}$ that were constructed in the previous step. We again use a divide and conquer technique to reduce the problem of maximizing over n_f piecewise linear functions to sequence of $\log n_f$ problems, each of maximizing over two piecewise linear functions. An analysis similar to mergesort shows that the overall computational cost of the process is

$O(r\,n_f\,\log(r\,n_f))$.

When $F_1^*(\cdot)$ (Equation 2.5) involves $O(r\,n_f)$ linear pieces at arbitrary points, we can compute $F_1^*(x)$ in $O(\log(r\,n_f))$ time, using a binary search to find the appropriate segment and a constant amount of time to compute the value given this segment. As we are computing this maximum value, we can also specify which feature represents this segment.

We can compute $F_i^*(x)$ for $i = 2, 3, \ldots, d$ in a similar way. Consequently, when we compute the maximum of two functions, we also store the d highest functions at every point. Note that the amount of memory storage required here increases linearly in d.

Hence, each lookup function can be constructed in $O(r\,n_f \log(r\,n_f))$ time, requires $O(r\,n_f)$ memory to store, and most importantly, can be queried (to find the optimal value for a specific x) in $O(\log(r\,n_f))$ time. Note that a naïve implementation that merely tests all the features in the last level of the tree to construct the leaves (called "OPTDT" below) will require a time linear in $r\,n_f$, whereas our technique has complexity logarithmic in $r\,n_f$.

Collectively, these results show the following theorem:

Theorem 1. *Given any labeled dataset S over n_f r-ary variables and binary class labels, the* OPTNBDT *algorithm of Figure 2.3 will compute the depth-d decision tree that has the highest classification accuracy under the Naïve Bayes assumption. Moreover, it requires $O(n_f|S| + (r\,n_f)^{d-1} d \log(r\,n_f))$ time and $O(d(r\,n_f)^d)$ space.*

2.3 Empirical Results

We performed an extensive set of experiments to compare the performance of OPTNBDT to various learners, both state-of-the-art decision tree learners, and Naïve Bayes systems. This section first describes the datasets and experimental setup, then the experimental results and finally our analyses.

Experimental setup: The experiments are performed using nine datasets from UCI Machine Learning Repository [24] as well as our own Prostate Cancer dataset obtained from our colleagues at the Cross Cancer Institute. All the selected datasets have binary class labels. Table 2.1 characterizes these datasets.

The experiments are performed using 5-fold cross validation, performed 20 times; we report the average error rate and standard deviation across these trials. We compare the

Table 2.1: Datasets used in the fixed-depth decision tree experiments.

Dataset	Number of features	Number of instances	Number of feature values	Abbre-viation
Tic tac toe	10	985	27	Tic
Flare	10	1066	31	Flr
Hepatite	13	80	26	Hep
Crx	13	653	30	Crx
Vote	16	435	48	Vot
Lymphography	18	145	50	Lym
Chess	36	3196	73	Chs
Connect-4	42	5000	126	Con
Prostate cancer	47	81	126	Prs
Promoters	58	106	228	Prm

classification performance and running time of OPTNBDT with different depths — here 2,3, and 4 — to the following fixed-depth decision trees:

- 3-ID3 [48] is a decision tree learning algorithm that uses an entropy based measure as its splitting criterion, but stops at depth 3.
- 1-OPTDT [37] is a decision stump that splits the data with only one feature — *i.e.*, it is a depth-one decision tree
- 2-OPTDT is an optimal depth-2 decision tree
- 3-OPTDT is an optimal depth-3 decision tree.

Experimental results: Figure 2.4 shows the average classification error rate and the error bar corresponding to 95% confidence interval of each algorithm. We see that no algorithm is consistently superior over all the datasets. While deeper decision trees sometimes improve classification performance (classification accuracy increases with the depth of the tree for the `Lym`, `Chs`, and `Prm` datasets), deeper trees can cause overfitting, which may explain why shallower trees can be better: Decision stumps, which only split the data based on one feature, outperform the other trees for the `flr`, `crx`, and `vot` datasets.

Figure 2.4 shows that the classification error of 3-OPTNBDT is often lower than the errors from 3-ID3, which shows that the optimal Naïve Bayes-based decision trees can perform better than heuristic, entropy-based trees. In addition, Figure 2.4 shows that the classification error of 3-OPTNBDT is often similar to 3-OPTDT except for `Prm`

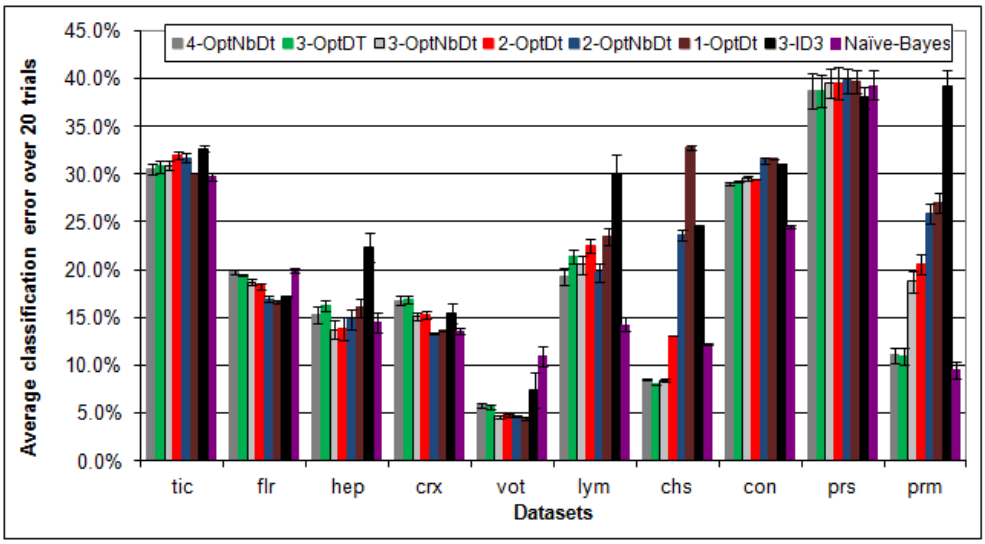

Figure 2.4: Average classification error for 4-OPTNBDT, 3-OPTNBDT, 2-OPTNBDT, 1-OPTDT, 2-OPTDT, 3-OPTDT, Naïve Bayes, and 3-ID3 algorithms.

dataset, which implies that Naïve Bayes assumption does not hurt the performance of 3-OPTNBDT most of the time. In fact, a paired t-test over these ten datasets show that 3-OPTNBDT is statistically better (more accurate) than 3-ID3 (at $p < 0.05$). However, the difference between 3-OPTNBDT and 3-OPTDT is not statistically significant (at $p < 0.05$). Similarly, the difference between 2-OPTNBDT and 2-OPTDT is not statistically significant (at $p < 0.05$). As expected, we found that (on average) the error improves for deeper trees, although this improvement is not statistically significant (at $p < 0.05$): 4-OPTNBDT (0.194) $<$ 3-OPTDT (0.197) $<$ 3-OPTNBDT (0.200) $<$ 2-OPTDT (0.209) $<$ 2-OPTNBDT (0.222) $<$ 1-OPTDT (0.235) $<$ 3-ID3 (0.257).

Figure 2.4 also includes the results of the Naïve Bayes classifier. While it works fairly well in many of these datasets, note that it is not a contender: recall that our goal is to produce an effective shallow decision tree, which involves only a (small) subset of the features, probed sequentially. However, the Naïve Bayes classifier will use all of the feature values, which must all be available.

Running Time: Now consider the running time of these algorithms, as well as 3-OPTDT, which implements the dynamic programming algorithm that produces the fixed-depth decision tree that is optimal over the training data, given the Naïve Bayes assumption — *i.e.*, this d-OPTDT algorithm explicitly constructs and tests all possible leaves up to depth d, whereas d-OPTNBDT uses a pre-computed opt-feature list (OFL) to efficiently

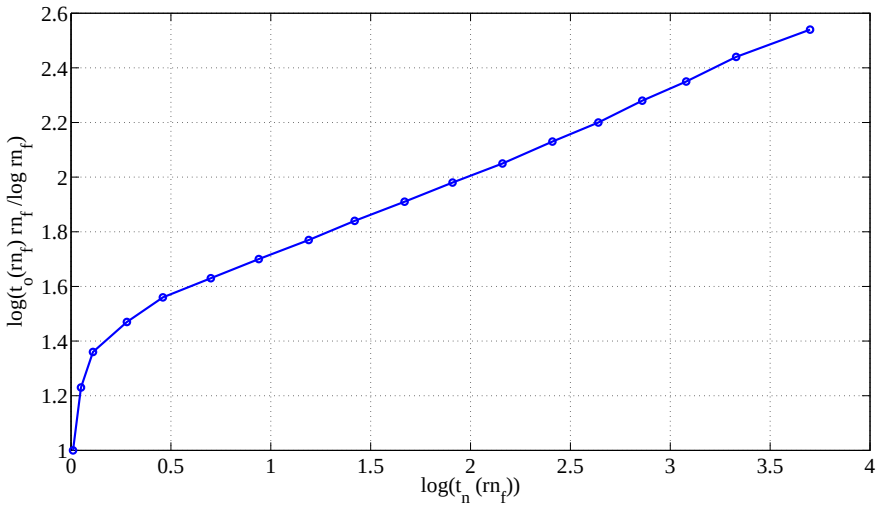

Figure 2.5: Comparing 3-OPTNBDT versus 3-OPTDT: $t_o(r\,n_f)$ and $t_n(r\,n_f)$ are the time that 3-OPTNBDT and 3-OPTDT require for n_f features, respectively.

construct the final (depth d) level of the decision tree.

The previous section proved that OPTNBDT is $O(\log(r\,n_f)/r\,n_f)$ times faster than OPTDT, where n_f is the number of features. To explore this claim empirically, we extracted eighteen artificial datasets from the Promoters (`Prm`) dataset, using $3i$ features, for $i = 1 : 18$. Figure 2.5 plots $\log(t_o(r\,n_f) \times r\,n_f/\log r\,n_f)$ versus $\log(t_n(r\,n_f))$, where $t_o(r\,n_f)$ (respectively, $t_n(r\,n_f)$) is the run-time that 3-OPTNBDT (respectively, 3-OPTDT) requires for n_f features. This confirms that OPTNBDT is significantly more efficient than OPTDT, especially when there are many features in the dataset.

Figure 2.6 shows the running time of the various algorithms: Each line corresponds to one of the algorithms, formed by joining a set of points whose value on the horizontal axis is the total number of feature values $\sum_i r_i = O(r\,n_f)$ of a particular dataset, and the value on the vertical axis is the (log of the) run time of the algorithm on that dataset. As expected, the 3-ID3 points are relatively independent of the number of features, although it depends on the number of instances in the training set; this explains the higher running time of ID3 on `con` and `chs` datasets. The other lines, however, appear fairly linear in this log plot, at least for larger values of $\sum_i r_i$; this is consistent with the Theorem 1. Finally, to understand why the 4-OPTNBDT timing numbers are virtually identical to the 3-OPTDT numbers, recall that 4-OPTNBDT basically runs 3-OPTDT to produce a depth-3 decision tree, then uses the OFL to compute the 4th level. These numbers show that the

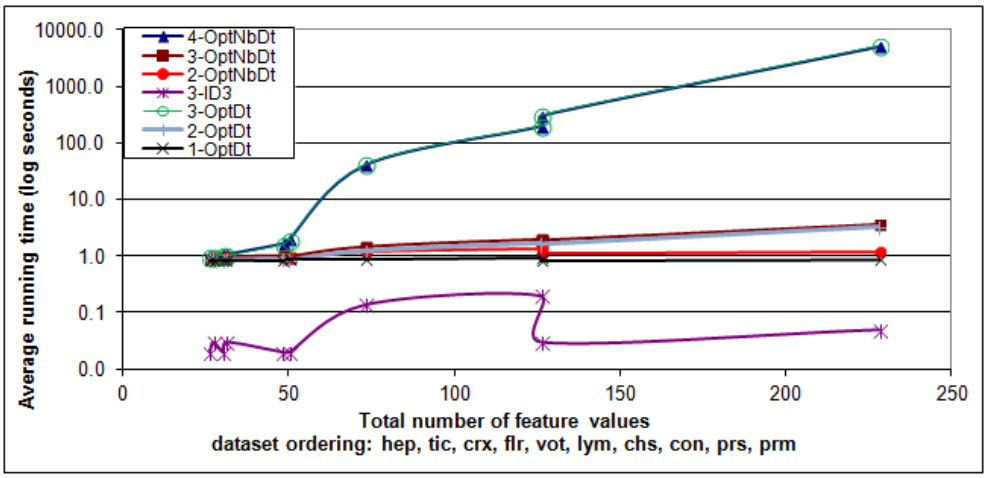

Figure 2.6: Run-time (log of seconds) of 4-OPTNBDT, 3-OPTNBDT, 2-OPTNBDT, 3-OPTDT, 2-OPTDT, 1-OPTDT, and 3-ID3 algorithms.

time required to compute OFL, and to use it to produce the final depth, is extremely small.

2.4 Summary

There are many situations where we need to produce a *fixed-depth decision tree* — *e.g.*, the bounded policy associated with per-patient capitation. This chapter has presented OPTNBDT, an algorithm that efficiently produces the optimal such fixed-depth decision tree, given the Naïve Bayes assumption — *i.e.*, assuming that the features are independent given the class. We proved that this assumption implies that the optimal feature at the last level of the tree essentially depends only on $x_{\pi,+} \in [0, 1]$, the posterior probability of the class label given the tests previously performed. We then describe a way to efficiently pre-compute which feature is best, as a function of this probability value, and then, when building the tree itself, to use this information to quickly assign the final feature on each path. We show that this results in a speedup of $O(n_f / \log n_f)$ compared to the naïve method of testing all the features in the last level, then provide empirical evidence, over a benchmark of ten datasets, supporting this claim. We also found that OPTNBDT is not just efficient, but (surprisingly, given its Naïve Bayes assumption) is often more accurate than entropy based decision tree learner like ID3.

Chapter 3

Active Learning Strategies

Active learning algorithms use various methods to query unlabeled instances. This section describes the query strategies and scoring functions that are commonly used in the literature.

3.1 Estimated Uncertainty Reduction

In this section we discuss query strategies that attempt to reduce the overall uncertainty of the model. In this strategy, the active learner computes the expected uncertainty/error of the model, resulted from querying the unlabeled instance x. It then queries the instance that causes the minimum expected uncertainty in other unlabeled instances. For example, Roy and McCallum [49] selected instances that reduce the expected error of the Naïve Bayes classifier over unlabeled instances. Guo and Greiner [50] used an algorithm that selects the instance that provides the maximum conditional mutual information about the labels of unlabeled instances. Our LMU algorithm extends the approach by Guo and Greiner [50] to the structured data (with conditional random fields as the classifier) in order to use it for its initial step of active learning.

Uncertainty reduction strategy has been successfully applied to several machine learning models including Naïve Bayes [49], Gaussian random fields [51], logistic regression [50], and support vector machines [52]. The shortcoming of this strategy is its computational cost, since it needs to consider not only every unlabeled instance in the pool, but also retrains the model for every possible value of the label. For some models, like Gaussian random fields, the incremental training procedure is fairly simple and practical, but for many others, the whole training must be repeated for the entire pool of unlabeled instances. Another attempt to reduce the computational cost of the uncertainty reduction

strategy involves subsampling the pool of unlabeled instances [49]. Alternatively, Guo and Greiner [50] only consider few iterations towards the optimum model parameter in their optimization process. Because of the high computational costs of this strategy, we only use it in the initial step of LMU (see Section 4.2).

3.2 Uncertainty Sampling

In uncertainty sampling, an active learner queries the unlabeled instance that has the highest uncertainty. There has been many suggestions in the literature for ways to measure the uncertainty of an instance. For example, in binary classification tasks using a probabilistic model, the most uncertain instance is the one whose probability of being positive is closest to 0.5 [10].

A more general criteria to determine the uncertainty of an unlabeled instance x is conditional entropy, defined as:

$$H(y|x,\theta) = -\sum_i P(y_i|x,\theta) \log P(y_i|x,\theta), \tag{3.1}$$

where θ is the vector of model parameters, and y_i are the possible labels for the instance. Instances with large conditional entropy have high uncertainty. For the binary classification task, highest conditional entropy happens with $P(y_i|x,\theta)$ being at 0.5. In multiclass scenario, $P(y_i|x,\theta)$ with uniform distribution (*i.e.*, all $P(y_i|x,\theta)$ are equal) yields the highest conditional entropy. Our LMU algorithm uses the entropy approach by extending the definition of entropy to structured data (Equation 4.12) and using conditional random fields to model $P(y_i|x,\theta)$.

Another approach, used in information extraction tasks [15], [53], is to query the instance whose most likely label $y_m = \arg\max_y P(y|x,\theta)$ has the smallest conditional probability $P(y_m|x,\theta)$.

Uncertainty sampling has also been applied to non-probabilistic models. Many researchers, working with support vector machines, select the instance closest to the boundary [12], [54], [55]. Lewis and Gale [10] have modified the outputs of a decision tree classifier to obtain a probabilistic model and use uncertainty sampling. For example, for a binary classification task, if the number of "$+$" instances within a branch is 3 and the "$-$" instances are 2, then $P(Y = +|x) = \frac{3}{5}$. Similar approach has been applied to nearest neighbor classifier, where the probability of each class is the proportion of neighbors

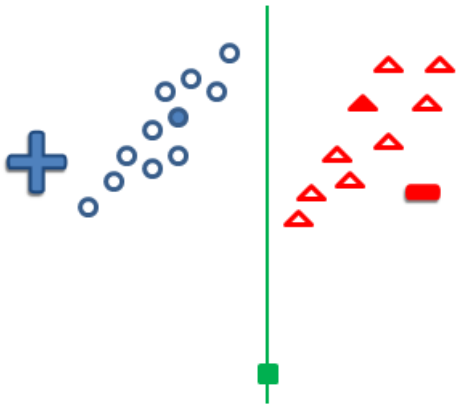

Figure 3.1: Selecting the square green point, which is the most uncertain data point, does not improve the classification performance of the linear classifier.

with similar class. Here for a binary classification task, if 3 out of 5 neighbors are $+$, then $P(Y = +|x) = \frac{3}{5}$.

The most uncertain approach typically works well if the current parameters nicely approximate the conditional probability distribution of the entire data. Unfortunately, these parameters are often problematic as they are based on a very small training set and sometimes they are prone to querying outliers [49]. Figure 3.1 shows a binary classification task where the data is separated by a linear classifier trained on two data instances, shown as the filled circle and filled triangle. Here the most uncertain instance located on the boundary line (shown as the green square) does not provide further information about the labels of unlabeled instances.

3.3 Entropy of Model Parameters

In Section 3.2, the conditional entropy of labels given the model parameters was used to query the unlabeled instances. Although this approach works well in practice, it highly depends on the inventory of the training data S_i at time i and the training procedure that results in θ_i^*. One may imagine a scenario where θ_i^* is not an optimal model parameter, e.g., as a result of a bad training data or simply being a local optimum. In this case the active learning algorithm would be misled by θ_i^* and we may end up querying uninformative instances in the next steps (like querying the square instance in Figure 3.1).

Therefore, we may consider a query strategy that does not rely on current model parameters. One query strategy is to compute the uncertainty of model parameters given the data, *i.e.*, $Uncertainty(\theta|D)$. This way we can compute the information content of

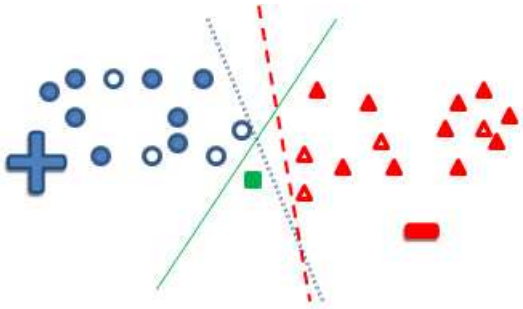

Figure 3.2: Query by committee strategy, here two of the three classifiers agree that the square data point has positive label.

the unlabeled data instances based on how uncertain the model parameters θ would be, after we train them on $D = (x, S_i)$, where x is the candidate point and S_i is the current set of labeled instances.

The uncertainty can be measured by computing the conditional entropy of the parameters given the data as:

$$Uncertainty(\theta|D) = H(\theta|D) = -\int_{-\infty}^{\infty} P(\theta|D) \log P(\theta|D) d\theta. \tag{3.2}$$

This definition of entropy is intended for continuous variables, whereas the previous definition of conditional entropy in Equation 3.2 is used for discrete data.

$H(\theta|D)$ is a measure of how uncertain the model parameters are after being trained on data D. Therefore, it can be used to query the instances that reduce the uncertainty of model parameters [56]. Based on our observations on some real-world datasets, reducing the uncertainty of model parameters may not result into a more accurate classifier.

3.4 Query by Committee

In this method, a committee (set) of classifiers are trained on the current labeled set, then they are used to classify all the unlabeled instances. Here, the unlabeled instance with the maximum disagreement among all classifiers is queried for labeling. One step of this active learning process is shown in Figure 3.2. Here solid, dashed and dotted lines represent three linear classifiers and the last two classifiers disagree with the solid line classifier on the label of the green square instance. Therefore, the green square instance is queried by the active learner.

In query by committee strategy (QBC), there is no general constraint on the size of the committee, as even small number of classifiers have been shown to work well [11],

[15], [57].

In one of the earliest works in active learning with QBC strategy, Cohn *et al.* [25] introduced selective sampling, which is a sequential process that updates two spaces sequentially – the current version space $\mathcal{H}_i$ (at ith iteration) that includes the set of hypotheses consistent with current training data, and the region of uncertainty R_i, which is the subset of the instances where different hypothesis in $\mathcal{H}_i$ give different labels. In each iteration, the algorithm queries a random unlabeled instance from R_i to obtain its label. It then forms $\mathcal{H}_{i+1}$ by eliminating all the hypotheses in $\mathcal{H}_i$ that are inconsistent with this newly labeled instance. The algorithm then forms R_{i+1} by removing from R_i any region if all of the hypotheses in $\mathcal{H}_{i+1}$ agree on the label of that region.

Balcan *et al.* [26] introduced the A^2 agnostic active learner, which uses thresholds for the error rate. At iteration i, let R_i be the region of uncertainty. A^2 queries a set of instances D_i from R_i and computes the lower bound $LB(D_i, h)$ and upper bound $UB(D_i, h)$ on the true error rate $L(h)$ for all the hypotheses in the version space. It then eliminates all the hypotheses whose lower bound is greater than the minimum upper bound. Iteration i is complete when enough samples D_i are gathered to eliminate at least half of the current region of uncertainty. Similar to selective sampling methods, if all hypotheses in the version space agree on some region of the instance space, this region can be eliminated from R_i.

Dasgupta *et al.* [27] proposed an algorithm that maintains two sets of labeled instances, S and S', where S contains instances that receive different labels from two minimum error hypotheses and this difference is less than a given threshold. Therefore, the instances in S are labeled by the oracle. If the difference between the empirical error of the minimum error hypotheses is greater than the given threshold, then the instances are in set S' and their labels are inferred. Note that S' can be viewed as the set of constraints, *i.e.*, in the next iteration we only consider classifiers that are consistent with the label of instances in S'.

Each of the algorithms in [25], [26] and [27] only considers classifiers that are consistent with the current labeled instances ("version space"), which is updated as more instances are labeled. If there is noise in the data, these methods might not produce the optimal classifier since there is the risk of eliminating the optimal classifier if it is not consistent with the current labeled instances.

Another group of algorithms [20], [28], [29] as well as *ISAL*, avoid using a version

space in their process. Balcan *et al.* [28] showed that for the margin-based classifiers with linear separators and perfectly separable data, active learning can obtain an exponential convergence rate if it only queries instances that are inside the margin. They used a rejection sampling strategy to query such instances.

Some active learning algorithms including *ISAL* use an unbiased estimation of error in their process. The recent works by Beygelzimer *et al.* [20] and Ganti *et al.* [29] use importance weights to cancel the estimation bias of the error. Those algorithms use rejection sampling to query the desired instances by assigning a non-zero weight to them. They also use an unbiased estimation of error when selecting the next instance to query. In contrast, *ISAL* samples directly from some appropriate distributions. Similarly, *ISAL* also uses an unbiased estimation of error when selecting the next sampling distribution by assigning a weight proportional to the ratio between the sampling distribution and the true distribution.

Chapter 4

Active Learning With Structured Data

As noted in the previous section, most of the existing active learning systems have been used primarily to learn a classifier that maps each instance to a *single (scalar) label*; even the imaging work mentioned above has focused on mapping each image to one of small sets of labels Y (object recognition). As images can often be recognized based on only a small set of extracted features, object recognition corresponds to a typical machine learning problem. However, many important learning problems involve predicting labels for structured data such as images. There have been relatively little active learning research related to *image segmentation*, which requires producing a more complicated label: such systems map an image of $n_r \times n_c$ pixels to $n_r \times n_c$ individual pixel-labels. Moreover, these pixel-labels are not independent of one another (*e.g.*, if one pixel is a tumor, it is more likely that its neighbors are, as well; see Figure 4.1). This forces a segmenter to consider the entire image ($n_r \times n_c$ pixels) as an instance, rather than label one pixel at a time. Hence, notions like uncertainty and information content must be defined for the entire image.

To label the structured data, probabilistic methods such as conditional random fields (CRF) [58] are commonly used. This section overviews our basic system for actively learning structured data with the focus on the image segmentation task. We first mention the training process for the underlying segmentation system, based on Conditional Random Fields (CRFs), that is designed to deal with variables that are organized in a 2-dimension grid [59]. We then describe our LMU system, which actively learns the parameters of the CRF.

We let $\mathbf{I}$ represent the set of $n_r \times n_c$ image pixels, $\mathbf{x} = \{x_i \,|\, i \in \mathbf{I}\}$ be an observed

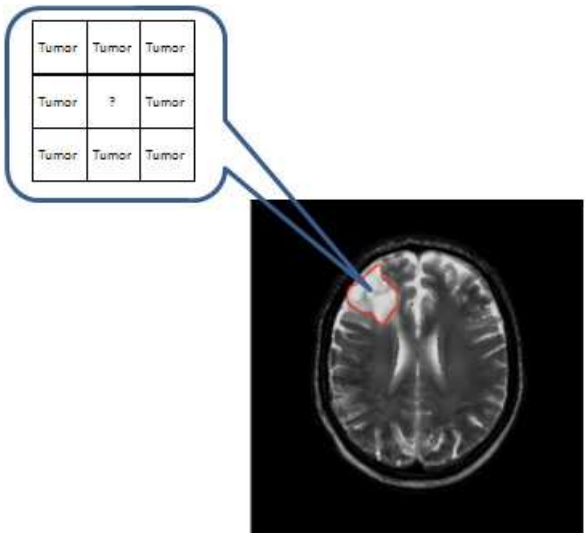

Figure 4.1: An unknown pixel that is surrounded by tumorous pixels.

input image, where each x_i is a vector describing pixel i (perhaps its intensity and texture) and $\mathbf{y} = \{y_i \,|\, i \in \mathbf{I}\}$ is the corresponding joint set of labels over all pixels of the image.

We will assume the segmentation is *binary* over $n_r \times n_c$ images, where each $y_i \in \{-1, +1\}$ (*e.g.*, is this pixel tumorous versus healthy), and the overall output $\mathbf{y}$ is $n_r \times n_c$ such bits.

At any time, our system has a pool of unsegmented instances U, as well as a (possibly empty) pool of segmented instances S. Later in section 4.3 where we describe our experiments, we will also have a set of unlabeled test images S_t. Note that these sets, S_t, S, U, are disjoint. Throughout this chapter, we will use "labeled" as a synonym for "segmented" and "unlabeled" for "unsegmented".

In this chapter we present LMU active learner [8] for image segmentation tasks. [1] Our active learner will sequentially select an unsegmented image $u \in U$, obtain its label (recall this label is a set of $|u|$ bits — one for each pixel in u), and then move this now-labeled image from U to S.

4.1 Conditional Random Field

A conditional random field (CRF) is a model of the conditional probability of a set of labels $\mathbf{y}$ given the observations $\mathbf{x}$, here given by

[1] A version of this chapter has been published in the proceedings of the ICML2009 conference.

$$P_\theta(\,\mathbf{y}\,|\,\mathbf{x}) = \frac{1}{\Omega_\theta(\mathbf{x})} \exp\left(\sum_{i\in I} \Phi_{\mathbf{w}}(y_i, \mathbf{x}) + \sum_{i\in I}\sum_{j\in N_i} \Psi_{\mathbf{v}}(y_i, y_j, \mathbf{x})\right), \qquad (4.1)$$

where N_i are the 4 neighbors of i (north, east, south, west);

$$\Phi_{\mathbf{w}}(y_i, \mathbf{x}) = \log(\frac{1}{1+\exp(-y_i \mathbf{w}^T g_i(\mathbf{x}))})$$

is the *association potential* of pixel i that uses the association parameters $\mathbf{w}$ obtained from training data, and $g_i(\mathbf{x})$ is the feature vector for pixel i;[2] $\Psi_{\mathbf{v}}(y_i, y_j, \mathbf{x}) = y_i y_j \mathbf{v}^T \mu_{ij}(\mathbf{x})$ is the interaction potential that captures the spatial correlation with neighboring pixels using the interaction parameters $\mathbf{v}$ obtained from the training data; $\mu_{ij}(x) = x_i - x_j$ is the difference between feature vectors in pixel i and j; and $\theta = [\mathbf{w}, \mathbf{v}]$ are the model parameters.

The normalizing factor

$$\Omega_\theta(\mathbf{x}) = \sum_{\mathbf{y}} \exp\left(\sum_{i\in I} \Phi_{\mathbf{w}}(y_i, \mathbf{x}) + \sum_{i\in I}\sum_{j\in N_i} \Psi_{\mathbf{v}}(y_i, y_j, \mathbf{x})\right) \qquad (4.2)$$

insures that the CRF produces a probability. Given a set of parameters θ, we can compute the label $\mathbf{y}^*$ for a given image $\mathbf{x}$: $\mathbf{y}^* = \arg\max_{\mathbf{y}} P_\theta(\,\mathbf{y}\,|\,\mathbf{x})$ (see Section 4.1.2). The challenge is learning the best values for these parameters, θ^*. The rest of this subsection discusses how to compute these optimal parameters from a set of labeled images S; then for this S and a single unlabeled image u, it then provides a useful approximation to this computation, to avoid its inherent intractability.

4.1.1 Training

Typical supervised CRF training involves finding the parameters

$$\theta^*_S = \arg\max_\theta \mathrm{LL}_S(\,\theta\,) \qquad (4.3)$$

that maximize the log of the posterior probability over training set of labeled images S

$$\mathrm{LL}_S(\,\theta^*\,) = \sum_{s\in S} \log P_{\theta^*}(\,\mathbf{y}^{(s)}\,|\,\mathbf{x}^{(s)}).$$

[2] Here, this $g_i(\mathbf{x})$ is just the information in x_i. In general, it could also include information from some adjacent pixels, via some smoothing operator, etc.

In our active learning framework, we need to consider the effect of adding one more image from unlabeled set to the training set. Since the label $y^{(u)}$ of an unlabeled image $x^{(u)}$ is not known before presenting it to an oracle, we use a conditional entropy term to express the likelihood associated with the unlabeled image as

$$\mathrm{LL}_u(\,\theta\,) \quad = \quad \sum_{\mathbf{y}} P_\theta(\,\mathbf{y}\,|\,\mathbf{x}^{(u)})\log P_\theta(\,\mathbf{y}\,|\,\mathbf{x}^{(u)}).$$

The overall objective function now consists of two terms: the first term deals with all labeled images in training set S and the second term is for an unlabeled image u as

$$\mathrm{LL}_{S+u}(\,\theta\,) \quad = \quad \mathrm{LL}_S(\,\theta\,) \;+\; \gamma\,\mathrm{LL}_u(\,\theta\,), \tag{4.4}$$

where the $\gamma \in \mathbb{R}$ parameter trades-off the two factors. We then seek the parameters θ^*_{S+u} that maximize Equation 4.4.

4.1.2 Useful Approximation

Given the lattice neighborhood structure of the CRF for image data, it is intractable to compute the normalizing factor $\Omega_\theta(\mathbf{x})$ in Equation 4.2 [60]. Following [59], [60], we incorporate the pseudo-likelihood approximation, which assumes that the joint probability distribution of all pixel labels $\mathbf{y}$ can be approximated by the product of "local probabilities" of each pixel, which is based on only the observations of x_i and the labels of the neighboring nodes y_{N_i}:

$$\begin{aligned}
\hat{P}_\theta(\mathbf{y}|\mathbf{x}) \quad &\approx \quad \prod_{i\in I}\hat{P}_\theta(y_i|y_{N_i},\mathbf{x}),\\
\hat{P}_\theta(y_i|y_{N_i},\mathbf{x}) \quad &= \quad \frac{1}{\Omega_i(\mathbf{x})}\exp\Big(\,\Phi_{\mathbf{w}}(y_i,\mathbf{x}) + \sum_{j\in N_i}\Psi_{\mathbf{v}}(y_i,y_j,\mathbf{x})\Big),
\end{aligned} \tag{4.5}$$

where this $\Omega_i(\mathbf{x})$ is a "local normalizing term", which deals only with the i^{th} pixel. Using the approximation in Equation 4.5, the log likelihood term for (respectively) training set S and a single unlabeled image u is:

$$\widehat{\mathrm{LL}}_S(\,\theta\,) \quad = \quad \sum_{s\in S}\sum_{i\in\mathbf{I}^{(s)}}\log\hat{P}_\theta(y_i^{(s)}|y_{N_i}^{(s)},x_i^{(s)}), \tag{4.6}$$

$$\widehat{\mathrm{LL}}_u(\,\theta\,) \quad = \quad \sum_{i\in\mathbf{I}}\sum_{y_i}\hat{P}_\theta(y_i|y_{N_i}^{(u)},x_i^{(u)})\times\log\hat{P}_\theta(y_i|y_{N_i}^{(u)},x_i^{(u)}). \tag{4.7}$$

Following Equation 4.4, we can add the above equations to form a single objective that combines the effect of the many labeled images S and an unlabeled data u:

$$\widehat{\mathrm{LL}}_{S+u}(\,\theta\,) \quad = \quad \widehat{\mathrm{LL}}_S(\,\theta\,) \;+\; \gamma\,\widehat{\mathrm{LL}}_u(\,\theta\,). \tag{4.8}$$

In our experiments, we set $\gamma = 1$. We also used conjugate gradient to optimize the objective function in Equation 4.8.

The $\widehat{\mathrm{LL}}_u(\theta)$ term from Equation 4.7 requires the label for unlabeled data, which unfortunately is not yet available, *i.e.*, $y_{N_i}^{(u)}$ is not known. An inference step is added to estimate the labels based on current parameters. The inference is based on iterative conditional probability (ICM) [61], which sets the label for each pixel as the maximum posterior probability:

$$y_i^* = \arg\max_{y_i} P_\theta(\, y_i \,|\, y_{N_i}, \mathbf{x}),$$

where for each pixel i, we assume that the labels of its neighbors y_{N_i} are fixed to their current estimate. We then use y_i^* to compute the label of its neighbors. We repeat this process until every y_i converges to its final value.

4.2 The LMU Active Learning Algorithm

At each time, given U and S, our LMU active learner [8] needs to select which unlabeled image $u \in U$ to give to the oracle for labeling. (Recall that this now-labeled u will then be added to the set of labeled images S.) One option is to choose the most uncertain image:

$$\mathrm{MU}(U, S) \quad = \quad \arg\max_{u \in U} H(\, \mathbf{Y}^{(u)} \,|\, \mathbf{x}^{(u)}, S\,),$$

where

$$H(\, \mathbf{Y} \,|\, \mathbf{x}, S\,) \;=\; -\sum_{\mathbf{y} \in \mathcal{Y}} P(\, \mathbf{y} \,|\, \mathbf{x}, S\,) \log P(\, \mathbf{y} \,|\, \mathbf{x}, S\,) \tag{4.9}$$

$$\approx\; -\sum_{\mathbf{y} \in \mathcal{Y}} \hat{P}_{\theta_S}(\mathbf{y}|\mathbf{x}) \log \hat{P}_{\theta_S}(\mathbf{y}|\mathbf{x}) \tag{4.10}$$

approximates the conditional entropy of the label $\mathbf{Y}$ given the image observations $\mathbf{x}$, based on the current conditional probability, which is based on the training data S. (To explain Equation 4.10: As we use the data S to produce the parameters θ_S, we identify $P(\, \mathbf{y} \,|\, \mathbf{x}, S\,)$ with $\hat{P}_{\theta_S}(\mathbf{y}|\mathbf{x})$; see Equation 4.3 and relevant approximation.)

The summation in Equation 4.10 is over all $|\mathcal{Y}| = 2^{n_r \times n_c}$ possible binary assignments to $n_r \times n_c$ pixels. We approximate this as the simple sum of the entropies of the labels y_i of each pixel of the image, $i \in \mathbf{I}$:

$$\hat{H}(\,\mathbf{Y}\,|\,\mathbf{x}, S\,) = \hat{H}(\,\mathbf{Y}\,|\,\mathbf{x}, \theta_S\,) \tag{4.11}$$

$$= -\sum_{i\in\mathbf{I}} \sum_{y_i\in\{\pm 1\}} \left[\hat{P}_{\theta_S}(y_i|x_i) \log \hat{P}_{\theta_S}(y_i|x_i)\right]. \tag{4.12}$$

We view this most-uncertain instance $\text{MU}(U, S)$ as being closest to the *boundary* between positive and negative instances. Having the label for such boundary points will help us to define the boundary more precisely and consequently increase the classification accuracy. Of course, our active learner has to start with an empty $S = \{\}$; here the associated $\theta_{\{\}}$ parameters are problematic. Moreover, this approach does not consider the distribution over $\mathbf{X}$, which means knowing more about this "boundary" point might not help identify that much with respect to this $\mathbf{X}$. The time complexity of this algorithm is $O(|U|)$ as it computes the uncertainty of all the unlabeled instances in each iteration.

This suggests an alternative approach: select the instance that would provide the maximum information about the labels of remaining unlabeled instances — *i.e.*, the instance that most reduces the uncertainty (RU) of the other unlabeled instances $U - \{u\}$:

$$\text{RU}(U, S) = \arg\max_{u\in U} H(\,\mathbf{Y}_U\,|\,\mathbf{x}_U, S\,) - H(\,\mathbf{Y}_U\,|\,\mathbf{x}_U, S + \mathbf{x}^{(u)}\,) \tag{4.13}$$

$$= \arg\min_{u\in U} H(\,\mathbf{Y}_U\,|\,\mathbf{x}_U,\ S + \mathbf{x}^{(u)}\,) \tag{4.14}$$

$$= \arg\min_{u\in U} \sum_{v\in U; v\neq u} H(\,\mathbf{Y}^{(v)}\,|\,\mathbf{x}^{(v)},\ S + \mathbf{x}^{(u)}\,) \tag{4.15}$$

$$\approx \arg\min_{u\in U} \sum_{v\in U; v\neq u} \hat{H}(\,\mathbf{Y}^{(v)}\,|\,\mathbf{x}^{(v)},\ \theta_{S+\mathbf{x}^{(u)}}\,). \tag{4.16}$$

Equation 4.14 follows from the observation that the first term of Equation 4.13 is constant for all instances u; Equation 4.15 uses the fact that the entropy of the set of independent images is just their sum; and Equation 4.16 again uses the approximate conditional entropy $\hat{H}$ defined in Equation 4.12. (This $\theta_{S+\mathbf{x}^{(u)}}$ is the solution that maximizes Equation 4.8.)

Note this RU approach works even for $S = \{\}$ and its time complexity is $O(|U|^2)$, which is computationally more expensive than MU approach ($O(|U|)$). Our actual LMU system uses both approaches: Given a set of unlabeled images U and no labeled instances $S = \{\}$, it first uses RU to find the first instance u_1 to label, then sets $S = \{u_1\}$. Thereafter, it uses MU to find the second, third, and further images. See Figure 4.2.

LMU(U: unsegmented images)

1: $S = \{\}$ *% S is initially empty*
% Compute $u_1 = RU(U, \{\})$
2: **for each** unlabeled image $u \in U$
3: $\theta_u = \arg\max_\theta \widehat{\mathrm{LL}}_u(\theta)$ *% Equation 4.7*
4: $s_u = 0$
5: **for each** other unlabeled image $v \in U$, $v \neq u$
6: s_u += $\hat{H}(\mathbf{Y}^{(v)} \mid \mathbf{x}^{(v)}, \theta_u)$ *% Equation 4.12*
7: $u_1 = \arg\min_u s_u$
8: $\mathbf{y}^{(u_1)} = \text{Oracle}(\mathbf{x}^{(u_1)})$ *% Get label*
9: $S = (\langle \mathbf{x}^{(u_1)}, \mathbf{y}^{(u_1)} \rangle)$
10: $\theta_S = \arg\max_\theta \widehat{\mathrm{LL}}_S(\theta)$ *% Equation 4.6*
11: $U = U - \{u_1\}$
12: **for** $i = 2, ...$
% $u_i = MU(U, S)$
13: **for each** $u \in U$
14: $t_u = \hat{H}(\mathbf{Y}^{(u)} \mid \mathbf{x}^{(u)}, \theta_S)$ *% Equation 4.12*
15: $u_i = \arg\max_u t_u$
16: $\mathbf{y}^{(u_i)} = \text{Oracle}(\mathbf{x}^{(u_i)})$ *% Get label*
17: $S = S + (\langle \mathbf{x}^{(u_i)}, \mathbf{y}^{(u_i)} \rangle)$
18: $\theta_S = \arg\max_\theta \widehat{\mathrm{LL}}_S(\theta)$ *% Equation 4.6*
19: $U = U - \{u_i\}$
20: **end**
21: **return** θ_S

Figure 4.2: Pseudo code for the LMU active learning algorithm.

Figure 4.3: Sample "sky" images from Geometric context dataset.

4.3 Experiments

To investigate the empirical performance of our active learning algorithm, we conducted a set of experiments on two challenging real-world problems: Finding the sky in the geometric context dataset and segmenting tumors in medical images. We also ran a scaling study, to see the influence of the size of each image on the number of images required to obtain good performance.

4.3.1 Finding The Sky in Color Images

The geometric context dataset [31] is a collection of 105 images, many very cluttered, that span a variety of natural, urban, and suburban sceneries. Our goal here is to find the sky within these images. Figure 4.3 shows three instances of images in this dataset. This task is challenging since the sky could be blue or white, clear or overcast, and worse, many scenes contain both sky and ocean, which are not easily separable based on their color. The original images were of various sizes; we downsized each to 64×64 pixels to make the size uniform, and to make our computations more tractable. We partitioned these images into the unlabeled-set U with 85 images and the test-set S_t with 20 images.

Features: We associate each pixel with twelve values: its 3 color intensities, its vertical position (as the sky is typically at the top of the image) and 8 texture values: We apply the MR8 filter banks [62] (which contain edge filters and bar filters at 6 different orientations, at 3 scales) to the region centered on each pixel, but record only the maximum filter response at each of the 6 orientations; we also include 2 isotropic features: Gaussian and a Laplacian of Gaussian.

In each iteration of the active learning process, our LMU system identifies one specific image u from U, which is removed from U, labeled by an oracle, then added to the labeled training set S. We then train a segmenter on this augmented training set $S+$"labeled" u to

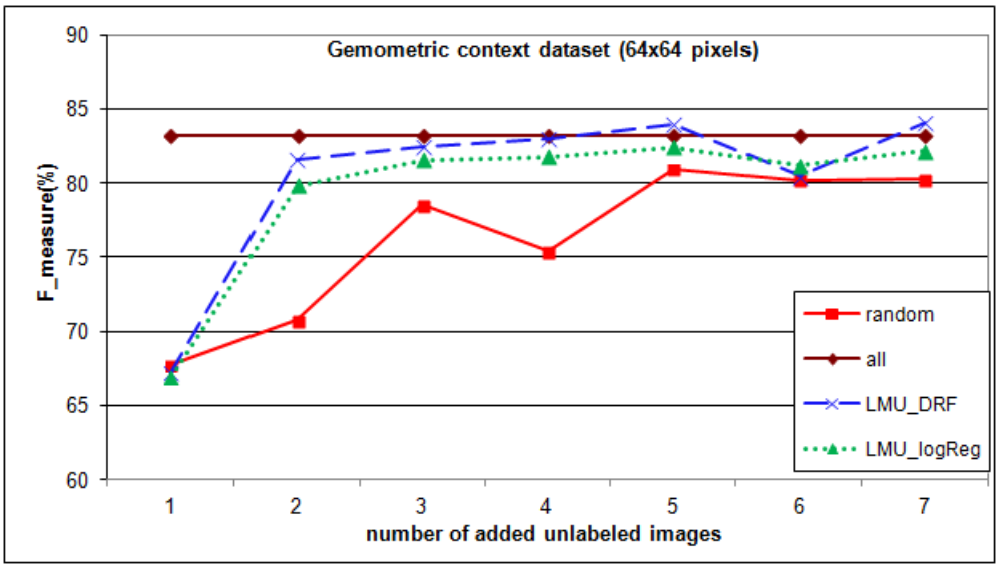

Figure 4.4: Accuracy of segmentation versus number of training data chosen from unlabeled set for geometric context dataset.

produce θ_{S+u}, then test this system on 20 images in the test-set S_t, recording the average F-measure

$$\text{F-measure} \quad = \quad \frac{2 \times (precision)(recall)}{(precision + recall)}$$

where $precision = tp/(tp+fp)$ and $recall = tp/(tp+fn)$, where tp, fp, fn are true positive, false positive and false negative respectively. Note this measure is problematic when $tp = 0$; see Footnote 3.

In Figure 4.4, the horizontal axis represents the number of added unlabeled images and vertical axis is the average F-measure. The "all" line shows the results of training on (the oracle-labeled versions of) *all* 85 unlabeled images. Checking the "LMU" line, we see that the first carefully-selected image alone produced a classifier whose F-measure was 67%, and this accuracy improved to 81% by using the second image. Note this is only 2% below the accuracy obtained by training on all unlabeled data; moreover, this is statistically indistinguishable at the $p < 0.05$ level, based on a paired t-test. (The accuracy of the second iteration was significantly better than the first — paired t-test $p < 0.05$.) There is no significant change between the second iteration and subsequent iterations, which means that the first two images chosen by LMU are good enough to train the classifier. Those two images are the left and middle images shown in Figure 4.3.

Of course, it is possible that *any two images* would be sufficient. To test this, we

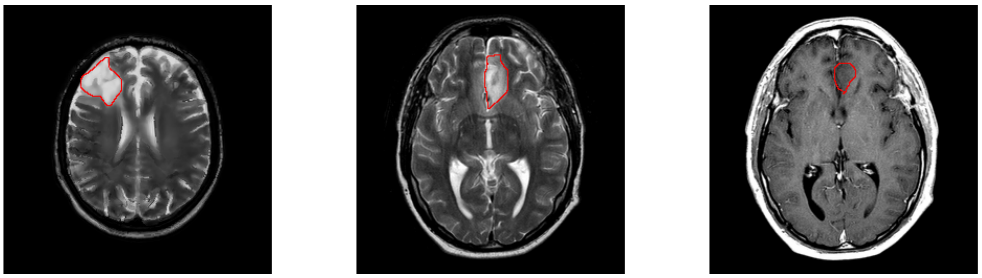

Figure 4.5: Sample images from brain tumor dataset, tumor is segmented in red.

randomly selected images for the training set; see the "random" line in Figure 4.4. Each point in this line is the average over 10 random choices. We see that this line is well below the LMU line. Moreover, the segmenter remains statistically inferior to the "all" line until observing (on average) 5 randomly-drawn images.

We also developed LMU_logReg, which actively learns the parameters of the logistic regression segmentor. Note that in logistic regression there is no interaction potential between every pixel and its neighbors. The results of applying LMU_logReg on geometric context dataset is shown in Figure 4.4. This figure shows that the logistic regression on the geometric context dataset has slightly lower accuracy compared to CRF, which implies that the interaction potential in Equation 4.1 has lower weight than the association potential.

4.3.2 Finding Tumors in Brain Scans

Here, we consider the challenge of finding tumors within a patient's brain — that is, labeling each pixel in a magnetic resonance (MR) image as either tumorous or non-tumorous. This task is crucial in surgical planning and radiation therapy, and currently requires a significant amount of manual work by human medical experts.

Here, we have 80 images (axial slices) from the brains of 16 patients. These are taken from different regions of the brain; in particular, no two images are adjacent to each other. We resized each image from 256×256 pixels to 64×64. Figure 4.5 shows three images from this dataset, with the tumor segmentation outlined in red. We again partition these images into an unlabeled set U, containing 71 images from 11 individuals, and a test set S_t, containing 9 images from the other 5 patients.[3]

Features: Most patient visits yield scans in 3 different MR modalities: T1, T2, and

[3] The F-measure score is problematic if any test image has no tumor, as here there can be no true positives ($tp = 0$). Hence, to simplify our analysis, our S_t includes only images that contain some tumor. However, the unlabeled set U includes some images that have no tumor.

T1c (that is, T1 after the patient has received a contrasting agent) [63]. We identify each pixel i with a vector of 4 values, including the T2 value and the difference between T1c and T1. As each brain is somewhat symmetric around the sagittal plane, we also include the symmetry feature by computing the difference between intensities of pairs of symmetrical pixels with respect to the sagittal plane, for both T2 and T1c - T1 modalities. So we compute four features for each pixel.

Figure 4.6 shows the results of actively selecting the training set. Actively training on one image in this dataset produces an average F-measure of 57%. This segmenter is as good as the one obtained using all of the data (paired t-test $p < 0.05$). (The specific image selected is the left one in Figure 4.5.) Training on the second LMU-selected image increased the average accuracy to 70%, which is higher than the 61% accuracy obtained using all of the data.

Note that some of the images in the unlabeled set U do not contain any tumor. On the other hand, the test set only contains images that have some tumor (see Footnote 3). This difference between the test domain and the training domain is known as "covariate shift" [64]. Therefore, the optimal model obtained by training on all the images in the dataset may not be the optimal for the test set. We conjecture that the active learner can obtain higher segmentation accuracy by querying images that better represent the images in the test set (*i.e.*, better represent the characteristics of the tumors).

Donmez and Carbonell [65] report a similar situation (albeit in active sampling for learning the ranking on the TREC 2004 dataset), noting that the performance of their active learning algorithm is sometimes better than the one obtained by training on all the data. Finally, as with the Sky data, on average, the segmenters produced using the first several (here three) *randomly* drawn images were all significantly inferior to the "all" segmenter.

The results of applying LMU_logReg on brain tumor dataset is shown in Figure 4.6. This figure shows that the performance of LMU_logReg is statistically significantly worse (at $p < 0.05$) than LMU in the brain tumor dataset. The reason could be the fact that the labels for each pixel in a brain image is very dependent on the labels of its neighboring pixels.

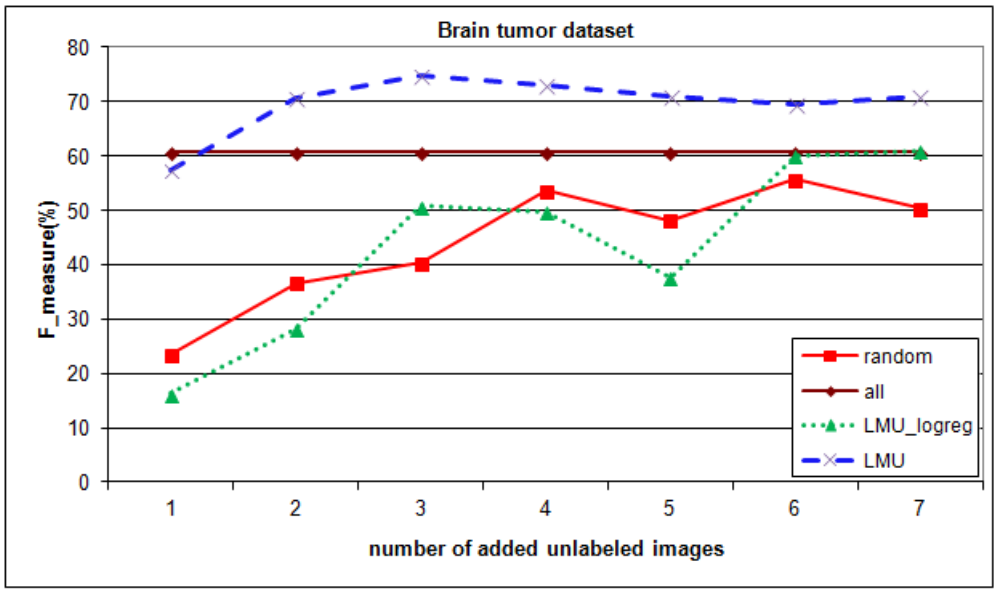

Figure 4.6: Accuracy of segmentation versus number of training data chosen from unlabeled set for brain tumor dataset.

4.3.3 Scalability Study

This subsection explores how our LMU scales with the size of the images. We therefore downsized the images in the geometric context dataset from 64×64 to 32×32, then repeated the same active learning process described above. The results, appearing in Figure 4.7, show essentially the same trend that appeared in Figure 4.4. Here, however, LMU required 7 images before it first obtained a segmenter whose performance was statistically "equivalent" to the one based on all of the data (paired t-test, $p < 0.05$).

4.3.4 Discussion

It is, at first, very surprising that one can produce an effective segmenter with so few images — here, only 2 for the (original) sky data, and 1 for the brain tumor data! Towards explaining this, note that each of these images is not really a single "point", but is actually $64 \times 64 \approx$ 4,000 pixels, and each oracle-label is actually providing around 4000 bits. Hence, the 2 sky images is essentially 8000 bits, which is a lot of information. The results in the scaling studies are consistent with this conjecture: Here, we required 7 carefully-selected images to obtain the information needed to do well; notice that this $7 \times (32 \times 32) \approx 2 \times (64 \times 64)$.[4]

[4]We are not claiming that 8000 bits is a magical number — instead, we are just observing that our active learner requires more small images than large images, which is consistent with the claim that the number of

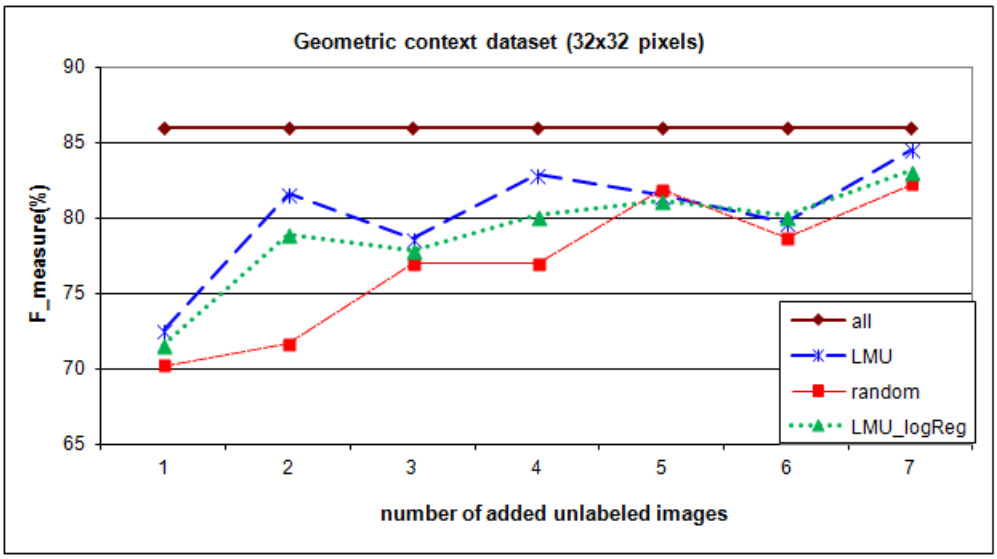

Figure 4.7: Accuracy of segmentation versus number of training data chosen from unlabeled set for geometric context dataset with images downsized to 32×32 pixels.

As further support, consider the segmenters based on randomly-drawn images. While they were, typically, worse than ones based on images specified by LMU, we found that we were still getting good segmenters using only a few such random images. Again, this is consistent with the view that the label of each image is supplying a great deal of information — even if the image is drawn randomly.

4.3.5 Other Results

The results shown above strongly suggest that very few images are sufficient to train a CRF-based segmentor, but only if they are well selected. We explored this claim over other datasets. Figure 4.8 shows the results of applying LMU on the set of sunflower objects from Caltech101 dataset [66]. Since the dataset is intended for object recognition task, it is relatively easy to segment the object from background, which is probably why LMU gets good accuracy after actively selecting only one image.

We also explored some approaches to reduce LMU's computational complexity. For example, LMU can start with a randomly selected image (LMU_randStart). Figure 4.9 shows the result of this algorithm on brain tumor dataset. We see that LMU_randStart may need more queries to obtain an accurate segmentor. Here, LMU_randStart requires 3 queries to have the accuracy obtained by training on all unlabeled data.

pixels being labeled seems significant.

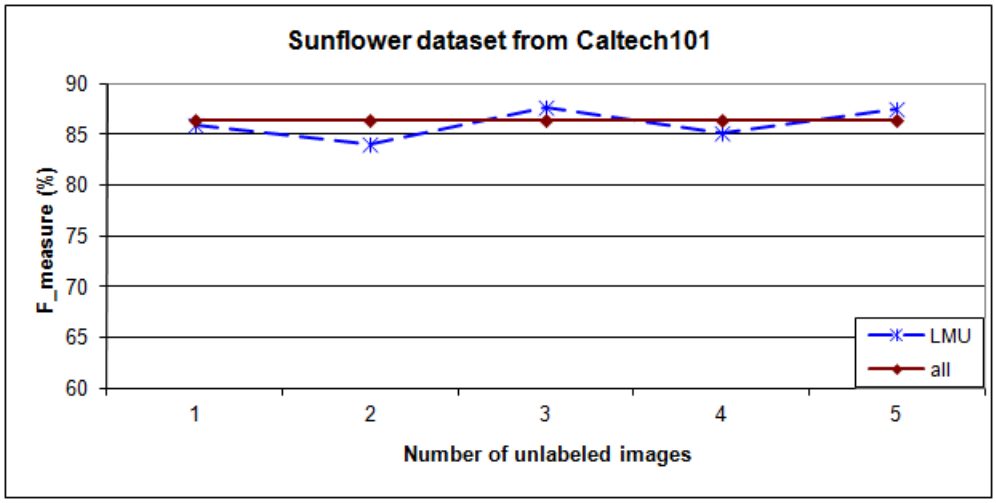

Figure 4.8: Accuracy of segmentation versus number of training data on the sunflower category in Caltech101 dataset.

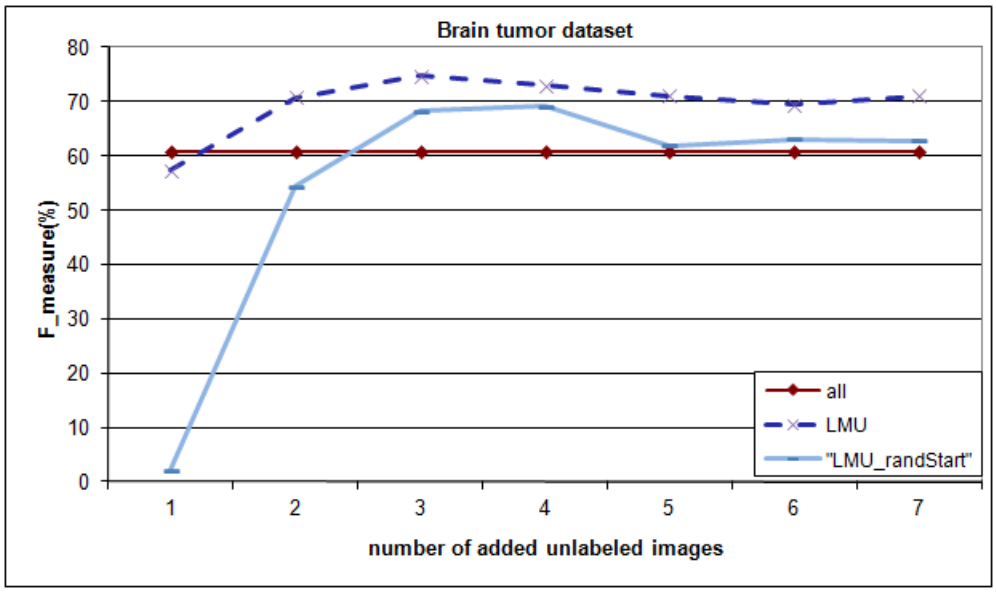

Figure 4.9: Accuracy of segmentation versus number of training data chosen from unlabeled set on brain tumor dataset for LMU and LMU with random start.

4.4 Summary

In this chapter we presented the LMU active learning algorithm developed for image segmentation tasks. LMU is one of the first studies that considers the challenge of actively learning the parameters of a machine learning model (CRF-based segmenter) for images. LMU combines two active learning strategies by first querying an image that most reduces the uncertainty of other images and then selecting the image with maximal uncertainty. Our experiments on real world datasets showed that this particular combination was effective for this task and it could produce an effective segmenter using very few segmented images. Our studies support the claim that it may be because the label for a single image contains a great deal of information; *i.e.*, corresponds to receiving many single-bit labels, in the standard active learning framework.

If some images in the dataset contain more information about the relation between image features and labels than others, then LMU is likely to produce an accurate segmentor with fewer labeled images than random sampling. On the other hand, if all images in the dataset contain the same amount of information about the relation between image features and labels, then LMU is likely to have similar performance to random sampling.

Chapter 5

Importance Sampling Active Learning

In this section we introduce importance sampling active learning algorithm (*ISAL*), which attempts to reduce the number of queried instances by drawing them from an appropriate distribution, which can be different from the underlying distribution.

One application of *ISAL* is in experimental design where sampling from the underlying distribution is difficult. For example, consider the task of tuning the parameters of a jet engine where we are interested in knowing the engine's breaking point. Figure 5.1 shows the probability density function for the cruising speed of an airplane. This figure shows that the average cruising speed of the plane is 500 mph and 700 mph is the minimum speed at which the engine is likely to fail.

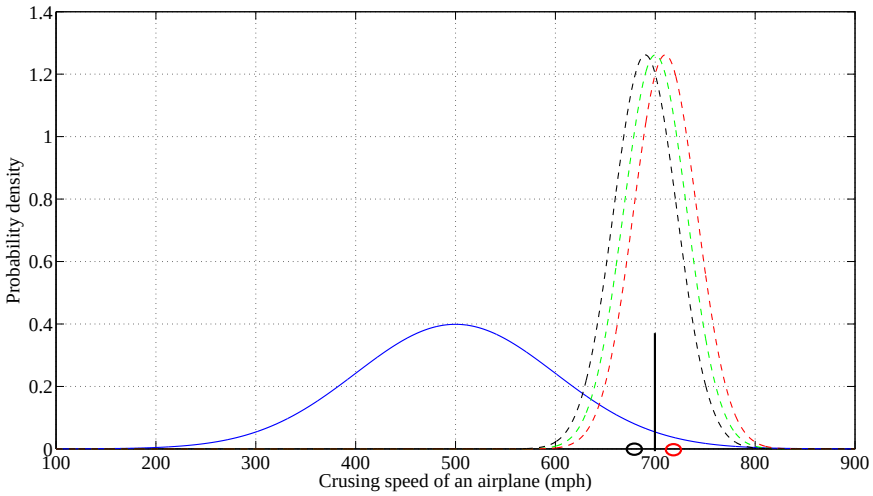

Figure 5.1: Hypothetical distribution for the cruising speed of an airplane (solid line). Dashed lines show the sampling distributions around the point that the engine fails.

Here we can use an engine simulator, change the parameters and record the outcome. The naïve approach is to randomly choose the values of the parameters from the underlying distribution. Given that the breaking point is relatively unlikely (in underlying distribution), it may take a long time to find the breaking point of the engine. Alternatively, by clever selection of points around the decision boundary, *ISAL* might be able to find the breaking point using few probes. For example in Figure 5.1, it would be a good choice to select two data points with different labels around the estimated decision boundary. Those two points are important because with a perfectly separable data, the decision boundary is located somewhere between them.

In the rest of this chapter we describe *ISAL* and analyze its performance.

5.1 Preliminaries

Let $\langle X, Y \rangle$ be jointly sampled from a distribution P over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$. The specification of the learning problem includes the set of hypotheses $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$. The performance of a hypothesis $h \in \mathcal{H}$, whose goal is to predict Y given X, is measured by

$$L(h) = \mathbb{E}\left[\ell(h(X), Y)\right], \tag{5.1}$$

where $\ell(h(X), Y) = \frac{1}{2}|h(X) - Y|$ and the expectation is over random variables X and Y. We will typically think of $\ell(a, b) = 1$ if $a \neq b$, and is 0 otherwise. Then $L(h) = \mathbb{P}\left(h(X) \neq Y\right)$.

Let $Z_1, \ldots, Z_n \in \mathcal{X} \times \mathcal{Y}$ be random variables such that $Z_i = \langle X_i, Y_i \rangle$, where the distribution of X_i is possibly governed by the learner and Y_i is generated from the marginal $P_{Y|X}$ given X_i: $Y_i \sim P_{Y|X}(\cdot|X_i)$. Let $h_n : \mathcal{X} \to \mathcal{Y}$ be a hypothesis obtained after training on $(Z_1, \ldots, Z_n)$. We will assume that the auxiliary random variables $\langle X, Y \rangle$, which represent the future test instance, are disjoint from $(Z_1, \ldots, Z_n)$. The value of label Y is predicted by $h_n(X)$.

For simplicity, we assume that P_X, the true marginal of X, is absolute continuous and its density (with respect to the Lebesgue measure) is $p(\cdot)$. Similarly, we assume that the marginal of X_i given the past instances, $P_{X_i|Z_1,\ldots,Z_{i-1}}$, has density q_i. In fact, during learning, the learner selects q_i.

We assume that all densities $q_i \in \mathcal{Q}$ have support that includes p's support, *i.e.*,

Algorithm 1 (Importance Sampling Active Learning (*ISAL*))

Input:

p is the density for true distribution of X.
$\mathcal{H}$ is the class of hypotheses.
n is the total number of iterations.
$\mathcal{Q}$ is the set of distributions where every $q \in \mathcal{Q}$ has the support that includes p's support.
$h'[x, y, S_i] = \arg\min_{h \in \mathcal{H}} \{\hat{L}(h|S_i) : h(x) \neq y\}$.
$\hat{L}(h|S_i)$ is defined in Equation 5.3.
$S_0 = \emptyset$.

1: **for** $i = 1, \ldots, n$
2: **if** (i==1) then $q_1 \leftarrow p$
3: **else** generate $q_i \leftarrow \arg\min_{q \in \mathcal{Q}} \mathbb{E}_{x \sim q}[\mathbb{1}\{\hat{L}_{i-1}(h'_{i-1}[x, h^*_{i-1}(x), S_{i-1}] \big| S_{i-1}) \neq 0\}]$
4: $x_i \leftarrow Draw(q_i(.))$ *% draw from distribution with density* q_i
5: request the label for x_i : y_i
6: $S_i = S_{i-1} \cup \{[q_i, x_i, y_i]\}$
7: $h^*_i = \arg\min_{h \in \mathcal{H}} \hat{L}(h|S_i)$
8: **end**
9: return h^*_n

Figure 5.2: Importance sampling active learning (*ISAL*) algorithm.

$\{x : p(x) \neq 0\} \subseteq \{x : q_i(x) \neq 0\}$. The set

$$S_i = \{(q_1, x_1, y_1), ..., (q_i, x_i, y_i)\} \tag{5.2}$$

represents the i triples of distribution q_j, draws $x_j \in \mathcal{X}$, and labels $y_j \in \mathcal{Y}$ for $j = 1, \ldots, i$.

5.2 Importance Sampling Active Learning Algorithm (*ISAL*)

The *ISAL* algorithm is shown in Figure 5.2. In iteration $i = 1, \ldots, n$, the algorithm draws an instance x_i from the distribution with density q_i, which (for $i > 1$) had the lowest expected empirical error of alternative hypothesis (line 4 of *ISAL*) in the $(i-1)$-th iteration.

Choosing the Candidate Point: *ISAL* will sequentially request the labels of instances $x_1, \ldots, x_n$ from the oracle, using the following approach by Beygelzimer *et al.* [20]: on the i-th iteration, first find a hypothesis h^*_i that has the minimum empirical error on the current labeled set, S_i. It then finds an alternative hypothesis h'_i that has minimum empirical error over all hypotheses that disagree with h^*_i on the label of the candidate point $x \in \mathcal{X}$. If the empirical error of h^*_i and h'_i are close to each other, then there is

uncertainty on the label of x and we ask an oracle to provide the label. However, instead of identifying a single candidate x, *ISAL* identifies a distribution q_i that puts high weight on instances that have highly uncertain labels.

The Sampling Distribution q_i: In the situation where the data is noise-free and the true model is linear ($\mathcal{H}_{Linear}$), we can directly compute the optimal such q_i (line 3 of Figure 5.2). Here, $q_i \leftarrow \mathcal{N}(\mu_i, \Sigma_i)$ for some relevant μ_i, Σ_i, which depends on the current margin.

Otherwise, the user provides the set of distributions $\mathcal{Q}$ as an input to *ISAL* based on his knowledge of the data. We assume that the cardinality of this set of distributions $\mathcal{Q}$ is small enough such that we can evaluate each, to find the best such q_i (line 3 of Figure 5.2). $\mathcal{Q}$ may include a set of Gaussian distributions $\mathcal{N}(\mu, \Sigma)$, where each $\mu = [\mu_1, \ldots, \mu_d]$, d is the dimension of the space, and $\Sigma = diag(\sigma_1, \ldots, \sigma_d)$, where for each $j = 1, \ldots, d$, it includes $\mu_j = c_j + m\tau_j$, where $m = 1, \ldots, M$ and $c_j \in \mathbb{R}$, $\tau_j \in \mathbb{R}^+$, $\sigma_j \in \mathbb{R}^+$ and $M \in \mathbb{N}$ are the parameters set by the user such that it covers the regions of interest (*e.g.,* boundary regions), known from heuristics. In addition, $\mathcal{Q}$ always includes the underlying distribution $p(\cdot)$ too.

Using Weighted Estimation of Error: In learning, a good empirical measure of the performance of a hypothesis is the number of mistakes that the hypothesis makes over all instances in a hold-out dataset. When x_i is drawn from q_i (line 4 of Figure 5.2), which is different from p, this measure would become biased. In order to remove the bias, we introduce importance weights and estimate the performance of a hypothesis h by Equation 5.3; see Figure 5.2.

$$\hat{L}(h|S) = \frac{1}{|S|} \sum_{(q,x,y) \in S} \frac{p(x)}{q(x)} \ell(h(x), y). \tag{5.3}$$

Definition 1. *Let $\hat{L}(h|S_i)$ be the empirical error of $h \in \mathcal{H}$ over a set of i instances $S_i = \{(q_1, x_1, y_1), \ldots, (q_i, x_i, y_i)\}$ (Equation 5.3) and $\mathcal{H}$ be a set of linear classifiers. Let $h_i^* = \arg\min_{h \in \mathcal{H}} \{\hat{L}(h|S_i)\}$. For an instance $x \in \mathcal{X}$, we define the alternative hypothesis*

$$h_i'[x, h_i^*, S_i] = \arg\min_{h \in \mathcal{H}} \{\hat{L}(h|S_i) : h(x) \neq h_i^*(x)\}. \tag{5.4}$$

Note that $h_i'[x, h_i^*, S_i]$ is itself a function, which takes an argument in $\mathcal{X}$ and returns a value in $\mathcal{Y}$. If there are more than one function that satisfies the condition $\arg\min_{h \in \mathcal{H}} \{\hat{L}(h|S_i) : h_i^*(x) \neq h(x)\}$, just pick one arbitrarily. In the case of threshold classifiers for one-dimensional data, we consider

$$h_i'[x^{(j)}, h_i^*, S_i](x) = \begin{cases} -1 & if\ x < x^{(j)} \\ -h_i^*(x) & if\ x = x^{(j)} \\ +1 & if\ x > x^{(j)}. \end{cases} \tag{5.5}$$

We will suppress h_i^*, S_i in $h_i'[x^{(j)}, h_i^*, S_i](x)$ when it is clear from context.

Lemma 1. *For any $h \in \mathcal{H}$ and set of n instances $S_n = \{(q_1, x_1, y_1), \ldots, (q_n, x_n, y_n)\}$, $\mathbb{E}\left[\hat{L}(h|S_n)\right] = L(h)$.*

Proof. By linearity of expectation,

$$\mathbb{E}\left[\hat{L}(h|S_n)\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\frac{p(X_i)}{q_i(X_i)}\ell(h(X_i), Y_i)\right].$$

Since q_i is computed based on $Z_1, \ldots, Z_{i-1}$ and each X_i is generated from q_i, a simple calculation gives that $\mathbb{E}\left[\frac{p(X_i)}{q_i(X_i)}\ell(h(X_i), Y_i)\right] = \mathbb{E}\left[\ell(h(X), Y)\right] = L(h)$. Therefore, $\mathbb{E}\left[\hat{L}(h|S_n)\right] = L(h)$. □

Definition 2. *(Weighted loss function) Let $\mathcal{Q}$ be the set of permissible densities, $X \in \mathcal{X}$ have the label $Y \in \mathcal{Y}$, and $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ be the class of hypotheses. Let p be the true probability density function of random variable X. Suppose x is drawn from a distribution with density $q \in \mathcal{Q}$ where q has support that includes p, and $\frac{p(x)}{q(x)} \leq r_{max},\ \forall x \in \mathcal{X}$, where r_{max} is a finite positive number. Define the class of weighted loss functions $\mathcal{F}$ to be*

$$\mathcal{F} = \{f : \mathcal{Q} \times \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \times \mathbb{R} \to \mathbb{R} : f(q, x, y, h, p) = \frac{p(x)}{q(x)}\ell(h(x), y)\}, \tag{5.6}$$

where $\ell(h(x), y) = \frac{1}{2}|h(x) - y|$.

5.2.1 Analysis of *ISAL* on Realizable One-Dimensional Data

In this section, we investigate the performance of *ISAL* on a one-dimensional dataset $\mathcal{X} \subseteq \mathbb{R}$ that can be perfectly separated by some simple threshold functions $\mathcal{H} = \{h_c : c \in \mathbb{R}\}$,

$$h_c(x) = \begin{cases} +1 & if\ x \geq c \\ -1 & if\ x < c, \end{cases} \tag{5.7}$$

(used by [25], [30], [67]). Figure 5.3 shows a few instances of the input space $\mathcal{X}$.

According to VC theory, in order to achieve an error rate of less than ϵ, a passive learner requires $O(\frac{1}{\epsilon})$ random draws from p [68] (ignoring the confidence term δ). Dasgupta [30] showed that a binary search with $O(\log \frac{1}{\epsilon})$ draws can achieve an error rate

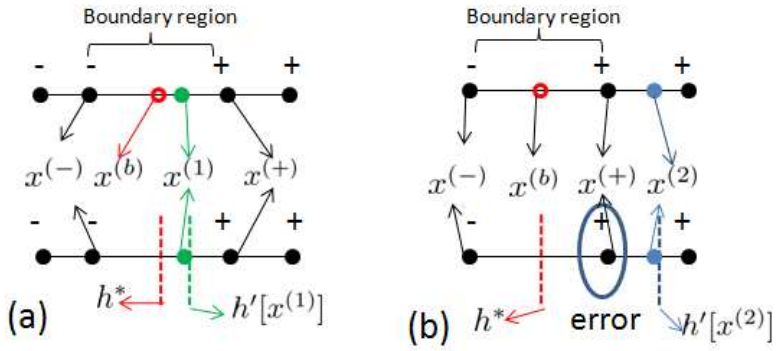

Figure 5.3: Two scenarios for *ISAL* algorithm on one-dimensional data, when the new instance is (a) $x^{(1)}$ inside boundary region or (b) $x^{(2)}$ outside of boundary region.

of less than ϵ on this one-dimensional case. As a special case, we also show that *ISAL* can achieve this accuracy with expected $O(\log \frac{1}{\epsilon})$ draws, although its application is not restricted to one-dimensional data.

Definition 3. *At iteration i of* ISAL, *let $x_i^{(-)}$ be the maximum valued instance in S_i of Equation 5.2 with -1 label, $x_i^{(+)}$ be the minimum valued instance in S_i with $+1$ label, and $(x_i^{(-)}, x_i^{(+)})$ be the boundary region with midpoint $x_i^{(b)} = \frac{x_i^{(-)}+x_i^{(+)}}{2}$; see Figure 5.3(a).*

We consider two scenarios for any drawn instance. In the first scenario, instance $x^{(1)}$ is located inside the boundary region, *e.g.*, the green point in Figure 5.3(a) and the alternative classifier $h_i'[x^{(1)}]$ corresponding to it has the boundary point shown in green. Note that in Figure 5.3(a), $\hat{L}(h_i^*|S_i) = \hat{L}(h_i'[x^{(1)}]\big|S_i) = 0$. In the second scenario, the new instance $x^{(2)}$ is located outside the boundary region (blue point in Figure 5.3(b)). Similarly, h_i^* and $h_i'[x^{(2)}]$ are shown as red and blue lines respectively. Here $\hat{L}(h_i'[x^{(2)}]\big|S_i) > 0$ because the decision boundary corresponding to $h_i'[x^{(2)}]$ is outside the boundary region. In summary, instances inside the boundary region have $\hat{L}(h_i'|S_i) = 0$ and points outside the boundary region have $\hat{L}(h_i'|S_i) > 0$.

Lemma 2 extends this analysis to a group of instances. It compares two distributions with densities $q^{(1)}$ and $q^{(2)}$, where $q^{(1)}$ has more weights inside the boundary region than $q^{(2)}$. Lemma 2 shows that *ISAL* selects $q^{(1)}$ rather than $q^{(2)}$.

Lemma 2. *Let $\mathcal{X} \subseteq \mathbb{R}$, $\mathcal{Y} = \{-1, +1\}$ and $x \in \mathcal{X}$ be perfectly separable by some hypothesis in $\mathcal{H}$. Define S_i, $x_i^{(-)}$, $x_i^{(+)}$, and $x_i^{(b)}$ using definition 3; and $\mathcal{H}$, $\hat{L}(h|S_i)$, h_i^*, $h_i'[x^{(j)}]$ using definition 1. At iteration i of ISAL, let $q^{(1)}$ be $\mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$, $q^{(2)}$ be $\mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$, where $\sigma_{min} < \sigma$ or $\eta \neq 0$. Let $x^{(1)} \in \mathcal{X}$ have distribution $q^{(1)}$ and*

$x^{(2)} \in \mathcal{X}$ *have distribution* $q^{(2)}$. *Then*

$$\mathbb{E}_{x^{(1)}\sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]\big|S_i) \neq 0\}] < \mathbb{E}_{x^{(2)}\sim q^{(2)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(2)}]\big|S_i) \neq 0\}]. \tag{5.8}$$

Proof. We have that

$$\mathbb{E}_{x^{(1)}\sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]\big|S_i) \neq 0\}] = \int_{-\infty}^{\infty} \mathbb{1}\{\hat{L}(h_i'[x]\big|S_i) \neq 0\} q^{(1)}(x)dx. \tag{5.9}$$

As we discussed earlier, $\hat{L}(h_i'[x]\big|S_i)$ is zero in the boundary region $x \in (x_i^{(-)}, x_i^{(+)})$, so we have

$$\begin{aligned}\mathbb{E}_{x^{(1)}\sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]\big|S_i) \neq 0\}] = &\int_{-\infty}^{x_i^{(-)}} 1 \times q^{(1)}(x)dx \\ &+ \int_{x_i^{(+)}}^{\infty} 1 \times q^{(1)}(x)dx. \end{aligned}\tag{5.10}$$

From Equation 5.10, we note that $\mathbb{E}_{x^{(1)}\sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]\big|S_i) \neq 0\}]$ is equivalent to the area under the curve of density function $q^{(1)}$ outside the boundary region. Therefore, we can write it as

$$\mathbb{E}_{x^{(1)}\sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]\big|S_i) \neq 0\}] = 1 - \int_{x_i^{(-)}}^{x_i^{(+)}} q^{(1)}(x)dx. \tag{5.11}$$

Similarly,

$$\mathbb{E}_{x^{(2)}\sim q^{(2)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(2)}]\big|S_i) \neq 0\}] = 1 - \int_{x_i^{(-)}}^{x_i^{(+)}} q^{(2)}(x)dx. \tag{5.12}$$

Therefore, to show that $\mathbb{E}_{x^{(1)}\sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]\big|S_i) \neq 0\}] < \mathbb{E}_{x^{(2)}\sim q^{(2)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(2)}]\big|S_i) \neq 0\}]$, we have to show that the area under the curve in the boundary region for $q^{(1)}$ is greater than the one for $q^{(2)}$.

The area under the curve in the boundary region is the probability that a random variable $X \in \mathcal{X}$ is inside the boundary region, *i.e.*, $\mathbb{P}\left(x_i^{(-)} < X < x_i^{(-)}\right)$.

First we compare $\mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ with all the distributions $\mathcal{N}(x_i^{(b)}, \sigma^2)$, where $\sigma > \sigma_{min}$ (see Figure 5.4). Then we compare $\mathcal{N}(x_i^{(b)}, \sigma^2)$ with all the distributions $\mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$, where $\eta \neq 0$. Finally, we extend the comparison between $\mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ and all the distributions $\mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$, where $\eta \neq 0$ and $\sigma > \sigma_{min}$.

Random variables $X^{(1)}$ and $X^{(3)}$ have distributions $\mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ and $\mathcal{N}(x_i^{(b)}, \sigma^2)$ respectively. We show that the instance $x^{(1)} \sim \mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ has higher probability of

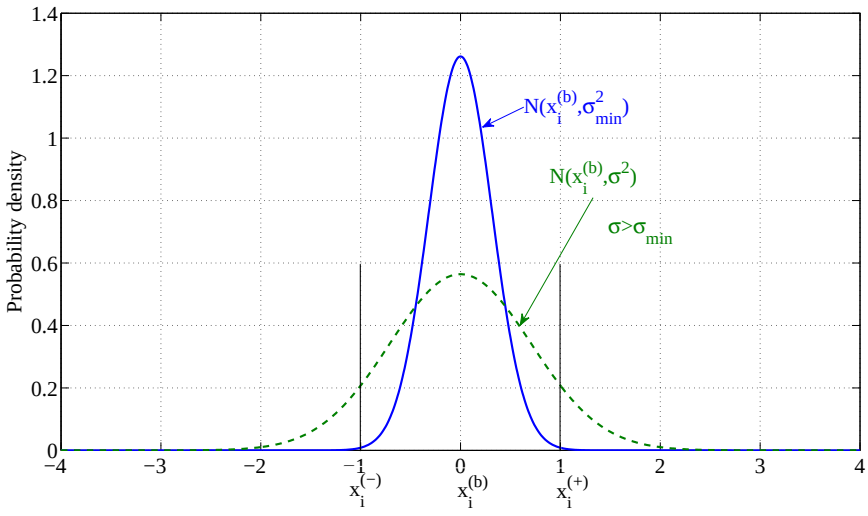

Figure 5.4: Normal distributions $\mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ and $\mathcal{N}(x_i^{(b)}, \sigma^2)$.

being inside the boundary region than the instance $x^{(3)} \sim \mathcal{N}(x_i^{(b)}, \sigma^2)$. Probability of $x^{(1)} \sim \mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ being inside the boundary region is

$$\mathbb{P}\left(x_i^{(-)} < X^{(1)} < x_i^{(+)}\right) = \Phi(\frac{x_i^{(+)} - x_i^{(b)}}{\sqrt{\sigma_{min}^2}}) - \Phi(\frac{x_i^{(-)} - x_i^{(b)}}{\sqrt{\sigma_{min}^2}}), \tag{5.13}$$

where $\Phi(.)$ is the standard normal cumulative probability function.

Similarly, probability of $x^{(3)} \sim \mathcal{N}(x_i^{(b)}, \sigma^2)$ being inside the boundary region is

$$\mathbb{P}\left(x_i^{(-)} < X^{(3)} < x_i^{(+)}\right) = \Phi(\frac{x_i^{(+)} - x_i^{(b)}}{\sqrt{\sigma^2}}) - \Phi(\frac{x_i^{(-)} - x_i^{(b)}}{\sqrt{\sigma^2}}). \tag{5.14}$$

Since $\sigma_{min} < \sigma$ we have that $\frac{x_i^{(+)} - x_i^{(b)}}{\sqrt{\sigma^2}} < \frac{x_i^{(+)} - x_i^{(b)}}{\sqrt{\sigma_{min}^2}}$. $\Phi(.)$ is monotonically increasing, which implies that $\Phi(\frac{x_i^{(+)} - x_i^{(b)}}{\sqrt{\sigma^2}}) < \Phi(\frac{x_i^{(+)} - x_i^{(b)}}{\sqrt{\sigma_{min}^2}})$. Similarly, $\Phi(\frac{x_i^{(-)} - x_i^{(b)}}{\sqrt{\sigma_{min}^2}}) < \Phi(\frac{x_i^{(-)} - x_i^{(b)}}{\sqrt{\sigma^2}})$. From Equations 5.13 and 5.14, we conclude that

$$\mathbb{P}\left(x_i^{(-)} < X^{(1)} < x_i^{(+)}\right) > \mathbb{P}\left(x_i^{(-)} < X^{(3)} < x_i^{(+)}\right). \tag{5.15}$$

We now show that *ISAL* selects $\mathcal{N}(x_i^{(b)}, \sigma^2)$ over $\mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$, where $\eta \neq 0$. Figure 5.5 shows two Gaussian distributions with the same variance but different means. The figure shows that when we shift the distribution by η, the area under the curve that is outside $(x_i^{(-)}, x_i^{(+)})$ grows faster than the area inside $(x_i^{(-)}, x_i^{(+)})$ due to the exponential nature of Gaussian density function. This implies that the area under the curve in the

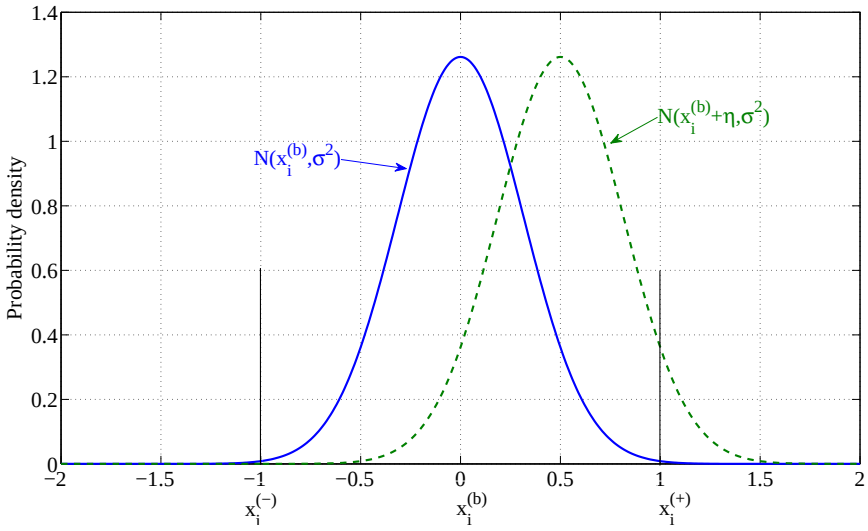

Figure 5.5: Normal distributions $\mathcal{N}(x_i^{(b)}, \sigma^2)$ and $\mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$.

boundary region for $\mathcal{N}(x_i^{(b)}, \sigma^2)$ is higher than the one for $\mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$, *i.e.*, for $x^{(2)} \sim \mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$ and $x^{(3)} \sim \mathcal{N}(x_i^{(b)}, \sigma^2)$, we conclude that

$$\mathbb{P}\left(x_i^{(-)} < X^{(3)} < x_i^{(+)}\right) > \mathbb{P}\left(x_i^{(-)} < X^{(2)} < x_i^{(+)}\right). \tag{5.16}$$

By combining Equations 5.15 and 5.16, we conclude that for $x^{(1)} \sim \mathcal{N}(x_i^{(b)}, \sigma_{min}^2)$ and $x^{(2)} \sim \mathcal{N}(x_i^{(b)} + \eta, \sigma^2)$, $\mathbb{P}\left(x_i^{(-)} < X^{(1)} < x_i^{(+)}\right) > \mathbb{P}\left(x_i^{(-)} < X^{(2)} < x_i^{(+)}\right)$. Consequently,

$$\mathbb{E}_{x^{(1)} \sim q^{(1)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(1)}]|S_i) \neq 0\}] < \mathbb{E}_{x^{(2)} \sim q^{(2)}}[\mathbb{1}\{\hat{L}(h_i'[x^{(2)}]|S_i) \neq 0\}]. \tag{5.17}$$

□

This implies that *ISAL* algorithm will select $q^{(1)}$ over $q^{(2)}$. Consequently, x_i will be drawn from $q^{(1)}$ rather than $q^{(2)}$. If the label of drawn instance x_i, y_i, is $+1$ then we update $x_{i+1}^{(+)} = x_i$ and $x_{i+1}^{(-)} = x_i^{(-)}$. Otherwise, $x_{i+1}^{(-)} = x_i$ and $x_{i+1}^{(+)} = x_i^{(+)}$. We continue this process until the size of the boundary region is less than ϵ. Note that $\sigma_{min} = 0$ implies that $q(x)$ is a delta function, thus the importance weight $\frac{p(x)}{q(x)}$ is always zero. To avoid this situation *ISAL* assumes that $\sigma_{min} > 0$.

5.2.1.1 Convergence Rate of *ISAL* on One-Dimensional Data

We are interested in bounding the number of iterations n required to shrink the size of the boundary region from its initial size $b_1 = x_1^{(+)} - x_1^{(-)}$ to less than $\epsilon > 0$. At iteration $i + 1$, the size of the boundary region is a function of $Z_i = (X_i, Y_i)$, *i.e.*,

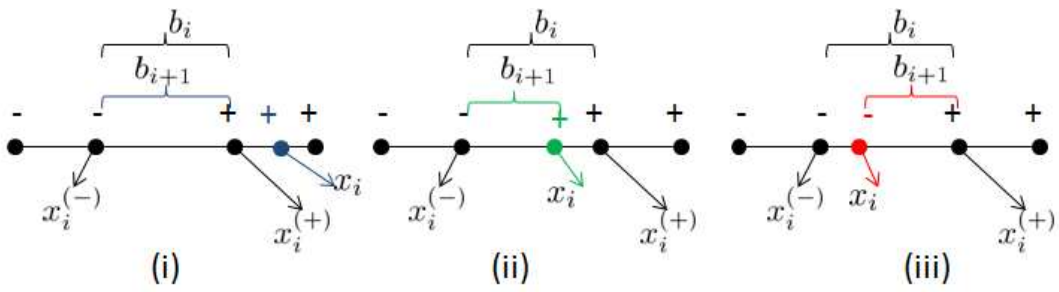

Figure 5.6: Change in the size of the boundary region from iteration i to $i+1$. (i) when x_i is outside of the boundary region, (ii) when x_i is inside the boundary region and is positive, (iii) when x_i is inside the boundary region and is negative.

$$B(Z_i) = \begin{cases} x_i^{(+)} - x_i^{(-)} & if \quad X_i \leq x_i^{(-)} \quad or \quad X_i \geq x_i^{(+)} \\ X_i - x_i^{(-)} & if \quad x_i^{(-)} < X_i < x_i^{(+)}, \; Y_i = +1 \\ x_i^{(+)} - X_i & if \quad x_i^{(-)} < X_i < x_i^{(+)}, \; Y_i = -1. \end{cases} \tag{5.18}$$

Figure 5.6 shows three different draws at iteration i, which are related to three cases of Equation 5.18. For example, in Figure 5.6(i) x_i is outside of the interval $(x_i^{(-)}, x_i^{(+)})$ so the boundary region does not change for the next iteration, *i.e.*, $x_{i+1}^{(-)} = x_i^{(-)}$ and $x_{i+1}^{(+)} = x_i^{(+)}$. However, in Figure 5.6(ii), x_i is inside the boundary region and is positive, therefore, $x_{i+1}^{(+)}$ will be updated to $x_{i+1}^{(+)} = x_i$. In addition, in Figure 5.6(iii) x_i is inside the boundary region and is negative, thus, $x_{i+1}^{(-)}$ will be updated to $x_{i+1}^{(-)} = x_i$.

The boundary region at iteration $i+1$, $(x_{i+1}^{(-)}, x_{i+1}^{(+)})$, is a function of Z_i and depends on the previous boundary region $(x_i^{(-)}, x_i^{(+)})$. Similarly, $(x_i^{(-)}, x_i^{(+)})$ is a function of Z_{i-1}. Therefore, Z_i and Z_{i-1} are dependent. In the same way, Z_{i-1} depends on Z_{i-2} and so on. Therefore, $Z_1, \ldots, Z_i$ are dependent random variables.

Lemma 3. *Let $\mathcal{X} \subseteq \mathbb{R}$, $\mathcal{Y} = \{-1, +1\}$, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Assume all $x \in \mathcal{X}$ can be perfectly classified by some hypothesis in $\mathcal{H}$. At iteration i of ISAL, let $(x_i^{(-)}, x_i^{(+)})$ be the boundary region, where $x_i^{(-)}$ and $x_i^{(+)}$ are defined in definition 3 and $b_i = x_i^{(+)} - x_i^{(-)}$. Let $q_i = \mathcal{N}(x_i^{(b)}, \sigma_i^2)$ be the density of the distribution for $X_i \in \mathcal{X}$. Let $B(Z_i)$ be the size of the boundary region at iteration $i+1$, as defined in Equation 5.18.*

Then the expected reduction in size of the boundary region at iteration $i+1$, given the previous boundary regions, is

$$\mathbb{E}\left[\frac{B(Z_i)}{b_i}\middle| Z_1, \ldots, Z_{i-1}\right] \leq \frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}\sigma_u}{\sqrt{\pi}}(1 - e^{\frac{-1}{8\sigma_u^2}}), \tag{5.19}$$

where Φ is the normal cumulative probability function and $\sigma_u = \frac{\sigma_i}{b_i}$.

Proof. Let $r(X_1, Y_1) = \frac{B(X_1,Y_1)}{b_1}$. By definition,

$$\mathbb{E}\left[r(X_1, Y_1)|x_1^{(-)}, x_1^{(+)}\right] = \int_{-\infty}^{\infty} q_1(x|x_1^{(-)}, x_1^{(+)}) \sum_{y\in\{-1,+1\}} P(y|x) r(x, y|x_1^{(-)}, x_1^{(+)}) dx. \tag{5.20}$$

We know that $P(Y_1 = +1|X_1 \geq x_1^{(+)}) = 1$ and $r(X_1, Y_1|x_1^{(-)}, x_1^{(+)}, X_1 \geq x_1^{(+)}) = 1$. In addition $P(Y_1 = -1|X_1 \leq x_1^{(-)}) = 1$ and $r(X_1, Y_1|x_1^{(-)}, x_1^{(+)}, X_1 \leq x_1^{(-)}) = 1$. Therefore,

$$\begin{aligned}
\mathbb{E}\left[r(X_1, Y_1)|x_1^{(-)}, x_1^{(+)}\right] =& \int_{-\infty}^{x_1^{(-)}} r(x, y|x_1^{(-)}, x_1^{(+)}) q_1(x|x_1^{(-)}, x_1^{(+)}) dx \\
&+ \int_{x_1^{(+)}}^{\infty} r(x, y|x_1^{(-)}, x_1^{(+)}) q_1(x|x_1^{(-)}, x_1^{(+)}) dx \\
&+ \int_{x_1^{(-)}}^{x_1^{(+)}} \sum_{y\in\{-1,+1\}} P(y|x) r(x, y|x_1^{(-)}, x_1^{(+)}) q_1(x|x_1^{(-)}, x_1^{(+)}) dx
\end{aligned} \tag{5.21}$$

$$\begin{aligned}
=& 2\Phi(\frac{x_1^{(-)} - x_1^{(b)}}{\sigma_1}) \\
&+ \int_{x_1^{(-)}}^{x_1^{(+)}} \sum_{y\in\{-1,+1\}} P(y|x) r(x, y|x_1^{(-)}, x_1^{(+)}) q_1(x|x_1^{(-)}, x_1^{(+)}) dx
\end{aligned} \tag{5.22}$$

$$\begin{aligned}
=& 2\Phi(\frac{x_1^{(-)} - x_1^{(b)}}{\sigma_1}) \\
&+ \int_{x_1^{(-)}}^{x_1^{(b)}} \sum_{y\in\{-1,+1\}} P(y|x) \frac{B(x, y|x_1^{(-)}, x_1^{(+)})}{b_1} q_1(x|x_1^{(-)}, x_1^{(+)}) dx \\
&+ \int_{x_1^{(b)}}^{x_1^{(+)}} \sum_{y\in\{-1,+1\}} P(y|x) \frac{B(x, y|x_1^{(-)}, x_1^{(+)})}{b_1} q_1(x|x_1^{(-)}, x_1^{(+)}) dx
\end{aligned} \tag{5.23}$$

$$\begin{aligned}
\leq& 2\Phi(\frac{x_1^{(-)} - x_1^{(b)}}{\sigma_1}) \\
&+ \frac{1}{b_1} \int_{x_1^{(-)}}^{x_1^{(b)}} (x_1^{(+)} - x) q_1(x|x_1^{(-)}, x_1^{(+)}) dx \\
&+ \frac{1}{b_1} \int_{x_1^{(b)}}^{x_1^{(+)}} (x - x_1^{(-)}) q_1(x|x_1^{(-)}, x_1^{(+)}) dx.
\end{aligned} \tag{5.24}$$

Note that the last inequality is obtained by knowing that $B(X_1, Y_1|x_1^{(-)}, x_1^{(+)}) \leq$

$x_1^{(+)} - X_1$, for $x_1^{(-)} < X_1 < x_1^{(b)}$ and, $B(X_1, Y_1|x_1^{(-)}, x_1^{(+)}) \leq X_1 - x_1^{(-)}$ for $x_1^{(b)} < X_1 < x_1^{(+)}$.

Throughout this proof, we assume that $x_1^{(-)}$ and $x_1^{(+)}$ are given and suppress them when it is clear from the context. We also use $Z_1 = (X_1, Y_1)$.

Now we compute the integrals in the right side of Equation 5.24 as

$$\int_{x_1^{(-)}}^{x_1^{(b)}} (x_1^{(+)} - x) q_1(x) dx = \int_{x_1^{(-)}}^{x_1^{(b)}} (x_1^{(+)} - x) \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x - x_1^{(b)})^2}{2\sigma_1^2}} dx. \tag{5.25}$$

Using a change of variable $u = \frac{x - x_1^{(b)}}{x_1^{(+)} - x_1^{(-)}}$, $\sigma_u = \frac{\sigma_1}{x_1^{(+)} - x_1^{(-)}} = \frac{\sigma_1}{b_1}$ (knowing that $b_1 = x_1^{(+)} - x_1^{(-)}$), we have

$$\int_{x_1^{(-)}}^{x_1^{(b)}} (x_1^{(+)} - x) q_1(x) dx = \frac{b_1}{\sqrt{2\pi}\sigma_u} \int_{-0.5}^{0} (0.5 - u) e^{-\frac{u^2}{2\sigma_u^2}} du \tag{5.26}$$

$$= \frac{b_1}{\sqrt{2\pi}\sigma_u} \left(\frac{\sqrt{2\pi}\sigma_u}{2} (\frac{1}{2} - \Phi(\frac{-1}{2\sigma_u})) - \sigma_u^2 e^{\frac{-1}{8\sigma_u^2}} + \sigma_u^2 \right) \tag{5.27}$$

$$= b_1 \left(\frac{1}{4} - \frac{1}{2}\Phi(\frac{-1}{2\sigma_u}) + \frac{\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}})}{\sqrt{2\pi}} \right). \tag{5.28}$$

Similarly,

$$\int_{x_1^{(b)}}^{x_1^{(+)}} (x - x_1^{(-)}) q_1(x) dx = b_1 \left(\frac{1}{4} - \frac{1}{2}\Phi(\frac{-1}{2\sigma_u}) + \frac{\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}})}{\sqrt{2\pi}} \right). \tag{5.29}$$

Therefore, Equation 5.24 can be written as

$$\mathbb{E}\left[r(Z_1)\right] \leq 2\Phi(\frac{-1}{2\sigma_u}) + 2 \left(\frac{1}{4} - \frac{1}{2}\Phi(\frac{-1}{2\sigma_u}) + \frac{\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}})}{\sqrt{2\pi}} \right)$$

$$= \frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}}), \tag{5.30}$$

where $\sigma_u = \frac{\sigma_1}{b_1}$ and $b_1 = x_1^{(+)} - x_1^{(-)}$.

In other words,

$$\mathbb{E}\left[\frac{B(Z_1)}{b_1}\right] \leq \frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}}). \tag{5.31}$$

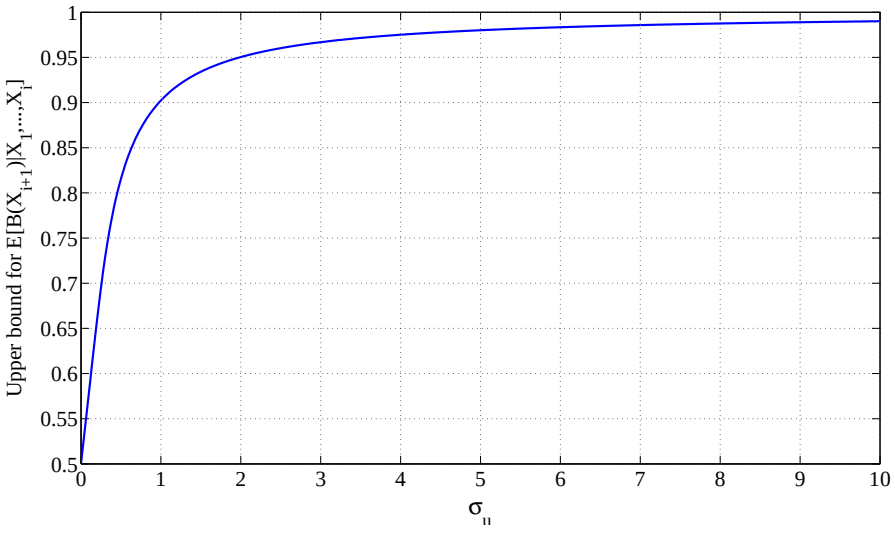

Figure 5.7: Upper bound for the expected reduction in size of boundary region versus the variance of sampling distribution.

Figure 5.7 shows the value of $\frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}})$ versus σ_u. As σ_u gets smaller, the upper bound gets closer to $\frac{1}{2}$. For example, if $\sigma_u < \frac{1}{2}$, *i.e.*, the standard deviation of q_1 is less than half the boundary size, then $\mathbb{E}\left[\frac{B(Z_1)}{b_1}\right] < 0.82$.

We also have

$$\lim_{\sigma_u \to \infty} \left(\frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}}) \right) = 1, \tag{5.32}$$

$$\lim_{\sigma_u \to 0} \left(\frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}}) \right) = \frac{1}{2}. \tag{5.33}$$

As $\left(\frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}}) \right)$ is a monotonically increasing function of σ_u, from Equations 5.31 and 5.32 we get, for any $\sigma_u > 0$, $\mathbb{E}\left[\frac{B(Z_1)}{b_1}\right] < 1$.

In general, at iteration i of *ISAL*, $z_1, \ldots, z_{i-1}$ were drawn in the previous iterations and $x_i^{(-)}$ and $x_i^{(+)}$ are now known.

The upper bound for $\mathbb{E}\left[B(Z_i)|Z_1, \ldots, Z_{i-1}\right]$ is computed as

$$\mathbb{E}\left[B(Z_i)|Z_1, \ldots, Z_{i-1}\right] \leq b_i \Phi(\frac{x_i^{(-)} - x_i^{(b)}}{\sigma_i}) + \int_{x_i^{(-)}}^{x_i^{(b)}} (x_i^{(+)} - x) q_i(x|z_1, \ldots, z_{i-1}) dx$$

$$+ \int_{x_i^{(b)}}^{x_i^{(+)}} (x - x_i^{(-)}) q_i(x|z_1, \ldots, z_{i-1}) dx + b_i(1 - \Phi(\frac{x_i^{(+)} - x_i^{(b)}}{\sigma_i})). \tag{5.34}$$

After computing the integrals in the right side of Equation 5.34, we get

$$\mathbb{E}\left[\frac{B(Z_i)}{b_i}\Big| Z_1, \ldots, Z_{i-1}\right] \leq \frac{1}{2} + \Phi(\frac{-1}{2\sigma_u}) + \frac{\sqrt{2}}{\sqrt{\pi}}\sigma_u(1 - e^{\frac{-1}{8\sigma_u^2}}), \tag{5.35}$$

where $\sigma_u = \frac{\sigma_i}{b_i}$. □

To have the same upper bound for $\mathbb{E}\left[\frac{B(Z_{i-1})}{b_{i-1}}\Big| Z_1, \ldots, Z_{i-2}\right]$ and $\mathbb{E}\Big[\frac{B(Z_i)}{b_i}\Big| Z_1, \ldots, Z_{i-1}\Big]$, for $i = 1, \ldots, n$, we make the following assumption:

Assumption 1. *At the i-th iteration of ISAL, we assume that $\frac{\sigma_i}{b_i} \leq \sigma_{up}$, where $\sigma_{up} \in \mathbb{R}^+$, σ_i is the standard deviation of the sampling distribution and b_i is the size of the boundary region.*

Theorem 2. *Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subseteq \mathbb{R}$, $\mathcal{Y} = \{-1, +1\}$ and assume that Assumption 1 holds, and that all $x \in \mathcal{X}$ can be perfectly classified by some hypothesis in $\mathcal{H}$. Let $(Z_1(p_1), \ldots, Z_n(p_n))$ be a parametric class of random variables with the property that $Z_i \sim p_i$. At iteration i of ISAL, let $(x_i^{(-)}, x_i^{(+)})$ be the boundary region, where $x_i^{(-)}$ and $x_i^{(+)}$ are defined in Definition 3. Fix an $\epsilon \in \mathbb{R}^+$ and assume $x_1^{(-)}$ and $x_1^{(+)}$ are given. Then the expected number of iterations required by ISAL to reduce the size of the boundary region to less than ϵ is $O(\log \frac{1}{\epsilon})$.*

Proof. $B(Z_{n-1})$ is the size of boundary region at iteration n as defined in Equation 5.18. The expected size of boundary region at iteration n can be written as

$$\mathbb{E}\left[B(Z_{n-1})\right] = b_1 \mathbb{E}\left[\frac{B(Z_1)}{b_1}\frac{B(Z_2)}{B(Z_1)} \cdots \frac{B(Z_{n-1})}{B(Z_{n-2})}\right] \tag{5.36}$$

$$= b_1 \int_{z_1} p_1(z_1) \mathbb{E}\left[\frac{B(Z_1)}{b_1}\frac{B(Z_2)}{B(Z_1)} \cdots \frac{B(Z_{n-1})}{B(Z_{n-2})}\Big| Z_1 = z_1\right] dz_1 \tag{5.37}$$

$$= b_1 \int_{z_1} p_1(z_1) \ldots \int_{z_{n-2}} p_{n-2}(z_{n-2}|z_1, \ldots, z_{n-3}) \times$$
$$\mathbb{E}\left[\frac{B(Z_1)}{b_1}\frac{B(Z_2)}{B(Z_1)} \cdots \frac{B(Z_{n-1})}{B(Z_{n-2})}\Big| Z_1 = z_1, \ldots, Z_{n-2} = z_{n-2}\right] dz_{n-2} \ldots dz_1. \tag{5.38}$$

$\frac{B(Z_1)}{b_1}\frac{B(Z_2)}{B(Z_1)} \cdots \frac{B(Z_{n-2})}{B(Z_{n-3})}$ can be pulled out of the conditional expectation as

$$\mathbb{E}\left[B(Z_{n-1})\right] = b_1 \int_{z_1} p_1(z_1) \dots \int_{z_{n-2}} p_{n-2}(z_{n-2}|z_1, \dots, z_{n-3}) \frac{B(z_1)}{b_1} \dots \frac{B(z_{n-2})}{B(z_{n-3})} \times \mathbb{E}\left[\frac{B(Z_{n-1})}{B(Z_{n-2})}\Big| Z_1 = z_1, \dots, Z_{n-2} = z_{n-2}\right] dz_{n-2} \dots dz_1. \tag{5.39}$$

From lemma 3 and assumption 1, we have (note that $B(z_{n-2}) = b_{n-1}$)

$$\mathbb{E}\left[\frac{B(Z_{n-1})}{B(Z_{n-2})}\Big| Z_1 = z_1, \dots, Z_{n-2} = z_{n-2}\right] \leq \frac{1}{2} + \Phi(\frac{-1}{2\sigma_{up}}) + \frac{\sqrt{2}\sigma_{up}}{\sqrt{\pi}}(1 - e^{\frac{-1}{8\sigma_{up}^2}}).$$

Let $\alpha_{up} = \frac{1}{2} + \Phi(\frac{-1}{2\sigma_{up}}) + \frac{\sqrt{2}\sigma_{up}}{\sqrt{\pi}}(1 - e^{\frac{-1}{8\sigma_{up}^2}})$, which means $\frac{1}{2} < \alpha_{up} < 1$. Therefore,

$$\mathbb{E}\left[B(Z_{n-1})\right] \leq b_1 \int_{z_1} p_1(z_1) \dots \int_{z_{n-2}} p_{n-2}(z_{n-2}|z_1, \dots, z_{n-3}) \times \frac{B(z_1)}{b_1} \frac{B(z_2)}{B(z_1)} \dots \frac{B(z_{n-2})}{B(z_{n-3})} \alpha_{up}\, dz_{n-2} \dots dz_1. \tag{5.40}$$

Note that $\frac{B(z_1)}{b_1}\frac{B(z_2)}{B(z_1)} \dots \frac{B(z_{n-3})}{B(z_{n-4})}$ can be pulled out of the last integral over z_{n-2} as:

$$\mathbb{E}\left[B(Z_{n-1})\right] \leq \alpha_{up} b_1 \int_{z_1} p_1(z_1) \dots \int_{z_{n-3}} p_{n-3}(z_{n-3}|z_1, \dots, z_{n-4}) \frac{B(z_1)}{b_1} \dots \frac{B(z_{n-3})}{B(z_{n-4})} \times \int_{z_{n-2}} p_{n-2}(z_{n-2}|z_1, \dots, z_{n-3}) \frac{B(z_{n-2})}{B(z_{n-3})} dz_{n-2} \dots dz_1. \tag{5.41}$$

By definition,

$$\mathbb{E}\left[\frac{B(Z_{n-2})}{B(z_{n-3})}\Big| Z_1 = z_1, ..., Z_{n-3} = z_{n-3}\right] = \int_{z_{n-2}} p_{n-2}(z_{n-2}|z_1, ..., z_{n-3}) \frac{B(z_{n-2})}{B(z_{n-3})} dz_{n-2}.$$

Therefore, the above equation can be written as:

$$\mathbb{E}\left[B(Z_{n-1})\right] \leq \alpha_{up} b_1 \int_{z_1} p_1(z_1) \dots \int_{z_{n-3}} p_{n-3}(z_{n-3}|z_1, \dots, z_{n-4}) \frac{B(z_1)}{b_1} \dots \frac{B(z_{n-3})}{B(z_{n-4})} \times \mathbb{E}\left[\frac{B(Z_{n-2})}{B(z_{n-3})}\Big| Z_1 = z_1, \dots, Z_{n-3} = z_{n-3}\right] dz_{n-3} \dots dz_1. \tag{5.42}$$

Again from lemma 3 and assumption 1, we have

$$\mathbb{E}\left[\frac{B(Z_{n-2})}{B(z_{n-3})}\Big| Z_1 = z_1, \dots, Z_{n-3} = z_{n-3}\right] \leq \alpha_{up},$$

where $B(z_{n-2}) = b_{n-1}$. Therefore,

$$\mathbb{E}\left[B(Z_{n-1})\right] \leq \alpha_{up}^2\, b_1 \int_{z_1} p_1(z_1) ... \int_{z_{n-3}} p_{n-3}(z_{n-3}|z_1, ..., z_{n-4}) \times \frac{B(z_1)}{b_1} ... \frac{B(z_{n-3})}{B(z_{n-4})} dz_{n-3} ... dz_1. \tag{5.43}$$

This process continues until we get to

$$\mathbb{E}\left[B(Z_{n-1})\right] \leq \alpha_{up}^n\, b_1. \tag{5.44}$$

To guarantee $\mathbb{E}\left[B(Z_{n-1})\right] \leq \epsilon$, we use its upper bound as

$$\alpha_{up}^n\, b_1 \leq \epsilon \tag{5.45}$$

$$\alpha_{up}^n \geq \frac{\epsilon}{b_1} \tag{5.46}$$

$$n \geq \log_{\alpha_{up}} \frac{1}{b_1} + \log_{\alpha_{up}} \epsilon. \tag{5.47}$$

Therefore, for a given $\frac{1}{2} < \alpha_{up} < 1$, the expected number of iterations required to obtain a boundary size of ϵ is $\log_{\alpha_{up}} \epsilon + \log_{\alpha_{up}} \frac{1}{b_1}$, which is equivalent to $\log_{\frac{1}{\alpha_{up}}} \frac{1}{\epsilon} + \log_{\frac{1}{\alpha_{up}}} b_1$, so can be written as $O(\log \frac{1}{\epsilon})$.

□

Equation 5.47 shows that the convergence rate of *ISAL* is $\log_{\alpha_{up}} \epsilon$, assuming that $b_1 = 1$. To have convergence rate faster than passive learning, we need $\log_{\alpha_{up}} \epsilon < \frac{1}{\epsilon}$. In other words, α_{up} should be less than ϵ^ϵ. For example, when $\epsilon = 0.01$ then α_{up} should be less than 0.955, which requires $\sigma_{up} \leq 2$ (see Figure 5.7).

5.2.1.2 Analysis of *ISAL* When the Sampling Distribution Partially Overlaps With Decision Boundary

In the scenario that the sampling distribution is not centered in the middle of the boundary region (*e.g.*, dashed curve in Figure 5.5), $\mathbb{E}\left[r(Z_i)|Z_1, \ldots, Z_{i-1}\right]$ is computed as

$$\begin{aligned}\mathbb{E}\left[r(Z_i)|Z_1, \ldots, Z_{i-1}\right] =& \mathbb{P}\left(X_i \leq x_i^{(-)}\right) \times 1 + \mathbb{P}\left(X_i \geq x_i^{(+)}\right) \times 1 \\ &+ \mathbb{P}\left(x_i^{(-)} < X_i < x_i^{(+)}\right) \times \alpha_i,\end{aligned} \tag{5.48}$$

where $r(Z_i) = \frac{B(Z_i)}{b_i}$ and $\frac{1}{2} < \alpha_i < 1$. Recall that the size of the boundary region does not change when $X_i \leq x_i^{(-)}$ or $X_i \geq x_i^{(+)}$. If the sampling distribution q_i has the property

that $q_i(x) \neq 0$ for all $x \in \mathcal{X}$ (*e.g.*, Gaussian distribution), then $\mathbb{P}\left(x_i^{(-)} < X_i < x_i^{(+)}\right) > c$ for some constant $c > 0$. Consequently,

$$\mathbb{E}\left[r(Z_i)|Z_1, \ldots, Z_{i-1}\right] < 1. \tag{5.49}$$

The above equation means if the sampling distribution q_i is not centered in the middle of the boundary region, the expected size of the boundary region at iteration $i+1$ is less than its size at iteration i. This characteristic of the algorithm implies that it converges to the optimal classifier even if the exact location of the boundary region is not known.

From Equation 5.34, $\mathbb{E}\left[r(Z_i)|Z_1, \ldots, Z_{i-1}\right]$ given $q_i = \mathcal{N}(\mu_i, \sigma_i^2)$, is

$$\begin{aligned}\mathbb{E}\left[r(Z_i)|Z_1, \ldots, Z_{i-1}\right] \leq & \Phi(\frac{x_i^{(-)} - \mu_i}{\sigma_i}) + \int_{x_i^{(-)}}^{x_i^{(b)}} (x_i^{(+)} - x) q_i(x|z_1, \ldots, z_{i-1}) dx \\ & + \int_{x_i^{(b)}}^{x_i^{(+)}} (x - x_i^{(-)}) q_i(x|z_1, \ldots, z_{i-1}) dx + 1 - \Phi(\frac{x_i^{(+)} - \mu_i}{\sigma_i})\end{aligned} \tag{5.50}$$

$$\begin{aligned}= & 1 + \frac{\sigma_i^2}{\sqrt{2\pi}}\Big(2e^{-\frac{(x_i^{(b)} - \mu_i)^2}{2\sigma_i^2}} - e^{-\frac{(x_i^{(+)} - \mu_i)^2}{2\sigma_i^2}} - e^{-\frac{(x_i^{(-)} - \mu_i)^2}{2\sigma_i^2}}\Big) \\ & + \Phi(\frac{x_i^{(+)} - \mu_i}{\sigma_i})(\mu_i - x_i^{(-)} - 1) \\ & + \Phi(\frac{x_i^{(-)} - \mu_i}{\sigma_i})(\mu_i - x_i^{(+)} + 1) \\ & + \Phi(\frac{x_i^{(b)} - \mu_i}{\sigma_i})(-2\mu_i + x_i^{(+)} + x_i^{(-)}).\end{aligned} \tag{5.51}$$

Note that the right side of Equation 5.51 depends on $x_i^{(+)}$, $x_i^{(-)}$, μ_i, σ_i; moreover, unlike Equation 5.34, it can not be simplified further. To ensure a convergence rate faster than passive learning, the right side of Equation 5.51 has to be less than ϵ^ϵ. Given $x_i^{(+)}$, $x_i^{(-)}$, μ_i, σ_i, one has to compute the value of α_i and verify that it is less than ϵ^ϵ.

If we normalize the data such that $x_i^{(-)} = -0.5$, $x_i^{(+)} = 0.5$, and μ_u and σ_u are the normalized versions of μ_i and σ_i respectively, then Equation 5.51 can be written as

$$\begin{aligned}\mathbb{E}\left[r(Z_i)|Z_1, \ldots, Z_{i-1}\right] \leq & 1 + \frac{\sigma_u^2}{\sqrt{2\pi}}\Big(2e^{-\frac{\mu_u^2}{2\sigma_u^2}} - e^{-\frac{(0.5 - \mu_u)^2}{2\sigma_u^2}} - e^{-\frac{(-0.5 - \mu_u)^2}{2\sigma_u^2}}\Big) \\ & + \Phi(\frac{0.5 - \mu_u}{\sigma_u})(\mu_u - 0.5) \\ & + \Phi(\frac{-0.5 - \mu_u}{\sigma_u})(\mu_u + 0.5) \\ & + \Phi(\frac{-\mu_u}{\sigma_u})(-2\mu_u).\end{aligned} \tag{5.52}$$

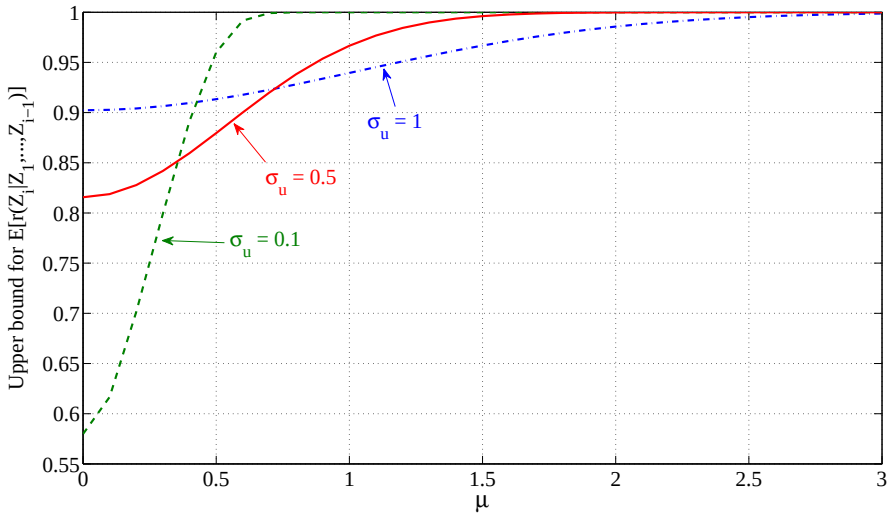

Figure 5.8: Upper bound for $\mathbb{E}\left[r(Z_i)|Z_1,\ldots,Z_{i-1}\right]$ when the sampling distribution is $\mathcal{N}(\mu_u,\sigma_u^2)$ and the data is normalized such that the middle of boundary region is the origin.

Figure 5.8 shows the upper bound for $\mathbb{E}\left[r(Z_i)|Z_1,\ldots,Z_{i-1}\right]$ versus μ_u for three values of σ_u. For example, $\mu_u = 0$ is the case where the distribution is centered in the middle of the boundary region. As μ_u moves further away from the middle of the boundary region, $\mathbb{E}\left[r(Z_i)|Z_1,\ldots,Z_{i-1}\right]$ gets closer to 1. Similar to the previous section, to obtain a faster convergence rate for *ISAL* where $\epsilon = 0.01$, $\mathbb{E}\left[r(Z_i)|Z_1,\ldots,Z_{i-1}\right]$ should be less than 0.955 in each iteration. For example, $\sigma_u = 0.1$ and $0 \leq \mu_u \leq 0.5$ satisfy this condition; see Figure 5.8.

5.2.2 Analysis of *ISAL* on Realizable Multi-Dimensional Data

In this section, we consider a commonly studied setting in the active learning literature [30] [28] [69], as described in Assumption 2.

Assumption 2. *Instances are located uniformly on a unit ball in $\mathbb{R}^d$. Here the Bayes classifier is a linear separator h_{Bayes} with parameter w_{Bayes} that goes through the origin (e.g., dashed line in Figure 5.9).*

In this analysis we consider the margin-based classifiers with linear separators. In margin-based classification, the optimization problem is

$$\begin{array}{ll} \textit{maximize} & b \\ \textit{subject to} & w^T \cdot x_j y_j \geq b, \quad j = 1,\ldots,i, \\ & ||w|| = 1, \end{array}$$

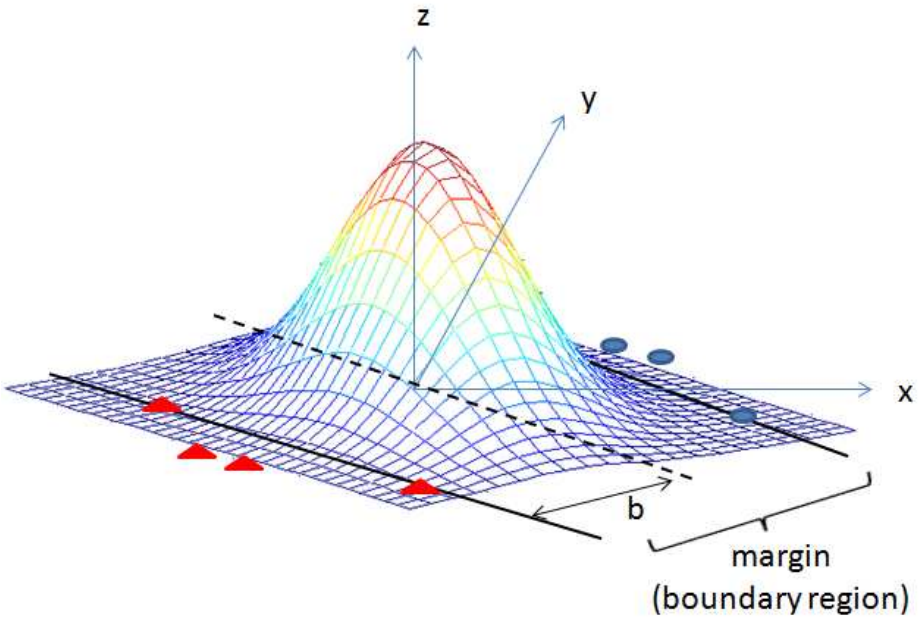

Figure 5.9: Two dimensional data with a margin-based classifier. *ISAL* samples from a distribution that has more weight inside the margin.

where w is the parameter of the linear separator and b is the margin parameter [36].

Definition 4. *Let $\mathcal{H}$ be the class of real measurable functions on $\mathcal{X}$. For $\alpha > 0$, a sequence $(x_1, \ldots, x_m) \in \mathcal{X}^m$ is said to be α-shattered by $\mathcal{H}$ if there is an $(r_1, \ldots, r_m) \in \mathbb{R}^m$ such that for each $(y_1, \ldots, y_m) \in \{-1, 1\}^m$, there is a function $h \in \mathcal{H}$ satisfying for all $i = 1, \ldots, m$,*

$$h(x_i) \geq r_i + \alpha \quad \textit{if} \quad y_i = 1,$$
$$h(x_i) \leq r_i - \alpha \quad \textit{if} \quad y_i = -1.$$

*The **fat shattering dimension** of $\mathcal{H}$ at scale α, denoted by $fat_\alpha(\mathcal{H})$, is the size of a largest α-shattered set [70].*

Intuitively, the fat shattering dimension is the size of the largest set that can be shattered with a function $h \in \mathcal{H}$, while leaving a margin of error α, *i.e.*, in all cases, all points are outside the margin. Figure 5.10 shows 3 points that are α-shattered by a linear classifier.

The sample complexity of PAC learning has a general lower bound of $\Omega(\frac{VC(\mathcal{H})}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta})$, where $VC(\mathcal{H})$ is the VC dimension of $\mathcal{H}$ [71]. The VC dimension of class of

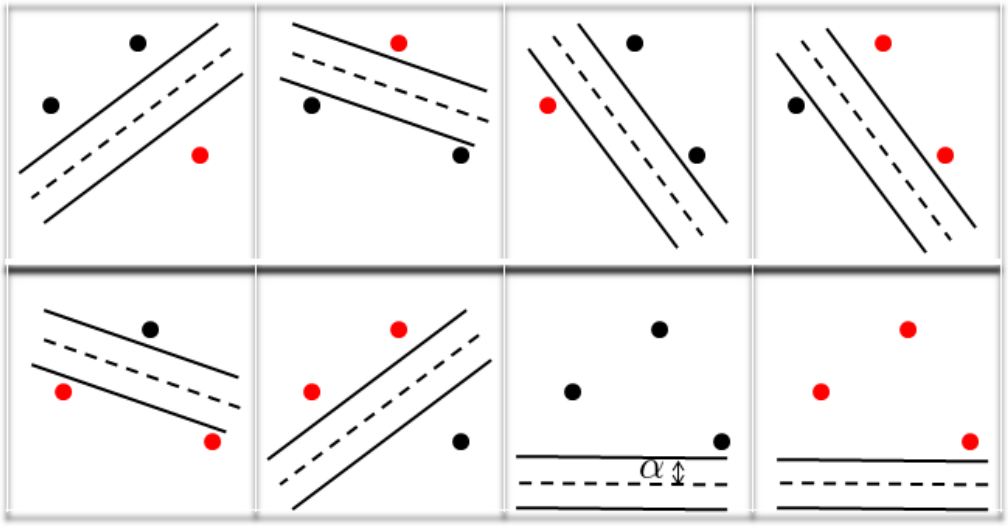

Figure 5.10: 3 points that are α-shattered by linear classifiers

hypotheses $\mathcal{H}$ over instance space $\mathcal{X}$ is the cardinality of largest set of points that can be shattered by $\mathcal{H}$ (corresponding to $\alpha = 0$ in definition 4). Under Assumption 2, the sample complexity of PAC learning has a lower bound of

$$n = \Omega(\frac{1}{\epsilon}(d + \log\frac{1}{\delta})) \tag{5.53}$$

since the VC dimension of homogeneous half spaces on a unit ball in $\mathbb{R}^d$ is d [68]. We consider a realizable case where $\mathbb{P}\left(w_{Bayes} \cdot xy \leq 0\right) = 0$, where w_{Bayes} is the parameter of the Bayes classifier, which is linear under assumption 2. At iteration i, h_i^* is the linear classifier that is consistent with the labeled set S_i. By standard VC bounds (Equation 5.53), a sample size of $\tilde{O}(2^i d)$ is required to ensure $L(h_i^*) \leq \frac{1}{2^i}$ (with high probability); this implies that $\tilde{O}(2^i d)$ new labeled instances should be added to ensure that $L(h_i^*)$ is halved [28]. The goal is to show that we can set the margin parameter b_i such that only a fraction of $\tilde{O}(2^i d)$ instances require a label from the oracle.

Balcan *et al.* [28] showed that, for the above settings and margin-based classifiers, active learning can obtain an exponential convergence rate if instances are queried inside [1] the margin. They proved that there exists a constant C such that, for any $\epsilon, \delta > 0$, and the margin parameter $b_i = \frac{\pi}{2^{i-1}}$ at iteration i, if the active learner samples $Cd^{\frac{1}{2}}(d\ln d + \ln\frac{i}{\delta})$ instances inside the margin, then with probability of at least $1-\delta$, after $\lceil \log_2 \frac{1}{\epsilon} \rceil$ iterations, it can find a classifier with error of at most ϵ. In other words, they showed that at iteration

[1] For x inside the margin we have $|w_i^* \cdot x| \leq b_i$.

i, with the margin parameters of $b_i = O(2^{-i})$, active learning obtains a convergence rate of $\tilde{O}(d^{\frac{3}{2}} \log \frac{1}{\epsilon})$. Consequently, they used a rejection sampling strategy to reject the instances outside the margin, and only request the label for instances inside the margin.

Theorem 3. *Let $q_i = \mathcal{N}(\mu_i, \Sigma_i)$, where μ_i is on the linear separator and Σ_i is such that more than half the q_i's weight is inside the margin. There exists a constant C such that, for any $\epsilon, \delta > 0$, if $r_{max} C d^{\frac{1}{2}}(d \ln d + \ln \frac{i}{\delta})$ instances are drawn from q_i at i-th iteration, with margin parameter $b_i = \frac{\pi}{2^{i-1}}$, then with probability of at least $1 - \delta$, after $n = \lceil \log_2 \frac{1}{\epsilon} \rceil$ iterations, we obtain a classifier $h_n^* = \arg\min_{h \in \mathcal{H}} \hat{L}(h|S_n)$ with error of at most ϵ.*

Proof. We actively sample the instances from the distribution q_i, whose weight inside the margin is more than half. Therefore, the instances drawn are more likely (with probability of $\zeta > \frac{1}{2}$) to be inside the margin. Asymptotically, if we sample n instances, then $n\zeta$ of them will be inside the margin. Based on the results by Balcan *et al.* [28] (Theorem 1), we know that, for margin-based classifiers, if at i-th iteration for some constant $C' > 0$, $C' d^{\frac{1}{2}}(d \ln d + \ln \frac{i}{\delta})$ instances are drawn inside the margin, then with probability at least $1 - \delta$, we find a classifier with the (expected) error of at most ϵ. By using the weighted loss function of definition 2 with the maximum weight of r_{max}, $\frac{r_{max} C'}{\zeta} d^{\frac{1}{2}}(d \ln d + \ln \frac{i}{\delta}) < 2 r_{max} C' d^{\frac{1}{2}}(d \ln d + \ln \frac{i}{\delta}) = r_{max} C d^{\frac{1}{2}}(d \ln d + \ln \frac{i}{\delta})$ instances are required in i-th iteration, where $C = 2C'$. □

Corollary 1. *There exits a constant C such that, for any $\epsilon, \delta > 0$, using ISAL algorithm with margin parameter $b_i = \frac{\pi}{2^{1+(i-1)\beta}}$ at iteration i, where $\beta = \log_2(1 + \frac{1}{d\sqrt{d}})$, after $n = r_{max} C(\lceil \log_2 \frac{1}{\epsilon} \rceil) d^{\frac{1}{2}}(d \ln d + \ln \frac{\lceil \log_2 \frac{1}{\epsilon} \rceil}{\delta})$ iterations, ISAL returns a classifier whose error is at most ϵ, with probability at least $1 - \delta$.*

Proof. Given that the data is uniformly distributed over a unit ball in $\mathbb{R}^d$, the error rate of any linear classifier h, which passes through the origin, is $L(h) = \frac{\theta(h, h_{Bayes})}{\pi}$, where $\theta(h, h_{Bayes})$ is the angle between h and h_{Bayes}. Therefore, $L(h_i^*) \leq 2^{-m}$ implies $||w_i^* - w_{Bayes}||_2 \leq 2^{-m}\pi$, where $m > 0$ and w_i^* and w_{Bayes} are the parameter of h_i^* and h_{Bayes} respectively. Consequently, both h_i^* and h_{Bayes} predict the same label for instances whenever $|w_i^* \cdot x| \geq 2^{-m}\pi$. As we mentioned before, the probability of an arbitrary $x \in \mathcal{X}$ such that $|w_i^* \cdot x| \leq 2^{-m}\pi$ is $\tilde{O}(2^{-m}\sqrt{d})$, because for $d > 1$, the projection of uniform random variables in the unit ball onto one-dimension can be approximated by a Gaussian distribution with variance $\frac{1}{d}$ [28]. If in the i-th iteration, we set $b_i = O(2^{-m})$ and draw $\tilde{O}(\frac{2^m}{\sqrt{d}})$ instances, then only $\tilde{O}(1)$ of the new instances have $|w_i^* \cdot x| \leq 2^{-m}\pi$,

i.e., require the label from an oracle. Then the error of h^*_{i+1} is $L(h^*_{i+1}) < \frac{1}{2^m(1+\frac{1}{d\sqrt{d}})}$. Let $\beta = \log_2(1+\frac{1}{d\sqrt{d}})$, then $L(h^*_{i+1}) \leq 2^{-(m+\beta)}$, *i.e.*, the error of h^*_{i+1} is reduced by a factor of $2^{-\beta} = \frac{1}{1+\frac{1}{d\sqrt{d}}}$ from the error of h^*_i. Similarly, $L(h^*_{i+2}) \leq 2^{-(m+2\beta)}$. Similar to Theorem 3, for some constant $C' > 0$, after $\frac{r_{max}C'}{\zeta} d^{\frac{1}{2}}(d\ln d + \ln \frac{\lceil \log_2 \frac{1}{\epsilon} \rceil}{\delta})$ iterations, $L(h^*_i)$ is at least halved. This implies that a total of $n = \frac{1}{\zeta}\sum_{i=1}^{\lceil \log_2 \frac{1}{\epsilon} \rceil} r_{max}C' d^{\frac{1}{2}}(d\ln d + \ln \frac{i}{\delta})$ iterations of *ISAL* is sufficient to find a classifier with the error of at most ϵ, with probability of at least $1-\delta$. We can bound n as

$$n = \frac{1}{\zeta}\sum_{i=1}^{\lceil \log_2 \frac{1}{\epsilon} \rceil} r_{max}C' d^{\frac{1}{2}}(d\ln d + \ln \frac{i}{\delta}) < \frac{r_{max}C'}{\zeta}(\lceil \log_2 \frac{1}{\epsilon} \rceil) d^{\frac{1}{2}}(d\ln d + \ln \frac{\lceil \log_2 \frac{1}{\epsilon} \rceil}{\delta}) \tag{5.54}$$

$$< 2r_{max}C'(\lceil \log_2 \frac{1}{\epsilon} \rceil) d^{\frac{1}{2}}(d\ln d + \ln \frac{\lceil \log_2 \frac{1}{\epsilon} \rceil}{\delta}) \tag{5.55}$$

$$= r_{max}C(\lceil \log_2 \frac{1}{\epsilon} \rceil) d^{\frac{1}{2}}(d\ln d + \ln \frac{\lceil \log_2 \frac{1}{\epsilon} \rceil}{\delta}) \tag{5.56}$$

□

For $d \geq 4$, a faster convergence rate for *ISAL* can be achieved as:

Theorem 4. *Let $q_i = \mathcal{N}(\mu_i, \Sigma_i)$, where μ_i is on the linear separator and Σ_i is such that more than half the q_i's weight is inside the margin. There exists a constant C such that, for any $d \geq 4$ and any $\delta > 0$, $0 < \epsilon < 1/4$, if $r_{max}C\sqrt{\ln(1+i)}(d\ln(1+\ln i) + \ln(\frac{i}{\delta}))$ instances are drawn from q_i at i-th iteration, with margin parameter $b_i = 2^{1-i}\pi d^{-1/2}\sqrt{5+\ln(1+i)}$, then with probability of at least $1-\delta$, after $n = \lceil \log_2 \frac{1}{\epsilon} \rceil - 2$ iterations, we obtain a classifier $h^*_n = \arg\min_{h\in\mathcal{H}} \hat{L}(h|S_n)$ with error of at most ϵ.*

Proof. We actively sample the instances from the distribution q_i, whose weight inside the margin is more than half. Therefore, the drawn instances are more likely (with probability of $\zeta > \frac{1}{2}$) to be inside the margin. Asymptotically, if we sample n instances, then $n\zeta$ of them will be inside the margin. Based on the results by Balcan *et al.* [28] (Theorem 2), we know that for margin-based classifiers, if at i-th iteration for some constant $C' > 0$, $C'\sqrt{\ln(1+i)}\left(d\ln(1+\ln i) + \ln(\frac{i}{\delta})\right)$ instances are drawn inside the margin, then with probability at least $1-\delta$, we find a classifier with the error of at most ϵ.

By using the weighted loss function of definition 2 with the maximum weight of r_{max}, $\frac{r_{max}C'}{\zeta}\sqrt{\ln(1+i)}\left(d\ln(1+\ln i) + \ln(\frac{i}{\delta})\right) \leq 2r_{max}C'\sqrt{\ln(1+i)}(d\ln(1+\ln i)$

$+\ln(\frac{i}{\delta})) = r_{max}C\sqrt{\ln(1+i)}(d\ln(1+\ln i)+\ln(\frac{i}{\delta}))$ instances are required in i-th iteration, where $C = 2C'$. □

Corollary 2. *There exits a constant C such that for $d \geq 4$ and any $\epsilon, \delta > 0$, $\epsilon < 1/4$, using ISAL algorithm with margin parameter $b_i = 2^{1-i}\pi d^{-1/2}\sqrt{5+\ln(1+i)}$ at iteration i, after*

$$n = r_{max}C\left(\left\lceil \log_2 \frac{1}{\epsilon}\right\rceil - 2\right)\sqrt{\ln\left(\lceil \log_2 \frac{1}{\epsilon}\rceil - 1\right)} \times \left[d\ln\left(1+\ln\left(\lceil \log_2 \frac{1}{\epsilon}\rceil - 2\right)\right) + \ln\left(\frac{\lceil \log_2 \frac{1}{\epsilon}\rceil - 2}{\delta}\right)\right] \tag{5.57}$$

iterations, ISAL returns a classifier whose error is at most ϵ, with probability at least $1-\delta$.

5.2.3 General Version of the Algorithm (*ISALg*)

ISAL algorithm of Figure 5.2 can be generalized by replacing (line 3) $\arg\min_{q\in\mathcal{Q}} \mathbb{E}_{x\sim q}[\mathbb{1}\{\hat{L}_{i-1}\ (h'_{i-1}[x, h^*_{i-1}(x)]\big|S_{i-1}) \neq 0\}]$ with $\arg\min_{q\in\mathcal{Q}} \mathbb{E}_{x\sim q}[\hat{L}_{i-1}(h'_{i-1}[x, h^*_{i-1}(x)]\big| S_{i-1})]$ as it might be impossible to have $\hat{L}_{i-1}(h'_{i-1}[x, h^*_{i-1}(x)]\big|S_{i-1}) = 0$ in the presence of label noise. This leads to the generalized version of the algorithm, *ISALg*; see Figure 5.11.

In this section, we analyze the performance of *ISALg* on perfectly separable one-dimensional data of Section 5.2.1 and in Section 5.2.4 we discuss the non-realizable case. For a distribution $q \in \mathcal{Q}$, we have that

$$\mathbb{E}\left[\hat{L}(h'_i[x]\big|S_i)\right] = \int_{-\infty}^{\infty} \hat{L}(h'_i[x]\big|S_i)q(x)dx. \tag{5.58}$$

Since $\hat{L}(h'_i[x]\big|S_i)$ is zero inside the boundary region $(x_i^{(-)}, x_i^{(+)})$, we have that

$$\mathbb{E}\left[\hat{L}(h'_i[x]\big|S_i)\right] = \int_{-\infty}^{x^{(-)}} \hat{L}(h'_i[x]\big|S_i)q(x)dx + \int_{x^{(+)}}^{\infty} \hat{L}(h'_i[x]\big|S_i)q(x)dx. \tag{5.59}$$

Let $q^{(1)} \in \mathcal{Q}$ be $\mathcal{N}(x_i^{(b)}, \sigma_i^2)$ (solid blue curve in Figure 5.13) and $q^{(2)} \in \mathcal{Q}$ be $\mathcal{N}(x_i^{(b)} + \eta, \sigma_i^2)$, where $\eta \in \mathbb{R}$ (dashed red curve in Figure 5.13). Note that $\hat{L}(h'_i[x]\big|S_i)$ is a non-decreasing function in $(x_i^{(+)}, \infty)$ and is also non-increasing in $(-\infty, x_i^{(-)})$ (Figure 5.13).

Algorithm 2 (General version of Importance Sampling Active Learning (*ISALg*) algorithm)

Input:

p is the density for true distribution of X.
$\mathcal{H}$ is the class of hypotheses.
n is the total number of iterations.
$h'[x, y] = \arg\min_{h \in \mathcal{H}} \{\hat{L}(h|S_i) : h(x) \neq y\}$.
$\hat{L}(h|S_i)$ is defined in Equation 5.3.
$S_0 = \emptyset$.

1: **for** $i = 1, \ldots, n$
2: **if** (i==1) then $q_1 \leftarrow p$
3: **else** generate $q_i \leftarrow \arg\min_q \ \mathbb{E}_{x \sim q} \left[\hat{L}_{i-1}\left(h'_{i-1}[x, h^*_{i-1}(x)]\right| S_{i-1}\right)\right]$
4: $x_i \leftarrow Draw(q_i(.))$ *% draw from distribution with density* q_i
5: request the label for $x_i : y_i$
6: $S_i = S_{i-1} \cup \{[q_i, x_i, y_i]\}$
7: $h^*_i = \arg\min_{h \in \mathcal{H}} \hat{L}(h|S_i)$
8: **end**
9: return h^*_n

Figure 5.11: General version of importance sampling active learning (*ISALg*) algorithm.

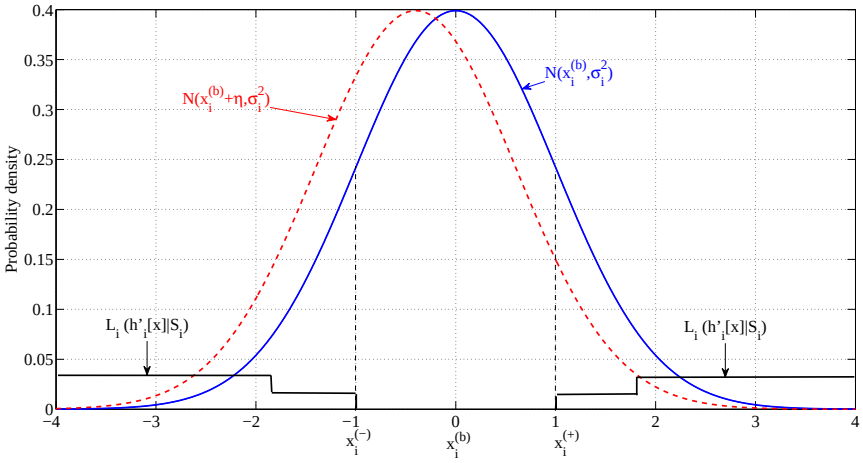

Figure 5.12: Normal distributions $\mathcal{N}(x_i^{(b)}, \sigma_i^2)$ and $\mathcal{N}(x_i^{(b)} + \eta, \sigma_i^2)$ when $\hat{L}(h'_i[x]\big|S_i)$ in $(x_i^{(+)}, \infty)$ region is similar to the one in $(-\infty, x_i^{(-)})$ region.

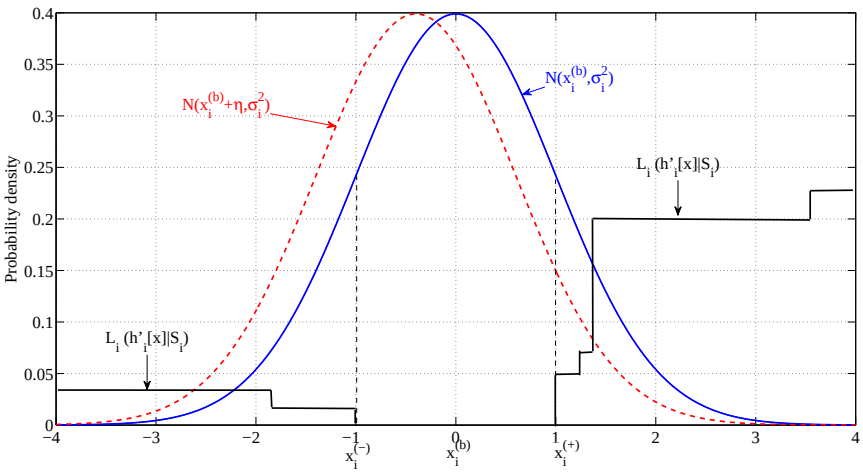

Figure 5.13: Normal distributions $\mathcal{N}(x_i^{(b)}, \sigma_i^2)$ and $\mathcal{N}(x_i^{(b)} + \eta, \sigma_i^2)$ when $\hat{L}(h'_i[x]\big|S_i)$ is greater in $(x_i^{(+)}, \infty)$ region compared to $(-\infty, x_i^{(-)})$ region.

Figure 5.12 shows an example for a scenario where $\hat{L}(h'_i[x]\big|S_i)$ in the interval $(x_i^{(+)}, \infty)$ is similar to $\hat{L}(h'_i[x]\big|S_i)$ in the interval $(-\infty, x_i^{(-)})$. Then the (solid) blue distribution will have $\mathbb{E}\left[\hat{L}(h'_i[x]\big|S_i)\right]$ smaller than the (dashed) red distribution, since its area under the curve outside the boundary region is smaller.

Now imagine a scenario where there are more positive instances in the training set than negative instances. For example, in Figure 5.13, we have $i = 6$ instances in the training set where four are positive, each corresponding to one step of $\hat{L}(h'_i[x]\big|S_i)$ in $(x_i^{(+)}, \infty)$ region, and two negative instances corresponding to two steps of $\hat{L}(h'_i[x]\big|S_i)$ in $(-\infty, x_i^{(-)})$ region. Figure 5.13 shows that $\hat{L}(h'_i[x]\big|S_i)$ in the interval $(x_i^{(+)}, \infty)$ is much bigger than $\hat{L}(h'_i[x]\big|S_i)$ in the interval $(-\infty, x_i^{(-)})$. As a result, the (dashed) red distribution will have smaller $\mathbb{E}\left[\hat{L}(h'_i[x]\big|S_i)\right]$ than the (solid) blue distribution because its area under the curve in the high error region of $(x_i^{(+)}, \infty)$ is smaller.

This means that if $\mathcal{Q}$ only contains $q^{(1)}$ and $q^{(2)}$, the next instance will be drawn from the (dashed) red distribution, *i.e.*, $x_i \sim q^{(2)}$. Consequently, it is more likely that the new instance be drawn from $(-\infty, x_i^{(b)})$, which means that $\hat{L}(h'_i[x]\big|S_i)$ will increase in the $(-\infty, x_i^{(-)})$ region (one step will be added to $\hat{L}(h'_i[x]\big|S_i)$ in that region). For example, the new instance x_i in Figure 5.14 has added one step to $\hat{L}_{i+1}(h'_{i+1}[x]\big|S_{i+1})$ in the $(-\infty, x_{i+1}^{(b)})$ region. It also has updated $x_{i+1}^{(-)}$ to x_i given that $y_i = -1$.

In summary, *ISALg* is balancing the empirical error of h' in both sides of boundary region. Once the empirical error of h' in the left side of the boundary region is similar

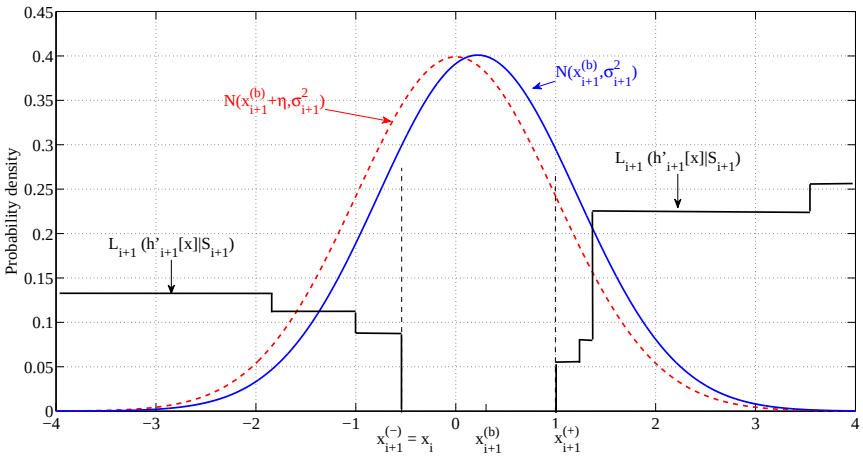

Figure 5.14: The instance x_i has added one step to $\hat{L}_{i+1}(h'_{i+1}[x]\big|S_{i+1})$ in the $(-\infty, x^{(b)}_{i+1})$ region in an attempt to balance $\hat{L}_{i+1}(h'_{i+1}[x]\big|S_{i+1})$ in both sides of boundary region.

to the one on the right (at iteration $j > i$), then the distribution $\mathcal{N}(x^{(b)}_j, \sigma^2_{min})$, which is centered in the middle of the boundary region, will have $\min_{q\in\mathcal{Q}} \mathbb{E}\left[\hat{L}_j(h'_j[x]\big|S_j)\right]$ and will be chosen by *ISALg*.

Note that the empirical error $\hat{L}(h'_i[x]\big|S_i)$ is a random variable with unknown density function. In addition, $\hat{L}(h'_i[x]\big|S_i)$ is a piecewise linear function (one example is shown in Figure 5.14). Therefore, it is difficult to obtain a closed form formula for $\mathbb{E}\left[\hat{L}(h'_i[x]\big|S_i)\right]$. However, the convergence rate of *ISALg* on perfectly separable data is expected to be $O(\log \frac{1}{\epsilon})$ similar to *ISAL*.

5.2.4 Analysis of *ISAL* for Non-Realizable Case

Some of the recent works on active learning [20], [28], [72], [73] have considered the Tsybakov noise condition [74] along the decision boundary. This condition, describes a noise distribution using a detailed parameterization. A passive learning algorithm under this condition can obtain a convergence rate of between $O(\frac{1}{\epsilon})$ and $O(\frac{1}{\epsilon^2})$ [28]. We assume that the data instances are drawn uniformly from the unit ball in $\mathbb{R}^d$, the Bayes classifier h_{Bayes} is linear and the Bayes error is non-zero. Similar to the work by Balcan *et al.* [28], we assume the following noise condition, where there exists a known parameter $\beta > 0$ such that

$$P_X\left(\big|P(Y=+1|X) - P(Y=-1|X)\big| \geq 4\beta\right) = 1. \tag{5.60}$$

This assumption implies [28] that for $\alpha' = 0$

$$L(h) - L(h_{Bayes}) \geq \beta \min\left(1, \frac{4\theta(h, h_{Bayes})}{\pi}\right)^{1/(1-\alpha')}, \tag{5.61}$$

where $\theta(h, h_{Bayes})$ is the angle between linear classifiers h and h_{Bayes}. We analyze a more general case of $\alpha' \in [0, 1)$ where α' in Equation 5.61 characterizes the amount of label noise in the data, *i.e.*, how fast $\mathbb{P}(Y = +1|X = x)$ changes near the decision boundary [75], thus the user has no control over it. An $\alpha' = 0$ may happen when there are few instances near the decision boundary, since in practice these instances are more prone to label noise [72].

Figure 5.15 shows the importance sampling active learning algorithm for non-separable case (*ISALn*). In each iteration of *ISALn*, m_i instances are queried from q_i. The set of labeled instances S_i are updated to only include the m_i new instances. Note that h_i^* is within the radius ρ_i of h_{i-1}^*, *i.e.*, $h_i^* = \arg\min_{h \in diam(h_{i-1}^*, \rho_i)}\{\hat{L}(h|S_i)\}$, where $diam(h, \rho) = \{g \in \mathcal{H}, x \in \mathcal{X} | \mathbb{P}(h(x) \neq g(x)) \leq \rho\}$ is the set of all hypotheses $g \in \mathcal{H}$ whose probability of disagreement with h is less than some $\rho > 0$.

Theorem 5. *Let $d \geq 4$. Assume there exists a hypothesis h_{Bayes} such that Equation 5.61 holds. In each iteration of ISALn of Figure 5.15, let $q_i = \mathcal{N}(\mu_i, \Sigma_i)$, where μ_i is on the linear separator and Σ_i is such that more than half the q_i's weight is inside the margin. There exists a constant C, such that for any $\epsilon, \delta > 0$, and $\epsilon < \beta/8$, if $m_i = Cr_{max}\epsilon_i^{-2}(d + \ln\frac{i}{\delta})$ instances are drawn from q_i at iteration i of ISALn, where r_{max} is defined in Definition 2, $\epsilon_i = 2^{-\alpha'(i-1)-4}\ \beta/\sqrt{5 + \alpha' i \ln 2 - \ln\beta + \ln(1+i)}$, $b_i = 2^{-(1-\alpha')i}\pi d^{-1/2}\ \sqrt{5 + \alpha' i \ln 2 - \ln\beta + \ln(2+i)}$, $\rho_i = 2^{-(1-\alpha')i-2}\pi$ for $i > 1$, $\alpha' \geq 0$, $\rho_1 = \pi$, then, with probability of at least $1 - \delta$, after $n = \lceil \log_2(\beta/\epsilon) \rceil$ iterations, ISALn returns a classifier with error of at most ϵ.*

Proof. The proof is similar to Theorem 3 in [28]. However, *ISALn* uses the weighted loss function of Definition 2, thus, at i-th iteration for some constant $C' > 0$ and $\xi > 1/2$, it needs $m_i = \frac{C'}{\zeta} r_{max}\epsilon_i^{-2}\ (d + \ln\frac{i}{\delta}) \leq 2C' r_{max}\epsilon_i^{-2}(d + \ln\frac{i}{\delta}) = Cr_{max}\epsilon_i^{-2}(d + \ln\frac{i}{\delta})$ instances to obtain $L(h_i^*|U) - L(h_{Bayes}|U) \leq \frac{2\epsilon_i}{\sqrt{\pi}}$, with probability $1 - \frac{\delta}{(i+i^2)}$, where $U = \{x \in \mathcal{X} : |w_{i-1}^* \cdot x| \leq b_{i-1}\}$, w_{i-1}^* is the parameter of h_{i-1}^* and $C = 2C'$. □

Note that *ISALn* has to query $\sum_{i=1}^{n} m_i = O(r_{max}\epsilon^{-2\alpha'}\ln(1/\epsilon)(d + \ln(n/\delta)))$ instances inside the margin to achieve an error rate of ϵ. Therefore, for $\alpha' = 0$, active learning can achieve exponential convergence rate, however, for $\alpha' \in (0, 1)$, only a polynomial improvement can be achieved over passive learning.

Algorithm 3 (Importance Sampling Active Learning algorithm, non-separable case (*ISALn*))

Input:

p is the density for true distribution of X.

$\mathcal{H}$ is the class of hypotheses.

n is the total number of iterations.

$m_i > 0$ is the number of queries in iteration i.

For $h \in \mathcal{H}$ and $\rho > 0$, $diam(h, \rho) = \{g \in \mathcal{H} | \mathbb{P}\left(h(x) \neq g(x)\right) \leq \rho\}$, where $x \in \mathcal{X}$ and ρ is the radius around hypothesis h.

$h'[x, y] = \arg\min_{h \in \mathcal{H}} \{\hat{L}(h|S_i) : h(x) \neq y\}$.

$\hat{L}(h|S_i)$ is defined in Equation 5.3.

$S_0 = \emptyset$.

h_0^* is drawn randomly from $\mathcal{H}$.

1: **for** $i = 1, \ldots, n$

2: **if** (i==1) then $q_1 \leftarrow p$

3: **else** generate $q_i \leftarrow \arg\min_q \ \mathbb{E}_{x \sim q} \left[\hat{L}_{i-1}\left(h'_{i-1}[x, h^*_{i-1}(x)] \middle| S_{i-1}\right)\right]$

4: **for** $j \leftarrow 1, 2, \ldots, m_i$ *% draw m_i instances from q_i*

5: $x_{i_j} \leftarrow Draw(q_i(.))$

6: request label for x_{i_j}: y_{i_j}

7: **end**

8: $S_i = \{[q_i, x_{i_1}, y_{i_1}], \ldots, [q_i, x_{i_{m_i}}, y_{i_{m_i}}]\}$

9: $h_i^* = \arg\min_{h \in diam(h^*_{i-1}, \rho_i)} \{\hat{L}(h|S_i)\}$

10: **end**

11: return h_n^*

Figure 5.15: Importance sampling active learning (*ISALn*) algorithm for non-separable case.

5.3 Bound for the Importance Weighted Error

As previously mentioned, *ISAL* uses importance weighted loss function to identify the sampling distribution. Theorem 6 shows the upper bound for $\sup_{h\in\mathcal{H}}\left|L(h)-\hat{L}(h|S_n)\right|$. It also proves that the error $L(h_n^*)$ of the hypothesis returned by *ISAL*, is close to the best possible error $\inf_{h\in\mathcal{H}} L(h)$.

Theorem 6. *Let $x\in\mathcal{X}$ and $y\in\mathcal{Y}$. Fix a class of hypotheses $\mathcal{H}$ and let $L(h)$ be the expected error of hypothesis $h\in\mathcal{H}$. Let n be the number of instances queried and $h_n^*=\arg\min_{h\in\mathcal{H}}\{\hat{L}(h|S_n)\}$, where $\hat{L}(h|S_n)$ is the weighted empirical error of h on the queried set S_n. Let r_{max} be the maximum weight of the loss, defined in Definition 2. Then with probability of at least $1-\delta$, we have*

$$L(h_n^*)-\inf_{h\in\mathcal{H}} L(h)\leq \sup_{h\in\mathcal{H}}\left|L(h)-\hat{L}(h|S_n)\right| \tag{5.62}$$

$$\leq 2r_{max}\inf_{\alpha>0}\left\{4\alpha+12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H})\log(\frac{2en}{\alpha})}{n}}\right\}+r_{max}\sqrt{\frac{8\ln\frac{2}{\delta}}{n}}. \tag{5.63}$$

This section describes the steps to prove Theorem 6.

Given that $\hat{L}(h_n^*|S_n)-\inf_{h\in\mathcal{H}} L(h)=\sup_{h\in\mathcal{H}}(\hat{L}(h_n^*|S_n)-L(h))$ and $\hat{L}(h_n^*|S_n)\leq\hat{L}(h|S_n)$ for all $h\in\mathcal{H}$, we see

$$\hat{L}(h_n^*|S_n)-\inf_{h\in\mathcal{H}} L(h)\leq\sup_{h\in\mathcal{H}}|\hat{L}(h|S_n)-L(h)| \tag{5.64}$$

$$=\sup_{h\in\mathcal{H}}|L(h)-\hat{L}(h|S_n)|. \tag{5.65}$$

Similarly,

$$L(h_n^*)-\hat{L}(h_n^*|S_n)\leq\sup_{h\in\mathcal{H}}(L(h)-\hat{L}(h|S_n))$$

$$\leq\sup_{h\in\mathcal{H}}|L(h)-\hat{L}(h|S_n)|. \tag{5.66}$$

Summing up the two inequalities (5.65) and (5.66), we get:

$$L(h_n^*)-\inf_{h\in\mathcal{H}} L(h)\leq 2\sup_{h\in\mathcal{H}}\left|L(h)-\hat{L}(h|S_n)\right| \tag{5.67}$$

$$=\frac{2}{n}\sup_{h\in\mathcal{H}}\left|\sum_{i=1}^{n}\mathbb{E}\left[\frac{p(X_i)}{q_i(X_i)}\ell(h(X_i),Y_i)\right]-\sum_{i=1}^{n}\frac{p(X_i)}{q_i(X_i)}\ell(h(X_i),Y_i)\right|. \tag{5.68}$$

In the rest of this section we will obtain the bound for $\sup_{h\in\mathcal{H}}\left|L(h)-\hat{L}(h|S_n)\right|$.

Lemma 4. *(Hoeffding-Azuma inequality [76]). Let $g : \mathcal{U}^n \to \mathbb{R}$ be a function of n random variables such that, for some nonnegative constant c, for all $u_i \in \mathcal{U}$ and $u_i' \in \mathcal{U}$,*

$$|g(u_1, \ldots, u_n) - g(u_1, \ldots, u_{i-1}, u_i', u_{i+1}, \ldots, u_n)| \leq c, \quad 1 \leq i \leq n. \tag{5.69}$$

If $V = g(U_1, \ldots, U_n)$, where U_is are Martingale sequence, then the Hoeffding-Azuma inequality holds:

$$P\left(|V - \mathbb{E}[V]| \geq \kappa\right) \leq 2e^{-\frac{\kappa^2}{2nc^2}}. \tag{5.70}$$

Our main example of such a function is the term in the right hand side of Equation 5.68, which is

$$V = \frac{1}{n} \sup_{h \in \mathcal{H}} \left| \sum_{i=1}^{n} \mathbb{E}\left[\frac{p(X_i)}{q_i(X_i)} \ell(h(X_i), Y_i)\right] - \sum_{i=1}^{n} \frac{p(X_i)}{q_i(X_i)} \ell(h(X_i), Y_i) \right|. \tag{5.71}$$

Note that V in Equation 5.71 satisfies the condition of Lemma 4 with $c = 2r_{max}/n$, because for every i, $\frac{p(X_i)}{q_i(X_i)}$ is at most r_{max} and $\ell(h(X_i), Y_i)$ is at most 1.

Therefore, by applying Lemma 4 to V in Equation 5.71 we get:

for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\frac{1}{n} \sup_{h \in \mathcal{H}} \left| \sum_{i=1}^{n} \mathbb{E}\left[\frac{p(X_i)}{q_i(X_i)} \ell(h(X_i), Y_i)\right] - \sum_{i=1}^{n} \frac{p(X_i)}{q_i(X_i)} \ell(h(X_i), Y_i) \right| \leq$$
$$\mathbb{E}\left[\frac{1}{n} \sup_{h \in \mathcal{H}} \left| \sum_{i=1}^{n} \mathbb{E}\left[\frac{p(X_i)}{q_i(X_i)} \ell(h(X_i), Y_i)\right] - \sum_{i=1}^{n} \frac{p(X_i)}{q_i(X_i)} \ell(h(X_i), Y_i) \right|\right] + r_{max}\sqrt{\frac{8 \ln \frac{2}{\delta}}{n}}. \tag{5.72}$$

The above equation can also be written as

$$\sup_{h \in \mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \leq \mathbb{E}\left[\sup_{h \in \mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right|\right] + r_{max}\sqrt{\frac{8 \ln \frac{2}{\delta}}{n}}. \tag{5.73}$$

Equation 5.73 allows us to focus on the expected value of $\sup_{h \in \mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right|$, which can be bounded by a symmetrization device. In the rest of this section we obtain the bound for the expectation term in the right hand side of Equation 5.73.

Since the active learning model is a repeated process, we can represent it in the form of a tree [77]. Let $\mathcal{T}(\mathcal{Z}, n)$ be the set of all $\mathcal{Z}$-valued binary trees of depth n. A sequence of $(z_1, \ldots, z_n)$ of n mappings $z_i : \{\pm 1\}^{i-1} \to \mathcal{Z}$ defines a $\mathcal{Z}$-valued tree of depth n for a given set $\mathcal{Z}$ [77]. For a function $f : \mathcal{Z} \to \mathbb{R}$, define $f(z)$ to be the mapping $(f \circ z_1, \ldots, f \circ z_n)$. For a tree $z \in \mathcal{T}(\mathcal{Z}, n)$ and a class of functions $\mathcal{F}$, $\mathcal{F}(z) = \{f(z) :$

$f \in \mathcal{F}\}$ is the projection of $\mathcal{F}$ on z. A path of length n in the tree is the sequence $\xi = (\xi_1, \ldots, \xi_n) \in \{\pm 1\}^n$ and throughout this section we refer to $z_i(\xi_1, \ldots, \xi_{i-1})$ as $z_i(\xi)$.

Definition 5. *The **sequential Rademacher complexity** [33] of a function class $\mathcal{F}$ is defined as*

$$\mathcal{R}_n(\mathcal{F}) = \sup_{z \in \mathcal{T}(\mathcal{Z},n)} \mathbb{E}_\xi \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \xi_i f(z_i(\xi)) \right], \tag{5.74}$$

where $\xi = (\xi_1, \ldots, \xi_n)$ is the sequence of i.i.d. Rademacher random variables.

During the proof of Theorem 2 in [77], the following inequality is proved:

Theorem 7. *(from [77]) Let*

$$\mathcal{D}_n(\mathcal{F}) = \sup_{p_1 \in P} \mathbb{E}_{z_1 \sim p_1} \sup_{p_2 \in P} \mathbb{E}_{z_2 \sim p_2} \cdots \sup_{p_n \in P} \mathbb{E}_{z_n \sim p_n} \left[\sup_{f \in \mathcal{F}} \left\{ \sum_{i=1}^{n} \int f(z_i') dp_i(z_i') - \sum_{i=1}^{n} f(z_i) \right\} \right] \tag{5.75}$$

where P is an arbitrary collection of distributions. Then $\mathcal{D}_n(\mathcal{F}) \leq 2\mathcal{R}_n(\mathcal{F})$, assuming that $(z_1, \ldots, z_n) \to \sup_{f \in \mathcal{F}} \sum_{i=1}^{n} f(z_i)$ is $(p_1, \ldots, p_n)$-integrable, for any $(p_1, \ldots, p_n) \in P^n$.

Note that in the proof of Theorem 2 in [77], Fubini's theorem is implicitly used multiple times. Thus the condition of Theorem 7 ensures that the conditions of Fubini's theorem is satisfied.

Let $\mathcal{F}$ be the class of weighted loss functions of Definition 2. The term $\mathbb{E}\{\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h|S_n)|\}$ in Equation 5.73 can be written as

$$\mathbb{E}\left[\sup_{h \in \mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \right] = \mathbb{E}\left[\frac{1}{n} \sup_{f \in \mathcal{F}} \left| \sum_{i=1}^{n} \mathbb{E}\left[f(Q_i, X_i, Y_i) \right] - \sum_{i=1}^{n} f(Q_i, X_i, Y_i) \right| \right], \tag{5.76}$$

where Q_i is $(Q_1, X_1, Y_1, \ldots, Q_{i-1}, X_{i-1}, Y_{i-1})$-measurable, $X_i \sim Q_i$, $Y_i \sim P_{Y|X}(\cdot|X_i)$, and $Q_1, \ldots, Q_n \in \mathcal{Q}$ are (randomly generated) densities. We assume $\sup_{\substack{x \in \mathcal{X} \\ q \in \mathcal{Q}}} \frac{p(x)}{q(x)} \leq r_{max}$, thus $\mathbb{E}\left[f(Q_i, X_i, Y_i) \right] = \int f(q, x, y) dS_i(q, x, y)$, where S_i is the joint distribution of (Q_i, X_i, Y_i). Note that S_i is random and in particular $(Q_1, X_1, Y_1, \ldots, Q_{i-1}, X_{i-1},$

$Y_{i-1})$-measurable. Therefore, by letting $\mathcal{M} = \{R \otimes P | R \in \mathcal{V}(Q)\}$, where $\mathcal{V}(Q)$ is the space of probability measures over Q, we have

$$\mathbb{E}\left[\sup_{h\in\mathcal{H}}\left|L(h) - \hat{L}(h|S_n)\right|\right] = \sup_{S_1\in\mathcal{M}} \mathbb{E}_{(Q_1,X_1,Y_1)\sim S_1} \sup_{S_2\in\mathcal{M}} \mathbb{E}_{(Q_2,X_2,Y_2)\sim S_2} \cdots$$
$$\sup_{S_n\in\mathcal{M}} \mathbb{E}_{(Q_n,X_n,Y_n)\sim S_n}\left[\frac{1}{n}\sup_{f\in\mathcal{F}}\left|\sum_{i=1}^{n}\int f(q,x,y)dS_i(q,x,y) - \sum_{i=1}^{n} f(Q_i,X_i,Y_i)\right|\right]. \tag{5.77}$$

By Theorem 7, the last term in Equation 5.77 is bounded by $2\mathcal{R}_n(\mathcal{F})$, since the integrability condition holds, because $\sup_{f\in\mathcal{F}}\sum_{i=1}^{n} f(q_i,x_i,y_i) \leq nr_{max}$ is bounded.

Definition 6. *(from [77]) Let $\mathcal{K}$ be the class of functions $k : \mathcal{X} \to \mathbb{R}$. Let $\mathbf{x} = \{x_1, ..., x_n\}$, where $x_i \in \mathcal{X}$ for $i = 1, \ldots, n$, be a set of n points. A set of vectors $\boldsymbol{V} \subset \mathbb{R}^n$ is an α-cover, with respect to the p-norm of $\mathcal{K}$ on $\mathbf{x}$ if for all $k \in \mathcal{K}$ there exists a $\mathbf{v} \in \boldsymbol{V}$ such that:*

$$\left(\frac{1}{n}\sum_{i=1}^{n}|v_i - k(x_i)|^p\right)^{\frac{1}{p}} \leq \alpha. \tag{5.78}$$

*We define p-**norm covering number** $\mathcal{N}_p(\alpha, \mathcal{K}, \mathbf{x})$ as the size of minimal such cover $\boldsymbol{V}$, i.e.,*

$$\mathcal{N}_p(\alpha, \mathcal{K}, \mathbf{x}) = \min\{|\boldsymbol{V}| : \boldsymbol{V} \text{ is an } \alpha\text{-cover, under the p-norm, of } \mathcal{K} \text{ on } \mathbf{x}\}.$$

Also define $\mathcal{N}_p(\alpha, \mathcal{K}, n) = \sup_{\mathbf{x}} \mathcal{N}_p(\alpha, \mathcal{K}, \mathbf{x})$. In other words, $\mathcal{N}_p(\alpha, \mathcal{K}, n)$ is the maximum covering number over $\mathbf{x}$.

We use the following two lemmas from [77] in our proofs.

Lemma 5. *Suppose $\mathcal{H}$ is a class of $[-1, 1]$-valued functions on $\mathcal{X}$. Then for any $\alpha > 0$, any $n > 0$, and any set $\mathbf{x} = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$,*

$$\mathcal{N}_1(\alpha, \mathcal{H}, \mathbf{x}) \leq \mathcal{N}_2(\alpha, \mathcal{H}, \mathbf{x}) \leq \mathcal{N}_\infty(\alpha, \mathcal{H}, \mathbf{x}) \leq \left(\frac{2en}{\alpha}\right)^{fat_\alpha(\mathcal{H})} \tag{5.79}$$

Lemma 6. *Fix a class $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{Z}}$ and a function $\phi : \mathbb{R} \times \mathcal{Z} \to \mathbb{R}$. Assume, for all $z \in \mathcal{Z}$, $\phi(\cdot, z)$ is a Lipschitz function with a constant C. Then*

$$\mathcal{R}_n(\phi(\mathcal{H})) \leq C \cdot \mathcal{R}_n(\mathcal{H})$$

where $\phi(\mathcal{H}) = \{z \to \phi(h(z), z), h \in \mathcal{H}, z \in \mathcal{Z}\}$.

Corollary 3. *Suppose $\mathcal{H}$ is a class of $[-1,1]$-valued functions on $\mathcal{X}$. Then for any $\alpha > 0$, and any $n > 0$*

$$\mathcal{N}_2(\alpha, \mathcal{H}, n) \leq \left(\frac{2en}{\alpha}\right)^{fat_\alpha(\mathcal{H})} \tag{5.80}$$

Proof. Lemma 5 is true for any set $\mathbf{x} = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$. This includes the set whose covering number is $\mathcal{N}_2(\alpha, \mathcal{K}, n) = \sup_{\mathbf{x}} \mathcal{N}_2(\alpha, \mathcal{K}, \mathbf{x})$. □

5.3.1 Bounding Sequential Rademacher Complexity

Sequential Rademacher complexity $\mathcal{R}_n(\mathcal{H})$ can be bounded using Lemma 7.

Lemma 7. *Let $\mathcal{H} \subseteq [-1,1]^{\mathcal{X}}$. We have*

$$\mathcal{R}_n(\mathcal{H}) \leq \inf_\alpha \left\{ 4\alpha + 12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H}) \log(\frac{2en}{\alpha})}{n}} \right\}.$$

Proof. From Theorem 9 and Definition 11 of [77], we have[2]

$$\mathcal{R}_n(\mathcal{H}) \leq \inf_\alpha \left\{ 4\alpha + 12 \int_\alpha^1 \sqrt{\frac{\log \mathcal{N}_2(t, \mathcal{H}, n)}{n}} \mathrm{d}t \right\}. \tag{5.81}$$

By replacing $\mathcal{N}_2(t, \mathcal{H}, n)$ in (5.81) with its bound from (5.80), we get

$$\mathcal{R}_n(\mathcal{H}) \leq \inf_\alpha \left\{ 4\alpha + 12 \int_\alpha^1 \sqrt{\frac{fat_t(\mathcal{H}) \log(\frac{2en}{t})}{n}} \mathrm{d}t \right\}. \tag{5.82}$$

The term inside the integral of Equation (5.82) is monotonically decreasing with respect to t, as both $fat_t(\mathcal{H})$ and $\log(\frac{2en}{t})$ decrease as t increases. Therefore the integrand in (5.82) can be bounded by its maximum value, which happens at $t = \alpha$ for $0 < \alpha \leq 1$, *i.e.,*

$$\mathcal{R}_n(\mathcal{H}) \leq \inf_\alpha \left\{ 4\alpha + 12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H}) \log(\frac{2en}{\alpha})}{n}} \right\}. \tag{5.83}$$

□

Lemma 8. *Pick any $n \geq 1$ and let $\alpha > 0$. Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Fix a set of permissible densities $\mathcal{Q}$ and a class of hypotheses $\mathcal{H}$. Let $\mathcal{F}$ be the class of weighted loss functions*

[2] Note that our definition of sequential Rademacher complexity in definition 5 has a normalizing factor of $\frac{1}{n}$.

defined in Definition 2. Then the sequential Rademacher complexity of class of weighted loss functions $\mathcal{F}$ is bounded by

$$\mathcal{R}_n(\mathcal{F}) \leq r_{max} \times \inf_{\alpha} \left\{ 4\alpha + 12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H}) \log(\frac{2en}{\alpha})}{n}} \right\} \tag{5.84}$$

where $fat_\alpha(\mathcal{H})$ denotes the fat shattering dimension of $\mathcal{H}$ at scale α.

Proof. Using Lemma 6 with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\phi(t,(x,y)) = \frac{p(x)}{q(x)}|t-y|$, we have $\mathcal{R}_n(\mathcal{F}) \leq r_{max}\mathcal{R}_n(\mathcal{H})$. This is because $\frac{p(x)}{q(x)}|t-y|$ is r_{max} Lipschitz in t for any x and y, where $r_{max} = \max_{x\in\mathcal{X}} \frac{p(x)}{q(x)}$.

Then Lemma 7 shows that $\inf_\alpha \left\{ 4\alpha + 12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H}) \log(\frac{2en}{\alpha})}{n}} \right\}$ is the bound for $\mathcal{R}_n(\mathcal{H})$, which implies that its product with r_{max} is the bound for $\mathcal{R}_n(\mathcal{F})$ too. □

Proof of Theorem 6. From Equation 5.67 we have

$$L(h_n^*) - \inf_{h\in\mathcal{H}} L(h) \leq \sup_{h\in\mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right|. \tag{5.85}$$

Also from Equation 5.73, with probability of at least $1-\delta$, we have that

$$\sup_{h\in\mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \leq \mathbb{E}\left[\sup_{h\in\mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \right] + r_{max}\sqrt{\frac{8\ln\frac{2}{\delta}}{n}}. \tag{5.86}$$

By combining the above two equations, with probability of at least $1-\delta$, we have

$$L(h_n^*) - \inf_{h\in\mathcal{H}} L(h) \leq \mathbb{E}\left[\sup_{h\in\mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \right] + r_{max}\sqrt{\frac{8\ln\frac{2}{\delta}}{n}}. \tag{5.87}$$

From Equation 5.77 we have that

$$\mathbb{E}\left[\sup_{h\in\mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \right] \leq 2\mathcal{R}_n(\mathcal{F}). \tag{5.88}$$

Inserting the bound of Equation 5.88 into right hand side of Equation 5.87 yields:

with probability of at least $1-\delta$

$$L(h_n^*) - \inf_{h\in\mathcal{H}} L(h) \leq 2\mathcal{R}_n(\mathcal{F}) + r_{max}\sqrt{\frac{8\ln\frac{2}{\delta}}{n}}. \tag{5.89}$$

Using Lemma 8, we can bound $\mathcal{R}_n(\mathcal{F})$ by $r_{max} \inf_\alpha \left\{ 4\alpha + 12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H}) \log(\frac{2en}{\alpha})}{n}} \right\}$, which results in

$$L(h_n^*) - \inf_{h\in\mathcal{H}} L(h) \leq \sup_{h\in\mathcal{H}} \left| L(h) - \hat{L}(h|S_n) \right| \tag{5.90}$$

$$\leq 2r_{max} \inf_{\alpha} \left\{ 4\alpha + 12(1-\alpha)\sqrt{\frac{fat_\alpha(\mathcal{H})\log(\frac{2en}{\alpha})}{n}} \right\} + r_{max}\sqrt{\frac{8\ln\frac{2}{\delta}}{n}}. \tag{5.91}$$

□

5.4 Experiments

This section describes our empirical evaluation of *ISALg* on noise-free data as well as data with various amounts of label noise. These experiments focus on binary classification tasks with $\mathcal{Y} = \{-1, +1\}$.

5.4.1 Neurophysiology Data

An efficient data collection procedure is essential in experimental designs. For example, in neurophysiology experiments where the goal is to characterize a neural system, we are interested in the response of a cell to different stimuli. Following the work of Lewi *et al.* [56] to design a neurophysiology experiment, we simulate the receptive field of a visually sensitive neuron. Generalized linear models are often used in the neurophysiology experiments to model the space. The number of spikes, which is the likelihood of the response ν, is related to the firing rate λ, where

$$\lambda = \mathbb{E}_{\nu|x,\theta}[\nu] = \exp(\theta^T \cdot x). \tag{5.92}$$

The relationship between the stimulus x and the response of the neuron is given by a Poisson distribution where

$$\log P(\nu|x,\theta) = \log \frac{e^{-\lambda\Delta t}(\lambda\Delta t)^\nu}{\nu!}, \tag{5.93}$$

where Δt is the length of the time frame for measuring the firing rate, λ.

A Gabor function is considered for the receptive field of the neuron, which is a proxy model of a $V1$ simple cell. Simulated responses are generated by sampling from Equation 5.93. We assume that the stimulus x is in the form of a vector obtained by rasterizing a $m \times m$ image and θ is a vector obtained by rasterizing a $m \times m$ Gabor function. For

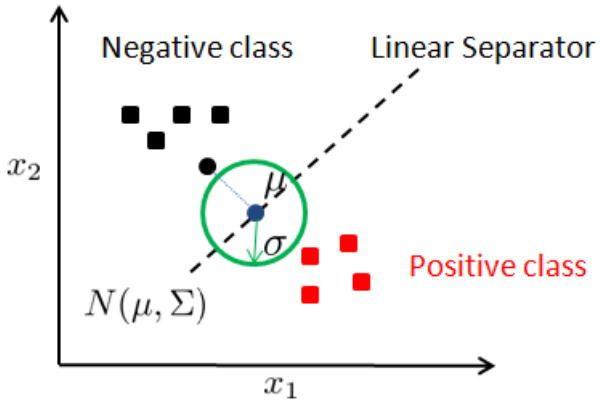

Figure 5.16: *ISALg* generates the sampling distribution (green oval) for a two dimensional dataset.

these experiments, we consider the case of $m = 3$ ($d = 9$ features) and $m = 4$ ($d = 16$ features).

We assign the response of zero (no response) to -1 class and other responses are in $+1$ class. Similar to [56], we draw the intensity of each pixel in the stimulus x from a Gaussian distribution, here $\mathcal{N}(128, 10)$ where $x < 0$ are set to 0 and $x > 256$ are set to 256.

5.4.2 Experimental Setup

Throughout the experiments, we use support vector machines (SVM) as the classifier. To consider the weighted loss function of Definition 2 in our implementation, we modified the code of LIBSVM software [78].

The sampling distribution q_i in each iteration is $\mathcal{N}(\mu_i, \Sigma_i)$, where μ_i is obtained by selecting a point randomly from the underlying distribution and projecting it onto the current linear separator. Note that before applying *ISALg*, we first normalize the data such that all the features have the same empirical marginal variance. The covariance matrix Σ_i depends on the margin at iteration i (see the experiments below).

Figure 5.16 shows a two dimensional dataset that are separated by a linear separator. In this figure, *ISALg* draws a point randomly from the underlying distribution (black dot), then projects it onto the separator to get μ (blue dot). It then centers a Gaussian distribution at μ, $\mathcal{N}(\mu, \Sigma)$. A contour for this Gaussian is shown by green oval.

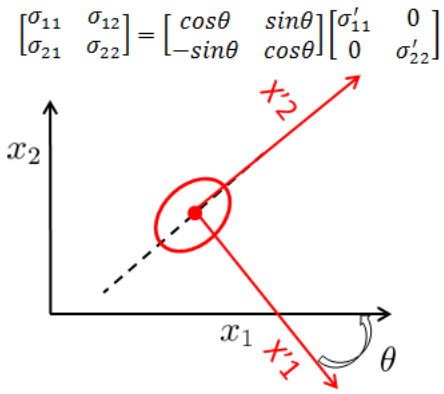

Figure 5.17: An example scenario of a Gaussian, parallel to the linear separator, to be rotated with angle θ.

5.4.3 Remarks on the Sampling Distribution of *ISALg*

In this section we investigate the difference between using the Gaussian distribution parallel to the linear separator and a non-parallel Gaussian for *ISALg*. Figure 5.17 shows a scenario for 2-dimensional data where the linear separator is shown as dashed line.

We could let the red Gaussian with covariance parallel to the separator, be the sampling distribution for *ISALg*. We estimate the covariance matrix by sampling some instances around the separator. Then, the covariance matrix can be numerically estimated by randomly selecting a point (red point in Figure 5.18) on the separator as the center of the Gaussian (μ_i).

We select an arbitrary direction uniformly on the separator and pick a point (blue point in Figure 5.18) with a distance of $dist_1 \sim \mathcal{N}(0, b_i^2)$ from μ_i, where b_i is half of the size of margin at iteration i. Next we select a direction perpendicular to the separator and pick a point (green point in Figure 5.18) with a distance of $dist_2 \sim \mathcal{N}(0, b_i^2)$ from the separator. We repeat this process for 1000 times to obtain 1000 instances, then estimate their covariance matrix.

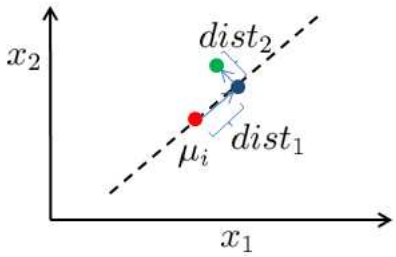

Figure 5.18: The process of estimating the covariance matrix for *ISALg* parallel.

However, this approach increases the sample complexity of *ISALg* and hence may not be suitable for applications where sample efficiency is important. Therefore, because of sample inefficiency of the above-mentioned approach, we may approximate the covariance matrix by only considering the diagonal elements of the matrix.

5.4.4 *ISALg* for Experimental Design

In this section we study the application of *ISALg* in the experimental design setting where the active learner can generate any data point it wants, using the neurophysiology data of Section 5.4.1.

5.4.4.1 *ISALg* on Data Without Label-Noise

In the first active learning algorithm as a baseline, instances are sequentially sampled from the underlying distribution to train the classifier (random sampling). The average test error over 50 trials on noise-free data is shown as dashed line in Figure 5.19 for $d = 9$ (3×3 image) and Figure 5.20 for $d = 16$ (4×4 image); see Section 5.4.1. The performance of *ISALg* that uses a Gaussian distribution with a diagonal covariance matrix is shown as solid blue line in the figures (*ISALg* diagonal). In addition, the performance of *ISALg* with a Gaussian distribution whose mahalanobis ellipsoids are parallel to the separator is also shown in the figures (*ISALg* parallel). Here, *ISALg* randomly samples the first four instances from the underlying distribution. *ISALg* diagonal samples the rest of instances from $\mathcal{N}(\mu_i, \Sigma_i)$, where μ_i is randomly selected on the current separator, and Σ_i is the a diagonal matrix whose diagonal elements are half the size of current margin squared, *i.e.*, b_i^2. For *ISALg* parallel, we estimate the covariance matrix that is parallel to the separator as described in Section 5.4.3 where $dist_1$ and $dist_2$ are drawn from $\mathcal{N}(0, b_i^2)$.

Figures 5.19 and 5.20 also show the active learning algorithm of [20], referred to as IWAL2 here. IWAL2 draws instance from the underlying distribution, and only accepts those instances whose label uncertainty is above a given threshold. We use the parameters used in [20].

We also propose the BI (boundary instance) active learning algorithm, that in each iteration, randomly draws an instance on the current decision boundary (obtained by training on current labeled set S_i). Note that the current decision boundary changes as more instances are added to the labeled set. In addition, BI algorithm uses a biased estimation

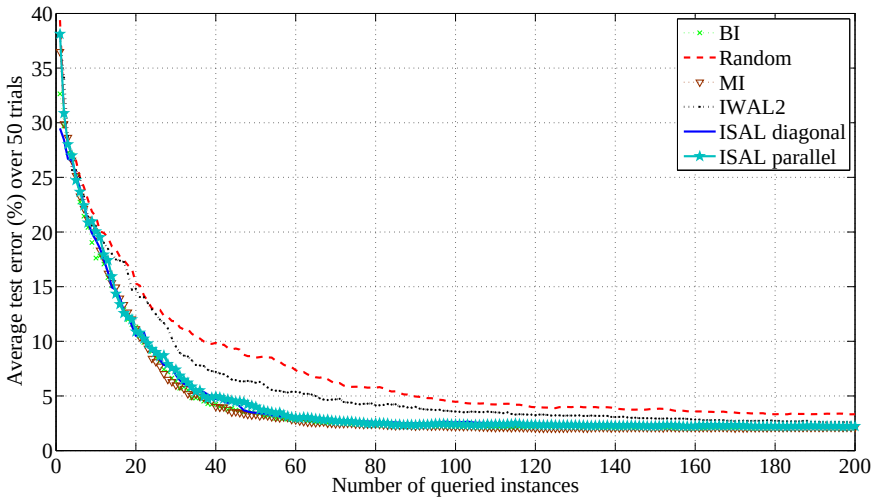

Figure 5.19: Average test error versus number of queried instances for *ISALg* and four other active learning algorithms on neurophysiology data with 3×3 image as stimulus, using SVM.

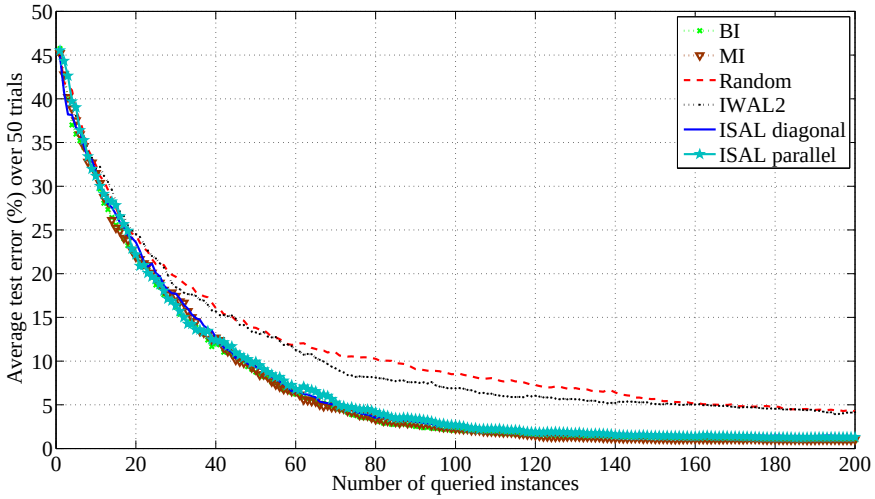

Figure 5.20: Average test error versus number of queried instances for *ISALg* and four other active learning algorithms on neurophysiology data with 4×4 image as stimulus, using SVM.

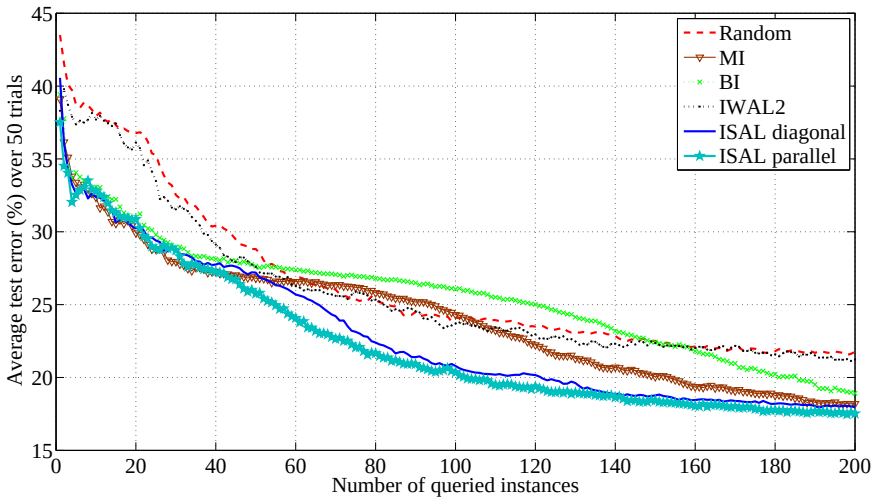

Figure 5.21: Average test error versus number of queried instances for *ISALg* and four other active learning algorithms on neurophysiology data that has 15% label noise with 3×3 image as stimulus, using SVM.

of the error. The fifth active learning algorithm in Figures 5.19 and 5.20 is MI (margin instance algorithm, similar to the approach in [28]), which draws instances inside the current margin. Similar to BI, the estimation of error by MI is biased.

Figures 5.19 and 5.20 show that *ISALg*, BI and MI have similar performances by converging quickly to the minimal test error. However, the performance of each of them is statistically better than the random sampling and IWAL2 (paired t-test, $p < 0.05$).

5.4.4.2 *ISALg* on Data With Label Noise

Figures 5.21 and 5.22 show the performance of the active learning algorithms on the neurophysiology data that contains 15% label noise (*i.e.*, the label of each instance is flipped 15% of the time), for $d = 9$ and $d = 16$ respectively (see Section 5.4.1).

Figures 5.21 and 5.22 show that *ISALg* converges faster than the other active learning algorithms to the optimal classifier. Furthermore, its performance is statistically better then the other four algorithms (paired t-test, $p < 0.05$). The reason is that *ISALg* draws the instances from a distribution q_i, which has more weight inside the margin, however, random sampling and IWAL2 draw instances outside the margin too. Recall that in margin-based classifiers, having an instance on the wrong side of the margin is not desirable for the learning (this can happen because of the label noise). On the other hand, having label noise inside the margin, is more tolerable. Based on our observation

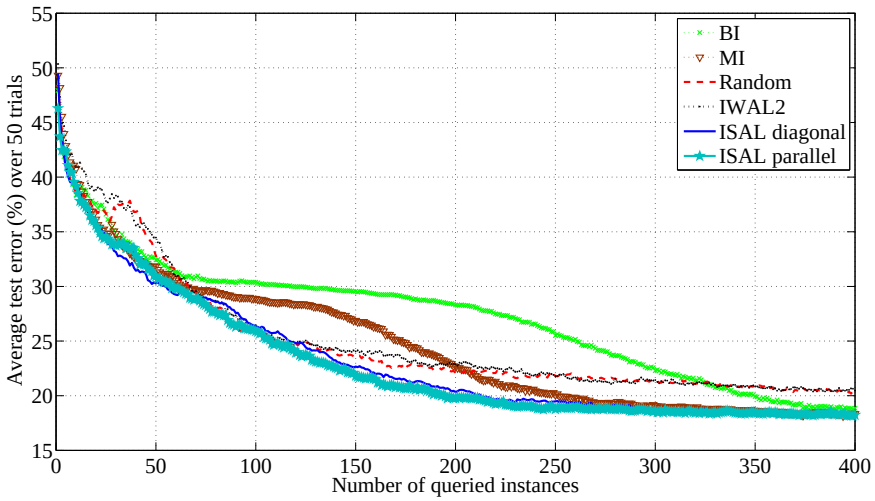

Figure 5.22: Average test error versus number of queried instances for *ISALg* and four other active learning algorithms on neurophysiology data that has 15% label noise with 4×4 image as stimulus, using SVM.

of the data, the average distance[3] to the optimal separator for instances drawn by *ISALg* is approximately half of the ones for the instances drawn by random sampling and IWAL2.

Figures 5.21 and 5.22 show that as the current decision boundary becomes more accurate, *i.e.*, gets closer to the optimal classifier, the performance of *ISALg* improves significantly.

To evaluate the performance of *ISALg* algorithm in the non-realizable case, we added various amounts of label noise to the neurophysiology data of Section 5.4.1. Figures 5.23 and 5.24 show the average test error over 50 trials on $n = 200$ queried instances versus different amounts of label noise for *ISALg* algorithm, BI, IWAL2, MI and sampling from underlying distribution. As Figures 5.23 and 5.24 show, *ISALg* outperforms other algorithms on data with large amounts of label noise.

5.4.5 *ISALg* for Pool-based Data

In this section, we discuss *ISALp*, which is the extension of *ISALg* to pool-based data. Note that *ISALg* algorithm of Figure 5.11 has to be modified such that it only queries instances from the pool of data. We also implemented the highly referenced algorithm by Tong and Koller [12] designed for SVM, called MU here, that queries the instance closest to the current linear separator. The distance of a point to the linear separator is obtained

[3]We measure the distance for a point (x, y) to the optimal separator (with parameter w) using $|w^T \cdot x|$.

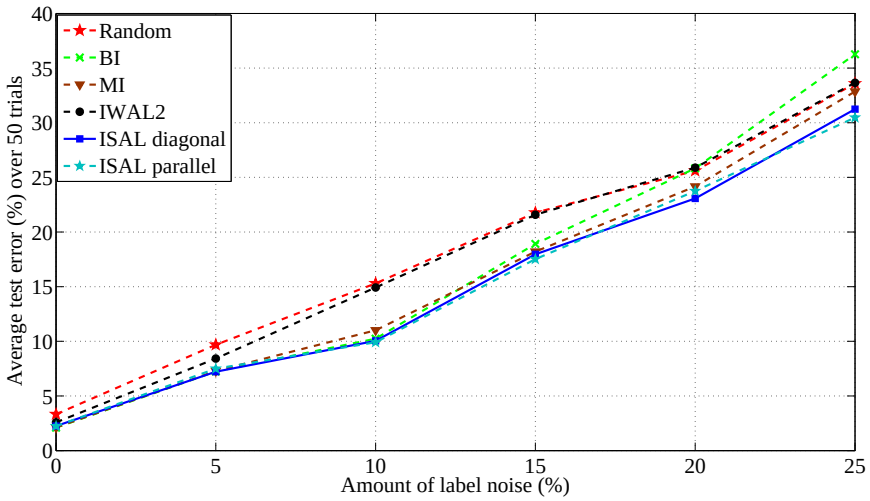

Figure 5.23: Average test error versus various amounts of label noise for *ISALg* and four other active learning algorithms on neurophysiology data with $n = 200$ and 3×3 image as stimulus.

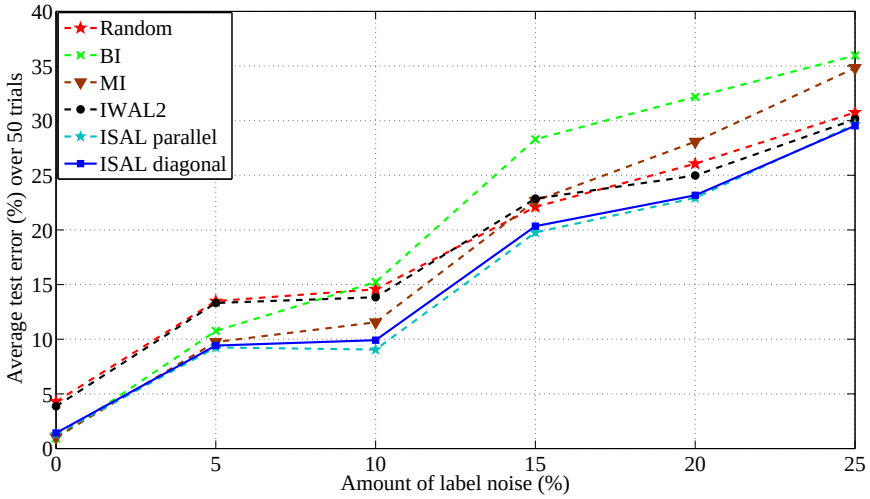

Figure 5.24: Average test error versus various amounts of label noise for *ISALg* and four other active learning algorithms on neurophysiology data with $n = 200$ queries and 4×4 image as stimulus.

by projecting that point onto the separator.

In early steps of active learning, the number of queried instances is limited. Thus, it is likely that the current separator is not a good approximation to the optimal separator. As such, *ISALp* initially assigns a probability to each instance. This probability is proportional to the distance of an instance to the current separator, *i.e.*, for an instance x_i, it equals

$$P(x_i) = \frac{e^{\frac{-\Delta_i^2}{\sigma_p^2}}}{\sum_j e^{\frac{-\Delta_j^2}{\sigma_p^2}}} \tag{5.94}$$

where Δ_i denotes the distance of the instance x_i to the separator, and σ_p is the bandwidth parameter. A small value for σ_p assigns a higher probability to instances near the decision boundary.

After querying a certain number of instances, the current separator is a good approximation of the margin, thus, *ISALp* switches to the MU algorithm. For example, the switching point can be obtained using the 5-fold cross validation error on the training data. In these experiments, we use a cross validation error of 20% as the switching point. We also use $\sigma_p = 1$.

In addition, we compare the above algorithms with LMU_logReg, which is a variant of LMU algorithm of Chapter 4 that actively learns the parameters of the logistic regression classifier for non-structured data. We also report the results of randomly selecting the instances to train the logistic regression classifier (Random_logReg).

In this section, we investigate the performance of the above algorithms on the following datasets:

- neurophysiology data: we generated two pools of $10,000$ instances from neurophysiology data with $d = 9$, one without label noise and one with 15% label noise.
- wine dataset [24] where we classified the data into high quality (for ratings more than 6) versus low quality wine. This dataset has 11 features and 3661 instances.
- vehicle dataset [24] with 18 features and 846 instances where we classified bus and van as $+1$ and the rest as -1.

For each dataset, 20% of the data is set aside as the test set. We also normalize each feature by subtracting its mean and dividing by its standard deviaion.

Figures 5.25 to 5.28 show the performance of *ISALp*, MU, LMU_logReg and random sampling on these four datasets. Note that in all the datasets, active learning converges faster to the optimal classifier than the random sampling. In addition, the performance of *ISALp* is initially better than MU in the two datasets that are not perfectly separable by a linear classifier (*i.e.*, neurophysiology data with 15% label noise and the wine dataset). We conjecture that the probabilistic approach used by *ISALp* in the initial steps contributes to its improvement over the MU. For example, on the noisy neurophysiology data, *ISALp* achieves test error that is about 7% better than MU for 100 queried instances. On the other hand, on perfectly separable data (*i.e.*, neurophysiology data with no label noise and the vehicle dataset), *ISALp* performs as well as the MU algorithm. This is because the current separator quickly converges to optimal classifier, *e.g.*, on the noise-free neurophysiology data, both algorithms reach 20% test error after querying 10 instances.

By comparing the performance of Random and Random_logreg in Figures 5.26 to 5.28, we observe that the classification error obtained from SVM is lower than logistic regression. The only exception is in realizable neurophysiology data where logistic regression performs slightly better. We conjecture that the confidence that SVM has in its predictions (by considering a margin around the decision boundary) results in the lower classification error. Therefore, the difference between the error rate of *ISALp* and LMU_logReg is due to their underlying learning model. In the scenario that the underlying learning models have similar error rates (*e.g.*, realizable neurophysiology data in Figures 5.25), the active learners *ISALp* and LMU_logReg also have similar error rates.

5.5 Summary

We presented an importance sampling active learning algorithm *ISAL*, where instances are queried from an appropriate distribution. We showed that *ISAL* can be effective in reducing the cost of labeling, in scenarios where new instances can be generated upon active learner's request. *ISAL* provides an active learning framework to learn classifiers in many situations. It uses importance weights to reduce the bias of error estimation, while controlling the variance of the estimation. It has the possibility of converging to the best of the available classifiers, unlike some active learning approaches that maintain a version space. In addition, it can be more sample-efficient since it directly samples from an appropriate distribution. Our empirical results show that it achieves fast convergence rate to the optimal error with SVM classifier on both noise-free and noisy data.

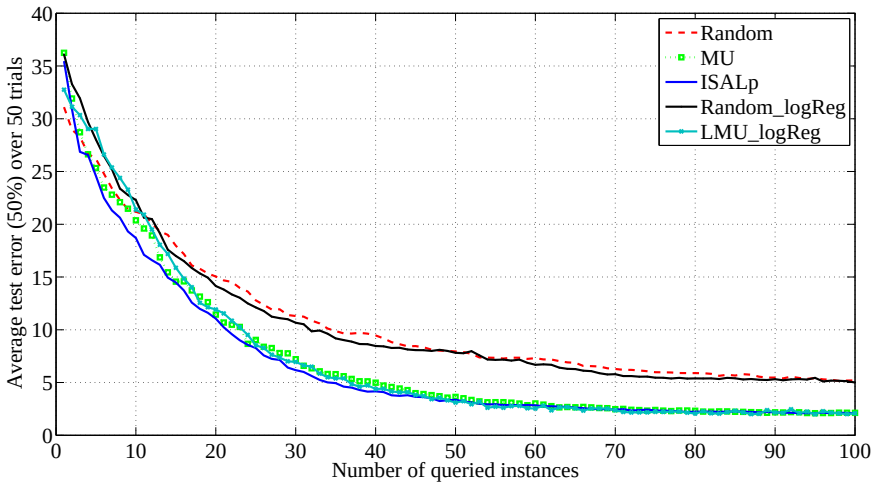

Figure 5.25: *ISALp* versus random sampling, MU, and LMU_logreg on a pool of $10,000$ neurophysiology data without label noise.

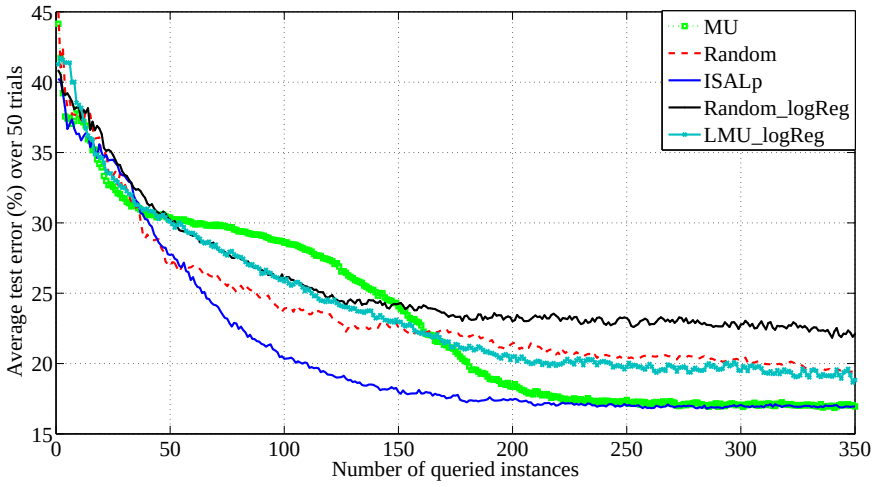

Figure 5.26: *ISALp* versus random sampling, MU, and LMU_logreg on a pool of $10,000$ neurophysiology data with 15% label noise.

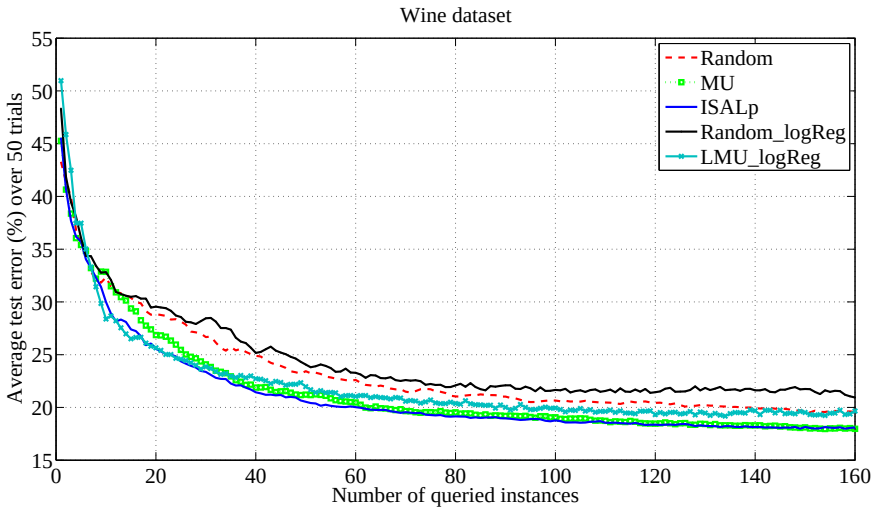

Figure 5.27: *ISALp* versus random sampling, MU, and LMU_logreg on wine dataset.

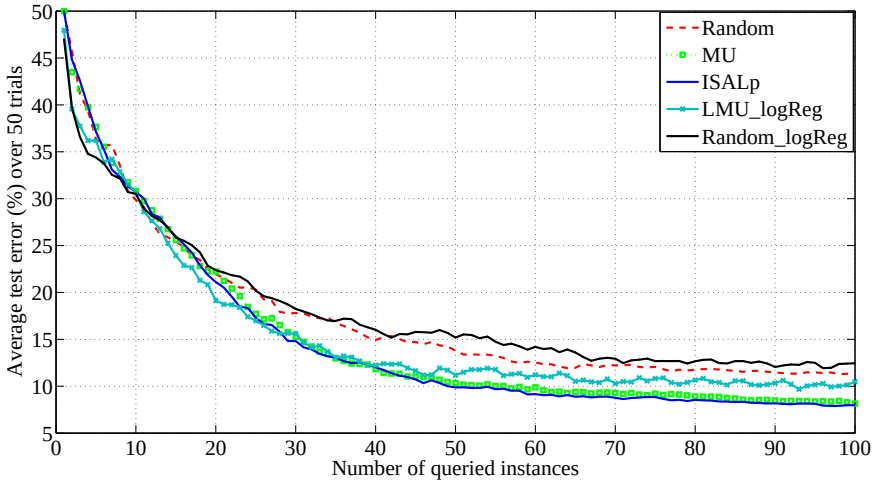

Figure 5.28: *ISALp* versus random sampling and MU on vehicle dataset.

Chapter 6

Conclusion

Supervised machine learning approaches are used extensively to solve real-world challenges. Sometimes these methods are given a limited budget to spend per instance before returning a label. Such scenario corresponds naturally to decision trees. The near optimal fixed-depth decision tree presented here improves the computational cost of obtaining such tree while yielding relatively high accuracy.

In addition, the use of supervised machine learning methods are limited by requiring labeled training data, which are often hard to obtain. However, there are often plenty of unlabeled instances available; this motivates the use of active learning to request the label of a small set of (sequentially) specified unlabeled data points.

The active learning methods proposed here aim to reduce the overall cost of acquiring labeled data by allowing the learner to effectively choose the informative instances that help to produce an accurate classifier.

6.1 Future Directions

Here, we discuss several future directions inspired by the work of this thesis.

- In Chapter 2 we discussed how to use the *opt-feature list* to fill in the features at the *last level* of the tree, However, our techniques are not specific to this final row. These ideas can be extended to pre-compute an optimal tree list with the final depth-d' subtree at the end of a path given the label posterior, rather than just the final internal node; *i.e.*, so far, we have dealt only with $d' = 1$. While this will significantly increase the cost of the precomputational stage, it should provide a significant computational gain when growing the actual tree.

- A second extension to OPTNBDT is to develop further tricks that allow us to efficiently build and compare a *set* of related decision trees — perhaps trees that are based on slightly different distributions, obtained from slightly different training samples. This might be relevant in the context of some ensemble methods [79], such as boosting or bagging, or in the context of budgeted learning of bounded classifiers [47].

- Our approach in OPTNBDT uses a simple cost model, where every feature has unit cost. An extension would allow different features to have different costs, where the goal is to produce a decision tree whose "cost depth" is bounded.[1]

- In Chapter 4 we found that LMU can be effective for the image segmentation task by producing a segmenter using very few segmented images. It may be possible to apply these ideas (and perhaps LMU) to other structured data, such as text, where the label of each word in a sentence is dependent on the label of the other words in the sentence.

- The current version of LMU assumes that all the images have the same labeling cost. There could be scenarios where the labeling cost varies from different sources. For example, the cost of segmenting tumors in human brain by a radiation oncologist can be more than that of a medical student. Here, the active learner could request the label of some images from the medical student, and others, from the radiation oncologist, with the understanding that it is important for the active learner to be cost-efficient while producing an accurate classifier.

- The *ISAL* algorithm of Chapter 5 uses a linear separator as its classifier. We anticipate further improvements by extending *ISAL* to other class of separators on non-linear data. This involves finding an appropriate sampling distribution for the given class of separators such that *ISAL* maintains its sample efficiency. One solution is to transform the data into a new space such that the data becomes linearly separable in this new space. This involves learning the appropriate transform function from the current space to the latent space.

- One application of *ISAL* can be in situations where instances from one class are

[1] The "cost depth" of a leaf of a decision tree is the sum of the costs of the tests of the nodes connecting the root to this leaf; and the "cost depth" of a tree is the maximal cost depth over the leaves. Note "cost depth" equals "depth" if all features have unit cost.

less frequent than other classes. In these scenarios, measures such as F-measure are typically used to evaluate the performance of a classifier. Here, one may consider modifying *ISAL* to use the F-measure instead of the current weighted loss function. Consequently, the convergence rate analysis for the *ISAL* should be modified to reflect these changes.

6.2 Summary of Contributions

The first part of this dissertation presents:

- OPTNBDT, a novel algorithm to produce the near optimal fixed-depth decision tree under Naïve Bayes assumption. We prove that under the Naïve Bayes assumption the optimal feature at the last level of the tree depends only on $P(Y = +|\pi)$, the posterior probability of the class label given the tests previously performed. Therefore, in a pre-processing step we precompute a list of features to use in the final layer. OPTNBDT uses this list to quickly assign a final feature based on a given path in the tree. We prove that OPTNBDT is computationally more efficient than the naïve method of testing all the features in the last level and provide empirical evidence to support this claim. Our empirical results illustrate that OPTNBDT is often more accurate than ID3 entropy-based decision tree.

We also extend the research on active learning in the following ways:

- *Active learning for image segmentation tasks*. Chapter 4 focuses on developing an effective active learner LMU for image segmentation. While there are now many results in active learning, this is one of the first studies that considers the challenge of actively learning the parameters of a machine learning model (CRF-based segmenter) for images. While LMU system is based on standard "parts" — selecting the image with maximal uncertainty, and the one that most reduces the uncertainty of other images — we found that this particular combination was effective for this task and it could produce an effective segmenter using very few segmented images.

- *Importance sampling active learning algorithm*. In chapter 5 we introduce a new active learning algorithm *ISAL*, based on importance sampling technique, where instances are queried from an appropriate distribution. *ISAL* provides an active learning framework that can be applied to many classifiers. We show that *ISAL*

can effectively reduce the cost of labeling, a useful feature for scenarios with high label costs. In addition, *ISAL* does not maintain a version space, which means it always has the possibility of converging to the best in class classifier. It uses importance weights to reduce the bias of error estimation, while controlling the variance of the estimation. It also is more sample-efficient by directly sampling from an appropriate distribution.

Bibliography

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[2] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[3] L. Breiman, J. Friedman, J. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[4] P. Auer, R. C. Holte, and W. Maass. Theory and applications of agnostic PAC-learning with small decision trees. In *ICML*, 1995.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[6] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, 2001.

[7] Cha Zhang and Tsuhan Chen. An active learning framework for content based information retrieval. *IEEE trans. on Multimedia, Special Issue on Multimedia Database*, 4:260–268, 2002.

[8] Alireza Farhangfar, Russell Greiner, and Csaba Szepesvári. Learning to segment from a few well-selected training images. In *ICML*, 2009.

[9] Sudheendra Vijayanarasimhan and Kristen Grauman. Multi-level active prediction of useful image annotations for recognition. In *NIPS*, 2008.

[10] David Lewis and William Gale. A sequential algorithm for training text classifiers. In *Proceedings of the Special Interest Group on Information Retrieval (ACM-SIGIR)*, volume 29, pages 13–19, 1994.

[11] Andrew Kachites Mccallum. Employing em in pool-based active learning for text classification. In *ICML*, 1998.

[12] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In *Journal of Machine Learning Research*, pages 999–1006, 2000.

[13] Steven Hoi, Rong Jin, Jianke Zhu, and Michael Lyu. Large-scale text categorization by batch mode active learning. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 633–642, New York, NY, USA, 2006. ACM.

[14] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *ICML*, 1999.

[15] 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 (EMNLP)*, 2008.

[16] Gokhan Tur, Dilek Hakkani-Tur, and Robert E. Schapire. Combining active and semisupervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, 2005.

[17] Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.

[18] Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. The automation of science. *Science*, 324(5923):85–89, 2009.

[19] Wei Chu, Martin Zinkevich, Lihong Li, Achint Thomas, and Belle Tseng. Unbiased online active learning in data streams. In *KDD*, 2011.

[20] Alina Beygelzimer, Daniel Hsu, John Langford, and Tong Zhang. Agnostic active learning without constraints. In *NIPS*, 2010.

[21] Vikram Krishnamurthy. Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Transactions on Signal Processing*, 50(6):1382–1397, 2002.

[22] Hwanjo Yu. SVM selective sampling for ranking with application to data retrieval. In *KDD*, pages 354–363, 2005.

[23] Ido Dagan and Sean Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, 1995.

[24] A. Asuncion and D.J. Newman. UCI machine learning repository, 2010. http://archive.ics.uci.edu/ml.

[25] David Cohn, Richard Ladner, and Alex Waibel. Improving generalization with active learning. In *Machine Learning*, pages 201–221, 1994.

[26] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *ICML*, 2006.

[27] Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In *NIPS*, 2007.

[28] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *COLT*, 2007.

[29] Ravi Ganti and Alexander Gray. Upal: Unbiased pool based active learning. In *AI Statistics*, 2012.

[30] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS*, 2005.

[31] Martin Szummer, Pushmeet Kohli, and Derek Hoiem. Learning CRFs using graph cuts. In *ECCV*, pages 582–595, 2008.

[32] ChiHoon Lee, Shaojun Wang, Matthew Brown, Albert Murtha, and Russell Greiner. Segmenting brain tumors using pseudo-conditional random fields. In *MICCAI*, pages 359–366, 2008.

[33] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Beyond regret. *Journal of Machine Learning Research - Proceedings Track 19*, pages 559–594, 2011.

[34] Alireza Farhangfar, Russell Greiner, and Martin Zinkevich. Near optimal fixed-depth decision tree. In *International Symposium on Artificial Intelligence and Mathematics*, 2008.

[35] Bruce Buchanan, Tom Mitchell, Reid Smith, and C. Richard Johnson. Models of learning systems. *Encyclopedia of Computer Science and Technology*, 11:24–51, 1978.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009.

[37] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[38] D. Dobkin, D. Gunopoulos, and S. Kasif. Computing optimal shallow decision trees. In *International Symposium on Artificial Intelligence and Mathematics*, Miami, FL, 1996.

[39] P. D. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning (ICML-2000)*, 2000.

[40] R. Greiner, A. Grove, and D. Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139:137–174, September 2002.

[41] Charles Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. Decision trees with minimal costs. In *ICML*, 2004.

[42] Victor Sheng, Charles Ling, Ailing Ni, and Shichao Zhang. Cost-sensitive test strategies. In *AAAI*, 2006.

[43] Pavel Brazdil Alberto Freitas, Altamiro Costa-Pereira. Cost-sensitive decision trees applied to medical data. *Lecture Notes in Computer Science, Proceedings of the 9th international conference on data warehousing and knowledge discovery*, 4654:303–312, 2007.

[44] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[45] D. D. Lewis. Naïve Bayes at forty: The independence assumption in information retrieval. In *ECML*, 1998.

[46] H. Chan and A. Darwiche. Reasoning about bayesian network classifiers. In *UAI*, 2003.

[47] A. Kapoor and R. Greiner. Learning and classifying under hard budgets. In *ECML*, 2005.

[48] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[49] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, 2001.

[50] Yuhong Guo and Russ Greiner. Optimistic active learning using mutual information. In *IJCAI*, 2007.

[51] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

[52] Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher, Roman Englert, and Yuval Elovici. Improving the detection of unknown computer worms activity using active learning. In *KI '07: Proceedings of the 30th annual German conference on Advances in Artificial Intelligence*, pages 489–493, Berlin, Heidelberg, 2007. Springer-Verlag.

[53] Aron Cullota and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–751, 2005.

[54] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *ICML*, 2000.

[55] Colin Campbell, Nello Cristianini, and Alex Smola. Query learning with large margin classifiers. In *ICML*, 2000.

[56] Jeremy Lewi, Robert Butera, and Liam Paninski. Sequential optimal design of neurophysiology experiments. *Neural Computation*, (21):619–687, 2009.

[57] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT*, 1992.

[58] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[59] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *ICCV*, pages 1150–1157, 2003.

[60] Chi hoon Lee, Feng Jiao, Shaojun Wang, Dale Schuurmans, and Russell Greiner. Learning to model spatial dependency: Semi-supervised discriminative random fields. In *NIPS*, 2006.

[61] Julian Besag. On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society, Series B*, 48:259–302, 1986.

[62] Manik Varma and Andrew Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *ECCV*, volume 3, pages 255–271, 2002.

[63] Val M. Runge. *Clinical MRI*. SAUNDERS, 2002.

[64] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[65] Pinar Donmez and Jaime G. Carbonell. Optimizing estimated loss reduction for active sampling in rank learning. In *ICML*, 2008.

[66] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, April 2007.

[67] Maria-Florina Balcan and Avrim Blum. A pac-style model for learning from labeled and unlabeled data. In *COLT*, 2005.

[68] Philip Michael Long. On the sample complexity of pac learning halfspaces against the uniform distribution. *IEEE Transactions on Neural Networks*, 6:1556–1559, 1995.

[69] Steve Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, 2007.

[70] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48:464–497, 1994.

[71] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 3(82):247–261, 1989.

[72] Steve Hanneke. Rates of convergence in active learning. *Annals of Statistics*, 39 (1):333–361, 2011.

[73] Vladimir Koltchinskii. Rademacher complexities and bounding the excess risk in active learning. *Journal of Machine Learning Research*, 11:2457–2485, 2010.

[74] Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.

[75] Rui Castro and Robert Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54(5):2339–2353, 2008.

[76] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19:357–367, 1967.

[77] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Random averages, combinatorial parameters, and learnability. *CoRR*, abs/1006.1138, 2010.

[78] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[79] Thomas G. Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.