The Bridge Between Theory and Applications: Examining Bias in Nonlinear Classification

a thesis presented
by
Travis McVoy
to
The Department of Computer Science

in partial fulfillment of the requirements
for the degree of
Bachelor of Arts
with honors
in the subjects of
Mathematics and Computer Science

Skidmore College
Saratoga Springs, New York
May 2025

Thesis advisor: Professor Tom O'Connell Travis McVoy

The Bridge Between Theory and Applications: Examining Bias in Nonlinear Classification

Abstract

There is a theorem (No Free Lunch) that tells us that no universal learner exists. In other words, there is no algorithm or learning scheme that can perform arbitrarily well on any learning task. In order to learn, then, we as scientists must induce some sort of bias into the model. For example, if one suspects a linear relationship in their data, they can bias their model toward linear functions. The same principle applies if one expects a polynomial relationship.

A related idea to No Free Lunch is the Bias-Variance Tradeoff. In essence, the more expressive one's model is, the more often it could potentially be wrong—leading to more variability in accuracy. Conversely, if one has introduced extreme bias into their model, the variability of their model's accuracy will be minimal—which is desirable if and only if the model accuracy is high. For instance, if the data has a polynomial relationship and the model is using a linear predictor, the variability in the model's accuracy will probably be low, but the accuracy will also be poor.

In this work, we aim to provide progress towards an answer to the question, "How much bias is enough? How far can we push the bias-variance tradeoff before our accuracy is too variable?"

In practice, the bias one would induce into a model is just specialized domain knowledge, and acquiring that knowledge may be expensive or even impossible. Thus, it is helpful if we can guarantee a reliable model with as little apriori knowledge as possible. For instance, in binary classification if we know our data is linearly separable, that alone is enough to achieve good results. This work examines how we can achieve analogous results in the case of nonlinearly separable data.

Our results are comprised of numerous experiments that examine the importance of bias in machine learning. One of the major themes of these experiments is structure. In particular, we partitioned nonlinear classification in $\mathbb{R}^2$ into several different categories so as to exploit the inherent structure of different kinds of data sets. In the future we hope to provide theoretical guarantees to accompany our empirical findings.

How To Read This Thesis

We assume some familiarity with a variety of different mathematical disciplines. Most prominently, comfort with probability and set theory is a necessity. Some exposure to analysis and topology would not hurt, but is not necessary. As for the skeleton of the thesis, we provide the following outline.

- Chapter 1 provides the learning theoretic ideas such as PAC learning, No Free Lunch, and VC dimension necessary for a discussion of bias in the field of machine learning. This chapter is long and difficult, but we urge the reader to understand hypothesis classes (Definition 1.4), the definitions surrounding probably approximately correct learning (especially Definition 1.11 and Definition 1.12), and empirical risk minimization Definition 1.8 at a minimum.

- Chapter 2 introduces the importance of structure. In short, linear data is not easy to learn because it is linear, but because we are assuming it is linear and that our assumption holds. If we make assumptions about the structure of nonlinear data and those assumptions hold, nonlinear classification is not all that different from linear classification.

- The most important ideas in Chapter 2 are contained in Section 2.2 where we discuss loops. Loops are so essential to our work that we dedicate an entire chapter to them in Chapter 3. In Chapter 3 we discuss different kinds of loops and how certain assumptions about data labeled by loops represents stronger or weaker apriori knowledge of the dataset.

- In Chapter 4 we explain in detail the experiments we will conduct to examine the ideas discussed in Chapter 3.

- Finally, in Chapter 5 we analyze the results of our experiments from Chapter 4.

A Remark About Figures

Some of our figures will probably be comically large on a printed copy. It was not clear to us how large figures needed to be in order to be pleasantly legible. Consequently, we generally made figures much larger than they may need to be. Additionally, the use of color is quite important in our plots as we make use of heat maps to examine variability. In an initial print, we found that some of the finer details of some heat maps were not captured by the printer. Tom will have an electronic copy of this work, should the committee need it.

Acronyms

We make use of several acronyms that we summarize here for the convenience of the reader.

- **ERM**: Empirical Risk Minimization. ERM is what models use to learn.
- **PAC**: Probably Approximately Correct. PAC is an uncertainty quantification.
- **NFL**: No Free Lunch. This refers to an important theorem about the nonexistence of a universal learner.
- **VC**: VC is shorthand Vapnik and Chervonenkis, the names of two contributors to the VC dimesnion, a major idea in learning theory.
- **FTL**: Fundamental Theorem of Learning. FTL states that a finite VC dimension is equivalent to learnability.

Acknowledgments

First and foremost, I would like to thank my advisor, Tom O'Connell for his patience, guidance, and support. Over the past few years I have spent hundreds of hours with Tom via coursework, various research credits, and of course, this thesis. His contributions to my progress as a scholar cannot be overstated. Most significant are his affinity for advanced mathematics in everything he teaches and his willingness to offer students flexibility and freedom to explore creative projects. An honorable mention is given to Tom's wonderful sense of humor. Without Tom this work would not exist, I likely would not have pursued the computer science major, and I doubt I would have pursued graduate study. Tom, I am forever in your debt. On behalf of all your students, I thank you for everything you have done for the computer science curriculum and for the endless—if I am not mistaken, Tom advised three! of the five theses in computer science this year on top of other projects like one credit research courses and project intensive courses like machine learning and AI—research opportunities that you provide.

I would also like to thank Patrick Daniels, Greg Malen, and Chris Seaton for their guidance surrounding measure theoretic and topological ideas. Patrick kindly offered to support my interests in measure theory despite never having had me as a student. His teachings inspired the idea for grid classifiers, a major result of this work. Greg and Chris assisted my thesis by providing some instruction on topology basics, which proved useful when discussing the structure of nonlinear data.

I reserve my final thanks to my parents who have continually provided support in my academic endeavors. To this day, I still remember being quizzed on multiplication tables by my mom as I swung on a swing set—a memory I will cherish for a lifetime.

Thank you all so much for the work you have done to support me and the work you will continue to do for future students.

Contents

It is an error to believe that rigor is the enemy of simplicity. On the contrary, we find it confirmed by numerous examples that the rigorous method is at the same time the simpler and the more easily comprehended.

David Hilbert

1 Learning Theory Fundamentals

In what follows we will summarize major concepts from Valiant's celebrated probably approximately correct (PAC) framework[13] and important theoretical extensions like the No Free Lunch Theorem[15] and Vapnik–Chervonenkis (VC) theory[14]. We believe it is to the reader's benefit to try to understand proofs of these fundamental results. To that end, we have made efforts to adapt proofs from graduate texts/notes[10,11,9] and provide extra commentary on details conventionally left to the reader.

1.1 PAC Learning

The goal of the PAC learning framework is to quantify uncertainty via two parameters, a confidence choice and an error tolerance. In simple terms, the results surrounding PAC learning determine the conditions necessary in order to achieve a learner that is "mostly correct most of the time."

1.1.1 The Learner's Input and Output

Definition 1.1 *The learner's input, denoted by S, is a finite **sequence** of n training pairs $\{(x_i, y_i)\}_{i=1}^{n}$ from $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the domain in which our data exists and $\mathcal{Y}$ is the set of*

all possible labels an element in $\mathcal{X}$ could be assigned to.

We call S a sequence, not a set, because the order in which a learner receives data could be important. Notice also that we have intentionally not specified the structure of $\mathcal{X}$ and $\mathcal{Y}$. The sets $\mathcal{X}$ and $\mathcal{Y}$ will change given the nature of the learning task. For much of this document, $\mathcal{X}$ will be $\mathbb{R}^2$ and $\mathcal{Y}$ will be the set of binary labels $\{0, 1\}$, but $\mathcal{X}$ will change when we discuss higher dimensions and $\mathcal{Y}$ will change when we discuss multiclass classification.

Definition 1.2 *The output of a learner is a function $h : \mathcal{X} \to \mathcal{Y}$. We call h a hypothesis, predictor, or classifier, as it is the function we use to make conclusions about unseen data.*

The reader may wonder why we call the output function h, instead of f. We reserve f for what is called the "true labeling function".

Definition 1.3 *The function $f : \mathcal{X} \to \mathcal{Y}$ is responsible for the true labels of our data. That is, f is the unknown function that always satisfies $f(x_i) = y_i$ when instances are labeled deterministically* [1].

The informal goal of a learner is to return a hypothesis that is "as similar to f as possible." The learner achieves this goal in part by searching through what is called a hypothesis class.

Definition 1.4 *The set of all functions a learning model can output is called a **hypothesis class**, and is denoted by $\mathcal{H}$.*

Observe that the choice of hypothesis class has serious ramifications on the learner's performance. In particular, if f is not included in $\mathcal{H}$—which is entirely within reason if we do not have highly specialized domain knowledge prior to learning—then it may not be possible to perfectly label all instances.

[1] In this work, we will assume labels are always deterministic, but know that that is not always the case. Noisy data, for instance, can be interpreted as data that is labeled stochastically.

1.1.2 The Risk of the Learner

In order for a learner to learn, it must have a method of assessing the quality of a hypothesis and its ability to predict or classify something. We assess a single instance with what is called a loss function.

Definition 1.5 *Given a hypothesis class $\mathcal{H}$ and a learning space*[2] *$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, we define a loss function ℓ to be any function mapping pairs (h, z) to a nonnegative real number:*

$$\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+.$$

Different learning tasks have different loss functions. For our work, the only loss function of importance is the $0, 1$ loss:

$$\ell_{0,1}(h, (x, y)) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y. \end{cases}$$

The $0, 1$ loss simply tracks whether a single instance is labeled correctly. Since loss functions only assess a single instance, they alone are not sufficient in assessing the error of a hypothesis as a whole. For a general assessment of a hypothesis, we define the notion of risk.

Definition 1.6 *The **true risk** incurred by a hypothesis h is defined to be the expected loss of h with respect to a joint probability distribution $\mathcal{D}$ over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$:*

$$L_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[\ell(h, z)].$$

The true risk is what we as theorists use to make arguments about learning schemes, but it is not actually useful in practice because we will never know the distribution $\mathcal{D}$. Since we cannot compute the true risk, we define the empirical risk via the empirical expectation.

[2] The reader may be curious as to whether there is any significance to denoting the learning space as $\mathcal{Z}$ instead of just $\mathcal{X} \times \mathcal{Y}$. The main reason for this is brevity. We will be referring to the learning space frequently throughout the document and notation can sometimes get quite cluttered if we write out $\mathcal{X} \times \mathcal{Y}$ instead of $\mathcal{Z}$ or (x, y) instead of z. Ultimately, though, the reader should be comfortable with either notation as they are equivalent.

Definition 1.7 *The **empirical risk** incurred by a hypothesis h is defined to be the expected loss over the sample S:*

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, z_i).$$

Observe that risk with the respect to the $0, 1$ loss is interpreted as the probability of mislabeling an instance. That is, in the discrete expectation

$$\mathop{\mathbb{E}}_{z \sim \mathcal{D}}[\ell(h, z)] = \sum_{z \in \mathcal{Z}} \ell_{0,1}(h, z) \mathbb{P}(Z = z),$$

a correctly classified instance receives zero weight in the sum whereas an incorrectly classified instance is weighted by its probability of being sampled. Similar statements apply for a continuous distribution.

1.1.3 Empirical Risk Minimization

Recall that we (informally) introduced learning as a search for a function that is "similar" to some true labeling function f. The formal goal of learning is to minimize $L_{\mathcal{D}}(h)$, the true risk of a hypothesis with respect to a distribution $\mathcal{D}$ and learning space $\mathcal{Z}$. Since we cannot compute the true risk directly, the next best attempt of risk minimization is to minimize empirical risk.

Definition 1.8 *A hypothesis that minimizes empirical risk with respect to a training sequence S is denoted h_S. Formally, h_S is given by*

$$h_s \in \mathop{\textit{argmin}}_{h \in \mathcal{H}} L_S(h).$$

Empirical risk minimization (ERM) is an effective technique for minimizing true risk under certain conditions, but it is also subject to failure if we are not careful. Consider a hypothesis h_{bad} that memorizes the training data:

$$h_{bad}(x_i) = \begin{cases} y_i & \text{if } x_i \in S \\ 0 & \text{otherwise.} \end{cases}$$

There are several problems with h_{bad} even though h_{bad} achieves zero empirical risk. First, the probability that h_{bad} misclassifies an instance is equal to the probability of sampling a positively[3] labeled instance that we have not seen before. This probability could be high, which would lead to very poor generalization[4]. Moreover, imagine that our learner is attmepting to learn to classify security threats, where a positive instance denotes, "yes, this is a threat." In such a scenario, h_{bad} will *never* identify an unseen threat, rendering the learning of h_{bad} pointless. In short, h_{bad} drastically overfits to the training data.

1.1.4 Avoiding Overfitting

Overfitting can be avoided with the appropriate use of bias. If we have some domain specific knowledge, we may suspect what the optimal hypothesis may be. In such a case, we may want to bias our learner by choosing a hypothesis class that represents our prior knowledge/beliefs. Meaning, we should restrict the hypothesis class $\mathcal{H}$. If, for example, we suspect there is a linear relationship between $\mathcal{X}$ and $\mathcal{Y}$, we could restrict $\mathcal{H}$ to some set of linear functions to avoid a polynomial overfit (Fig. 1.1).

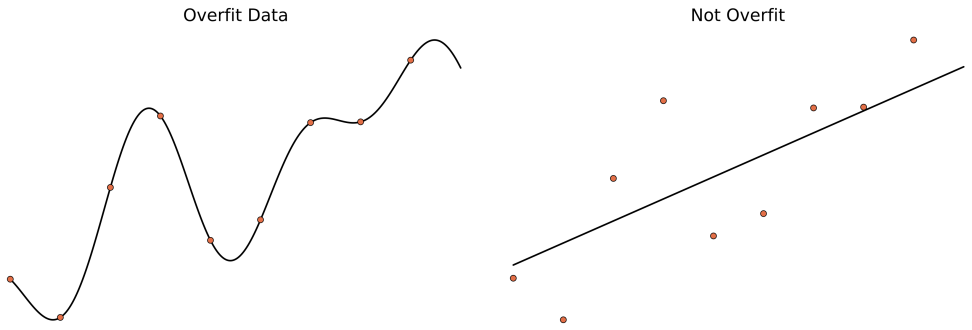


Figure 1.1: Polynomial Overfits Data

We could also restrict $\mathcal{H}$ by forcing it to be finite (or if it already is finite, to be smaller than it was before).

[3] We will interchange "labeled as 1" with "positively labeled" throughout. Admittedly, this is a bit confusing because zero is not negative in the mathematical sense. Note, though, that when we say positive and negative, we are really referring to more general notions of yes/no or good/bad.

[4] Generalization is another way of describing the learner's performance. If the learner has high risk, it does not generalize well to unseen data.

1.1.5 Realizable Learning

PAC learning is founded upon the characterization of generalization via a confidence parameter δ and an error tolerance ϵ. For an example, suppose our choices were $\epsilon = \delta = 0.1$. A learner would then be a realizable PAC learner if it labeled 90% of instances correctly 90% of the time after running ERM on sufficiently many samples. The problem with realizable PAC learning is that it is not always possible with a reasonable choice of $\mathcal{H}$. Realizable PAC learning requires the realizability assumption—a very bold assumption—to hold.

Definition 1.9 *The realizability assumption assumes the existence of a hypothesis $h^* \in \mathcal{H}$ for which $L_{\mathcal{D}}(h^*) = 0$ with respect to the $0, 1$ loss. In other words, the probability of sampling an instance mislabeled by h^* is zero.*

While this may be a dubious assumption, it is still worth consideration as realizable PAC learning allows for desirable theoretical guarantees. Namely, if realizability holds for a *finite* hypothesis class, we can guarantee arbitrarily low error with a sufficiently large sample. To prove this, we first require a formal definition of PAC learnability and sample complexity.

Definition 1.10 *The **sample complexity** of a hypothesis class, denoted $n_{\mathcal{H}}$, is a function of ϵ, δ that denotes the minimal number of instances needed in a sample to PAC learn over $\mathcal{H}$.*

To (realizably) PAC learn over $\mathcal{H}$ is to satisfy the following definition.

Definition 1.11 *A hypothesis class $\mathcal{H}$ is realizably PAC learnable if there exists a function $n_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm $\mathcal{A}$ with the following property:*

For every choice of $\epsilon, \delta \in (0, 1)$ and for every $\mathcal{D}$ over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, if the realizability assumption holds with respect to $\mathcal{H}$ and $\mathcal{D}$, then when running $\mathcal{A}$ on $n \geq n_{\mathcal{H}}(\epsilon, \delta)$ examples generated by $\mathcal{D}$, $\mathcal{A}$ returns an h that satisfies $L_{\mathcal{D}}(h) \leq \epsilon$.

Note that realizable learning is restricted to the $0, 1$ loss, but again, that is the only loss we will consider in this work. We now prove that finite hypothesis classes are realizably learnable.

Proposition 1.1 *If $\mathcal{H}$ is a finite, realizable hypothesis class, then $\mathcal{H}$ is realizably PAC learnable via ERM and has sample complexity*

$$n_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil .$$

Proof. First, let us outline the core of our argument. Since realizability holds, there is an $h^* \in \mathcal{H}$ for which $L_{\mathcal{D}}(h^*) = 0$. Hence, with probability 1, ERM will find at least one hypothesis with zero empirical risk on any sample we draw. The question of importance is whether zero empirical risk implies low true risk. If h^* is the only hypothesis with zero empirical risk, then clearly, ERM will always pick h^* and low risk is achieved. Assume, then, that there exist multiple hypotheses with zero empirical risk and that some of them have low true risk and some do not.

If we can bound the probability of drawing a sample with zero empirical risk and high true risk to be at most δ, then by complementation, the probability of a good sample (zero empirical risk and low true risk) is at least $1 - \delta$. To achieve our bound, we formalize our argument by defining two sets

$$\mathcal{H}_{bad} = \{h \in \mathcal{H} : L_{\mathcal{D}}(h) > \epsilon\}$$

and

$$S_{bad} = \{S \in (\mathcal{X} \times \mathcal{Y})^n : \exists h \in \mathcal{H}_{bad} \text{ for which } L_S(h) = 0\} .$$

The set $\mathcal{H}_{bad}$ is the set of hypotheses which violate our error tolerance and S_{bad} is the set of samples with n instances for which empirical risk is zero and true risk is poor. Our goal is to bound the probability of drawing sample from S_{bad}. Observe that we can write S_{bad} as a union over the h in $\mathcal{H}_{bad}$:

$$S_{bad} = \bigcup_{h \in \mathcal{H}_{bad}} \{S \in (\mathcal{X} \times \mathcal{Y})^n : L_S(h) = 0\} .$$

Now let $S \sim \mathcal{D}^n$ denote the IID sampling of n instances from $\mathcal{D}$ and then we have

$$\begin{aligned}
\mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}[S \in S_{bad}] &= \mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}\left[\bigcup_{h\in\mathcal{H}_{bad}} \{S \in (\mathcal{X}\times\mathcal{Y})^n : L_S(h) = 0\}\right] && (1.1)\\
&\leq \sum_{h\in\mathcal{H}_{bad}} \mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}[L_S(h) = 0] && (1.2)\\
&= \sum_{h\in\mathcal{H}_{bad}} \prod_{i=1}^{n} \mathop{\mathbb{P}}_{x\sim\mathcal{D}}[h(x_i) = f(x_i)] && (1.3)\\
&\leq \sum_{h\in\mathcal{H}_{bad}} \prod_{i=1}^{n} (1-\epsilon) && (1.4)\\
&\leq |\mathcal{H}|(1-\epsilon)^n. && (1.5)
\end{aligned}$$

For clarity, we provide some additional explanation. Eq. (1.2) follows from the inclusion-exclusion bound. That is, for two sets A, B, we have

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$$

which implies $\mathbb{P}(A \cup B) \leq \mathbb{P}(A) + \mathbb{P}(B)$. A similar argument holds for more than two sets. The step from Eq. (1.2) to Eq. (1.3) follows from IID sampling and the fact that zero empirical risk happens if and only if $h(x) = f(x)$ for all x in the sample. Since each hypothesis we are considering is a bad hypothesis, the risk of h is greater than ϵ. Recall that risk with respect to the $0, 1$ loss is simply the probability of misclassifying an instance and therefore, Eq. (1.4) follows from complementary probability. Lastly, Eq. (1.5) follows from the fact that $|\mathcal{H}_{bad}| \leq |\mathcal{H}|$. To conclude the proof, we use the fact that $1 - \epsilon \leq e^{-\epsilon}$ to write

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}[S \in S_{bad}] \leq |\mathcal{H}|(1-\epsilon)^n \leq |\mathcal{H}|e^{-\epsilon n}$$

and then we pick an n such that $\delta \geq |\mathcal{H}|e^{-\epsilon n}$. Rearranging yields

$$\epsilon n \geq \ln(|\mathcal{H}|/\delta) \quad \Rightarrow \quad n \geq \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil.$$

We have now shown that that if we use $\left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$ or more samples when running ERM, the probability of obtaining a sample with zero empirical risk and poor true risk is at most δ. Now, notice that a sample cannot have low and high true risk at the same time. Thus, the probability of obtaining a sample with zero risk and low true risk is at least $1 - \delta$, and therefore, $\mathcal{H}$ is PAC learnable with a minimum number of samples

$$n_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil.$$

□

1.1.6 Agnostic Learning

If realizability does not hold, we cannot guarantee that there always exists a hypothesis in $\mathcal{H}$ that achieves zero risk. Learning without the realizability assumption is called agnostic learning, and requires a slight modification of our perspective on what it means to successfully learn.

Definition 1.12 *A hypothesis class $\mathcal{H}$ is agnostically[5] PAC learnable with respect to learning space $\mathcal{Z}$ and loss function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ if there exists a function $n_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property:*

For every choice of $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{Z}$, when running a learning algorithm $\mathcal{A}$ on $n \geq n_{\mathcal{H}}(\epsilon, \delta)$ IID instances sampled from $\mathcal{D}$, the algorithm $\mathcal{A}$ returns an $h \in \mathcal{H}$ such that with probability at least $1 - \delta$

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon.$$

Just as we proved finite hypothesis classes are realizably learnable, we can prove finite classes are agnostically learnable. We will, however, require a few additional ideas. Namely, we need to discuss what it means for a sample to be ϵ representative and how ϵ representative samples relate to a property called uniform convergence.

[5] Notice that if realizability holds, agnostic learning and realizable learning are equivalent because $\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') = 0$ in the realizable setting.

Definition 1.13 *A training sequence S from a learning space $\mathcal{Z}$ and distribution $\mathcal{D}$ is called ϵ-representative if for all $h \in \mathcal{H}$,*

$$|L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$$

holds with respect to $\mathcal{Z}$ and $\mathcal{D}$.

Intuitively, an ϵ representative sample is a sample that is "informative enough" to make the empirical risk and the true risk roughly equivalent. While this does not necessarily mean our sample is large, the most generic way to achieve an ϵ representative sample is to make the sample as large as possible. We now show that obtaining an ϵ representative sample is sufficient for agnostic learning.

Lemma 1.1 *If a sample S is $\epsilon/2$ representative with respect to a learning space $\mathcal{Z}$, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$, then any ERM hypothesis h_S satisfies*

$$L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

Proof. Assume S is $\epsilon/2$ representative. We can then write

$$|L_{\mathcal{D}}(h_S) - L_S(h_S)| \leq \frac{\epsilon}{2}.$$

When we drop the absolute value, we obtain

$$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\epsilon}{2}.$$

Since h_S is an ERM hypotehsis, Definition 1.8 implies

$$L_S(h_S) + \frac{\epsilon}{2} \leq L_S(h) + \frac{\epsilon}{2}$$

for any $h \in \mathcal{H}$. Further, since S is $\epsilon/2$ representative, we have

$$|L_{\mathcal{D}}(h) - L_S(h)| = |L_S(h) - L_{\mathcal{D}}(h)| \leq \frac{\epsilon}{2} \tag{1.6}$$

Combining Eq. (1.6) with all that precedes it yields

$$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\epsilon}{2} \leq L_S(h) + \frac{\epsilon}{2} \leq L_{\mathcal{D}}(h) + \frac{\epsilon}{2} + \frac{\epsilon}{2} = L_{\mathcal{D}}(h) + \epsilon$$

for all $h \in \mathcal{H}$. To conclude the proof, observe that since the above holds over all h, we have $L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$, so indeed, if S is $\epsilon/2$ representative, then agnostic learning is achieved with ERM.

□

We now need to determine how to ensure a sample is ϵ representative. Formally, we need to prove that we can achieve the uniform convergence property.

Definition 1.14 *We say that a hypothesis class $\mathcal{H}$ has the **uniform convergence property** with respect to a learning space $\mathcal{Z}$ and a loss function ℓ if there exists a function $n^{UC}_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{Z}$, if S is a sample of $n \geq n^{UC}_{\mathcal{H}}(\epsilon, \delta)$ examples drawn IID according to $\mathcal{D}$, then, with probability $1 - \delta$, the sequence S is ϵ representative.*

To prove that a sample enjoys the uniform convergence property, we make use of a concentration inequality called **Hoeffding's Inequality**.

Theorem 1.1 *Let $\theta_1, \ldots, \theta_n$ be a sequence of IID random variables such that $\mathbb{E}[\theta_i] = \mu$ and $\mathbb{P}[a \leq \theta_i \leq b] = 1$. That is, all random variables are bounded. Then, for any $\epsilon > 0$,*

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n}\theta_i - \mu\right| > \epsilon\right] \leq 2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right).$$

Before we begin a proof, let us draw some parallels between Hoeffding's Inequality and the notions of risk we are already familiar with. In particular, notice that our loss function is a random variable due to the fact that instances $z = (x, y)$ are drawn randomly with IID sampling. Moroever, the notions of risk are empirical and theoretical expectations with respect to the loss, just as

$$\frac{1}{n}\sum_{i=1}^{n}\theta_i \qquad \text{and} \qquad \mu$$

are to θ_i. Thus, Hoeffding's Inequality tells us that, for some fixed h,

$$\mathbb{P}\left[|L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\right] \leq 2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right) \tag{1.7}$$

when the loss function ℓ is bounded. Certainly, we need to consider more than just a single fixed hypothesis, but Eq. (1.7) paves the path to do so.

Theorem 1.2 *If $\mathcal{H}$ is a finite hypothesis class chosen for a learning space $\mathcal{Z}$ and $\ell : \mathcal{H} \times \mathcal{Z} \to [a, b]$ is a bounded loss function, then $\mathcal{H}$ enjoys the uniform convergence property with repsect to $\mathcal{Z}$ and has sample complexity*

$$n_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{(b-a)^2 \ln(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil.$$

Moreover, the class $\mathcal{H}$ is agnostically PAC learnable using ERM with sample complexity

$$n_{\mathcal{H}}(\epsilon, \delta) \leq n_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \left\lceil \frac{2(b-a)^2 \ln\left(2|\mathcal{H}|/\delta\right)}{\epsilon^2} \right\rceil.$$

Proof. In the realizability setting, we bounded the probability of drawing a sample for which our risk tolerance was violated. We begin in a similar manner, only this time, we wish to bound the probability of drawing a sample for which our uniform convergence tolerance is violated. Define the set of such samples to be

$$S_{bad} = \left\{S : \exists h \in \mathcal{H} \text{ for which } |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\right\}.$$

Like before, we write S_{bad} as a union, only now we need to consider all hypotheses (because we want convergence uniformly over $\mathcal{H}$) so we write

$$S_{bad} = \bigcup_{h \in \mathcal{H}} \left\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\right\}.$$

Applying both the inclusion-exclusion bound and Hoeffding's Inequality (and specifically, Eq. (1.7)), we obtain

$$
\begin{aligned}
\mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}[S \in S_{bad}] &\leq \sum_{h\in\mathcal{H}} \mathop{\mathbb{P}}_{S\sim\mathcal{D}^n}[\{S : |L_S(h) - L_\mathcal{D}(h)| > \epsilon\}] && \text{[Union Bound]} \\
&\leq \sum_{h\in\mathcal{H}} 2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right) && \text{[Hoeffding]} \\
&= |\mathcal{H}|2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right).
\end{aligned}
$$

We now choose δ such that $|\mathcal{H}|2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right) \leq \delta$ and rearrange

$$
\frac{2|\mathcal{H}|}{\delta} \leq \exp\left(\frac{2n\epsilon^2}{(b-a)^2}\right) \quad \Rightarrow \quad \ln\left(\frac{2|\mathcal{H}|}{\delta}\right) \leq \frac{2n\epsilon^2}{(b-a)^2}
$$

to find that if we use

$$
n \geq \left\lceil \frac{(b-a)^2 \ln(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil
$$

instances, then the probability of sampling an S such that $|L_S(h) - L_\mathcal{D}(h)| \leq \epsilon$ is at least $1-\delta$. Since $n_\mathcal{H}$ is the minimum such number that achieves the uniform convergence property,

$$
n^{\text{UC}}_\mathcal{H}(\epsilon,\delta) \leq \left\lceil \frac{(b-a)^2 \ln(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil
$$

and the first part of the proof is finished. The second part follows quickly by plugging $\epsilon/2$ into $n^{\text{UC}}_\mathcal{H}$. That is, from Lemma 1.1, we know that if our sample is $\epsilon/2$ representative, then we achieve $L_\mathcal{D}(h_S) \leq \min_{h\in\mathcal{H}} L_\mathcal{D}(h) + \epsilon$. We have previously shown that a sample with $n^{\text{UC}}_\mathcal{H}(\epsilon,\delta)$ instances is ϵ representative, so choosing $n^{\text{UC}}_\mathcal{H}(\epsilon/2,\delta)$ instances for our sample size yields the desired result

$$
n_\mathcal{H}(\epsilon,\delta) \leq n^{\text{UC}}_\mathcal{H}(\epsilon/2,\delta) \leq \left\lceil \frac{2(b-a)^2 \ln\left(2|\mathcal{H}|/\delta\right)}{\epsilon^2} \right\rceil
$$

and the proof is complete.

□

1.2 The No Free Lunch Theorem

We have previously discussed the importance of bias and how we can use it to avoid overfitting. Our first try at avoiding overfitting involved restricting hypotheses to some finite class. Two natural follow up questions are, "Do we need bias in order to learn?" and "Are hypothesis classes of infinite size learnable?" The second question we will address in a later section, but it turns out the two questions are not unrelated, as they are both answered by restricting our view to some subset C of our domain $\mathcal{X}$.

1.2.1 No Free Lunch

We now turn to the question, "Do we need bias to learn?" Inductive bias can be interpreted as prior knowledge or beliefs about the task we are trying to learn—for if we did not have prior knowledge/beliefs, how would we know what choice of $\mathcal{H}$ to make? In what follows we will address the necessity of such prior knowledge. That is, we will prove there is no universal learning algorithm.

The key idea behind our argument is this: if there was a universal learner, then said learner would have to achieve PAC learnability on any task we already know is PAC learnable. We have already shown that realizable binary classification is PAC learnable. Therefore, if we can argue that for any algorithm A (including the supposed "universal learner") there exists binary classification tasks where A can[6] fail, then we have shown that no such universal learner exists. This is exactly what we will do, but to do so, we start with a lemma:

Lemma 1.2 *Let A be any learning algorithm for the task of binary classification with respect to the 0-1 loss over a (finite) domain $\mathcal{X}$. Denote the size of the training sequence S by m. Let C be some subset of $\mathcal{X}$ with size $2n$. If A receives a sequence S of n instances (not necessarily distinct) from $C \times \{0,1\}$ and returns a function $A(S) : C \to \{0,1\}$, it holds that*

$$\max_{i \in [2^{2n}]} \mathop{\mathbb{E}}_{S \sim \mathcal{D}_i} [L_{\mathcal{D}_i}(A(S)] \geq 1/4.$$

[6] The word "can" is extremely important here. We aren't arguing that any algorithm fails. Rather, we are arguing that for any algorithm, there exists SOME task(s) on which it fails, but that's okay because in this case (binary classification) a different algorithm A' could learn that task. In other words, algorithms biased towards different results will perform differently on different tasks.

Proof. Observe that there are $T = 2^{2n}$ possible functions from C to $\{0, 1\}$:

	$f_i(c_1)$	$f_i(c_2)$	$\cdots$	$f_i(c_{2n})$
f_1	0	0	$\cdots$	0
f_2	0	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
f_T	1	1	$\cdots$	1

Next, for each function f_i, let $\mathcal{D}_i$ uniformly distribute instances over $C \times \{0, 1\}$ according to f_i. That is,

$$\mathcal{D}[(x, y)] = \begin{cases} 1/|C| & \text{if } y = f_i(x) \\ 0 & \text{otherwise.} \end{cases}$$

Notice that there are $k = (2n)^n$ possible sequences of n examples from C. We denote these sequences by $S_1, \ldots, S_k$ and we let S_j^i denote the sequence of labeled instances:

$$S_j^i = \big((x_1, f_i(x_1), \ldots, (x_n, f_i(x_n)\big).$$

Under this notation, if the distribution is $\mathcal{D}_i$ then the possible training sequences A can receive are $S_1^i, \ldots S_k^i$, and because $\mathcal{D}_i$ is uniformly distributed according to f_i, all of $S_1^i, \ldots S_k^i$ are equally likely to be sampled. Hence,

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^n} [L_{\mathcal{D}_i}(A(S))] = \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i) P(S_j^i) = \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i).$$

We now introduce a bound and the maximum function using what I call the Max-Min-Average (MMA) property. Namely, for some set of values, the maximum must be greater than or equal to the average, which must be greater than or equal to the minimum. Using the MMA property will allow us to rewrite our work and make progress.

Namely, we have

$$\begin{aligned}\max_{i\in[T]} \frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}\left(A\left(S_j^i\right)\right) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}\left(A\left(S_j^i\right)\right) \\ &= \frac{1}{k}\sum_{j=1}^{k}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}\left(A\left(S_j^i\right)\right) \\ &\geq \min_{j\in[k]} \frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}\left(A\left(S_j^i\right)\right).\end{aligned} \tag{1.8}$$

Remark 1.1 *In the above, the intermediate step is necessary. Notice that our maximum is over i whereas our minimum is over j. We achieve these bounds simply by flipping the sums and respective coefficients in the equality in the middle.*

Now observe that for any S_j, there must be some corresponding set of values $v_1, \ldots, v_p$ in C that do not appear in S_j, because $|S_j| = n < 2n = |C|$. If $|S_j| = n$ then $p = n$ so clearly $p \geq n$. That is, there are at least as many elements in C that are not in S_j as there are in S_j. Thus, for every function $h : C \to \{0, 1\}$ and every i we have

$$L_{\mathcal{D}_i}(h) = \sum_{x\in C} \mathbb{1}_{[h(x)\neq f_i(x)]} P(Z = (x, f(x)))$$

but by the definition of $\mathcal{D}_i$, we have $P(Z = (x, f(x))) = 1/|C|$ so the above can be rewritten as

$$L_{\mathcal{D}_i}(h) = \frac{1}{2n}\sum_{x\in C} \mathbb{1}_{[h(x)\neq f_i(x)]}.$$

Next, notice that even though $p \geq n$, it is never true that $p = 2n$. Since S needs to consist of at least one unique instance in C, the largest p can be is $2n - 1$. Hence,

$$L_{\mathcal{D}_i}(h) = \frac{1}{2n}\sum_{x\in C} \mathbb{1}_{[h(x)\neq f_i(x)]} \geq \frac{1}{2n}\sum_{r=1}^{p} \mathbb{1}_{[h(v_r)\neq f_i(v_r)]} \geq \frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{[h(v_r)\neq f_i(v_r)]} \tag{1.9}$$

where the last inequality follows because $p \geq n$ so $1/2n \geq 1/2p$. We can now do some work that will allow us to eventually rewrite Eq. (1.8) and complete the proof. Namely, combining our work from Eq. (1.9) with Eq. (1.8) we have

$$\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i) \geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{[A\left(S_j^i\right)(v_r)\neq f_i(v_r)]}.$$

Some rearrangement yields

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{[A\left(S_j^i\right)(v_r)\neq f_i(v_r)]} &= \frac{1}{2p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{[A\left(S_j^i\right)(v_r)\neq f_i(v_r)]} \\ &= \left(\frac{1}{2}\right)\frac{1}{p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{[A\left(S_j^i\right)(v_r)\neq f_i(v_r)]}\end{aligned}$$

which allows us to apply MMA to the last equality and get

$$\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \geq \frac{1}{2}\min_{r\in[p]}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{[A\left(S_j^i\right)(v_r)\neq f_i(v_r)]}. \tag{1.10}$$

From here, we are almost finished. We just need a way to evaluate the sum on the right hand side of the above. Since the minimum function is being applied, v_r is fixed. This allows us to partition $f_1, \ldots, f_T$ into two sets: the set of functions f_i where $f_i(v_r) = 1$ and the set of functions f_i where $f_i(v_r) = 0$. The use of this comes from the following fact: for every $f_i(v_r) = 0$, there is one and only one $f_{i'} \in f_1, \ldots, f_T$ that agrees with f_i on all points in C except v_r. Formally, for every $f_i(v_r) = 0$ there is one and only one $f_{i'}(v_r) = 1$ for which $f_i(x) = f_{i'}(x)$ for all x besides v_r. Moreover, because the only place where f_i and $f_{i'}$ differ is v_r and, by definition, v_r is not in either of S_j^i or $S_j^{i'}$, it must be true that S_j^i and $S_j^{i'}$ are the same sequence. Hence, the algorithm A will return the same function on both S_j^i and $S_j^{i'}$ so

$$\mathbb{1}_{[A\left(S_j^i\right)(v_r)\neq f_i(v_r)]} + \mathbb{1}_{[A\left(S_j^{i'}\right)(v_r)\neq f_{i'}(v_r)]} = 1$$

because (WLOG) $f_i(v_r) = 0$ and $f_{i'}(v_r) = 1$ so one (and only one) of them must be

correct. We can now rewrite Eq. (1.10) by grouping all the S_j^i and $S_j^{i'}$ pairs (of which there are $T/2$) together in the sum on the right. That is,

$$\frac{1}{2} \min_{r \in [p]} \frac{1}{T} \sum_{i=1}^{T} \mathbb{1}_{[A\left(S_j^i\right)(v_r) \neq f_i(v_r)]} = \frac{1}{2} \cdot \frac{1}{T} \cdot \frac{T}{2} = \frac{1}{4}.$$

Again, the min function will just fix our v_r so all our previous work still applies and the above is valid. Putting it all together, we have

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^n} [L_{\mathcal{D}_i}(A(S))] = \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}\left(A\left(S_j^i\right)\right) \qquad \text{[Uniformity of } \mathcal{D}_i\text{]} \tag{1.11}$$

$$\Rightarrow \max_{i \in [T]} \mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^n} [L_{\mathcal{D}_i}(A(S))] \geq \min_{j \in [k]} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}\left(A\left(S_j^i\right)\right) \quad \text{[MMA from (1.8)]} \tag{1.12}$$

$$\Rightarrow \max_{i \in [T]} \mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^n} [L_{\mathcal{D}_i}(A(S))] \geq 1/4 \qquad \text{[Because (1.10) evaluates to } 1/4\text{]} \tag{1.13}$$

which completes the proof because $T = 2^{2n}$.

□

We can now use Lemma 1.2 to prove the No Free Lunch theorem.

Theorem 1.3 (No Free Lunch) *Let A be any learning algorithm for the task of binary classification with respect to the* 0-1 *loss over a (finite) domain $\mathcal{X}$. Denote the size of the training sequence S by n. If n is any number smaller than $|\mathcal{X}|/2$, then there exists a distribution over $\mathcal{X} \times \{0, 1\}$ such that:*

1. *There exists a function $f : \mathcal{X} \to \{0, 1\}$ with $L_{\mathcal{D}}(f) = 0$ and,*
2. *with probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^n$, we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.*

In other words, when $m < |\mathcal{X}|/2$, there exists conditions for which A fails to learn.

Proof. Start by noticing that because $m < |\mathcal{X}|/2$, we can indeed find a subset C of size $2n$ that agrees with our work in Lemma 1.2. Meaning, we choose the $\mathcal{D}_i$ and f_i in Lemma 1.2 that come from

$$\max_{i \in [T]} \mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^n} [L_{\mathcal{D}_i}(A(S))] \geq 1/4$$

and extend them to all of $\mathcal{X}$. That is, for some subset C of $\mathcal{X}$ with size $2n$, let the distribution over $\mathcal{X}$ be

$$\mathcal{D}[(x, y)] = \begin{cases} 1/|C| & \text{if } f(x) = y \text{ and } x \in C \\ 0 & \text{otherwise} \end{cases}$$

where $f(x) = f_i(x)$ for $x \in C$, and (WLOG) $f(x) = 0$ if $x \notin C$. Notice that what we assign to any $x \notin C$ is irrelevant because an $x \notin C$ will never be be drawn when sampling according to $\mathcal{D}$. To show the first statement in No Free Lunch holds, observe that $L_{\mathcal{D}}(f)$ is clearly 0 because our probability distribution only allows us to sample correctly labeled instances from C. Now, from our previous work in Lemma 1.2, we have

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^n} [L_{\mathcal{D}}(A(S))] \geq 1/4.$$

We now make use of a corollary from Markov's inequality, which is proven in [10] (see Lemma B.1):

$$\mathbb{P}[Z > 1 - a] \geq \frac{\mu - (1 - a)}{a}.$$

In this case, we have

$$\mathbb{P}\left[L_{\mathcal{D}}(A(S)) \geq 1/8\right] \geq \frac{\mathop{\mathbb{E}}_{S \sim \mathcal{D}^n} [L_{\mathcal{D}}(A(S))] - 1/8}{7/8} \geq \frac{1/4 - 1/8}{7/8} = 1/7$$

so indeed, $\mathbb{P}\left[L_{\mathcal{D}}(A(S)) \geq 1/8\right] \geq 1/7$ which proves the second statement in No Free Lunch and finishes the proof.

□

1.3 VC Theory

VC theory is quite difficult—even more so than everything we have done thus far. While proofs are important, the difficulty of the proofs surrounding VC theory are such that they perhaps take away from the story we are telling more than they add, given the time our audience will have to read this work. Thus, we will not work through the typical proofs one would be exposed to when studying VC theory. The curious reader may find such proofs in *Understanding Machine Learning* [10]. We will, however, provide a brief summary of the major ideas that make VC theory so useful.

Definition 1.15 (Restriction of $\mathcal{H}$ to C) *Let $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to $\{0,1\}$ and let $C = \{c_1, \ldots, c_m\} \subset \mathcal{X}$. The restriction of $\mathcal{H}$ to C, denoted $\mathcal{H}_C$, is the set of functions from C to $\{0,1\}$ that are induced by $\mathcal{H}$ on C. That is, $\mathcal{H}_C$ is a subset of $\{0,1\}^{|C|}$ that contains all binary labelings of C when applying a hypothesis on C:*

$$\mathcal{H}_C = \{(h(c_1), \ldots, h(c_n)) : h \in \mathcal{H}\}.$$

The case where a restriction results in the **shattering** of a set C is of particular importance.

Definition 1.16 (Shattering) *If the restriction of $\mathcal{H}$ to a set $C \subset \mathcal{X}$ results in every possible binary labeling of C then we say $\mathcal{H}$ shatters C. That is, if $\mathcal{H}_C$ contains every function from C to $\{0,1\}$, then $|\mathcal{H}_C| = 2^{|C|}$ and C is shattered by $\mathcal{H}$.*

Though we will not rigorously establish why due to the difficulty of the proof, shattering allows us to characterize learnability of binary classification via a quantity called the VC dimension.

Definition 1.17 (VC Dimension) *The VC dimension of a hypothesis class $\mathcal{H}$, denoted by $VC(\mathcal{H})$ is the maximal size of a set $C \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$. If $\mathcal{H}$ can shatter sets of arbitrarily large size, then we say $\mathcal{H}$ has an infinite VC dimension.*

Remark 1.2 *If $VC(\mathcal{H}) = d$ then $\mathcal{H}$ shatters any subset C for which $|C| \leq d$.*

Notice that because $VC(\mathcal{H}) = d$, then for any subset $C \subset \mathcal{X}$ of size d, restricting $\mathcal{H}$ to C generates *all* binary labelings of the elements of C. Thus, for any[7] $C' \subset C$, it follows that $\mathcal{H}$ is also capable of shattering C'. We denote this visually with a table. In particular, for $|C| = d = VC(\mathcal{H})$ the set $|\mathcal{H}_C|$ contains all binary labelings of d elements.

	$f_i(c_1)$	$f_i(c_2)$	$\cdots$	$f_i(c_{2n})$
f_1	0	0	$\cdots$	0
f_2	0	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
f_T	1	1	$\cdots$	1

and for some subset $C' \subset C$, imagine we chopped off part of the table. For instance if $|C'| = n$ with $n < d$ then just consider all columns from 1 to n. The rows will contain some repeated elements, but clearly, all functions from C' to $\{0, 1\}$ will be contained in columns 1 to n. We now make use of NFL and the above definitions to see why the VC dimension of a hypothesis class is so important.

Corollary 1.1 *Let $\mathcal{H}$ be a hypothesis class from $\mathcal{X}$ to $\{0, 1\}$. Let n be a training sequence size for S. Assume that there exists a set $C \subset \mathcal{X}$ of size $2n$ that is shattered by $\mathcal{H}$. Then, for any algorithm A there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$ and a predictor $h \in \mathcal{H}$ such that $L_{\mathcal{D}}(h) = 0$ but with probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^n$ we have $L_{\mathcal{D}}(A(S)) \geq 1/8$.*

Directly from the Corollary 1.1, we have

Theorem 1.4 *If $\mathcal{H}$ is a hypothesis class of infinite VC dimension then $\mathcal{H}$ is not PAC learnable.*

Proof. Since $\mathcal{H}$ has infinite VC dimension, then for any training size we pick, there exists a shattered size of $2n$, which implies the failure of learnability from Corollary 1.1.

□

[7] Note: We say that for any $\mathcal{H}$, the empty set is shattered by $\mathcal{H}$.

At this point we might wonder, does a finite VC dimension imply learnability? Not only does a finite VC dimension imply learnability, this result is so important that it has earned the title, "The Fundamental Theorem of Learning".

Theorem 1.5 (FTL) *Let $\mathcal{H}$ be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0, 1\}$ and let the loss function be the 0-1 loss. Then, the following are equivalent:*

1. *$\mathcal{H}$ has the uniform convergence property.*
2. *Any ERM rule is a successful agnostic PAC learner for $\mathcal{H}$.*
3. *$\mathcal{H}$ is agnostic PAC learnable.*
4. *$\mathcal{H}$ is PAC learnable if the realizability assumption holds.*
5. *Any ERM rule is a successful PAC learner for $\mathcal{H}$ if the realizability assumption holds.*
6. *$\mathcal{H}$ has a finite VC dimension.*

Recall the algebraic and probabilistic cartwheels we did to make provable statements about finite hypothesis classes. Imagine that for every infinite hypothesis class we had to do similar cartwheels. That would be unfortunate. Instead, we need only show that a hypothesis class has finite VC dimension. With that, we are ready to begin.

When approaching a problem you do not know how to solve, turn it into a problem you do know how to solve.

The Art of Problem Solving

2

Partitioning the Family of Nonlinear Classification Tasks

The Fundamental Theorem of Learning (FTL) equates learnability with a finite VC Dimension. While this is a very powerful theoretical tool, it can be abused if we are not careful. Linear predictors, for instance, have a finite VC dimension but obviously, they do not generalize well to nonlinear data. Does this contradict the FTL? No. The FTL makes statements about learning in both the realizable and agnostic setting. If the data is nonlinearly separable, then realizability fails with respect to linear predictors and therefore the finite VC dimension of linear predictors implies they are agnostic learners with respect to nonlinear data.

Notice that agnostic learners are only as strong as the strongest hypothesis in their hypothesis class. In other words, being an agnostic learner could be meaningless, and indeed, we can see this visually in Fig. 2.1. Namely, if the data is nonlinear, but still loosely resembles a line, linear predictors might be a viable approximation choice. Conversely, if the data is nonlinearly separable by some function that does not resemble a straight line at all, then clearly, we need something stronger than linear predictors.

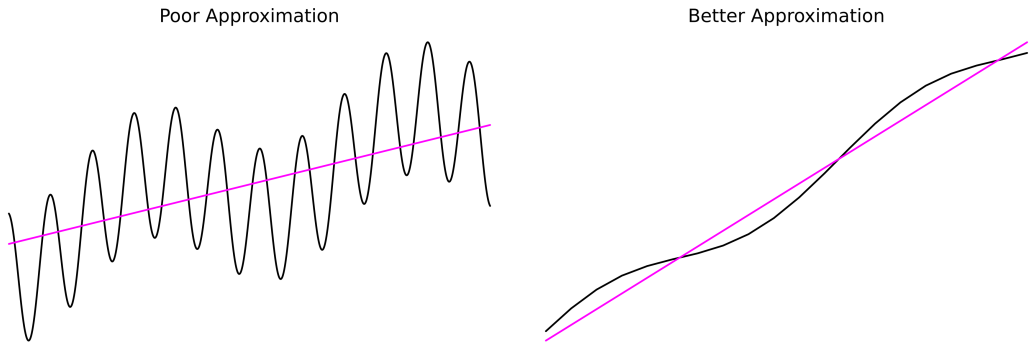


Figure 2.1: Linear Approximations of Nonlinear Curves

To make sense of how we might construct a stronger model in the nonlinear setting, we begin by partitioning all nonlinear classification tasks in $\mathbb{R}^2$ into three categories: curve classifiers, loop classifiers, and no man's land. By partitioning tasks into categories, we enable the exploitation of geometric structure. Meaning, classifying nonlinear data without any knowledge of what the true labeling function may be is simply too generic a task. We need to have some sense of how the space is separated.

2.1 Curve Classifiers

A curve classifier in $\mathbb{R}^2$ is comprised of a nonlinear, injective function and an inequality constraint.

Definition 2.1 *Consider some nonlinear function $f : \mathbb{R} \to \mathbb{R}$ that both separates $\mathbb{R}^2$ into two spaces and is injective. For some instance $\mathbf{x} \in \mathbb{R}^2$, the label of $\mathbf{x}$ is given by one of the curve classifiers $\mathbb{1}_{[x_2 \geq f(x_1)]}$ or $\mathbb{1}_{[x_2 \leq f(x_1)]}$ where $\mathbb{1}_{[condition]} = 1$ if $condition$ is true.*

Curve classifiers are perhaps the easiest of the three categories to extend to higher dimensions. The important property to maintain is that of unique outputs. If we preserve unique output from a given input, the extension of curve classifiers to $\mathbb{R}^3$ and beyond is a manifold that does not pass through or intersect with itself. Ultimately, though, curve classifiers are not of prime interest to us because the central theme of our work is examining what happens when we decrease or increase bias. In the case of curve classifiers, it is not

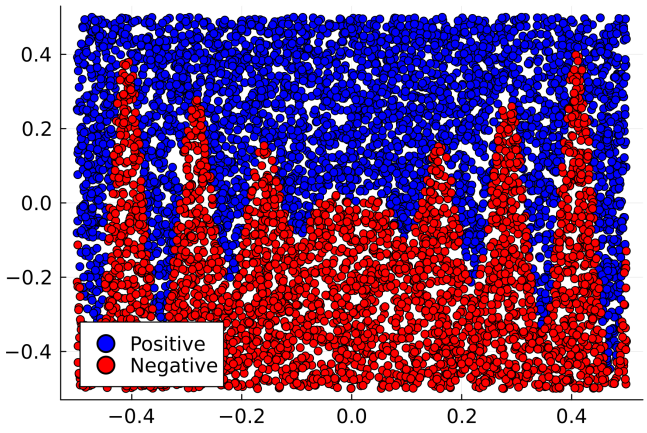


Figure 2.2: Curve Classifier Example

clear what increasing or decreasing bias would even mean. The Weierstrass Approximation Theorem[8] tells us that over an interval, any function can be well approximated by a polynomial—suggesting that maybe we can increase or decrease bias by changing the degree and coefficients of a polynomial, but we leave this to future work.

2.2 Loop Classifiers

Loops in $\mathbb{R}^2$ are obtained by taking some curve on an interval and gluing together its endpoints.

Definition 2.2 *A "single" loop is any continuous function* $f : [a, b] \to \mathbb{R}^2$ *satisfying*

$$f(j) = f(k)$$

if and only if $j = a$ *and* $k = b$*. That is,* f *is injective everywhere except on the endpoints. Note that "single" is in quotes because certain functions can create multiple clusters like the single loop in Fig. 2.3.*

Observe that the above is not a definition for a loop classifier, but just a loop itself, and a single-loop at that (we will talk about loop classifiers and multiple-loops soon). It is worthwhile to discuss what is or is not a single-loop in a bit more detail, because we can get our-

selves into trouble if we are not careful about subtleties. In particular, notice that the above definition does not allow for loops to intersect with themselves anywhere other than the "endpoints". An example of an invalid intersection is given in Fig. 2.3

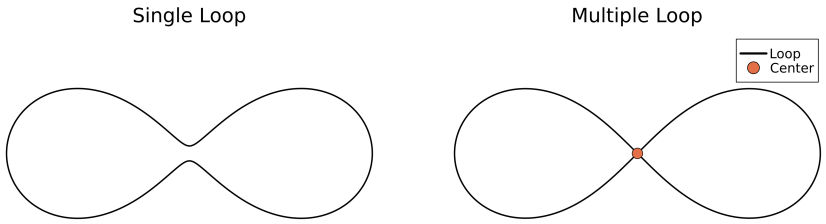


Figure 2.3: Single vs Multiple Loop

To see why the curve on the right of Fig. 2.3 is not a single-loop, suppose that the center is where the endpoints lie. We claim that the endpoints are not the only way to reach the center. Start at the center and walk along the loop as indicated by the arrows. You will reach the center again prior to covering the whole loop, implying an intersection. Formally, if the curve on the right in Fig. 2.3 is given by some mapping $g : [a, b] \to \mathbb{R}^2$, then there exists real numbers $a < j < k < b$ and intervals $[a, j]$, $[j, k]$, $[k, b]$ such that the image of $[a, j]$, $[j, k]$, $[k, b]$ under g all contain the center, which violates the injective property in Definition 2.2.

2.2.1 Polar Coordinates

Prior to defining loop classifiers, we need to discuss polar coordinates and their role in Definition 2.2. Specifically, definition Definition 2.2 does not require that the polar interpretation of the loop is restricted to $0 \leq \theta \leq 2\pi$. Suppose, for instance, that data were labeled according to a single-loop that indefinitely swirls around itself like that in Fig. 2.4. In order to reach certain points on the loop, we will certainly need $\theta > 2\pi$.

2.2.2 Single-Loop Classifiers

A single-loop classifier labels an instance as positive if the instance is inside or on the loop, and negative otherwise. By positive and negative, we mean "yes/no" or "true/false". Since

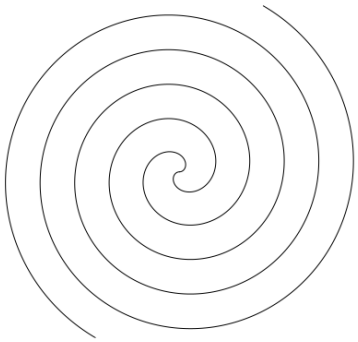


Figure 2.4: Nonending Swirly Loop (Fermat's Spiral[2])

it is conventional to represent 1 as true and 0 as false, a negative instance has a value of 0, despite the fact that 0 is neither negative nor positive. Recalling that the boolean operator $\mathbb{1}_{[condition]}$ returns 1 if a condition is true, and 0 otherwise, we obtain the definition of a loop classifier using polar coordinates.

Definition 2.3 *Consider an instance* $\mathbf{x}$*. We say that* $\mathbf{x}$ *is positively labeled if the radius* r *of* $\mathbf{x}$ *is less than or equal to* r_θ *where* r_θ *is the radius of the loop at the given* θ *value. That is, a loop classifier is given by* $\mathbb{1}_{[r \leq r_\theta]}$.

2.2.3 Multiple-Loop Classifiers

One would think that multiple-loop classifiers follow fairly intuitively from single-loop classifiers. One would be mistaken. Let us first start with a simple example. Imagine we place a torus in $\mathbb{R}^2$. Which areas are positively labeled and which areas are negatively labeled? We can't say "inside the loop" anymore because we need to make a choice as to which loop (and we need to decide what to do about any overlapping regions). In Fig. 2.5, for instance, we could interpret "inside the loop" three different ways.

The three interpretations we could use are 1) that "inside the loop" means being inside the circle with the smaller radius (left plot), 2) being inside the circle with larger radius *and* outside of the circle with the smaller radius (middle), or 3) as being anywhere inside either of the circles (right). While we could simply make a choice and stick with it, that does not generalize well.

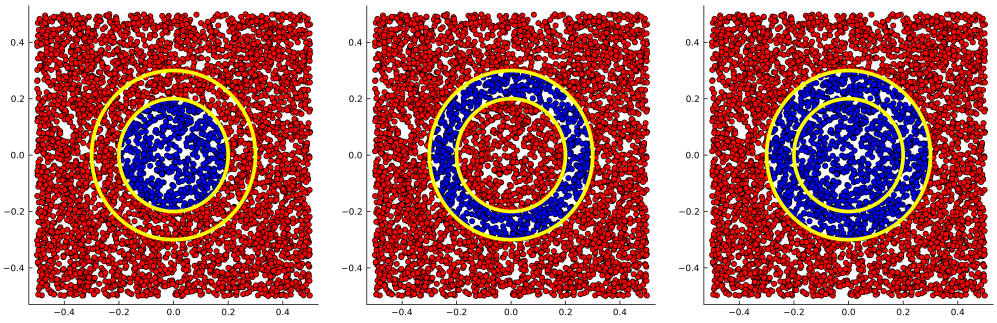


Figure 2.5: Three Interpretations of "Inside the Loop"

If we allow for any continuous function where the endpoints are the same to be a loop worthy of consideration, we allow for arbitrary self-intersection. To see why this is not ideal, we take a brief stroll down knot theory lane. With respect to this commentary, knots exist in $\mathbb{R}^3$. In Fig. 2.6, we can see that there are "discontinuities" in the drawings, denoting the occurrence of the string of the knot passing under or above itself.

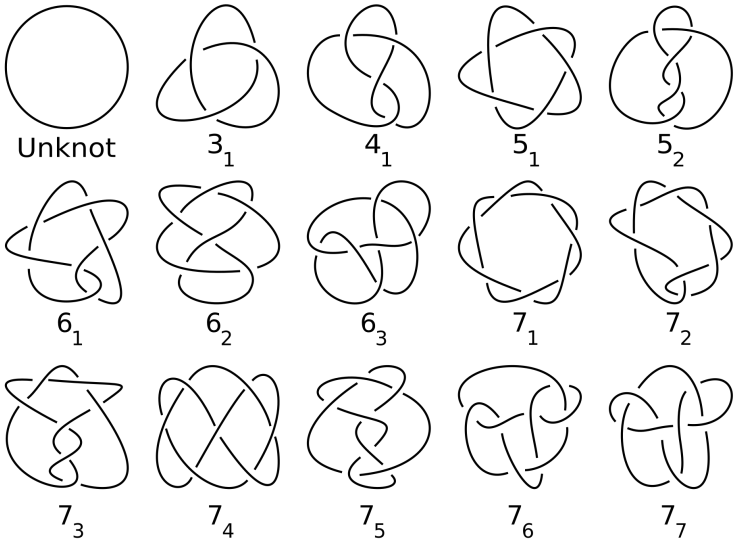


Figure 2.6: Knots (Image Source: Wikipedia[5])

Imagine we project the knots onto $\mathbb{R}^2$ so that the "discontinuities" are now intersection points. Suppose also that we treat the projections like loops. Now try to separate $\mathbb{R}^2$ using

the knots. It is not at all clear what "separate" means. Do we simply ignore the fact that there are multiple distinct bounded regions on a single knot and classify everything "inside the knot" as belonging to one class? Or do we play or a sort of leap frog game, similar to what we did in interpretations one and two of the torus in Fig. 2.5?

Admittedly, if we really wanted to, we could devise a set of assumptions about multiple-loop classifiers in general, but for the most part, we will focus our attention on data with more pleasant structure, and for good reason. When data is labeled by a multiple-loop classifier with several different bounded regions, the problem is essentially a multiclass classification problem. Yes, the data still has binary labels, but the classifier divides the space up into more than two regions, so one could assign a different class to each region. Multiclass classification is considerably more difficult than binary classification, and therefore, generic multiple-loop classifiers are a bit beyond the reach of this project. Does this mean we will ignore multiple-loop classifiers entirely? No.

2.2.4 Pleasant Multiple-Loop Classifiers

Consider the folium[2] curve in Fig. 2.7 and observe that at the intersection point at the center the curve violates the injective property required in Definition 2.2.

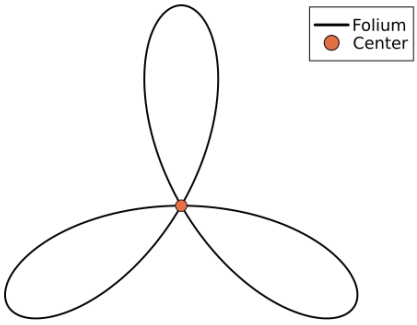


Figure 2.7: Folium Curve

Since injectivity does not hold, the folium curve is a multiple-loop classifier, but it is not an unpleasant one. There is a very clear way to define how a folium curve would classify

data. Namely, we can write a folium in polar coordinates with

$$r(a, b, \theta) = -b \cos\theta + 4a + \cos\theta \cdot \sin^2\theta \tag{2.1}$$

and therefore, folium classifiers obey Definition 2.3, despite the fact that they are not single-loop classifiers.

There exist other multiple-loop curves for which there is a clean interpretation of data would be labeled. Notice that in Fig. 2.8 we have two single-loop curves.

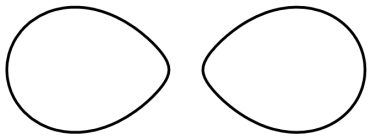


Figure 2.8: Cassinian ^{2} Curve

It is not unreasonable to argue that Definition 2.3 implies that points inside the two ovals in Fig. 2.8 are positively labeled. We can generalize this notion of multiple-loop classifiers being comprised of disjoint single loops.

Definition 2.4 *Consider a multiple-loop curve C that can be written as the union of n single-loop curves where the areas of the single-loops curves are disjoint. Let (r_k, θ_k) be the radius and angle of the polar coordinate representation of an instance $\mathbf{x} \in \mathbb{R}^2$ such that the radius r_k and angle θ_k is calculated with respect to the center of single-loop k. Now denote r_{θ_k} to be the radius of loop k at angle θ_k. The multiple-loop labeling function associated with C is given by*

$$\mathbb{1}_{[(r_1 \leq r_{\theta_1}) \vee (r_2 \leq r_{\theta_2}) \vee \cdots \vee (r_n \leq r_{\theta_n})]}.$$

Observe that if we wanted to, we could rewrite the folium curve as a union of three single-loop curves and then a folium classifier obeys Definition 2.4.

Before we conclude our discussion of multiple-loop classifiers, we should draw attention to an important distinction between single-loop and multiple-loop classifiers. Earlier we mentioned that multiclass classification is difficult and that said difficulty is one of the reasons that we are not fond of multiple-loop classifiers. One might wonder, if single-loop classifiers are allowed to have multiple clusters, does this introduce the multiclass classification difficulty we have briefly mentioned? No. The key difference is that a single loop, be it with multiple clusters or not, is still just one region. Hence, identifying the curve defining the loop (or even just a decent approximation of it) is enough to classify the entire learning space. In the multiple-loop setting, we cannot exploit boundaries to the same degree. That is, finding the boundary of a single-loop in the multiple-loop setting does not necessarily tell us anything about the other single-loops.

2.2.5 Loops in Higher Dimensions

Like curve classifiers, loop classifiers can be extended to higher dimensions. In this case, we might think of a loop classifier in $\mathbb{R}^n$ as some manifold homeomorphic[6] to an n sphere where anything inside or on the manifold is positively labeled and anything outside is negatively labeled. Like curve classifiers, we make a distinction to avoid considerations of objects with complicated qualities like self-intersection or objects that pass through themselves. In the two-dimensional case, self-intersection caused ambiguities, but it is actually worse in higher dimensions. To see how, consider a Klein bottle (Fig. 2.9). Certain interpretations of the phrase "everything inside positive, everything outside negative" lead to contradictory labelings. For instance, imagine the white space surrounding the bottle is negative. If we follow the white space into the whole, down the handle, and back up we will end up inside the bottle, implying that the inside of the bottle is both inside and outside at the same time.

2.3 No Man's Land

For completeness, we will briefly discuss the classifiers that live in what we call "No Man's Land (NML)." A NML classifier is any nonlinear classifier that does not lead to contradic-

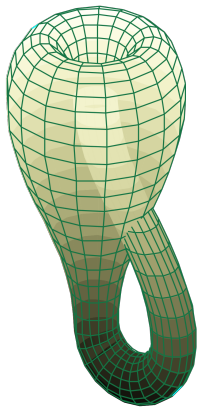


Figure 2.9: Klein Bottle (image source: Wikipedia [12])

tory labeling and does not obey definitions Definition 2.4 and Definition 2.2. That is, if one can construct a self-intersecting object that passes through itself and somehow does not force a point to be both positive and negative at the same time, one has constructed a NML classifier. Similarly, if one wanted a classifier that uses combinations of loops and/or curves, that would also be a NML classifier. NML classifiers are not something we will discuss much at all because they are simply too generic. Since NML do not follow some type of structure, it is likely impossible to even try to approximate learning without domain specific knowledge.

The man who moves a mountain begins by carrying away small stones.

Confucius

3

Loop Classifiers and Apriori Knowledge

Thus far, we have introduced learning theoretic ideas and partitioned nonlinear classification into more manageable categories. We have not, however, made any progress towards answering the big question, "To what degree can we maximize prediction accuracy while minimizing apriori knowledge?" In this chapter, we outline a series of loop classifiers that model progressively weaker apriori knowledge. Doing so will give us a framework to inspect the importance of making appropriate assumptions prior to learning. In Chapter 4, we will conduct experiments using the framework we have built up in this chapter.

3.1 The Spectrum of Apriori Knowledge

Imagine we could quantify the strength of apriori knowledge and put it on a line as shown in Fig. 3.1.

Figure 3.1: Spectrum of Knowledge

Neither end of the spectrum is particularly interesting. If we know everything there is to

know (the true labeling function and/or the distribution of the data), then learning is not necessary. If we know nothing, then No Free Lunch kicks in and learning is infeasible. Our goal is to get as close to "knowing nothing" as possible while still achieving decent (roughly 90% accuracy) generalization. Admittedly, the 90% number is arbitrary. Depending on the scenario, that number may be much higher than is reasonably achievable in practice, but it could also be far too low. If we are trying to detect cancer, for instance, a 10% failure rate is bad.

Rather than trying to jump to the right end of the spectrum right away, we think it will be helpful to start in the realizable setting, where we know as much as there is to know without knowing everything. There are many realizable loop classifiers we can consider, but we have chosen squares as the ideas present in rectangular loops will be helpful throughout the progression. To get a sense of how this progression will unfold, we give a rough sketch on the spectrum in Fig. 3.2.

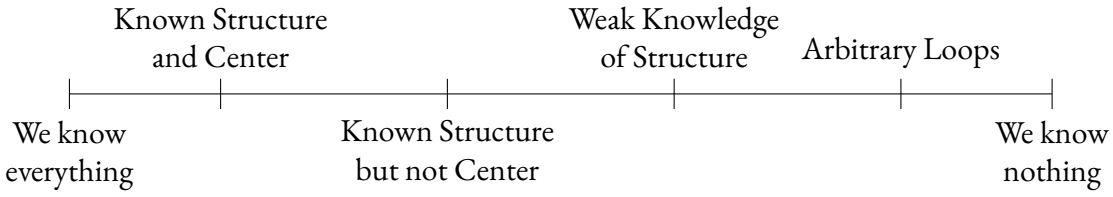


Figure 3.2: Sketch of Progression From Left to Right

Knowing the structure and center of the loop is akin to knowing the loop is a circle or hexagon centered at, say, the origin. Learning is then as simple as searching for a proper radius length (Section 3.2). Knowing the structure but not the center is slightly more challenging as we now need to estimate where the loop lies in space, but for a properly constructed learning scheme, this might (emphasis on might) still be a realiazable setting (Section 3.3). Learning begins to get challenging when we remove some of our knowledge of the structure of the loop. Imagine, for instance, that we don't know the exact structure of the loop, but we somehow know that it is convex (or convexish). If we know the shape is convex or nearly convex, perhaps we can approximate it with some hypothesis class of convex shapes (Section 3.4). Finally, we make things about as challenging as possible by removing almost all knowledge entirely; if all we know is that the true classifier is a loop, we

can no longer exploit convex geometry *and* we have the additional obstacle of estimating where the loop is in space (Section 3.5).

3.2 Squares With Known Centers

In Section 3.2 and we will construct learning settings in which data is nonlinearly separable but still realizably learnable. The importance of considering nonlinear classification in the realizable setting is two-fold. First, the realizable setting is as easy as easy gets. This setting acts as the ceiling for what a learner can achieve and therefore gives us something to compare to as we make learning progressively more challenging. Second, we want to draw attention to the fact that nonlinear classification is not necessarily "more difficult" than linear classification per se.

In the case of linearly separable data, knowledge of the linear structure is enough to achieve a realizable (arbitrarily strong) learner provided one has access to a sufficiently large sample. Nonlinear data is not actually any different. If one happens to know the exact structure of their data, then realizability holds and learning is as easy as in the linearly separable setting. In other words, if your data is labeled by a circle, searching for the appropriate radius size is no harder than searching for the value of the slope and intercept of a line that separates linear data. To see this phenomenon in real time, we introduce the concentric squares setting and the square learning space.

3.2.1 Defining the Learning Space

Unless specified otherwise, the reader should assume from this point on that labels are binary, $\mathcal{Y} = \{0, 1\}$, and that in all learning settings, the domain space is contained within the unit square centered at the origin.

Definition 3.1 *We call the learning space given by the unit square at the origin $\mathcal{X}_{sq}$, and it is formally given by*

$$\mathcal{X}_{sq} = \{(x_1, x_2) : x_1, x_2 \in [-0.5, 0.5]\},$$

which reads, "The domain space 'x square' is the set of all ordered pairs (x_1, x_2) where both x_1 and x_2 belong to the closed interval between -0.5 and 0.5.

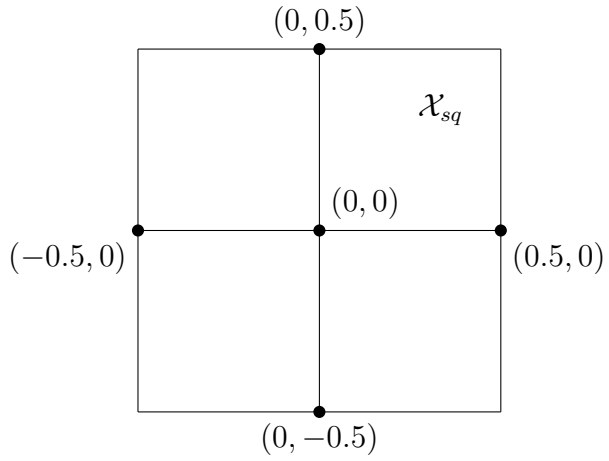


Figure 3.3: $\mathcal{X}_{sq}$ Learning Space

Notice that this restriction allows for a very pleasant interpretation of sampling if data is sampled uniformly over the learning space. Namely, if data is sampled uniformly over $\mathcal{X}_{sq}$, the probability of sampling a point in some area of $\mathcal{X}_{sq}$ is simply the area itself (this will be important later when we discuss experiments). Another nice feature that about $\mathcal{X}_{sq}$ is that it is very natural to search through using concentric squares, which is why we begin our progression in Fig. 3.2 with squares.

3.2.2 Defining Square Classifiers

We denote a loop classifier where the loop is a square with $Q_{b,\mathbf{c}}$ where $\mathbf{c}$ is the center of the square and b is a bound that determines the size of the square[1]. Namely, b is the length of the perpendicular bisector between the center and a side of the square (see Fig. 3.4) and it then follows that the length of the side square is $2b$.

We choose to define the square in this way because it results in a clean definition of the labeling function. If the square is centered at the origin, for instance, the labeling function

[1] The reader may wonder why we use Q instead of S. Recall S is what we use to denote training sequences. The letter Q seemed like the next best choice...

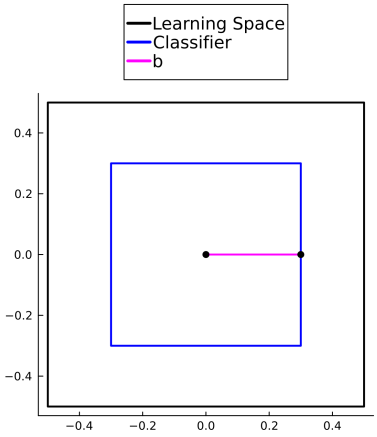


Figure 3.4: Bound b Given By Perpendicular Bisector

$Q_{b,\,(0,0)}$ is simply given by

$$Q_{b,\,(0,0)} = \begin{cases} 1 & \text{if } -b \leq x_1 \leq b \quad \text{AND} \quad -b \leq x_2 \leq b \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Recalling that $\mathbb{1}_{[condition]} = 1$ if and only if the condition is true, we can write Eq. (3.1) more succinctly with

$$Q_{b,\,(0,0)}(\mathbf{x}) = \mathbb{1}_{[|x_1| \leq b]} \cdot \mathbb{1}_{[|x_2| \leq b]} \tag{3.2}$$

Notice that the symmetry of squares is actually restrictive enough to simplify Eq. (3.2) even further

$$Q_{b,\,(0,0)}(\mathbf{x}) = \mathbb{1}_{[\max(|x_1|,|x_2|) \leq b]}. \tag{3.3}$$

Remark 3.1 *The utility of Eq.* (3.3) *and Eq.* (3.2) *is that they generalize to other learning scenarios. Eq.* (3.2) *generalizes to rectangles that are not squares if we have two different bounds and Eq.* (3.3) *extends to higher dimensions. If the learning space exists in* $\mathbb{R}^n$ *and the classifier we are considering is given by some hyper cube in* $\mathbb{R}^n$*, the labels are determined by* $\mathbb{1}_{[\max(|x_1|,\ldots,|x_n|) \leq b]}$.

In the case that the square is not centered at the origin, we simply account for the shift of the center by shifting the instance coordinates accordingly to achieve an equivalent result. Summarizing all the above and generalizing to squares with any center in $\mathcal{X}_{sq}$ yields the following definition.

Definition 3.2 *A square classifier $Q_{b,\mathbf{c}}$ in $\mathcal{X}_{sq}$ is given by $Q_{b,\mathbf{c}}(\mathbf{x}) = \mathbb{1}_{[\max(|x_1-c_1|,|x_2-c_2|)\leq b]}$.*

When we conduct experiments, we will compare two realizable settings: one in which we know the center of the loop in advance and one in which we do not. In the case where we know the center in advance, we will fix the center at the origin. To clean up notation a bit, we will introduce a a modified version of Definition 3.2 in which we drop the center from the subscript.

Definition 3.3 *A square classifer whose center is at the origin shall henceforth be denoted by $Q_b(\mathbf{x})$ with $Q_b(\mathbf{x}) = \mathbb{1}_{[\max(|x_1|,|x_2|\leq b)]}$.*

3.2.3 The Concentric Squares Hypothesis Class

We now formalize the realizability hinted at in Fig. 3.2. In particular, we define an assumption that, when true, results in a clear choice for a realizable hypothesis class.

Assumption 3.1 *If we have reason to believe data is labeled by a square centered at the origin, then we are assuming that the true labeling function is some Q_b with an unknown bound b.*

Though it may not be immediately obvious, this is an extremely strong assumption and if it holds true, there exists distributions for which we can PAC learn with a sample as small as 10(ish) instances. To get a sense of how ridiculous this is, we provide a visual (Fig. 3.5) of a sample of 10 instances sampled uniformly over $\mathcal{X}_{sq}$ and labeled by some Q_b.

Despite the sparseness of the data set, if we pick PAC parameters $\delta = 0.2$ and $\epsilon = 0.1$, the sample in Fig. 3.5 should[2] be sufficient for an ERM learner. That is, if the true classifier is indeed a square at the origin and we happen to correctly assume so in advance, on average,

[2] We will discuss the sample complexity of learning square classifiers in detail in Chapter 5.

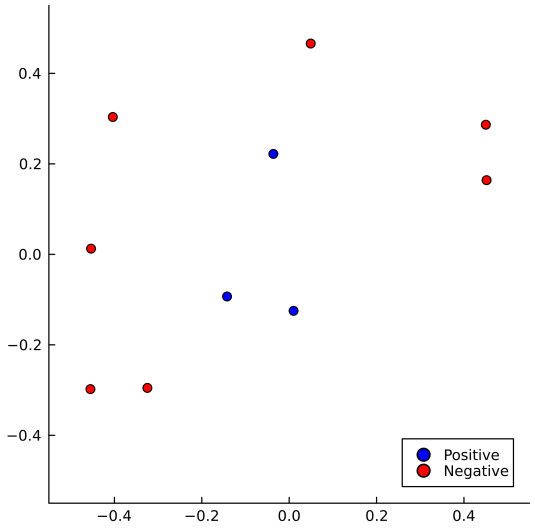


Figure 3.5: Sample of 10 Instances

we can be at least 90% accurate at least 80% of the time with a sample as seemingly meaningless as the one above.

The clear choice of $\mathcal{H}$ when learning under assumption Assumption 3.1 is a set of Q_b functions with varying b values. If we allow for b to be any real value between 0 and 0.5, then our hypothesis class is infinite. We could try to prove the set of all Q_b has a finite VC dimension, but we have already made efforts to prove statements about finite hypothesis classes. For now, we choose to be consistent with our work in Section 1.1 by choosing a finite hypothesis class. We arbitrarily choose to increment b from 0.001 to 0.5. That is, we chose to have $|\mathcal{H}| = 500$. To informally justify this choice, observe in Fig. 3.6 what happens as we increase our hypothesis space to a size of 500.

By inspection, it would seem that 500 hypotheses is likely to be sufficiently expressive for decent generalization. Under the choice to make $\mathcal{H}$ finite, we define the class like so.

Definition 3.4 *The finite class of concentric squares with bounds ranging from* 0.001 *to* 0.5 *in increments of* 0.001 *is given by*

$$\mathcal{H}_{sq} = \left\{ Q_b : b \in \{0.001k\}_{k=1}^{500} \right\}.$$

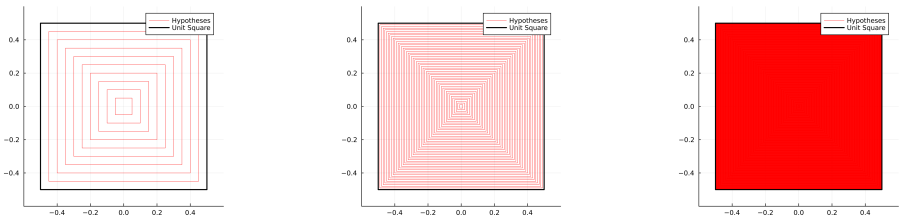


Figure 3.6: Concentric Squares Hypothesis Classis with Size $|\mathcal{H}| = 10$ (left), $|\mathcal{H}| = 50$ (middle), and $|\mathcal{H}| = 500$ (right).

Remark 3.2 *We go out of our way to write $\mathcal{H}_{sq}$ instead of $\mathcal{H}$ because we will be defining several different hypothesis classes in the coming sections. If we use $\mathcal{H}$ for all of them, commentary would get needlessly confusing.*

We have now adequately introduced square classifiers in which the center is not known. The next step in our progression of apriori knowledge is to examine squares classifiers when we do not know the center of the true labeling function.

3.3 Squares with Unknown Centers

In the previous setting we defined a finite hypothesis class of concentric squares centered at the origin, and this exercise has its purpose, but clearly, $\mathcal{H}_{sq}$ is subject to poor generalization if the true labeling function is not centered at the origin (and even poorer generalization if the loop is not a square). In this section, we handle the problem of a lack of knowledge of the true labeling loop's location in space. Poor knowledge of the true labeling loop's structure will be addressed in Section 3.4 and Section 3.5.

3.3.1 Families of Hypothesis Classes

If our apriori knowledge is such that we suspect that the true labeling function is a square, but we do not know where that square is, then Assumption 3.1 no longer holds. We now modify it ever so slightly and adjust our hypothesis class accordingly.

Assumption 3.2 *If we have reason to believe data is labeled by a square, we make the assumption that data is labeled by some $Q_{b,\mathbf{c}}$ with $b, \mathbf{c}$ unknown prior to training.*

As you can see, the assumption is almost exactly the same, but by removing knowledge of $\mathbf{c}$ we have actually removed significance bias from our model. To see just how significant, we construct what we will denote as a "family of hypothesis classes".

Definition 3.5 *A family $\mathcal{F}$ of hypothesis classes is a collection of hypothesis classes $\mathcal{H}_1, \mathcal{H}_2, \ldots$ where each $\mathcal{H}_i$ is unique. That is, for some hypothesis h, if $h \in \mathcal{H}_i$ then $h \notin \mathcal{H}_j$ for $j \neq i$.*

Remark 3.3 *If the meaning of Definition 3.5 is not entirely clear yet, that is fine. We will use Definition 3.5 in several different sections, so it is intentionally vague. Do not spend a bunch of time on understanding the above definition in a vacuum as it will make much more sense in context, as we will soon see.*

When we try to learn under Assumption 3.2, we need some way of throwing out the squares whose centers "disagree" with the training data. If, for instance, the training data have some cluster of positive instances in the upper left, we do not want to consider squares with centers in the lower right.

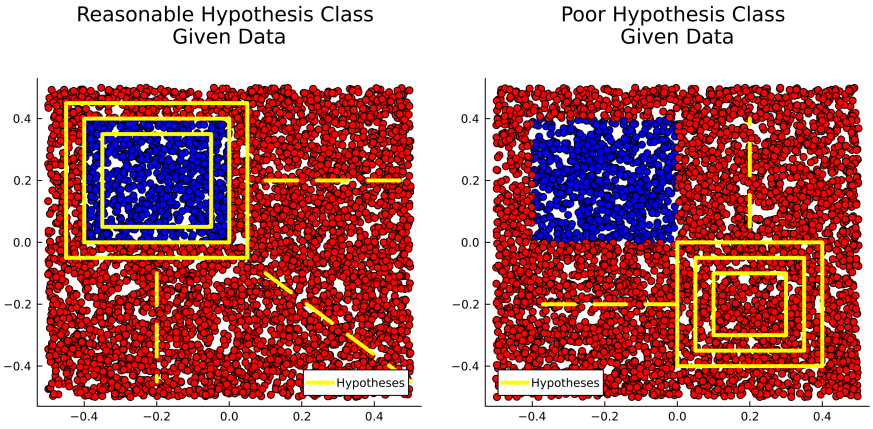


Figure 3.7: Reasonable vs. Poor Hypothesis Class Choices

Hypothesis class families give us a way to describe what it means "throw out" hypotheses with bad centers. Consider, for instance, the following family.

Definition 3.6 *Let $\mathcal{F}_{sq}$ be the family of hypothesis classes where each hypothesis class*

$$\mathcal{H}_{\mathbf{c}} = \{Q_{b,\mathbf{c}} : b > 0\}$$

consists of concentric square classifiers centered at $\mathbf{c}$*. That is, $\mathcal{F}_{sq}$ is a set of sets given by*

$$\mathcal{F}_{sq} = \{\mathcal{H}_{\mathbf{c}} : \mathbf{c} \in \mathcal{X}_{sq}\} = \{\{Q_{b,\mathbf{c}_1} : b > 0\}, \{Q_{b,\mathbf{c}_2} : b > 0\}, \ldots\}.$$

Imagine we have some way of using the training data to estimate the center of the cluster of positive instances. Call the estimate $\hat{\mathbf{c}}$. Instead of considering the infinitely many classes of hypotheses in $\mathcal{F}_{sq}$, we could limit our model only to $\mathcal{H}_{\hat{\mathbf{c}}}$. In other words, we could "throw out" any hypothesis class $\mathcal{H}_{\mathbf{c}}$ where $\mathbf{c} \neq \hat{\mathbf{c}}$.

Before we move on, let us quickly make an observation about a subtle detail of Definition 3.6. Notice that the only restriction we applied to b in $\mathcal{H}_{\mathbf{c}}$ is that $b > 0$ whereas in Definition 3.4 we restricted b to be some member of the finite sequence $\{0.001k\}_{k=1}^{500}$. We are releasing the restricting for two reasons. First, if we bound b from above at all, it does not make sense to make that bound smaller than 1. To see why, consider some square with a center in any of the four corners of $\mathcal{X}_{sq}$. If we limit b to be some finite number less than 1, we are assuming that certain labeling functions will *never* occur, which is bold without justification. If, for example, we limit b to be 0.5 as we did in Definition 3.4, then we are assuming samples like that in Fig. 3.8 are impossible.

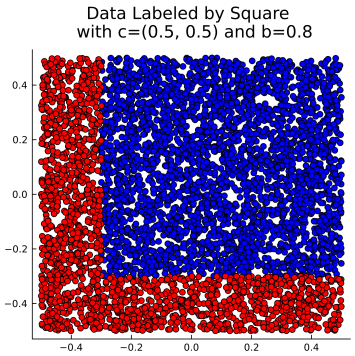


Figure 3.8: Square with $b > 0.5$

3.3.2 To Realize or Not To Realize, That is the Question

It is not clear whether $\mathcal{F}_{sq}$ is realizably learnable. Given that $\mathcal{F}_{sq}$ contains all labeling squares $Q_{b,c}$ in $\mathcal{X}_{sq}$ and Assumption 3.2 presumes we are looking for a labeling square, we might hope that $\mathcal{F}_{sq}$ is realizably learnable if Assumption 3.2 holds. In order for $\mathcal{F}_{sq}$ to be realizably learnable, realizably estimating the center of the true labeling square being one of them.

Regardless of whether $\mathcal{F}_{sq}$ is realizable, one thing is certain: not knowing the center of the true labeling square removes a lot of the bias we induced into the hypothesis class in Section 3.2.3. Consequently, learning square classifiers without knowledge of the square's location in space should have a larger (perhaps significantly larger) sample complexity. All the same, we suspect that squares with unknown centers will have lower sample complexity than more complex shapes. Put differently, knowledge of the loop's structure is more important than knowledge of the loop's location in space. This brings us to our next checkpoint in the knowledge spectrum: learning with poor knowledge of the structure of the loop.

3.4 Convex(ish) Loops

We have now discussed what it means to know as much as possible prior to learning (knowledge of loop structure and location as seen in Section 3.2) and what it means to know "almost as much as possible" (knowledge of loop structure but not location Section 3.3). From here, we make our first step into murkier waters. What happens if we do not know much about the structure of the loop prior to training? In particular, what if all we know is that the loop is convex? That is, the loop could be a rectangle, a regular polygon, an ellipsoid, etc. If the shape is convex, we could try to construct some family of hypothesis classes where each hypothesis class represents a convex polygon (not necessarily regular) with n sides. The problem with this approach is that it is not clear how to avoid overfitting without the use of an astronomical sample size. If we do not limit the learner in some way, it is entirely possible to return a wildly misguided model as depicted in Fig. 3.9.

Perhaps we can try to approximate an unknown convex shape with a shape that is flexible enough to mold to training data, but not flexible enough to overfit? Does such a shape

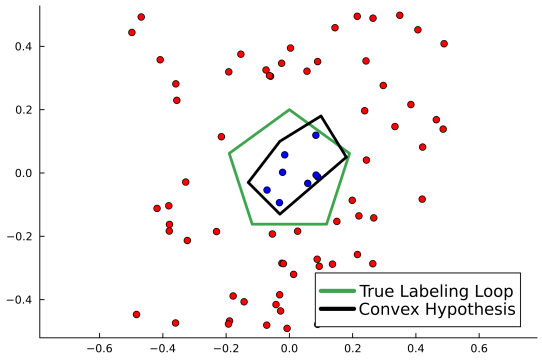


Figure 3.9: Overfit Polygon vs. True Labeling Loop

exist? Sort of. If we use rectangles, we might get a decent[3] approximation for shapes that are convex, or convex(ish). To see what we mean by "convex(ish)" consider shapes that were once convex but are not convex after a perturbation of some kind, as opposed to something that is "not convex at all" like the shapes shown in Fig. 3.10

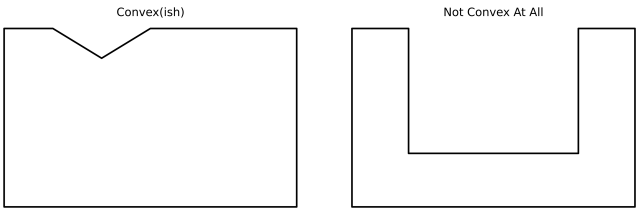


Figure 3.10: Convex(ish) vs. Not Convex At All

The reader might be a bit skeptical about rectangles as an approximation of loops, and rightfully so. Rectangles approximate loop classifiers about as well they cover the loop. We will rigorously define what this means in Section 3.4.1, but first, let's define rectangular classifiers. Recall that in Eq. (3.2) we said that a square classifier centered at the origin can be written as $Q_{b,(0,0)}(\mathbf{x}) = \mathbb{1}_{[|x_1|\leq b]} \cdot \mathbb{1}_{[x_2 \leq b]}$. Rectangular classifiers are quite similar only now we have two bounds instead of one.

Definition 3.7 *A rectangular classifier $R_{\mathbf{b},\mathbf{c}}$ is given by*

$$R_{\mathbf{b},\mathbf{c}(\mathbf{x})} = \mathbb{1}_{[|x_1 - c_1| \leq b_1]} \cdot \mathbb{1}_{[|x_2 - c_2| \leq b_2]}$$

[3] We are using the word "decent" *very* loosely here.

where $\mathbf{c} \in \mathcal{X}_{sq}$ *and* $b_1, b_2 > 0$.

As you might expect, we can extend the family of square classifiers (Definition 3.6) to rectangular classifiers.

Definition 3.8 *Let* $\mathcal{H}_{Rect,\mathbf{c}} = \{R_{\mathbf{b},\mathbf{c}} : b_1, b_2 > 0\}$ *be a set of rectangles that all have center* $\mathbf{c}$. *The family of rectangle hypothesis classes is given by*

$$\mathcal{F}_{Rect} = \{\mathcal{H}_{Rect,\mathbf{c}} : \mathbf{c} \in \mathcal{X}_{sq}\}.$$

3.4.1 How Good Are Rectangles Really?

There is (at least) one problem with using rectangles to approximate unknown shapes: we have now left the land of realizability. There is now an unwavering upper bound to how strong our model can be, and depending on the needs of the user, the rectangle approach may fail to be strong enough even if the learner finds the best hypothesis it can choose from. While it may be obvious that rectangles as an approximation of unknown loops can only be so strong, rigorously explaining why in a meaningful way is not as easy as one would expect. We will first talk about the geometry of approximating loops with rectangles, and then we will highlight why probabilistic reasoning renders geometry somewhat irrelevant.

Consider a loop L (just a curve, not a classifier) with area $0 < \alpha \leq 1$ and a rectangle R (just a shape, not a classifier) with nonzero area. Suppose that R overlaps with L. Call the area where the rectangle area and the loop area overlap Ω. One might think that the best rectangular approximation of a loop classifier is the rectangular classifier whose rectangle maximizes Ω. That is, one might think the rectangle on the right of Fig. 3.11 is desirable to the rectangle on the left, but this is not necessarily correct.

To see why a smaller Ω region might be ideal, suppose that the marginal distribution is a bivariate normal density[7] with a small variance and a mean centered in the smaller Ω region in Fig. 3.11. That is to say, suppose the marginal distribution is such that some huge percentage, say 99%, of the data is inside the smaller Ω region, as depicted in Fig. 3.12.

Given the bivariate we are considering in Fig. 3.12, compare the two Ω regions in Fig. 3.11 and observe that the smaller Ω region is actually ideal because it covers the critical cluster of the bivariate normal entirely and will therefore label points with 99% accuracy. Contrast

Figure 3.11: Two Examples of Overlapping Region Ω

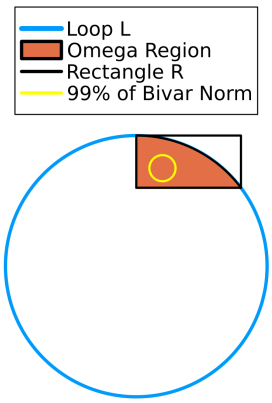


Figure 3.12: Bivariate Example

this with the larger Ω region which does not cover the critical cluster and will therefore mislabel points 99% of the time, despite being a better approximation of the loop L.

3.4.2 Revisiting the Knowledge Specturm

Recall in Fig. 3.2 that we wanted to try to construct various learning settings in which the assumptions necessary to learn in a given setting are progressively weaker as we move along the spectrum. In the realizable setting, we knew where the true labeling loop is in space *and* the type of structure the loop has. We then relaxed the assumption of apriori knowledge of the loop's location. Both of these scenarios have meaningful assumptions. In the case where we have weak knowledge of structure and no knowledge of location, the assumption that represents our apriori beliefs is not exactly meaningful, but we will write it down for

completeness.

Assumption 3.3 *If we have reason to believe a rectangular classifier might be a viable approximation with respect to our learning goals, we are making a multifaceted assumption. First, we are assuming there exists a hypothesis h_Ω in one of the classes in $\mathcal{F}_{Rect}$ such that the area outside the Ω region has low probability* ***and*** *that we have a reliable method to locate the hypothesis class containing h_Ω.*

We will do some experiments with rectangular classifiers, but really, the use of thinking about rectangles is not clear until we allow ourselves to use more rectangles than one, as we will see in the next section.

3.5 Arbitrary Loops

In the previous section, we discussed the possibility of approximating loop classifiers with rectangular classifiers, but what if the loop is not even remotely rectangular? What if there are multiple loops? To put it in terms of our spectrum of knowledge, what if all we know in advance is that the data is labeled by a loop of some kind, be it either a single loop classifier (Definition 2.2) or a multiple loop classifier (Definition 2.4)? We need something more expressive than rectangles to handle the lack of apriori knowledge. Perhaps we can re-purpose our previous work on rectangles. Maybe if a rectangle fails to be a decent global approximation, it could be a decent local approximation.

Call the region containing all positive instances in our training sequence ρ. Imagine we use rectangles not to approximate the true labeling loop itself, but instead to find ρ (left of Fig. 3.13). As we can see, the rectangle we used to find ρ is a poor global approximation of the true labeling loop in that it might do well outside of the borders, but . However, if we divide up the rectangle in Fig. 3.13 into a grid of smaller rectangles, we can get what appears to be several strong *local* approximations. Call each smaller rectangle a cell and then count the number of instances in the cell. If there are more positive instances than negative instances, that cell is assigned a positive label, and vice versa.

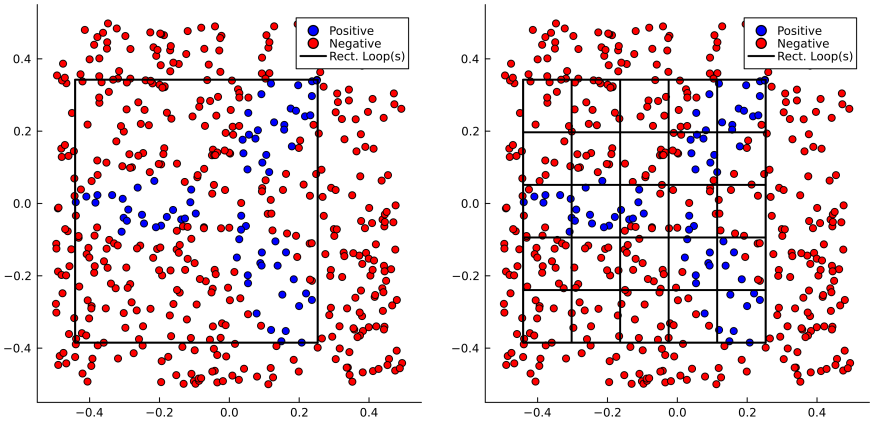


Figure 3.13: Locating Borders (Left) and Dividing Up Borders (Right)

3.5.1 Defining Grid Classifiers

The process we will now describe is how we might label data with what we are calling "grid classifiers". Defining grid classifiers in a rigorous way will require several different definitions, but in essence, the main pieces of information are the values determining the border of the grid and a systematic way of dividing up the region within the border into smaller cells. We start with the border.

Definition 3.9 *Consider a set B of four values $left, right, high, low$ such that the four values all exist in the interval $[-0.5, 0.5]$ and let S be a training sequence with at least one positive instance. Denote the set of positively labeled instances in S as S^+. We say that an instance* **x***, be it labeled positive or negative, is inside the border B if* **x** *satisfies*

$$(left \leq x_1 \leq right) \quad AND \quad (low \leq x_2 \leq high)$$

where $left$ is the x_1 coordinate of the leftmost positively labeled instance, $right$ is the x_1 coordinate of the rightmost positively labeled instance, $high$ is the x_2 coordinate of the topmost positively labeled instance, and low is the x_2 coordinate of the bottommost positively labeled instance. Formally, if x_{ij} is the jth coordinate of the ith positively labeled instance, then

$$left = \min_{1 \leq i \leq |S^+|}(x_{i1}) \quad right = \max_{1 \leq i \leq |S^+|}(x_{i1}) \quad high = \max_{1 \leq i \leq |S^+|}(x_{i2}) \quad low = \min_{1 \leq i \leq |S^+|}(x_{i2}).$$

Notice that we have not mentioned the center of the border. In the previous sections, defining a center has been helpful in our representation of the geometric structures we have considered. We could define the center of the border, but we would be using the values in B to do so, and it seems rather absurd to make the above definition even longer than it already is by introducing a center and the corresponding vertical and horizontal bounds. Cells are similar in this respect.

Definition 3.10 *Consider a border defined by some set of values B. We divide the region contained within the border into n^2 cells of identical area where n is number of divisions along each axis. That is, if we let $H = \frac{right-left}{n}$ represent the horizontal component of a cell and $V = \frac{high-low}{n}$ be the vertical component of a cell, then the boundary defined by B has n^2 cells of area $H \cdot V$. We call the region within the border B divided into n^2 cells a grid. Lastly, note that we index cells the way you would index a matrix (see Fig. 3.14).*

An instance is labeled positively by a grid classifier if it is both inside the grid's border and inside a positively assigned cell (where cell assignment is completed during training). Determining what cell an instance belongs to deserves its own discussion, and we will have that discussion soon. As for cell labeling, that will be discussed later in Chapter 4. For now, assume that we have access to a matrix $\mathbf{G}$ such that each entry $\mathbf{G}_{i,j}$ is either 0 or 1, determining the label of $Cell_{i,j}$ and that we know how to determine whether an instance x is inside a given cell. Under those assumptions, we are ready to define grid classifiers.

Definition 3.11 *Given a border B and a corresponding $n \times n$ grid matrix $\mathbf{G}$, the label of an instance $\mathbf{x}$ is given by the grid classifier*

$$g_{B,n,\mathbf{G}}(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{G}_{i,j} \cdot \mathbb{1}_{[\mathbf{x} \in Cell_{i,j}]}. \tag{3.4}$$

Denote the border values of $Cell_{i,j}$ with $left_i, right_i, low_j, high_j$ and then Eq. (3.4) is formally given by

$$g_{B,n,\mathbf{G}}(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{G}_{i,j} \cdot \mathbb{1}_{[left_i \leq x_1 \leq right_i]} \cdot \mathbb{1}_{[low_j \leq x_2 \leq high_j]}. \tag{3.5}$$

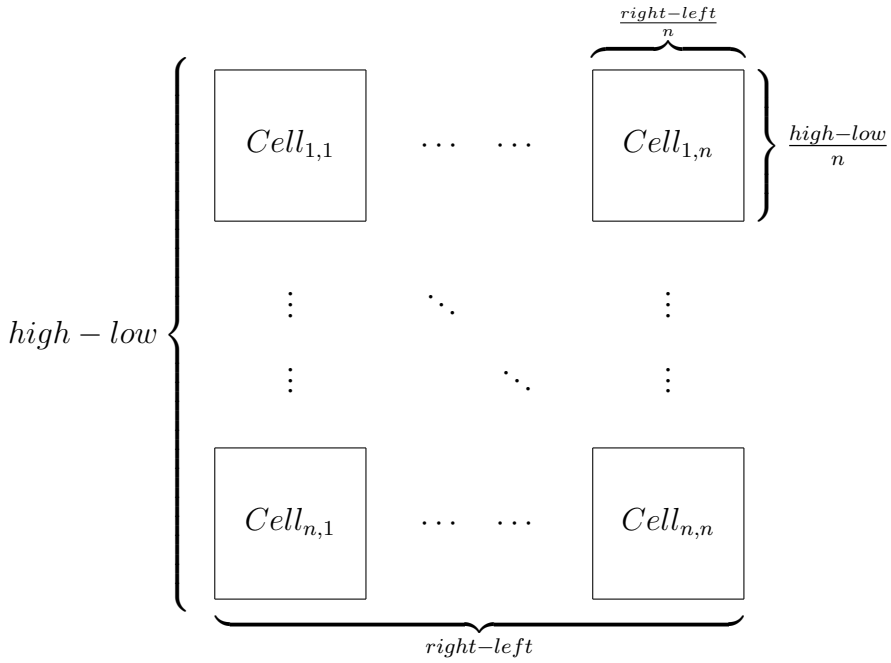


Figure 3.14: Grid of n^2 cells where each cell has a horizontal length $\frac{right-left}{n}$ and a vertical length $\frac{high-low}{n}$

Definition 3.11 is rather obtuse, so let us describe its meaning in a bit more detail. First, notice that we need to know $B, n,$ and $\mathbf{G}$ in order to classify an instance so even though $g_{B,n,\mathbf{G}}$ is ugly, removing the subscripts creates even uglier ambiguities. Next, observe that the grid is defined by a sequence of vertical and horizontal values so the boolean functions in Eq. (3.5) are just a modification of Definition 3.9 where we are now considering the borders of a single cell instead of the whole region.

We now address the issue of how to identify whether an instance belongs to a given sell. To see how we determine the indices i, j an instance belongs to, let us first consider a one dimensional example. Suppose we divide up a line segment over the interval $[0, b]$ into n evenly spaced chunks of length ℓ as shown in Fig. 3.15. Now consider a point p that exits somewhere in the interval. If we index from 1, then p belongs to the $\lfloor p/\ell \rfloor + 1$ chunk (provided $p \neq n\ell$). For the case where the interval is some $[a, b]$ we account for the shift with $(p - a)/\ell$.

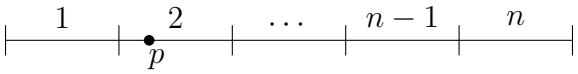


Figure 3.15: Example where $p/\ell > 1$ so p belongs to chunk 2

3.5.2 The Family of Grid Hypothesis Classes

We have previous discussed families of hypothesis classes. In this case, we sort of have a family of families. We first need to pick a border, and then we need to pick a grid size. Once our border and grid size has been chosen, our hypothesis class consists of all 2^{n^2} different grid classifiers.

Definition 3.12 *For a given border B and a grid size, we define the grid hypothesis class $\mathcal{H}_{B,n}$ to be the set of $n \times n$ grid classifiers formed using border B:*

$$\mathcal{H}_{B,n} = \left\{ g_{B,n,\mathbf{G}} : \mathbf{G} \textit{ is one of the } 2^{n^2} \textit{ possible grid assignments} \right\}.$$

We can then define a family of hypothesis classes by considering all B and n choices

Definition 3.13 *We define $\mathcal{F}_{grid}$ as a set of sets representing the family of grid learners where $\mathcal{F}_{grid}$ is given by*

$$\mathcal{F}_{grid} = \{\mathcal{H}_{B,n} : B \in \mathcal{X}_{sq}, n > 0\}.$$

3.5.3 The End of the Apriori Knowledge Spectrum

At this point, we are almost done formulating the ideas discussed on the spectrum in Fig. 3.2. That is, we have discussed what sort of hypotheses and assumptions are associated with learning loops when we know the structure and location of a loop classifier, when we know only the structure, and when we have incomplete knowledge of the structure of the loop. We now finish our commentary on learning loops in which both structure and location are about as unknown as unknown can be.

Assumption 3.4 *If our apriori knowledge is such that our only beliefs about our data is that it is classified by an arbitrary loop, we are assuming that there exists a hypothesis in one of the*

classes in $\mathcal{F}_{grid}$ that is strong enough for our learning needs. Specifically, we are assuming that we have enough data to 1) get a reliable estimate of the border B defining the necessary grid classifier and 2) that we have enough data to get a sufficient estimate of a grid matrix $\mathbf{G}$ that achieves relatively low error.

It does not matter beautiful your theory is. It does not matter how smart your are. If it does not agree with experiment, it is wrong.

Richard Feynman

4 Experiments

We have now built up hypotheses and their associated classes (or families) in the previous chapter to describe how we might try to learn to classify loops with varying apriori knowledge. In this chapter, we will outline what sort of experiments we will run to test our hypothesis classes and their associated assumptions.

4.1 Overview of Experiment Procedures

We want to get a sense of how our models perform when trying to learn the loops defined in Chapter 3. There will be some slight variation in our experiment procedures, but generally, we are interested in the sample complexity of our models and the performance as measured by the risk of the hypotheses they return.

4.1.1 Hypothesis Classes

Before we jump into the details of experimentation, let us briefly summarize Chapter 3. We will make use of the hypothesis classes we defined in great detail in the Chapter 3 so for the convenience of the reader, we summarize those classes here.

Loop	Hypothesis
Square at $(0,0)$	$Q_b(\mathbf{x}) = \mathbb{1}_{[\max(\lvert x_1\rvert,\lvert x_2\rvert)\leq b]}$
Square	$Q_{b,\mathbf{c}}(\mathbf{x}) = \mathbb{1}_{[\max(\lvert x_1-c_1\rvert,\lvert x_2-c_2\rvert)\leq b]}$
Convex(ish)	$R_{\mathbf{b},\mathbf{c}}(\mathbf{x}) = \mathbb{1}_{[\lvert x_1-c_1\rvert\leq b_1]} \cdot \mathbb{1}_{[\lvert x_2-c_2\rvert\leq b_2]}$
Arbitrary Loop	$g_{B,n,\mathbf{G}}(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{G}_{i,j} \cdot \mathbb{1}_{[left_i\leq x_1\leq right_i]} \cdot \mathbb{1}_{[low_j\leq x_2\leq high_j]}$

Table 4.1: Summary of Hypotheses

Loop	Family	Class
Square at $(0,0)$	N/A	$\mathcal{H}_{sq} = \{Q_b : b \in \{0.001k\}_{k=1}^{500}\}$
Square	$\mathcal{F}_{sq} = \{\mathcal{H}_{\mathbf{c}} : \mathbf{c} \in \mathcal{X}_{sq}\}$	$\mathcal{H}_{\mathbf{c}} = \{Q_{b,\mathbf{c}} : b > 0\}$
Convex(ish)	$\mathcal{F}_{Rect} = \{\mathcal{H}_{Rect,\mathbf{c}} : \mathbf{c} \in \mathcal{X}_{sq}\}$	$\mathcal{H}_{Rect,\mathbf{c}} = \{R_{\mathbf{b},\mathbf{c}} : b_1, b_2 > 0\}$
Arbitrary Loop	$\mathcal{F}_{Grid} = \{\mathcal{H}_{B,n} : B \in \mathcal{X}_{sq}, n > 0\}$	$\mathcal{H}_{B,n} = \{g_{B,n,\mathbf{G}} : \mathbf{G} \in \mathcal{G}\}$ $\mathcal{G}$ is set of all 2^{n^2} grids

Table 4.2: Summary of Hypothesis Classes/Families

4.1.2 Experiments in the Realizable Setting

In Section 3.2 we defined the concentric square hypothesis class, a finite class that is realizable under Assumption 3.1. Conducting experiments in the realizable setting gives us a sense of what the optimal performance of a learner both from a sample complexity and an accuracy perspective. In that sense, our experiments in the realizable setting exist not to "see what happens" but to give us a benchmark to compare to. In particular, we would like to compare how the sample complexity and risk performance weakens as we remove bias from our hypothesis classes. To that end, we begin our experiments in the realizable setting by first estimating the sample complexity of training ERM learners on the hypothesis class $\mathcal{H}_{sq}$. We will estimate the sample complexity using a binary search heuristic comprised of two procedures: empirical PAC verification and the search procedure itself.

Recall from Definition 1.11 that we say a hypothesis class $\mathcal{H}$ is PAC learnable via ERM if there exists a sample size $n_{\mathcal{H}}(\epsilon, \delta)$ such that the true risk of the ERM hypothesis $L_{\mathcal{D}}(h_s)$ is at most ϵ with at least $1-\delta$ confidence. In order to achieve $1-\delta$ confidence, then for every N trials we run, at most δN of them should violate our risk tolerance ϵ. Of course, δN is not necessarily an integer, hence we round up to $\lceil \delta N \rceil$.

Algorithm 1 Empirical PAC Verification

1: Fix some true labeling function f.
2: Generate a sample S of n instances.
3: Run the ERM algorithm on S.
4: Record whether $L_{\mathcal{D}}(h_S)$ exceeds our choice of ϵ.
5: Repeat over N total samples.
6: Check if at most $\lceil \delta N \rceil$ samples violated our error tolerance ϵ.

Remark 4.1 *Algorithm 1 Will be useful in multiple experiments, hence the generic nature of the Pseduo code. With respect to our experiments pertaining to $\mathcal{H}_{sq}$, the true labeling function will be some square classifier Q_b at the origin and our ERM algorithms will be those listed in Section 4.3.1.*

Getting back to estimating sample complexity, we can use Algorithm 1 by running it on varying sample sizes as illustrated in Algorithm 2.

Algorithm 2 Estimating Sample Complexity

1: Let f be the same labeling function as used in the empirical PAC verification.
2: Let $n' = n - 1$ where $n = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$.
3: Over many trials, do the following:
4: **while** empirical PAC learnability holds **do**
5: Repeat the empirical verification process using a sample of size n'.
6: Use binary search to decrease n'.
7: **end while**
8: Record n' and start another trial.
9: **return** the average value of n' over all trials.

Let us take a second to understand Algorithm 2. First, we set $n = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$ because Proposition 1.1 (realizability of finite hypothesis classes) tells us that a finite, realizable hypothesis class $\mathcal{H}$ has sample complexity of at most $n = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$. Second, let us discuss the binary search component. We are looking for the first instance where empirical PAC verification fails and we know that it succeeds if it is given $n = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$ instances. Hence, we can set $n' = n - 1$ as our initial upper bound, 1 as our initial lower bound and then binary search can quickly find a sample size where PAC learning happened to fail. That is,

if PAC learning succeeds, we cut the sample size in half and try again whereas if it fails, we add 50% of our sample size and try again. Repeating this process until $low \geq high$ acts as a *heuristic* to estimate the true sample complexity.

We can then repeat the entire heuristic many times over and use the average as our best estimate. In particular, if we interpret the value where empirical PAC learning fails as random variable (which it is, albeit a convoluted one), then the law of large numbers[7] tells us that as we repeat the binary search process over many trials, we should get closer and closer to the true sample complexity.

To conclude our overview of experiments in the realizable setting, recall that we mentioned that in addition to sample complexity estimates we would like to get a sense of the risk values ERM learners return. To see how well the ERM learners learn, we can simply keep a growing list of the risk values when running Algorithm 1. Then, when Algorithm 1 terminates, we can produce histograms of said risk values, giving us a sense of the distribution of the random variable[1] $L_{\mathcal{D}}(h_S)$.

4.1.3 Experiments in the Agnostic Setting

Experiments in the agnostic setting are slightly different than those in the realizable setting in that it does not make sense to try to estimate the sample complexity. Why? In the agnostic setting, there is a limit to how accurate our model can be regardless of the size of the sample. Thus, it would be more appropriate to estimate the complexity of uniform convergence (which is dependent on sample size), but that would require checking each hypothesis in the chosen class, and our agnostic classes are infinite. Rather than estimate the sample complexity using Algorithm 2, we will record average risk values over many different sample sizes. While this does not act as a direct substitute for Algorithm 2, it will give us information about the capabilities of our models.

[1] The true risk of a hypothesis is defined as an expected value. Note, though, that the ERM hypothesis will differ from sample to sample. In that sense, true risk is both an expected value and a random variable at the same time.

4.2 Data Generation

In order to generate data, we need a marginal distribution $\mathcal{D}_{\mathcal{X}_{sq}}$ that dictates how points are distributed throughout $\mathcal{X}_{sq}$, regardless of the true labeling function. We make the choice to use the uniform distribution for $\mathcal{D}_{\mathcal{X}_{sq}}$ and we make this choice for two reasons. First, it lends for a very pleasant geometric interpretation of probability as the probability of sampling a point in a region of space is just the area of the space itself.

Second, if our apriori knowledge is such that we have no knowledge of the true underlying marginal distribution, we are forced to assume a uniform distribution. Assuming otherwise would be inducing bias into our model that could be misplaced. In practice, if one does have some knowledge of the true underlying marginal distribution, they could use that to modify the algorithms we introduce in the coming sections for better results. Finally, to generate data we need a labeling function. Labeling functions will change based on the learning setting we are examining, and we highlight this in greater detail in the coming sections.

4.2.1 Square Loops

We represent the scenarios in which we have apriori knowledge of the true labeling loop's structure with square classifiers. Recall that in Definition 3.4 we defined $\mathcal{H}_{sq}$ to be finite. The purpose of this is to make use of the result that the sample complexity of finite hypothesis classes in which the realizability assumption holds has a known upper bound when using ERM. Consequently, if our data generation is such that $\mathcal{H}_{sq}$ is realizable, then we can use ERM and search for a lower bound of the sample complexity of $\mathcal{H}_{sq}$. Therefore, when generating data for $\mathcal{H}_{sq}$, we randomly distribute points in $\mathcal{X}_{sq}$ and then label them according to some square classifier centered at the origin with a bound $b \in \{0.001k\}_{k=1}^{500}$, forcing realizability to hold.

For square classifiers in which the location of the center is unknown prior to training, data generation is quite similar. In this case, though, we are generating labels for the data distributed uniformly in $\mathcal{X}_{sq}$ with a labeling function in $\mathcal{F}_{sq}$.

4.2.2 Convex(ish) and Arbitrary Loops

Data generation for loops with structure that is more complex than squares is not all that different than what we have previously described in that we are still distributing points through $\mathcal{X}_{sq}$ first and applying a label to each point second. The difference now is that we need to choose what sort of functions to use to label points. For the sake of time, we will use a folium curve for both the convex(ish) and arbitrary loop settings. The folium curve given by Eq. (2.1) can produce a wide variety of different shapes given different choices of a and b, allowing us to test different kinds of loop structures with minimal implementation cost. In particular, as we make different choices of a and b we can get closer or further away from a convex shape:

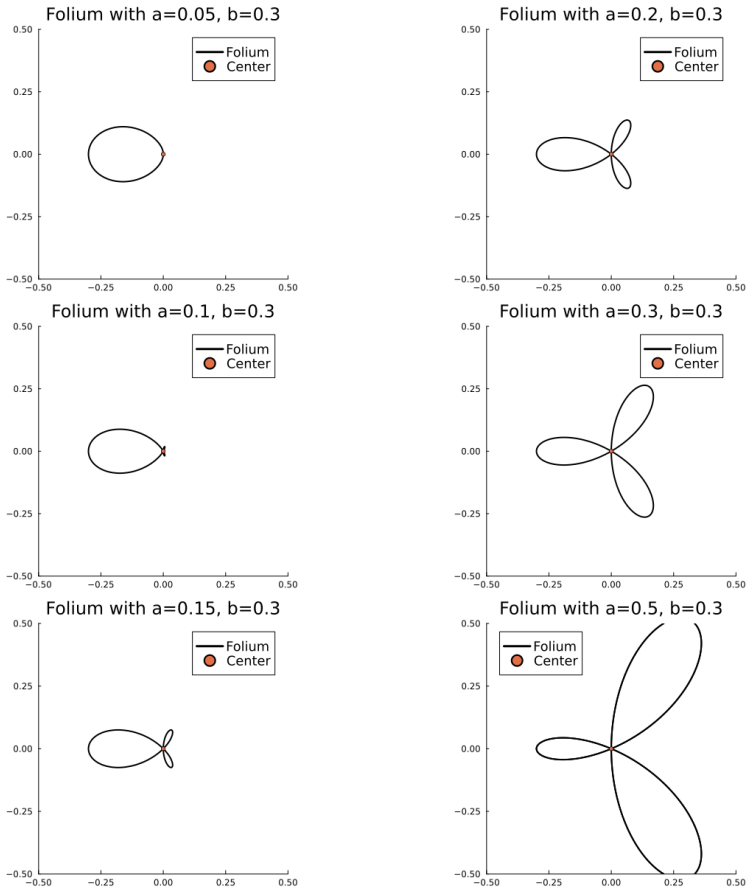


Figure 4.1: Folium Loops From Various Parameter Choices

Observe that the choice to use a folium curve to label data for our experiments in the convex(ish) *and* arbitrary loop settings has a subtle but important caveat. Assumptions 3.3 (convex loops) and 3.4 (arbitrary loops) are associated with different levels of apriori knowledge and therefore should be used for different types of data. The point of using a folium curve for both settings is that a folium curve is rich enough that we can progressively morph it to be more or less appropriate for either assumption, as shown in Fig. 4.1.

4.3 Training Algorithms and Risk Evaluation

ERM is typically what one would use to train a machine learning model. In the case of learning loops, we will be constructing our own algorithms that attempt to minimize risk as much as possible, but do not necessarily achieve zero empirical risk—and there is reason for this. Given that the whole point of this work is to examine what happens as we construct models with less and less inductive bias, achieving zero empirical risk could lead to overfitting in some of our learning schemes. In the following sections, we will discuss how we plan to train for each of the learning settings we have discussed throughout this work.

4.3.1 Training Algorithms for Concentric Squares at the Origin

As discussed in Section 4.1.2, we can empirically verify PAC learnability by randomly generating data, training on the data over many trails, and confirming that a pre-determined number of trials reaches a specified accuracy goal. What we did not discuss in Section 4.1.2 is how to train.

In the realizable setting with a finite hypothesis class, ERM could be as simple as looping through the hypothesis class, calculating the number of instances a given hypothesis mislabels, and iterating until we find a hypothesis with zero mislabeled sample points. Notice, though, there might be multiple hypotheses that achieve zero empirical risk. If that is the case, how do we know which one is best? To answer this question, we will introduce three different ERM algorithms. The first algorithm we will consider is what we call the "Positive" algorithm because it biases towards positive labeling.

Observe that Positive ERM returns the square classifier with the largest bound b for which empirical risk is minimized. Since the internal area within the square is all labeled

Algorithm 3 "Positive" ERM

```
Initialize the bound b to be the largest bound our hypothesis class allows.
while empirical risk is not minimized do
    if Q_b correctly classifies entire sample S then
        return Q_b
    else
        Decrement b to next largest bound.
    end if
end while
```

positively, by making the returned hypothesis square as large as possible, we are biasing towards positive labeling. Biasing towards negative labeling is similar; when we bias towards negative labeling we return the smallest square classifier with zero empirical risk.

Algorithm 4 "Negative" ERM

```
Initialize the bound b to be the smallest bound our hypothesis class allows.
while empirical risk is not minimized do
    if Q_b correctly classifies entire sample S then
        return Q_b
    else
        Increment b to next smallest bound.
    end if
end while
```

Let us now think about why upper and lower bounds for the size of b are useful in the training process. Consider a sample like the one in Fig. 4.2. Let the point $\mathbf{x}^+$ be the instance in the set of positively labeled instances that is furthest from the true labeling loop's center. [2] Let $\mathbf{x}^-$ be the point in the set of negatively labeled points that is closest to the center of the true labeling loop.

Clearly, any hypothesis existing in between $\mathbf{x}^+$ and $\mathbf{x}^-$ achieves zero empirical risk for the given sample, but what if we increase the sample size? Unless one of the instances $\mathbf{x}^+, \mathbf{x}^-$ is right on top of the true classifier, returning a hypothesis near $\mathbf{x}^+$ or $\mathbf{x}^-$ could lead to over-

[2] At the moment, the center of the true labeling loop will always be the origin because we are still discussing the learning setting in which we fix functions at the origin.

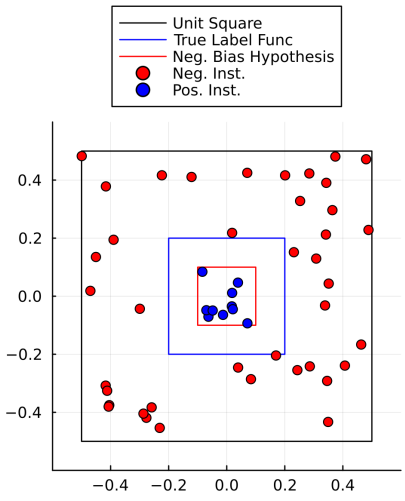


Figure 4.2: Gap Example

fitting. To try to avoid this, we can instead return the hypothesis exactly in between $\mathbf{x}^+$ and $\mathbf{x}^-$ using the "Midpoint" ERM scheme (which we will frequently refer to as "Mid").

Algorithm 5 "Midpoint" ERM

1: Let b_{high} be the bound that defines the hypothesis returned from the "Positive Bias" scheme.
2: Let b_{low} be the bound that defines the hypothesis returned from the "Negative Bias" scheme.
3: Define b_{mid} to be bound that is closest (or equivalent) to the midpoint between b_{low} and b_{high}.
4: **return** the hypothesis square defined by b_{mid}.

4.3.2 Risk Evaluation for Concentric Squares at the Origin

Risk evaluation is quite pleasant for concentric squares at the origin due to the fact that we are using the uniform distribution as the marginal distribution. Since the marginal is uniform and the true labeling square, denoted by f, and our hypothesis square, denoted by h, have the same center, the probability of mislabeling an instance is just the the area where f and h disagree. We can easily calculate that area with $|A(f) - A(h)|$ where $A(f), A(h)$

are the respective areas of f, h.

4.3.3 Training When Structure is Known, but Location is Not

Learning to classify data labeled by concentric squares at the origin exploits the knowledge of location. In this section, we examine how we can learn with slightly weaker apriori knowledge. Namely, we now learn under Assumption 3.2 where we suspect the true labeling function is a square but we do not know where that square is in space. To handle our lack of knowledge of location, we introduce methods for estimating the center of a loop.

Averaging Over the Positive Instances

One method we can use to try to estimate the center is to average over the positive instances. That is, if we have some set S^+ of positive instances from S, we can estimate the center with

$$\hat{\mathbf{c}} = (\hat{c_1}, \hat{c_2}) = \left(\frac{1}{|S^+|} \sum_{x_1 \in S^+}^{|S^+|} x_1, \frac{1}{|S^+|} \sum_{x_2 \in S^+}^{|S^+|} x_2 \right). \tag{4.1}$$

The reason we average along the positive instances and not the entire data set is that averaging along the entire data set gives us an estimate of the mean of the marginal distribution, and that mean might not be meaningful to us. For instance, if the marginal distribution is uniform (as it is in our experiments), then we should expect the average along all instances to approach $(0, 0)$ as the sample size increases, regardless of where the true center is. By averaging along the positive instances, we are limiting are focus to the loop itself without knowing where the loop is.

One problem with this approach is that it is subject to failure if the marginal distribution is such that sampling a large number of positive instances is unlikely. Suppose, for instance, that the true labeling loop is a square at the origin with a bound $b = 0.01$ and that the marginal distribution is uniform. The probability of sampling a positive instance is then $0.02^2 = 0.0004$ and therefore, even in a large sample, the number of positive instances will be quite small (or zero, even), leading to a poor empirical estimate. One way we can attempt to work around this issue is to exploit our apriori knowledge, as discussed in the next section.

Estimating the Center Using Extreme Points

We are currently operating under the assumption that the true labeling function is given by a square classifier. It then follows that the the positive point furthest to the left gives us our best possible estimate of the left border of the square classifier. Similarly, the rightmost positive gives us an estimate of the right border of the square classifier, as does the top most point for the top border, and the lowest point for the bottom border; we can use these extreme points to get an estimate of the center of the labeling loop.

Algorithm 6 Extreme Points Estimate (Positive Version)

1: Let $S^+_{x_1}$ be the set of x_1 values of the positive instances in our sample S.
2: Let $S^+_{x_2}$ be the set of x_2 values of the positive instances in our sample S.
3: Define $Top = \max(S^+_{x_2})$
4: Define $Bottom = \min(S^+_{x_2})$
5: Define $Left = \min(S^+_{x_1})$
6: Define $Right = \max(S^+_{x_1})$
7: Let $\hat{c_1} = \frac{Right - Left}{2}$ and $\hat{c_2} = \frac{Top - Bottom}{2}$.
8: **return** $\hat{\mathbf{c}} = (\hat{c_1}, \hat{c_2})$

The astute reader may question whether Algorithm 6 has made any progress over averaging along the positive instances. That is, we said that the problem with averaging along the positive instances is that it can fail when the number of positive instances is small. If the number of positive instances in our sample is small, how can we be sure that Top, $Bottom$, $Left$, $Right$ are actually close to the edges of the square classifier? For small sample sizes, we can not be sure, but that does not mean we have not made progress. Recall that in the positive, negative, and ERM algorithms we used upper and lower bounds to get a better estimate by taking their average. We can do the same here. We first need to find the negative instances closest to the square classifier, and we use the extreme points found in Algorithm 6 to do so.

Once we have estimates from the extreme negative and extreme positive points, we can take their average (Algorithm 8). We do this average process because the extreme points may not actually be that close to the border, but their average might. For instance, suppose that by chance the positive and negative points corresponding to the left edge of the border

Algorithm 7 Extreme Points Estimate (Negative Version)

1: Let $S^-_{x_1}$ be the set of x_1 values of the negative instances in our sample S.
2: Let $S^-_{x_2}$ be the set of x_2 values of the positive instances in our sample S.
3: Let $Top_+, Bottom_+, Left_+, Right_+$ be the extreme points found in Algorithm 6.
4: Define $Top(S^-_{x_2}) = \min\left(\left\{x_2 \in S^-_{x_2} : x_2 \geq Top_+\right\}\right)$
5: Define $Bottom(S^-_{x_2}) = \max\left(\left\{x_2 \in S^-_{x_2} : x_2 \leq Bottom_+\right\}\right)$
6: Define $Left(S^-_{x_1}) = \max\left(\left\{x_1 \in S^-_{x_1} : x_1 \leq Left_+\right\}\right)$
7: Define $Right(S^-_{x_1}) = \min\left(\left\{x_1 \in S^-_{x_1} : x_1 \geq Right_+\right\}\right)$
8: **return** $\hat{\mathbf{c}} = (\hat{c_1}, \hat{c_2}) = \left(\frac{Top(S^-_{x_2}) - Bottom(S^-_{x_2})}{2}, \frac{Right(S^-_{x_1}) - Left(S^-_{x_1})}{2}\right)$

are each roughly 0.1 units away from the border. When we average between them, we will be right on the border's edge, thereby giving us a better estimate of the center.

Algorithm 8 Extreme Points Estimate (Mid Version)

1: Let $Top_+, Bottom_+, Left_+, Right_+$ be the extreme points found in Algorithm 6.
2: Let $Top_-, Bottom_-, Left_-, Right_-$ be the extreme points found in Algorithm 7.
3: Define $Top_m = \frac{Top_+ + Top_-}{2}$
4: Define $Bottom_m = \frac{Bottom_+ + Bottom_-}{2}$
5: Define $Left_m = \frac{Left_+ + Left_-}{2}$
6: Define $Right_m = \frac{Right_+ + Right_-}{2}$
7: **return** $\hat{\mathbf{c}} = (\hat{c_1}, \hat{c_2}) = \left(\frac{Top_m - Bottom_m}{2}, \frac{Right_m - Left_m}{2}\right)$

Once we have center estimates, we can use modified versions of postive, negative, and mid ERM schemes. The modifications we would need to make are as follows. In the previous section, we had finitely many bounds to choose from. In this case, we have infinitely many so incrementing by some fixed value does not make sense. Instead, we can just skip to the next appropriate bound after mislabeling an instance. For example, if we are running positive bias and we misclassify a negative instance, we can skip from where we currently are to being right next to the negative instance we got wrong such that it is now labeled correctly.

One other change is that in the realizable setting we used a while loop because the realizability assumption guarantees zero empirical risk (and therefore, termination of the loop). In this case, a while loop could run forever so we simply loop through the entire sample,

jumping our bound from point to point as needed.

4.3.4 Risk Evaluation for Generalized Squares

Evaluating risk for generalized squares is much harder than one might expect it to be. In order to understand why, we will need several new definitions. We begin with the area of the true labeling loop inside $\mathcal{X}_{sq}$. Consider the two different labeling squares depicted in Fig. 4.3. Notice that the true labeling square, which we will denote as f, in the left example is entirely contained within $\mathcal{X}_{sq}$ whereas in the second example, only some of f is in $\mathcal{X}_{sq}$.

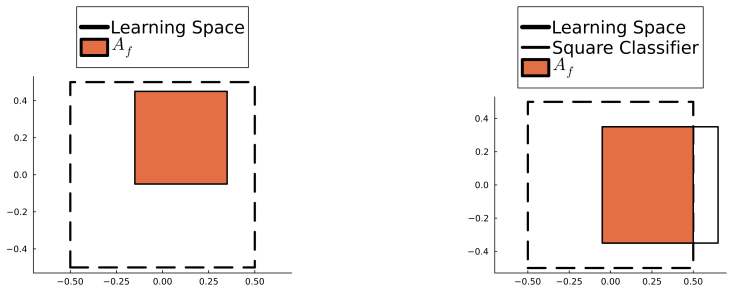


Figure 4.3: Two Examples of A_f. If f entirely in $\mathcal{X}_{sq}$ (left plot) then A_f equivalent to area of f. Otherwise (right plot) A_f less than area of f.

In order to calculate the risk of an arbitrary square classifier, we will first need A_f, the area of the label function f contained within $\mathcal{X}_{sq}$.

Claim 4.1 *Given a square classifer that acts as the true labeling function f, we say that the area A_f is the area of the square formed by f contained within $\mathcal{X}_{sq}$ and we find A_f with*

$$Height_f \cdot Width_f = (Top_f - Bottom_f) \cdot (Right_f - Left_f)$$

where $(Top_f - Bottom_f) \cdot (Right_f - Left_f)$ is given by

$$\big(\min(0.5, c_2 + b) - \max(-0.5, c_2 - b)\big) \cdot \big(\min(0.5, c_1 + b) - \max(-0.5, c_1 - b)\big)$$

and c_1 and c_2 are from the (c_1, c_2) pair describing the center of f and b is the bound as defined in Definition 3.2.

Proof. From inspection, it should be clear that the area A_f will always be contained in some rectangle. Thus, to find A_f we simply need to determine the height and width of the aforementioned rectangle. Given some area A_f, we find the height by subtracting the highest point Top_f from the lowest point $Bottom_f$. We define Top_f to be $\min(0.5, c_2 + b)$ because if the top of f is outside of $\mathcal{X}_{sq}$ then the highest point in $\mathcal{X}_{sq}$ is simply the top of $\mathcal{X}_{sq}$ itself. Similar commentary about $Bottom_f$, $Left_f$, and $Right_f$ completes the proof. □

The next step in understanding the true risk a square hypothesis is defining A_h, the area of the hypothesis contained inside $\mathcal{X}_{sq}$.

Definition 4.1 *We denote the area of a square hypothesis h that is contained within $\mathcal{X}_{sq}$ to be A_h. We find A_h in exactly the same way as A_f. That is,*

$$A_h = Height_h - Width_h.$$

We now need to think about where A_f and A_h overlap, as that is one of two areas where a square classifier has zero risk. In particular, consider the different regions highlighted in Fig. 4.4.

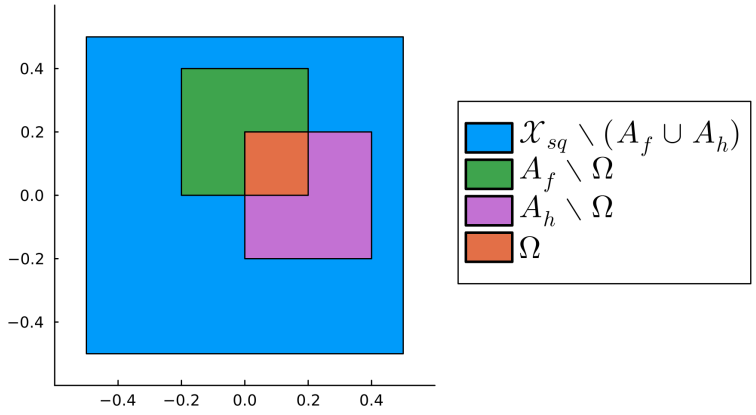


Figure 4.4: The Four Regions Needed to Evaluate Risk

As we have done before in Section 3.4.1, call the area where A_f and A_h overlap the Ω area. The areas $A_f \setminus \Omega$ and $A_h \setminus \Omega$ are respectively the region in A_f but not in Ω and the region in A_h but not in Ω. Finally, the last region $\mathcal{X}_{sq} \setminus (A_f \cup A_h)$ is everything that is outside of both A_f and A_h. We can see where f and h disagree in Table 4.3.

Region	$f(\mathbf{x})$	$h(\mathbf{x})$
$\mathbf{x} \in \Omega$	1	1
$\mathbf{x} \in \mathcal{X}_{sq} \setminus (A_f \cup A_h)$	0	0
$\mathbf{x} \in A_f \setminus \Omega$	1	0
$\mathbf{x} \in A_h \setminus \Omega$	0	1

Table 4.3: Summary of Four Regions in $\mathcal{X}_{sq}$

Because we chose to make the marginal distribution $\mathcal{D}_{\mathcal{X}_{sq}}$ be uniform, it then follows that the probability of h mislabeling an instance is the area $(A_h \setminus \Omega) \cup (A_f \setminus \Omega)$. We can find this region with $(A_f - \Omega) + (A_h - \Omega)$.

Remark 4.2 *We abused notation a bit in our discussion about risk for arbitrary squares. When referring to A_f, A_h, Ω we are sometimes referring to a set of points forming an area, sometimes referring to the numerical value of the area itself. In general, if we use arithmetic operations the reader may assume A_f, A_h, Ω are values and that when we use set operators like $\cup, \setminus$ we are referring to a set of points.*

4.3.5 Training and Evaluating Rectangular Classifiers

The algorithms necessary for training rectangular classifiers have actually already been discussed when we constructed algorithms to find extreme points. In particular, recall the points $Top, Bottom, Left, Right$ from Algorithm 6. A rectangular classifier biased towards negative labeling will simply classify everything that is in between $Left, Right$ and $Top, Bottom$ as positive, otherwise negative.

As for risk, the whole point of generating data with a uniform marginal distribution is that area is equivalent to risk and more importantly, that for nice shapes (squares), area is easy to calculate. Now that our true labeling loops have more complex structure, the only way to get an exact area calculation is through integration. Ultimately, though, if we

had any intention of integrating then we could have explored nonuniform data from the beginning. Hence, we are not going to get an exact value for the true risk of a rectangular classifier (or a grid classifier, for that matter). Instead, we will construct a meaningful test set comprised of a set of n^2 points evenly spaced out like a grid. Our risk estimate will then be the ratio between mislabeled points and all points.

4.3.6 Training and Evaluation of Grid Classifiers

In order to train grid classifiers, we first need to find the borders discussed in Definition 3.11, but we have actually already done this via rectangular classifiers. Rectangular classifiers are precisely the border we are looking for so we use the training scheme for rectangular classifiers when searching for a border to form our grid. As for the grid itself, there are multiple mini-procedures we need to discuss that occur during Algorithm 9.

Algorithm 9 Scan Grid

```
Initialize a hash table H to be empty.
Let S be our sample, B our border, G our untrained grid matrix.
for each instance x ∈ S do
    if x ∈ B then
        Locate the indices i, j such that x ∈ G_{i,j}
        if (i, j) ∉ H then
            add (i, j) to H
        end if
    end if
end for
return G, H
```

We would like the grid to be as expressive as possible without overfitting. To make the grid more expressive, we increase the number of cells using binary search, where a grid with empty cells is considered a failure and leads to decrementing and a grid with no empty cells results in incrementing. To avoid overfitting, we require that no cell is empty when the search terminates. In order to know whether we have any empty cells, we train each grid *while searching for the right number of cells*. Meaning, suppose that the number of cells is fixed to be n^2. To check if a cell is empty, we would have to loop through all instances

anyway, so why not train the grid matrix of size n^2 at the same time? We achieve both with Algorithm 9. Namely, if the hash recording all the indices we have seen before contains n^2 index pairs, then we know there are no empty cells and we can increment n. Otherwise, our grid is too expressive and we must decrement n.

As for cell labels, that can be done during Algorithm 9 as well. Let the grid matrix $\mathbf{G}$ be initialized to all zeroes. Now consider an instance $\mathbf{x}$ located at $\mathbf{G}_{i,j}$ in the grid. If $\mathbf{x}$ is a positive instance, we add one to the count corresponding to $\mathbf{G}_{i,j}$, otherwise we subtract one. When our scan is complete, assuming that there were no empty cells, any cell with count 0 had equally many positive and negative instances and is labeled by a coin flip. All other cells are labeled positive if their count is positive, negative otherwise.

Now that we have discussed the basic ideas behind our experiments from training to risk evaluation, we are finally ready to obtain some results.

If we knew what it was we were doing, it would not be called research, would it?

Albert Einstein

5 Results and Discussion

Everything we have been building up to has been in the spirit of getting closer to answering the questions, "How much bias is enough? What is the minimal amount of information we need to classify nonlinear data?" Have we answered these question? Well, we consider results without proof a nonanswer, so no, we have not answered the motivating questions. We have, however, obtained empirical results that might warrant future theoretical analysis.

In the following sections, we run the various algorithms described in Chapter 4 and inspect why their behavior agrees or disagrees with our initial expectations. Ultimately, we found that grid learners are completely capable of classifying nonlinear data in a bounded region with a uniform marginal distribution. Whether the strength of grid learners will remain if we extend to nonuniform distributions and unbounded learning spaces is not yet clear.

5.1 Concentric Squares at Origin

Not surprisingly, the sample complexity estimates of the finite, realizable hypothesis class $\mathcal{H}_{sq}$ were quite low. After running 1000 trials over several different ϵ and b values with a confidence choice $1 - \delta = 0.9$, we obtained the estimate plots shown in Fig. 5.1.

Notice what happens to the sample complexity of the ERM algorithms that are biased towards one of negative or positive labeling. When we make the bound of the true labeling function small, most of $\mathcal{X}_{sq}$ is associated with a negative label, so we should expect the negatively biased ERM scheme to have a smaller sample complexity (and it does). Similarly, when we make the bound large, most of $\mathcal{X}_{sq}$ is associated with a positive label so we should expect the positively biased ERM scheme to have a lower sample complexity (and it does).

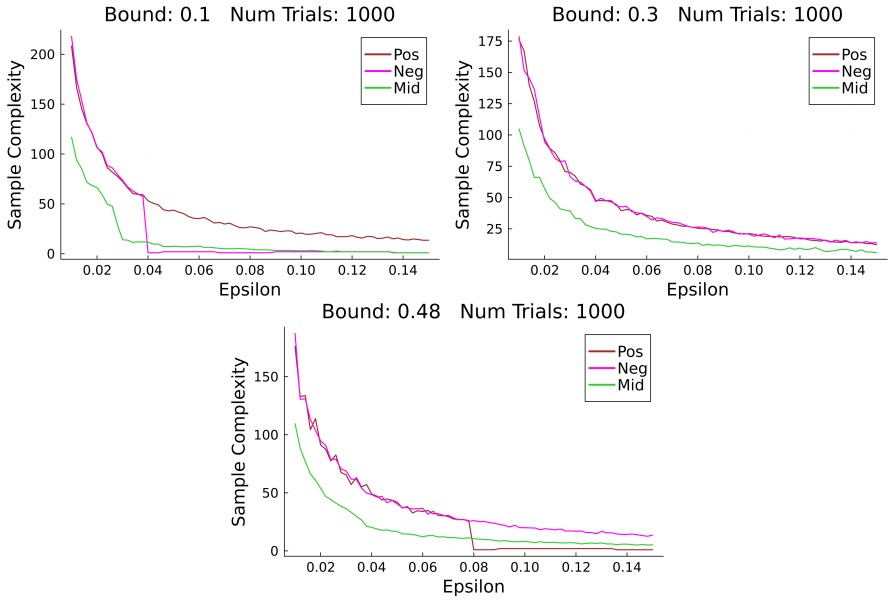


Figure 5.1: Sample Complexity Estimates for ERM Learners Choosing Classifiers from $\mathcal{H}_{sq}$

There are few outcomes that seem unusual at a glance, but are actually perfectly normal with further inspection. In particular, notice the dips in the plots associated with $b = 0.48$ and $b = 0.1$. The values $b = 0.1$ and $b = 0.48$ were the first values where we saw the bias induced into our Positive/Negative ERM schemes begin to outperform the Midpoint scheme. Does it seem reasonable that these are the values where biasing one way or the other makes a difference? Yes, it does, but the reason may not clear if we do not write out the arithmetic.

In the case where $b = 0.1$, the area of the true labeling square is $(2b)^2 = 0.04$, implying that 96% of $\mathcal{X}_{sq}$ is negatively labeled. Notice that the negative ERM sample complexity drops at risk tolerance 0.04, the same area as the true labeling square. In other words, when

our risk tolerance is forgiving enough and the bound is small, the negative ERM learner will have met our risk tolerance without training on any data at all. Similar remarks apply to the positive ERM scheme.

When the bound is $b = 0.48$, the area of the true labeling square is $2b^2 = 0.9216$ and therefore, about 8% of all of $\mathcal{X}_{sq}$ is negatively labeled. Since the positive ERM learner is initialized to start its search by labeling everything positively, the bound $b = 0.48$ implies that the initial hypothesis chosen by positive ERM violates obeys risk tolerance for values above 0.08 by default. And as we can see, when $b = 0.48$, the sample complexity of positive ERM plummets at $\epsilon = 0.08$, as we would expect it to.

A few paragraphs back, we mentioned that $b = 0.48, 0.1$ were the *first* values for which the bias induced into the positive/negative ERM algorithms begin to take effect, implying that there are other such values. Some readers may wish to know if we can visualize what happens as we slide the value of b. That is, can we display similar information to Fig. 5.1 without plotting many, many different plots? Yes! If we fix a bound and run Algorithm 1 (empirical PAC verification) over many different sample sizes, we can produce a heat map (Fig. 5.3) that communicates how much risk we might expect when training.

Remark 5.1 *We use different colors on the heat maps to denote differing scales. If we force each map to have the same scale, it is harder to see the variability on the midpoint map.*

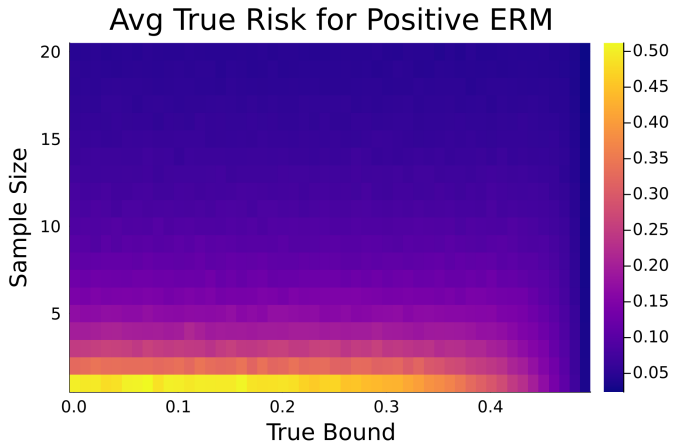


Figure 5.2: First of Three Heat Maps For $\mathcal{H}_{sq}$ Obtained Using 1000 Trials

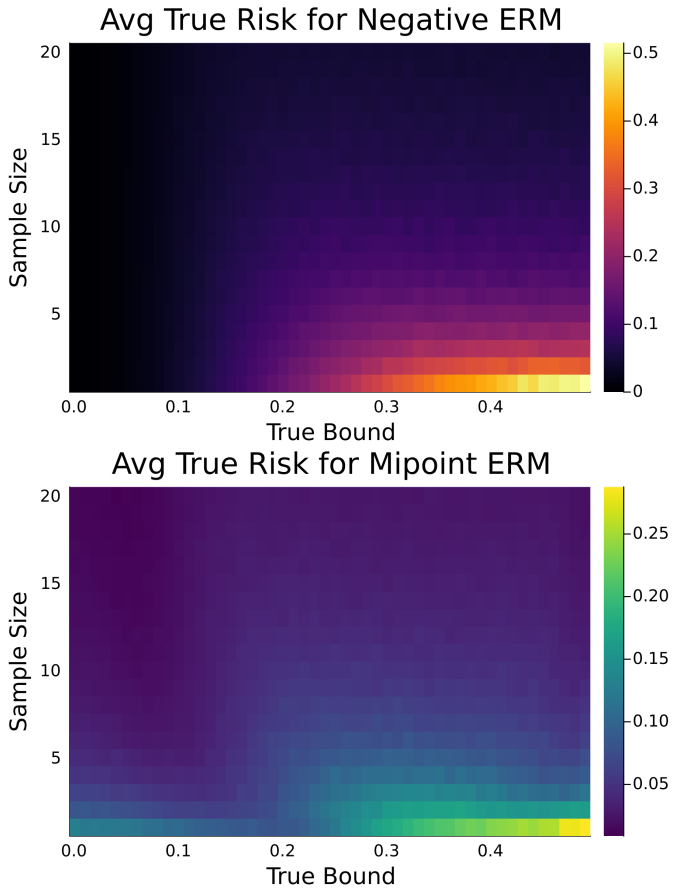


Figure 5.3: Second and Third of Three Heat Maps For $\mathcal{H}_{sq}$ Obtained Using 1000 Trials

Notice that at a sample size of 20, we are generally achieving an *average* risk of 0.05 or lower across all ERM algorithms, despite the fact that our sample complexity plots have a values of about 50 for a risk tolerance $\epsilon = 0.05$. We might wonder, then, do the heat plots and the sample complexity estimates contradict one another? No. The important detail here is that the heat plots are showing an average risk value whereas the sample complexity estimates require risk values to occur at a certain frequency to meet our confidence choice.

Between the biased induced into our ERM algorithms and the hypothesis class $\mathcal{H}_{sq}$, the fact that Assumption 3.1 (true labeling function is indeed a square at the origin) holds in these experiments results in frequently obtaining low or even zero risk hypotheses (even in smaller sample sizes). Hence, when we compute the average, we are getting a value that im-

plies more strength in the ERM learners than is actually there. We can see the importance of the confidence choice and how it can create a discrepancy between the heat map plots and the sample complexity estimates by plotting a histogram that shows the distribution of risk values found by Algorithm 1.

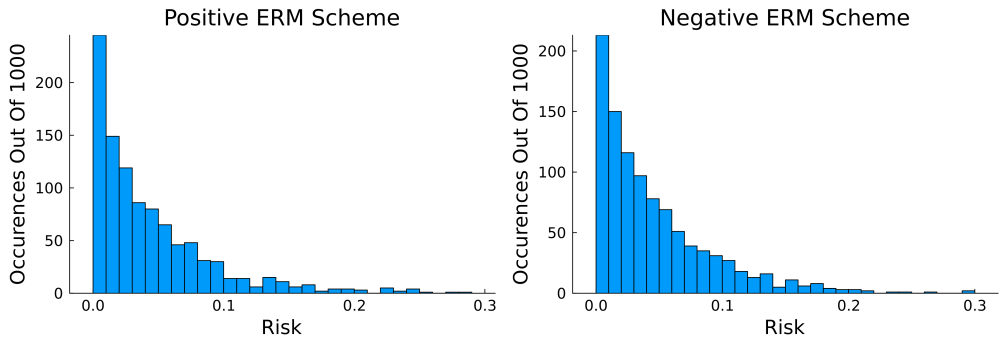


Figure 5.4: Risk Values Obtained When Training On Samples of Size 20 Generated by $Q_{0.3}$

As expected, for a bound of 0.3 (one of the bounds used in Fig. 5.6) the average risk seems to be roughly[1] 0.05. Notice, though, that if we do a quick sum[2] of the number of risk values at or below 0.05, we get conservatively get $250 + 150 + 125 + 100 + 100 = 725$ out of 1000 trials for the positive ERM values. Recall that in our sample complexity estimates, we set our confidence parameter to be $\delta = 0.1$, meaning we need at least 900 of the 1000 to have a risk value of 0.05 or below so clearly, we need a larger sample size than 20 to achieve the PAC learning threshold. Thus, the discrepancy between the heat plots and the sample complexity estimates is not an issue.

Before we move on from the realizable setting, let us briefly turn our attention to the risk values of the mid ERM algorithm. The heat map for the mid ERM algorithm has lower average risk values than either the positive or negative ERM algorithms. Moreover, the sample complexity is much lower for mid ERM than positive or negative. We might wonder, then, does mid ERM achieve zero risk more often than positive and negative ERM do?

Counterintuitively, mid ERM does not necessarily achieve 0 risk more often than positive or negative ERM. In the plots in Fig. 5.5 are results from an experiment with 1000 trials of samples of size 20 like before. Only this time, the mid values are such that there are

[1] We checked this with the computer, but one can confirm by quickly eyeballing the distribution.
[2] Again, this can be confirmed by eyeballing the height of the bins.

fewer zero risk occurrences (roughly 150) in the mid ERM results than there are in positive or negative ERM (roughly 200).

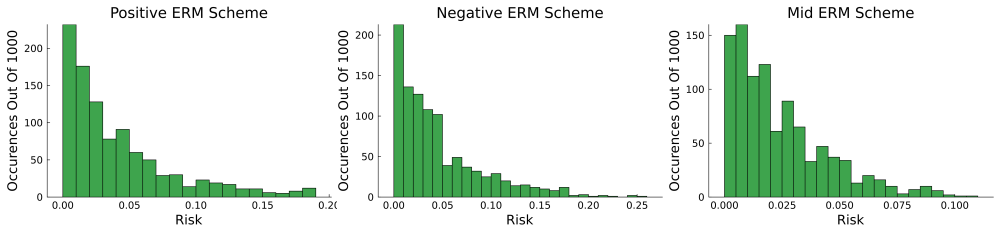


Figure 5.5: Risk Values Obtained When Training on Samples of Size 20 Generated by $Q_{0.3}$

The reason mid ERM achieves lower average risk value *and* lower sample complexity while simultaneously achieving fewer zero risk hypotheses has to do with the hypotheses chosen from positive and negative. In the histograms in Fig. 5.5, we see that positive and negative ERM both achieve about 200 zero risk hypotheses. The trick is that they do not achieve these hypotheses on the same sample.

Since the midpoint algorithm returns a hypothesis whose value is dependent on the values of positive and negative ERM hypotheses, if one and only one of them achieves a zero risk hypothesis, mid ERM is guaranteed (in that instance) to achieve a nonzero hypothesis. Other than that, mid ERM behaves exactly as we would expect, which is to say that it is a stronger learner than positive and negative ERM are. Even though mid ERM does not necessarily achieve zero risk as often positive or negative ERM, the variability of the risk values is much lower in the mid ERM distribution (notice that the worst risk mid ERM achieved was about 0.1 whereas the worst risk positive/negative ERM achieved was about 0.2).

5.1.1 Mid ERM and Support Vector Machines

Before we move on to the next learning setting, let us take a second to examine how strong $\mathcal{H}_{sq}$ is in combination with mid ERM when Assumption 3.1 holds.

In our sample complexity plots, we found that when training on data generated by $Q_{0.3}$ we found that 90% of the time, mid ERM achieved 99% accuracy with a sample size as small as 100 instances. Even at 50 and 25 instances mid ERM is still at least 96% accurate 90% of the time. Point being, if one did not know they were looking for a square, it would

not be clear at all from the samples depicted in Fig. 5.6 that the true labeling function is indeed a square.

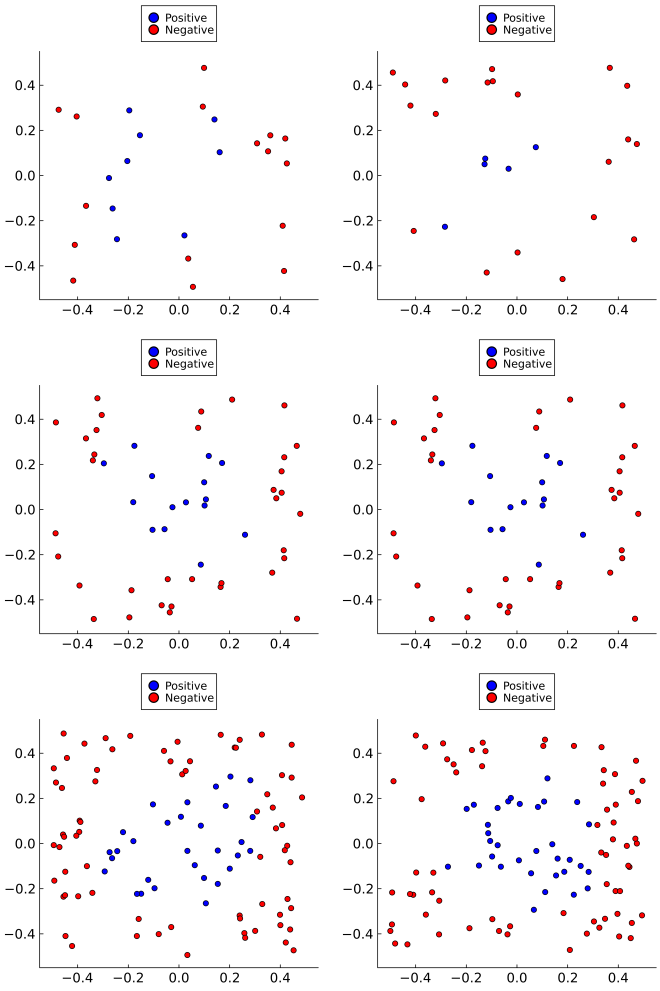


Figure 5.6: Samples of Size 25 (top), 50 (middle) 100 (bottom)

While it is true that the bias induced into $\mathcal{H}_{sq}$ is extreme, the positive and negative ERM schemes are not as strong as mid ERM, as we have discussed at length. The reason that mid ERM is so much stronger is that mid ERM is achieving a support vector machine [10], which should be cause for excitement given that the data is nonlinear and we are not transforming it. To see why mid ERM is an SVM, first recall that positive and negative ERM provide us with upper and lower bounds for the hypotheses we can pick. Thus, by taking their average,

we force a maximization of the margin between positive and negative points in the learning space. Another interesting point is that of linear separability. Hard-SVMs require data to be linearly separable, and certainly our data is not. How is, then, that we achieve a hard-SVM? Square classifiers actually act as a kernel. If we project our data into 3 space using the constraint $\max(|x_1|, |x_2|)$ defining Q_b then we achieve data that is linearly separable.

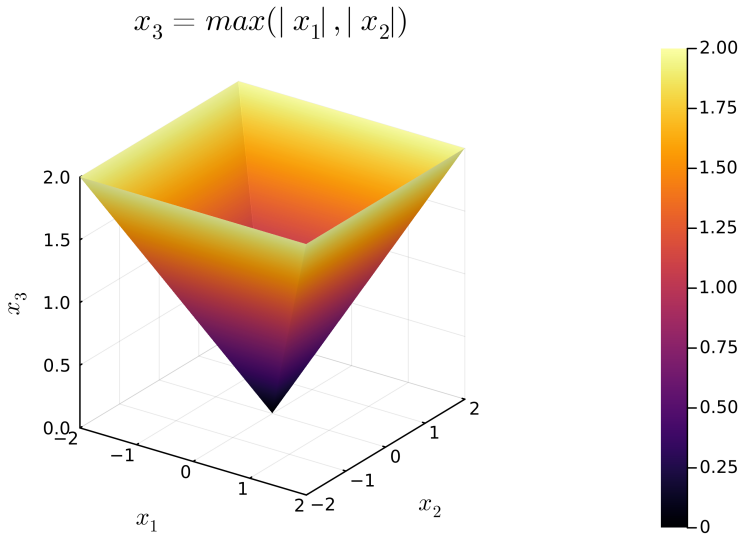


Figure 5.7: Projection of Data Into 3 Space

Imagine slicing the pyramid in Fig. 5.7 with a plane parallel to the x_1, x_2 plane. The x_3 value of the slice corresponds to the bound in the condition $\max(|x_1|, |x_2|) \leq b$. Thus, by slicing the pyramid, we separate all positive instances (those below/on the slice) from all negative instances (those above the slice).

5.2 Arbitrary Square Classifiers

We should expect the sample complexity of square classifiers to increase once we remove our apriori knowledge of the true labeling loop's location—and indeed it does. As shown in Fig. 5.8, where we used the same number of trials (1000) and confidence choice ($\delta = 0.1$) as we did in our experiments in Section 5.1, the sample complexity is much higher than that of square classifiers fixed at the origin.

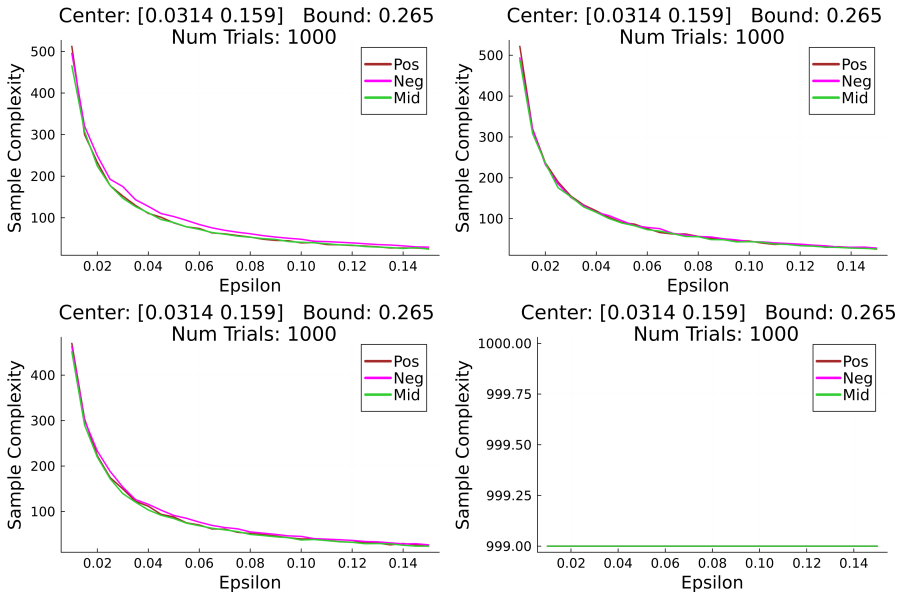


Figure 5.8: Sample Complexity Estimates After Estimating Center With Algorithm 6 (Top Left), Algorithm 7 (Top Right), Algorithm 8 (Low Left), and Averaging Along Positive Instances (Low Left)

There are several points of interests in Fig. 5.8. First and foremost, it should be clear that estimating the center dominates the sample complexity. We can be confident of this because the ERM schemes all have approximately the sample sample complexity estimates and the only commonality among all three is center estimation. The second most interesting detail in our estimates is that averaging along the positive instances does not provide a reliable estimate of the center. That is, the error attributed to the poor center estimate is such that we consistently fail to obey any risk tolerance at or below $\epsilon = 0.15$.

To get a sense of the performance of the estimate obtained by averaging along the positive instances, we conducted a mini-experiment in which we generated many different samples and estimated the center using all four methods several times over. The averages of the results are summarized in Table 5.1 and we can see that the method of averaging along the positive instances seems to be about an order of magnitude less precise than the other methods.

Sample Size	+ Extreme Pts.	− Extreme Pts.	Mid Extreme Pts.	Avg Along +
100	$\begin{bmatrix}-0.000374\\-0.000318\end{bmatrix}$	$\begin{bmatrix}-0.000210\\-0.000033\end{bmatrix}$	$\begin{bmatrix}-0.000292\\-0.000175\end{bmatrix}$	$\begin{bmatrix}0.001164\\0.000384\end{bmatrix}$
1000	$\begin{bmatrix}-0.000049\\-0.000000\end{bmatrix}$	$\begin{bmatrix}-0.000016\\0.000030\end{bmatrix}$	$\begin{bmatrix}-0.000032\\0.000015\end{bmatrix}$	$\begin{bmatrix}-0.000207\\-0.000187\end{bmatrix}$
10000	$\begin{bmatrix}-0.000002\\0.000005\end{bmatrix}$	$\begin{bmatrix}-0.000003\\0.000007\end{bmatrix}$	$\begin{bmatrix}-0.000002\\0.000006\end{bmatrix}$	$\begin{bmatrix}0.000038\\0.000025\end{bmatrix}$

Table 5.1: Summary of Center Estimates (up to 6 digits) For True Center $\begin{bmatrix}0\\0\end{bmatrix}$ Over 1000 Trials of Varying Sample Sizes

5.2.1 Violating Assumption 3.2

What happens if the true labeling square $Q_{b,\mathbf{c}}$ is such that the true center is inside $\mathcal{X}_{sq}$ but part of the square is outside $\mathcal{X}_{sq}$? Consider, for instance, the square classifier $Q_{0.4,(0.4,0.2)}$ centered at $(0.4, 0.2)$ with a bound 0.4. Such a classifier will generate rectangular data, such as the sample in Fig. 5.9.

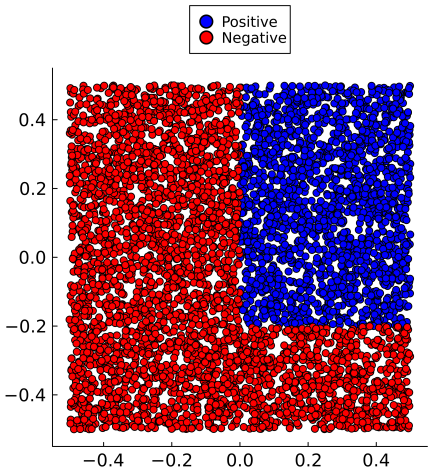


Figure 5.9: Sample of 5000 Instances Labeled By $Q_{0.4,(0.4,0.2)}$

Our original assumption about squares (Asm. 3.2) with unknown centers is simply that the true labeling loop is centered somewhere in $\mathcal{X}_{sq}$. Said assumption does not say anything

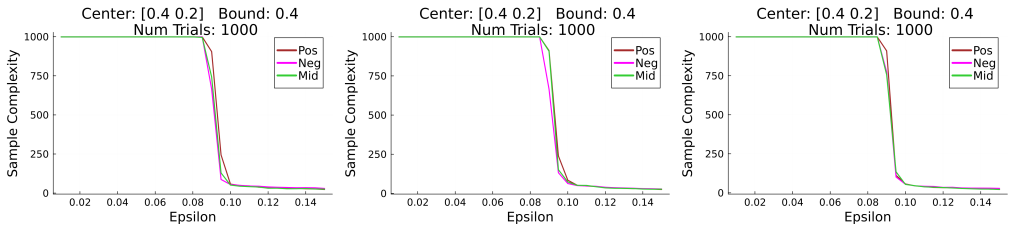


Figure 5.10: Sample Complexity Estimates For $Q_{0.4,(0.4,0.2)}$ Data With Confidence Choice $\delta = 0.1$

about squares that generate rectangular data. If our assumption was well formed, then the ERM algorithms we have constructed should be able to learn by choosing hypotheses from $\mathcal{F}_{sq}$. We can easily see, though, that is not the case.

When we plot sample complexity estimates (Fig. 5.10), we can see that our search terminates at the search limit 999 and stays there until there is an immediate drop, implying that learning fails due to the geometry of the data.

We can get a sense of why this failure occurs by examining the borders of the data. In particular, we know that the left most point is at least $0.4 - 0.04 = 0$ and the right most point is at most 0.5. Similarly, the top most point is at most $0.5 < 0.2 + 0.4$ and bottom most point is at least $0.2 - 0.4 = -0.2$ Hence, for samples with points near the border, the center estimate will be roughly $\left(\frac{0+0.5}{2}, \frac{0.5-0.2}{2}\right) = (0.25, 0.15)$. Ultimately, this misguided estimate is the root of the poor generalization. When negative ERM starts at the wrong center, it will inch forward, trying to correctly label as many positive instances as possible. In doing so, it will eventually settle at around 0.25. If we calculate the true risk of the classifier $Q_{0.25,(0.25,0.15)}$ we will get $0.09999\ldots$, which agrees perfectly with the sample complexity estimates. PAC learning fails when we try to pick a risk tolerance less than 0.1 because that is the best we can do.

5.3 Rectangular Classifiers

Admittedly, there is not much to discuss about rectangular classifiers. In short, if the true labeling loop is roughly rectangular (or only represents a very small fraction of the learning space), then rectangular classifiers provide a decent approximation (and otherwise they do not). We can provide a quantitative explanation of what "decent" means by examining a tri-

folium curve. Consider, for instances, three different parameter choices of the curve such that the first choice is a convex shape, the second is no longer convex but the rectangular approximation is small enough relative to the learning space that the error may be somewhat forgivable, and the final choice where a rectangle is simply not an appropriate approximation at all. We show these choices in Fig. 5.11.

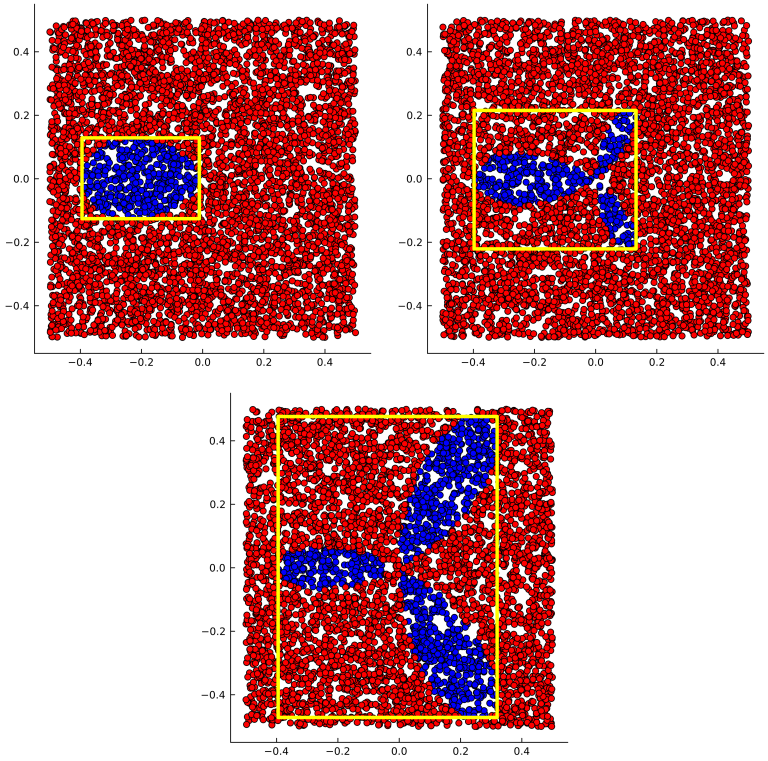


Figure 5.11: Rectangular Approximations of Folium Curves with Parameters $a = 0.1, b = 0.4$ (left), $a = 0.3, b = 0.4$ (right), and $a = 0.5, b = 0.4$ (bottom)

To quantify the strength of the approximations depicted in Fig. 5.11 we ran trials and recorded the risk values. As we can see in Fig. 5.12, the folium corresponding to a small convex shape is well approximated by a rectangle as the average risk is roughly 0.02 and the variance is small. As we strech the folium into a more complex curve, the approximation weakens. For choices $a = 0.3, b = 0.4$, the loop itself is clearly not approximated well by a rectangle, but the rectangles area is small enough that its error is still not egregious with an

average risk of roughly 0.145. Once the folium is both not at all convex and large relative to the learning space, approximation fails, obtaining less than 50% accuracy.

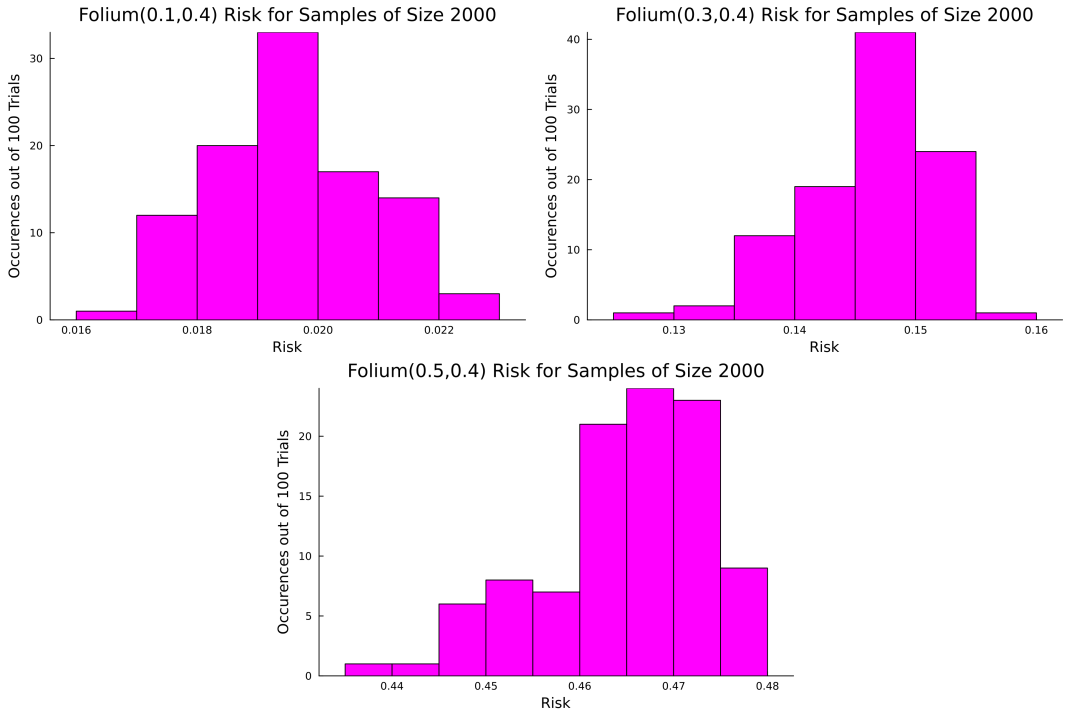


Figure 5.12: Risk Values of Folium Curves Depicted in Fig. 5.11

5.4 Grid Classifiers

As we saw previously, rectangular classifiers are not particularly strong in general, but they are still useful to us in that they lay the foundation for grid classifiers. Suppose for instance, that we add cells to the rectangular approximations in Fig. 5.11. Doing so yields unassigned grid matrices and as we can see in Fig. 5.13, if the cells of our matrices are small enough, it is reasonable to suspect that grid learners will obtain strong approximations of arbitrary loops with sufficiently large samples. Indeed, after training on the samples in Fig. 5.13, we obtain results that are quite promising (see Fig. 5.14).

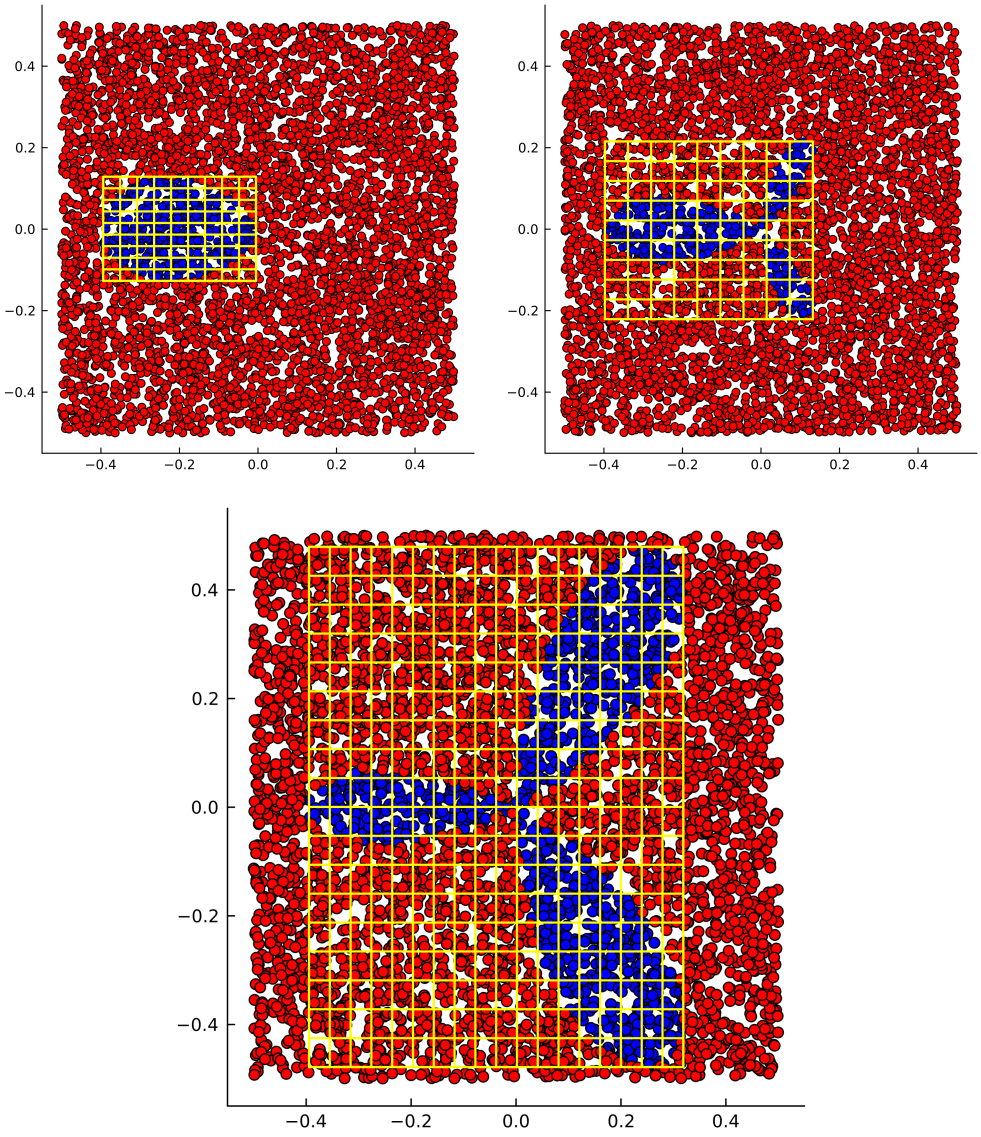


Figure 5.13: Grid Matrix Visualization for $Folium(0.1, 0.4)$ [left], $Folium(0.3, 0.4)$ [right], $Folium(0.5, 0.4)$ [bottom]

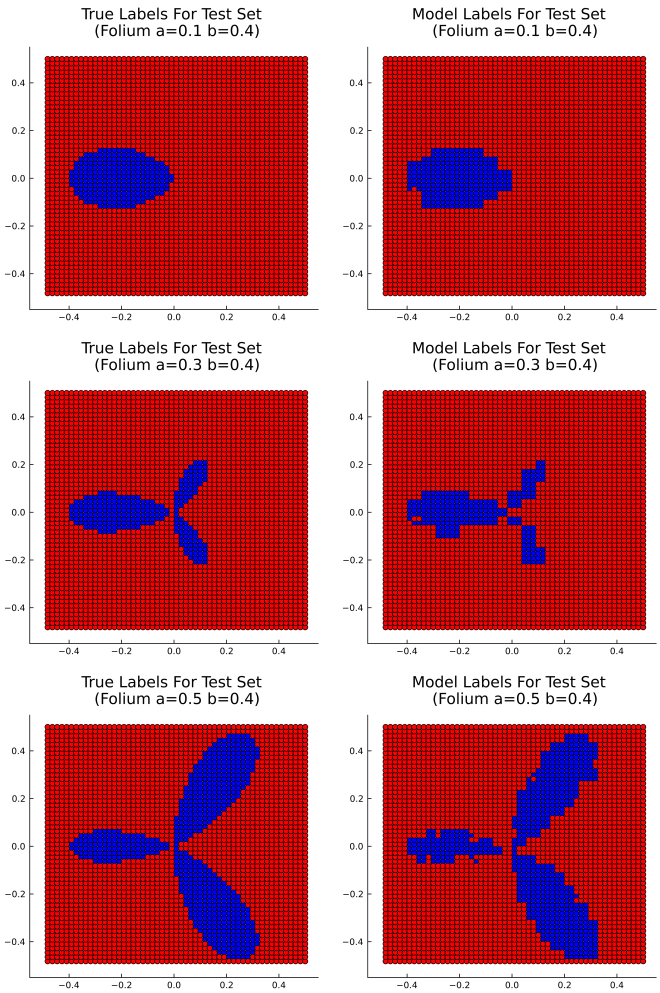


Figure 5.14: True Labels (Left) and Model Labels (Right) After Training

5.5 Conclusion

The reader may be left with a few very reasonable questions. 1) Do gird classifiers still perform well if we decrease the sample size? That is, the samples in Fig. 5.13 are large enough that we have densely filled the entire learning space. In practice, we may not have access to such a rich sample. 2) Thus far, our analysis of the performance of grid classifiers has been limited to three different labeling functions. What about other loops? 3) The family

$\mathcal{F}_{Grid}$ seems to be arbitrarily expressive. How is it that such a rich family of hypotheses is able to avoid overfitting? Formally, do grid classifiers have a finite VC dimension? 4) Can we extend grid classifiers to other learning settings, such those in higher dimensions, non-uniformly distributed data, or multiclass classification?

We can immediately give empirical answers to questions 1) and 2). Namely, we can create heat maps that show how grid classifiers perform on many different parameterizations of folium curves where each heat map is generated using a unique sample size.

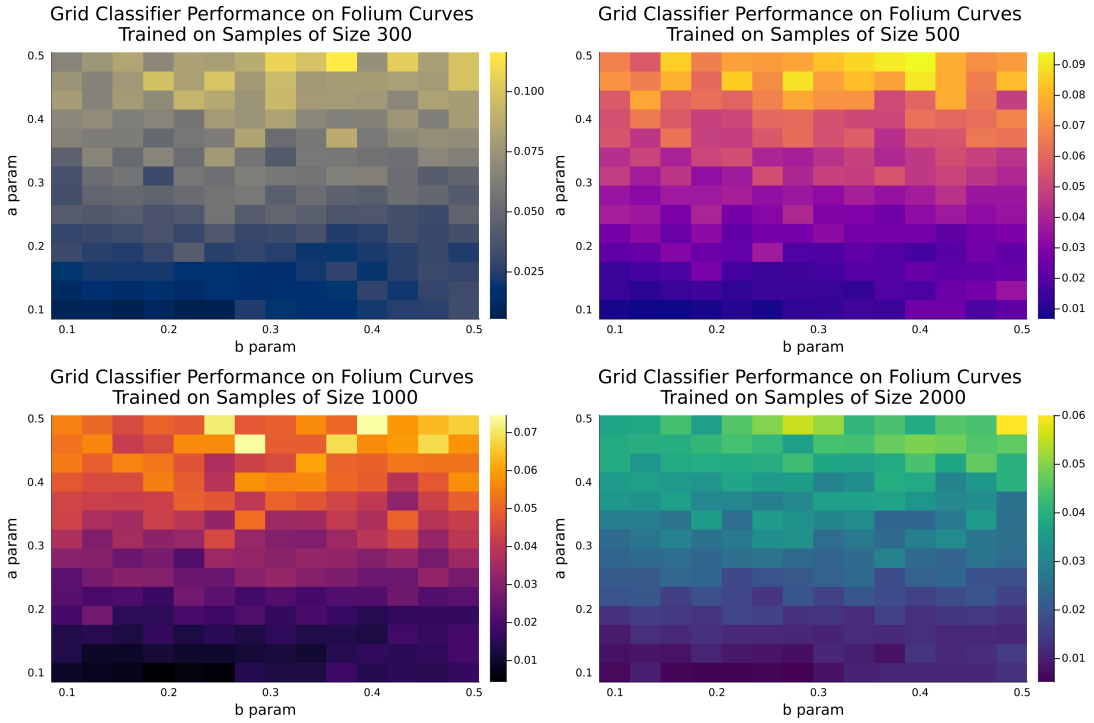


Figure 5.15: Grid Classifier Risk Maps

Notice that even with a sample as small as 300 we seem to be consistently hitting $\geq 90\%$ accuracy. Moreover, this holds over a wide variety of different folium curves, implying that grid classifiers have the potential to be very robust. Of course, this is all empirical and we are only testing grid classifiers on data whose marginal distribution is uniform over the learning space—which brings us to questions 3) and 4).

5.5.1 The VC Dimension of Grid Classifiers

The VC dimension of $\mathcal{F}_{Grid}$ is infinite, but we do not run an ERM algorithm on $\mathcal{F}_{Grid}$ directly. Rather, we first pick a border, and then a number of cells. Hence, the real question that we should be asking is whether a hypothesis class $\mathcal{H}_{B,n}$ of some fixed border and number of cells has a finite VC dimension. Clearly, if we have n divisions along the border, then there are n^2 cells. Each of those cells can classify an instance as either positive or negative. Thus, $\mathcal{H}_{B,n}$ shatters sets of size $|C| = n^2$. By default, every instance outside of the grid is, and always will be, labeled negatively. Thus, we cannot shatter sets with size $|C| > n^2$, implying $VC(\mathcal{H}_{B,n}) < \infty$.

Does this imply grid classifiers are agnostic learners? It is not clear. We are hesitant to say yes. Estimating the border is achieved by searching through the space of axis-aligned rectangles, which is a provably realizable class[10]. The issue is that axis-aligned rectangles, though similar to what we are calling borders is slightly different. Namely, it is assumed in the proof that the true labeling function is an axis-aligned rectangle.

Still, there is hope that searching for a border can be done with arbitrarily strong precision. Assuming that we can indeed get a realizable border, that is a border that perfectly captures the extreme points of the loop, all that remains is to make arguments about whether estimating the appropriate number of cells is agnostically learnable. If we can show that border estimate and determining cell numbers is agnostically learnable, then yes, grid classifiers are agnostic classifiers, and quite powerful ones at that.

Grid classifiers are dynamic. That is, we have choice as to how expressive we wish to make grid classifiers by picking a larger or smaller number of cells. This seems a lot like a realizable learner, but there is a catch. In order to get arbitrarily fine approximations, we need to have abtirarily many cells. Clearly, if we let $n \to \infty$ then $\mathcal{H}_{B,n}$ no longer has a finite VC dimension.

5.5.2 Higher Dimensions, Nonuniform Data, and Multiclass Classification, Oh My!

My hat is tipped to those that got the *Wizard of Oz* reference. Back to learning theory. Answering question 4) is a bit trickier than questions 1) through 3). We begin with higher

dimensions. The idea of grid classifiers is inspired by measure theory[8]. In measure theory, we cover sets on the real line with collections of intervals and this idea is readily extended to n dimensional space. So while we probably can extend grid learners to higher dimensions in theory, in practice we will run into memory problems. The grid matrix requires $O(n^d)$ space where d is the dimension of the learning space and n is the number of divisions along each dimension.

It is not clear whether grid classifiers are robust enough to handle data whose marginal distribution is nonuniform over the learning space. One might hope that we can gimmick our way around nonuniform data. That is, if there exists pockets of space in which the data is unlikely to be sampled, then perhaps we do not mind mislabeling those pockets. Whether this thinking actually works is not something we have determined, but we suspect it is subject to failure when we allow for unbounded learning spaces. Why? If the learning space is unbounded, then there might be distributions with infinitely many "pockets" that are unlikely to be sampled individually, but as a whole make up a significant portion of the marginal distribution.

The final question is that of multiclass classification. Notice that folium curves are generally comprised of multiple loops. That is, folium curves generally do not obey Definition 2.2 (single loops), yet grid classifiers did not have any problem learning the structure of a folium curve. Imagine that we took one of the pedals of the folium curve and assigned it a new class so that our labels are no longer binary. In this case, cells would be assigned such that the label of the cell is given by the instance that is most frequently populated in the cell. In this way, grid classifiers may be able to learn multiclass data, but we remain skeptical about how well this idea might generalize.

Multiclass learning is considerably more difficult than binary learning and has only recently seen major theoretical breakthroughs[1,3]. Moreover, the breakthroughs in multiclass learning use transductive learning[4] and it is still unknown whether transductive learning is substantially harder than PAC learning in the agnostic setting. Hence, though we are excited about the potential of grid classifiers, we suspect that a more rigorous analysis would lead to theoretical shortcomings. This leaves us with one of the major questions we wished to explore in this thesis. To what degree is a theoretical limitation meaningful in practice?

Through grid classifiers we have constructed a nonlinear classifier that shows promising

potential to learn one of the major categories in our partition of nonlinear classification. And yes, while we do suspect that there exist some distributions for which grid learners might fail, is that necessarily reason enough to discard them entirely? Maybe. Maybe not. Suppose that in theory there exists distributions that cause grid classifiers to fail. Do said theoretically challenging distributions exist frequently enough out in the wild for practitioners to care about their existence? If they are frequent, then we are back to square one. If they are not, we have made progress. In the future, we hope to collaborate with applied scientists and begin to answer these questions. For now, we conclude what has been a long, long expedition through learning theoretic ideas. Thank you for your time, and in case I do not see you, good afternoon, good evening, and good night.

References

[1] Asilis, J., Devic, S., Dughmi, S., Sharan, V., & Teng, S.-H. (2024). Open problem: Can local regularization learn all multiclass problems? In S. Agrawal & A. Roth (Eds.), *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research* (pp. 5301–5305).: PMLR.

[2] Barnett, J. H., Adams, P., Matthews, G., Sibley, T., & Dray, T. (1999). Fifty famous curves. Accessed: 2025-04-16.

[3] Brukhim, N., Carmon, D., Dinur, I., Moran, S., & Yehudayoff, A. (2022). A Characterization of Multiclass Learnability . In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)* (pp. 943–955). Los Alamitos, CA, USA: IEEE Computer Society.

[4] Dughmi, S., Kalayci, Y., & York, G. (2024). Is transductive learning equivalent to pac learning? *arXiv preprint arXiv:2405.05190v2*. Accessed: 2025-04-17.

[5] Jkasd (2008). Knot table. Accessed: 2025-04-15. Licensed under CC BY-SA 3.0 Unported.

[6] Munkres, J. R. (2000). *Topology*. Prentice Hall, Inc., 2 edition.

[7] Rice, J. A. (2006). *Mathematical Statistics and Data Analysis.* Belmont, CA: Duxbury Press., third edition.

[8] Royden, H. L. & Fitzpatrick, P. (2018 - 2010). *Real analysis / H.L. Royden, Stanford University, P.M. Fitzpatrick, University of Maryland, College Park.* Pearson modern classic. New York, NY: Pearson, fourth edition [2018 reissue]. edition.

[9] Schapire, R. E. (2008). Princeton's cos 511: Theoretical machine learning. https://www.cs.princeton.edu/courses/archive/spr08/cos511/schedule.html. Accessed: 2025-04-15.

[10] Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press.

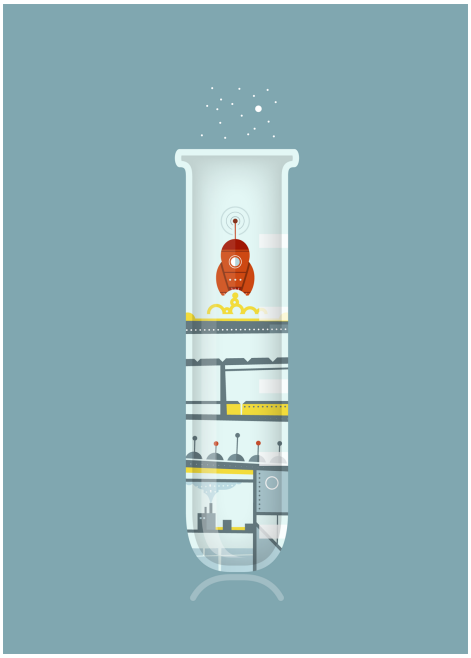
[11] Sharan, V. (2021). University of southern california theory of machine learning. https://vatsalsharan.github.io/fall21.html. Accessed: 2025-04-15.

[12] Tttrung (2005). Klein bottle. Image licensed under the GNU Free Documentation License. Accessed: 2025-04-16.

[13] Valiant, L. G. (1984). A theory of the learnable. *Commun. ACM*, 27(11), 1134–1142.

[14] Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5), 988–999.

[15] Wolpert, D. & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.



This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, *Science Experiment 02*, was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive AGPL license, and can be found online at github.com/suchow/Dissertate or from its lead author, Jordan Suchow, at suchow@post.harvard.edu.