Applying Rademacher-Like Bounds to Combinatorial Samples and Function Selection

Clayton Sanford

Senior Thesis

May 6, 2018

Abstract

This thesis examines measures of sample complexity as a means to evaluate the effectiveness and reliability of machine learning algorithms. We give an overview of VC-dimension, Rademacher complexity, and variations of those measures. We introduce the Cartesian Empirical Maximum Discrepancy (CEMD) framework, which applies Rademacher-like bounds to learning functions over combinatorial data. We use this method to evaluate algorithms for removing noise from audio files. We also introduce a progressive sampling algorithm that incorporates Rademacher complexity to the task of function selection, and we test that method on the problem of selecting between audio compression formats.

Research Collaborators: Cyrus Cousins, Eli Upfal

Thesis Committee: Eli Upfal, Katherine Kinnaird

Contents

1 Introduction and Motivation

Machine learning (ML) is one of the fastest growing fields both in computer science research and in the technology industry. Machine learning algorithms are rapidly improving in terms of accuracy, runtime, and generality, and those algorithms are applied to a broader range of applications. In particular, advances in deep learning have dramatically improved the accuracy and applicability of machine learning methods. Many of these methods achieve empirically impressive results, but they lack guarantees about the quality of those results. As ML is applied to domains with higher stakes — such as piloting self-driving cars, diagnosing medical conditions, and predicting the likelihood of an accident or illness for insurance — the need for statistical confidence in these an algorithm's predictions intensifies. In this thesis, we explore and expand upon theoretical bounds for how well ML algorithms perform on unknown data with the goal of better understanding the successes and limitations of ML algorithms.

Rigorous analysis of machine learning algorithms provides helpful probabilistic bounds that guarantee how likely an algorithm's success will generalize to a new set of samples. However, those bounds often require strong assumptions about the training and testing data. We examine a core assumption used in most machine learning theory: the requirement that training and testing samples are identically and independently distributed (i.i.d.). In particular, we extend a framework that analyzes the complexity of machine learning hypotheses for i.i.d. to samples to also be able to analyze *combinatorial data*.

A combinatorial sample (which is more rigorously defined later) is a data point that that consists of tuples of certain components. That is, a combinatorial data point can be defined solely by its components, and those components can be mixed and matched to create new data points. Rather than the data points being chosen i.i.d., the components are sampled i.i.d., and they are matched together in tuples to form combinatorial samples. For example, in a hypothetical app that predicts whether two individuals are romantically compatible, an algorithm would assign a compatibility score to each pair of people. The pair would be a combinatorial sample with each person as one of its two components. It is combinatorial because one of the people could be swapped out with another, and the new pair would still be a valid data point for compatibility.

Under traditional methods of analysis that bound the how well a compatibility algorithm generalizes to new data, each sample (or potential match) must be sampled i.i.d. at random. This means that each individual is unlikely to be drawn in more than one match. The hypothetical app creators would need to choose each pair independently from previous pairs, which means that the people they choose are unlikely to be considered for more than one pairing. In practical applications, this limits the effectiveness of using generalization bounds: often availability of data is the limiting factor, and samples must be created by recombining components of other data points. Our app creators can't obtain enough participants in their study to only consider each person

with only one other person, and they need to evaluate multiple pairings for each individual in order to have enough pairings without requiring too many participants. In this case, applying theory to real-world problems would be easier if we could relax the requirement that the combined samples are drawn i.i.d. to simply require that each component is drawn i.i.d. Then, we could apply bounds to algorithms which combine data in these ways, and provide better assurances about the quality of these results.

In this thesis, we discuss how data-dependent uniform convergence bounds can be expanded to the combinatorial domain and apply those concepts to algorithm selection problems in the audio domain. We introduce necessary background material in learning theory in Section 2. In Section 3 we introduce the Cartesian domain for learning problems and expand Rademacher-like bounds to that domain. We explore the audio denoising problem in Section 4 and discuss an architecture for applying our theoretical work to choosing an audio denoiser. We apply more standard Rademacher complexity bounds to the task of codec selection in Section 5 and give a sampling-based approach for determining the best function for a given encoding task. Finally, we examine open questions and inspiration for future work in Section 6.

2 Preliminaries

In order to discuss bounds on generalization for certain non-i.i.d. random variables, we first introduce several key measurements of learning complexity. In Section 2.1 we introduce Probably Approximately Correct (PAC) learning, the theoretical foundation of machine learning that defines what makes a concept "learnable." In Sections 2.2 and 2.3, we introduce the VC-Dimension and Rademacher Complexity, which are prominent measurements of learning complexity. Then, we discuss the Maximum Discrepancy in Section 2.4, a Rademacher-like bound that forms the basis of our framework. In Section 2.5, we give an overview of the Rademacher Chaos Complexity, which is a preexisting adaptation of Rademacher-like bounds to certain kinds of combinatorial data.

2.1 PAC-Learning

In machine learning, theoretical results underlie many practical advances in the field. Many past notable advances in the field couple impressive empirical performance with rigorous mathematical bounds on how well those algorithms perform. However, recent advances in deep learning have produced astounding results with little theoretical backing, so it is especially important now to continue advancing the theoretical side of machine learning to match the application-oriented side. In order to approach the task of bounding generalization of machine learning algorithms, we need a unified framework for considering the success of an algorithm. The most meaningful bounds for learning algorithms focus on *uniform convergence*, where all points of a function are guaranteed

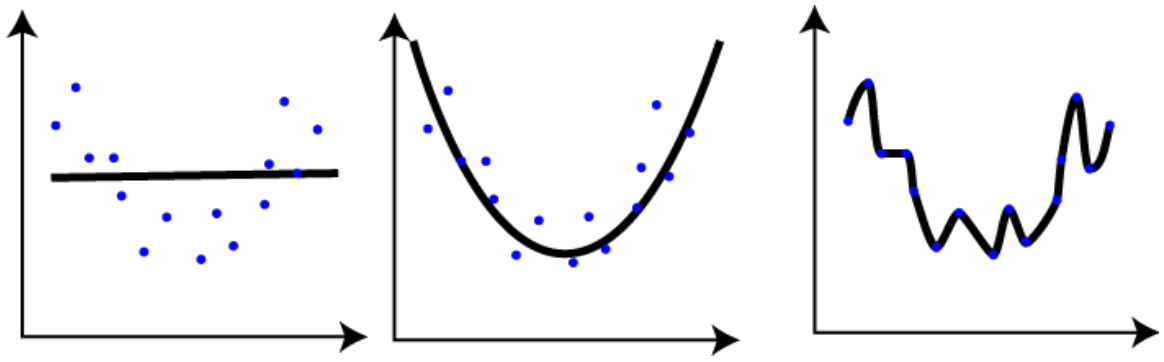

Figure 1: Trade-offs with the expressiveness of hypothesis classes in regression, from Johnson [2013].

to converge to some mean with at least a certain rate of convergence. These motivate the PAC-learning framework, which bounds the probability and severity of failure to converge to a correct hypothesis.

Valiant [1984] establishes a working definition of learnability by machines in his pivotal paper, "Theory of the Learnable." He uses the idea of a hypothesis class $\mathcal{H}$ to represent all possible hypotheses or decision-making processes that can be determined by the algorithm. For example, $\mathcal{H}$ for the perceptron algorithm is the set of linear separators passing through the origin. For linear regression algorithms with a set of basis functions, $\mathcal{H}$ contains all possible linear combinations of the basis function.

Definition 2.1. *A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$ that for any distribution $\mathcal{D}$ of samples, given $\epsilon, \delta > 0$ and access to poly$(\frac{1}{\epsilon}, \frac{1}{\delta})$ i.i.d. random examples from $\mathcal{D}$ with labels, with probability at least $1 - \delta$ returns a hypothesis $h \in \mathcal{H}$ with error (or probability of mis-classification) at most ϵ.*

In other words, given sufficiently many examples (depending on the desired ϵ and δ), there is no greater than δ probability that the samples were atypical enough to mislead the algorithm into producing a hypothesis with error greater than ϵ. The δ captures the "probably," and the ϵ captures the "approximately."

PAC-learning is the foundation upon which the rest of theoretical machine learning was built. PAC-learning provides a language for formal bounds. However, the PAC-learning framework is restrictive because it requires that any distribution of samples must be learnable. In this thesis, we examine data-dependent bounds that relax this requirement and allows us to formally assess learning algorithms in other contexts.

2.2 VC-Dimension

One of the most important problems that theoretical ML evaluates is how likely a model is to overfit to its data. *Overfitting* occurs when a ML algorithm creates a model that performs well on the given training data but fails to generalize to new testing samples. More *expressive* or *complex* ML models can fit more complicated trends in the data; however, they are more prone to overfitting. This means that complicated models require larger amounts of data to avoid overfitting. For example, support vector machines with radial basis kernels can perfectly fit any dataset (when the kernels are sufficiently narrow), but they often learn arbitrary patterns that do not generalize to new data. On the other hand, linear separators rarely perform much worse on testing data than training data, but they can only fit the most simple patterns in a dataset. By measuring the expressiveness of a ML model, we can predict the tendency of a model to overfit.

Figure 1 illustrates the trade-offs of using hypothesis classes of different complexities. The left-most panel uses an inexpressive hypothesis class that fails to learn the most interesting trends of the data. The center panel is just right: it learns the trend of the data without overfitting to the noise. The right-most panel uses too expressive of a hypothesis class for the data because it overfits to the data, and learns trends that do not generalize well to more data points.

Measurements like VC-Dimension, proposed by (and named after) Chervonenkis and Vapnik [1971], are used to measure the complexity of a hypothesis class. In general, a more complex hypothesis class learns hypotheses that can be more tailored to the data. This makes complex or expressive hypothesis classes more prone to overfitting. Measurements of learning complexity can be used to derive generalization bounds, which bound the difference between training and testing errors.

The VC-Dimension is based on combinatorial properties of a hypothesis class in the classification domain. It measures the expressiveness of the hypothesis class. To explain VC-Dimension, we first define several terms.

Definition 2.2. *A hypothesis class $\mathcal{H}$, with $h : \mathcal{X} \to \{-1, +1\}$ for $h \in \mathcal{H}$, **shatters** a set of n points $S \in \mathcal{X}^n$ if $\forall S' \subseteq S$, $\exists h \in \mathcal{H}$ such that:*

$$h(s) = \begin{cases} +1 & s \in S' \\ -1 & s \in S \setminus S' \end{cases}$$

That is, a hypothesis class $\mathcal{H}$ shatters a set of points if any labeling of those points has a hypothesis in H that correctly labels each point. We incorporate this definition into our definition of VC-Dimension.

Definition 2.3. *The **VC-Dimension** of a hypothesis class $\mathcal{H}$, $\mathrm{VC}(\mathcal{H})$ is the size of the largest set that can be shattered by H. That is:*

$$\mathrm{VC}(\mathcal{H}) = \max_d \{d \in \mathbb{N} : \ \exists S \in \mathcal{X}^d \text{ s.t. } \mathcal{H} \text{ shatters } S\}$$

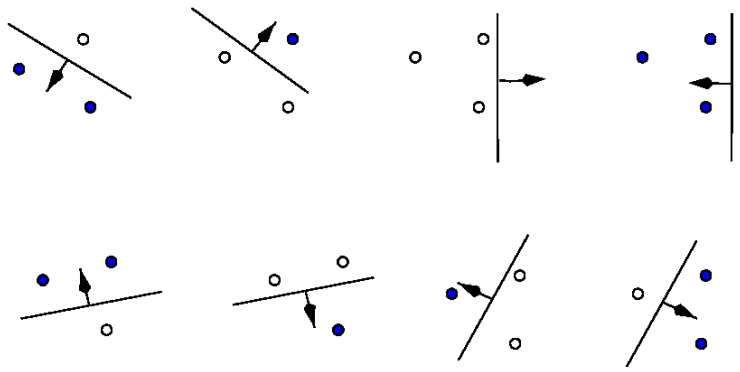

Figure 2: Linear separators in $\mathbb{R}^2$ shattering three points, from Burges [1998]

For example, Figure 2 illustrates why the VC dimension of 2-dimensional linear separators is 3. Linear separators in $\mathbb{R}^2$ can shatter a set of 3 points because there exists a linear separator that fits any labeling for these three points. The VC dimension of 2-dimensional linear separators is 3 because no four points in $\mathbb{R}^2$ can be shattered by them.

Based on the combinatorial properties of VC-Dimension, the generalization error of a hypothesis can be bounded. The *generalization error* represents how well a learned hypothesis generalizes to examples it has not seen yet, which is represented as the difference between the average loss on training examples and the expected or true loss on new examples. Because VC-Dimension is applied to the classification domain here, we use a *0/1 loss function* ℓ, which evaluates whether the algorithm correctly classified a given sample: $\ell(h(x), y) = \mathbb{1}\{h(x) \neq y\}$, for classification hypothesis $h \in \mathcal{H}$, example $x \in \mathcal{X}$, and label $y \in \{-1, +1\}$.

Theorem 2.1. *Let $\mathcal{H}$ be a hypothesis class. Let h^* represent the true hypothesis we try to learn. For any classification algorithm obtaining $h \in \mathcal{H}$ trained on m labeled samples, $\{(z_1, h^*(z_1)), \ldots, (z_m, h^*(z_m))\}$ drawn from distribution $\mathcal{D}$, the following holds with probability at least $1 - \delta$ for $\delta \in (0, 1)$:*

$$\underbrace{\mathop{\mathbb{E}}_{\mathcal{D}}\left[\ell(h(z), h^*(z))\right]}_{\textit{true loss}} - \underbrace{\frac{1}{m}\sum_{i=1}^{m}\ell(h(z_i), h^*(z_i))}_{\textit{training loss}} \leq \sqrt{\frac{1}{m}\left[\mathrm{VC}(\mathcal{H})\left(\log\left(\frac{2m}{\mathrm{VC}(\mathcal{H})}\right)+1\right) - \log\left(\frac{\delta}{4}\right)\right]}$$

The VC-dimension succeeds in quantifying the complexity of a hypothesis class and relating it to overfitting and generalization error. However, these combinatorial bounds are *distribution-free* — they ignore qualities of the distribution of the data — which leads to loose bounds. Alternatively, the Rademacher complexity is *distribution-dependent* and can therefore create more tight bounds.

2.3 Rademacher Complexity

Rademacher Complexity (RC) is a measure of the expressiveness of a function class. As introduced by Bartlett and Mendelson [2002], Rademacher Complexity takes the distribution of samples into account, allowing for tighter bounds on generalization than those based only on combinatorial properties of the hypothesis class, like those of VC-dimension.

For example, Rademacher Complexity outperforms VC-dimension when data are concentrated on a linear subspace and we wish to label the data with a linear classifier. VC bounds are dependent on the high-dimensional domain where the data *could* lie, while Rademacher Complexity bounds are dependent on the lower-dimensional subspace. By ignoring the distribution of the data, VC-dimension bounds are worse because they ignore the problem-specific information that the distribution conveys.

RC measures the ability of a hypothesis class to correlate to randomly labeled samples. By correlate, we mean that we find the hypothesis that makes the fewest number of mistakes when attempting to fit a labeled dataset. To do so, i.i.d. Rademacher random variables σ_i denote labels of -1 or $+1$ with equal probability for each example i. For each assignment of labels, the hypothesis that best correlates to the labels is selected.

We first define the Empirical Rademacher Complexity using the notation of Mitzenmacher and Upfal [2017], which measures the ability to correlate to a fixed set of samples.

Definition 2.4. *The **Empirical Rademacher Complexity** (ERC) for sample $S = \{z_1, \ldots, z_m\}$ over a hypothesis class $\mathcal{H}$ such that $h : S \to \mathbb{R}$ for $h \in \mathcal{H}$ is:*

$$\tilde{\mathfrak{R}}_m(\mathcal{H}, S) = \mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(z_i)\right]$$

where $\sigma \in \{-1, +1\}^m$ is a vector of random variables chosen i.i.d. and uniformly.

While the previous definition assesses the power of a function class to correlate to some sample of data, it still fails to take the distribution of that sample into account. To do so, we define the Rademacher Complexity, which computes the expectation over a distribution.

Definition 2.5. *The **Rademacher Complexity** (RC) over a hypothesis class $\mathcal{H}$ for distribution $\mathcal{D}$ is:*

$$\mathfrak{R}_m(\mathcal{H}, \mathcal{D}) = \mathop{\mathbb{E}}_{S\leftarrow\mathcal{D}^m}\left[\tilde{\mathfrak{R}}(\mathcal{H}, S)\right] = \mathop{\mathbb{E}}_{S\leftarrow\mathcal{D}^m}\left[\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(z_i)\right]\right]$$

where $S = \{z_1, \ldots, z_m\}$ is a set of m samples chosen i.i.d. from $\mathcal{D}$ and where $\sigma \in \{-1, +1\}^m$ is a vector of random variables chosen i.i.d. and uniformly.

From these definitions, tighter bounds than those induced by VC-dimension are induced. Below, we give probabilistic bounds on the generalization error based on both the ERC and the RC.

Theorem 2.2. *Let $\mathcal{H}$ be a set of functions representing the errors of hypotheses such that $h : \mathcal{X} \to [0, 1]$, $\forall h \in \mathcal{H}$, where $\mathcal{X}$ represents the feature space. Let $S = \{z_1, \ldots, z_m\}$ be a sample of $\mathcal{X}$ drawn from distribution $\mathcal{D}$. Then, the following each hold $\forall h \in \mathcal{H}$ and $\forall \delta \in (0, 1)$ with probability at least $1 - \delta$:*

$$\underbrace{\mathop{\mathbb{E}}_{\mathcal{D}}[h(z)]}_{\textit{true loss}} - \underbrace{\frac{1}{m}\sum_{i=1}^{m} h(z_i)}_{\textit{training loss}} \leq 2\tilde{\mathfrak{R}}_m(\mathcal{H}, S) + 3\sqrt{\frac{\ln(1/\delta)}{m}} \tag{1}$$

$$\mathop{\mathbb{E}}_{\mathcal{D}}[h(z)] - \frac{1}{m}\sum_{i=1}^{m} h(z_i) \leq 2\mathfrak{R}_m(\mathcal{H}, \mathcal{D}) + \sqrt{\frac{\ln(1/\delta)}{m}} \tag{2}$$

While these bounds are much tighter than those provided by VC-dimension, they are difficult to compute given the exponential number of terms that must be summed over and given the difficulty of finding the supremum from hypothesis class of potentially infinite size. To alleviate this, Massart's Lemma [Massart, 2000] bounds the ERC for finite-sized hypothesis classes, making it easier to use those generalization bounds.[1]

Lemma 2.1. ***[Massart's Lemma]*** *Assume that $|\mathcal{H}|$ is finite for hypothesis class $\mathcal{H}$. Given a sample $S = \{z_1, \ldots, z_m\}$, let:*

$$B = \max_{h \in \mathcal{H}} \sqrt{\sum_{i=1}^{m} f(z_i)^2}$$

Then:

$$\tilde{\mathfrak{R}}_m(\mathcal{H}, S) \leq \frac{B\sqrt{2 \ln |\mathcal{H}|}}{m}$$

For the generalization bound in Theorem 2.2 to hold, we require a hypothesis class of functions that output values on the interval $[0, 1]$; however, for these bounds to be useful, they must to be applicable in learning problems where completely different kinds of hypotheses are learned. The Contraction Lemma makes that possible by bounding the Rademacher Complexity when certain kinds of functions are applied to the hypothesis class [Shalev-Shwartz and Ben-David, 2014].

[1] Although we do not discuss those techniques in this thesis, Massart's Lemma can effectively be used to bound the ERC of infinite hypothesis classes that satisfy certain characteristics through the use of covering numbers.

Lemma 2.2. ***[Contraction Lemma]*** *If $\phi : \mathbb{R} \to \mathbb{R}$ is a ρ-Lipschitz function (that is, if $|\phi(x) - \phi(y)| \leq \rho|x - y|$), then:*

$$\mathfrak{R}_m(\phi \circ \mathcal{H}, \mathcal{D}) \leq \phi\mathfrak{R}_m(\mathcal{H}, \mathcal{D})$$

These results can be used to relate the Rademacher Complexities over more complicated learning problems by choosing a Lipschitz loss function and bounding the RC of that function applied to the hypothesis class.

2.4 Maximum Discrepancy

Maximum Discrepancy (MD) and Empirical Maximum Discrepancy (EMD) offer similar measures of complexity to Rademacher Complexity and Empirical Rademacher Complexity, which were also developed by Bartlett et al. [2002]. While Maximum Discrepancy bounds are less well-known than Rademacher Complexity, MD bounds better extend to our generalization from i.i.d. samples to combinatorial samples.

Maximum Discrepancy is calculated in a similar manner as Rademacher Complexity. They differ in that MD is given two sets S and S' of predetermined sizes, one that has $+1$ weights and one that has -1 weights. Both aim to find the function that best fits the weights assigned to the samples.

Definition 2.6. *The* ***Empirical Maximum Discrepancy*** *(EMD) for samples $S = \{z_1, \ldots, z_m\}$ and $S' = \{z'_1, \ldots, z'_m\}$ over a hypothesis class $\mathcal{H}$ such that $h : S \to \mathbb{R}$ for $h \in \mathcal{H}$ is:*

$$\tilde{\mathfrak{D}}_m(\mathcal{H}, S, S') = \sup_{h \in \mathcal{H}} \frac{1}{m} \left[\sum_{i=1}^{m} h(z_i) - \sum_{i=1}^{m} h(z'_i) \right]$$

We extend the empirical definition by choosing the sets from the distribution.

Definition 2.7. *The* ***Maximum Discrepancy*** *(MD) over a hypothesis class $\mathcal{H}$ for distribution $\mathcal{D}$ is:*

$$\mathfrak{D}_m(\mathcal{H}, \mathcal{D}) = \mathop{\mathbb{E}}_{S,S' \leftarrow \mathcal{D}^m} \left[\tilde{\mathfrak{D}}_m(\mathcal{H}, S, S') \right] = \mathop{\mathbb{E}}_{S,S' \leftarrow \mathcal{D}^m} \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \left[\sum_{i=1}^{m} h(z_i) - \sum_{i=1}^{m} h(z'_i) \right] \right]$$

where $S = \{z_1, \ldots, z_m\}$ and $S' = \{z'_1, \ldots, z'_m\}$ are sets of m samples chosen i.i.d. from $\mathcal{D}$.

Similar generalization bounds to those of Rademacher Complexity exist for the EMD as well. Unlike with Rademacher Complexity, we require that the number of samples is $2m$ to ensure that we can partition the samples into two equally-large sets of size m for the EMD computation. While this is not a substantial burden for EMD bounds, when we give a similar bound for our framework, this doubling plays a larger role.

Theorem 2.3. *Let $\mathcal{H}$ be a set of functions representing the errors of hypotheses such that $h : \mathcal{X} \to [0, 1]$, $\forall h \in \mathcal{H}$, where $\mathcal{X}$ represents the feature space. Let $S = \{z_1, \ldots, z_m\}$ and $S' = \{z_{m+1}, \ldots, z_{2m}\}$ be samples of $\mathcal{X}$ drawn from distribution $\mathcal{D}$. Then, the following each hold $\forall h \in \mathcal{H}$ and $\forall \delta \in (0, 1)$ with probability at least $1 - \delta$:*

$$\underbrace{\mathop{\mathbb{E}}_{\mathcal{D}}[h(z)]}_{\textit{true loss}} - \underbrace{\frac{1}{2m}\sum_{i=1}^{2m} h(z_i)}_{\textit{training loss}} \leq 2\tilde{\mathfrak{D}}_m(\mathcal{F}, S, S') + \frac{3}{2}\sqrt{\frac{\ln(1/\delta)}{m}} \tag{3}$$

EMD is easier to compute than ERC it does not need to take an expectation over values of σ. However, EMD is harder to approximate because it has no analogue to Massart's Lemma [2].

2.5 Rademacher Chaos Complexity

De la Peña and Giné [1999] measure the complexity of algorithms that learns from combinatorial data with the Rademacher Chaos Complexity (RCC). RCC analyzes learning algorithms whose inputs consist of unordered k-combinations of examples drawn i.i.d.

RCC can be used to bound the difference between training error and true error on algorithms that predict compatibility between a combination of elements. For example, one could predict the romantic compatibility between pairs of people, where both potential partners come from the same dating pool. RCC also lends itself well to graphical problems: each edge can be represented as a pair of vertices, so samples of edges can be expressed naturally as combined data. It can be used for link prediction and to predict edges in a graph.

While this model succeeds in expanding Rademacher-like bounds to some non-i.i.d. data sources, it fails to cover cases where the tuples of samples we learn on come from different sets: our Cartesian EMD framework addresses those issues. In addition, RCC also requires strict properties of the hypothesis class: notably, the labels output by every hypothesis must have an expected value of 0 for any given distribution over examples. These functions must still be centered around 0 in expectation given one component of combinatorial data, which is unusual to find. For this reason, it fails to find many practical applications in machine learning.

[2] Maximum Discrepancy, however, *does* have its own version of Massart's Lemma.

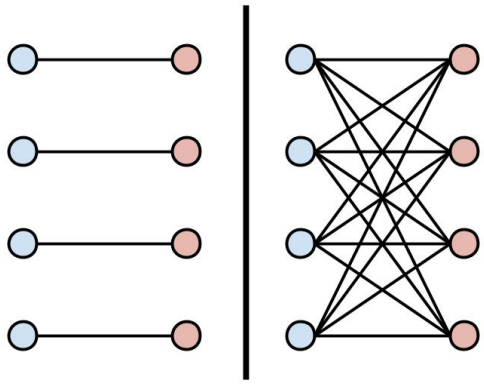

Figure 3: A comparison of combinatorial pairs drawn i.i.d. versus Cartesian-i.i.d.

3 Cartesian EMD

The VC-dimension, Rademacher Complexity, and Maximum Discrepancy bounds previously described all assume that samples used to train the learning algorithms are drawn i.i.d. Cousins [2018] introduces the Cartesian Empirical Maximum Discrepancy (CEMD) framework, which allows us to analyze algorithms that train on combinatorial data with i.i.d. components. This section introduces the model and discusses key theoretical results that are applied in the following section.

3.1 Cartesian Data

Cartesian data are samples which can be represented as a combination of distinct components. Those components are drawn separately from different distributions and combined to form a combinatorial sample.[3] Our model considers Cartesian data such that the components of each point are drawn i.i.d., but that the combined point may not be. We define what this means rigorously.

Definition 3.1. *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be domains for each sample component with respective distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$. For $1 \leq i \leq k$ and for some $m_i \in \mathbb{Z}^+$, let S_i be a set of m_i samples in $\mathcal{X}_i$ drawn i.i.d. from $\mathcal{D}_i$. Then $S \subseteq S_1 \times \cdots \times S_k$ is a* ***Cartesian-i.i.d. sample*** *of $\mathcal{X}_1 \times \cdots \times \mathcal{X}_k$. S is a* ***complete Cartesian-i.i.d. sample*** *if $S = S_1 \times \cdots \times S_k$, which we denote as $S \leftarrow \mathcal{D}^{\boldsymbol{m}}$, where $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_k$ and $\boldsymbol{m} = (m_1, \ldots, m_k)$.* [4]

[3] The words *combinatorial* and *Cartesian* tend to have similar meanings here. In general, we use Cartesian to refer to data points where components are drawn from different sources and combinatorial to describe any data points that can be represented as combinations of any components.

[4] $S \leftarrow \mathcal{D}^{\mathbf{m}}$ means that we sample m_i elements in $S_i \leftarrow \mathcal{D}_i$ for each $i \in \{1, \ldots, k\}$ and let $S = S_1 \times \cdots \times S_k$.

For $k = 2$, combinatorial data can be thought of intuitively as edges of a bipartite graph. The nodes in each partition represent data points of each component, and an edge linking a node from each partition represents the combination of those components. If the sample is complete, then the analogous graph is a complete bipartite graph.

Figure 3 demonstrates how Cartesian-i.i.d. combinatorial data allows for many more samples to be produced from the same components than when the samples are drawn i.i.d. for $k = 2$. For combinatorial data composed of two different components, this contrasts which data points are used if samples are drawn i.i.d. or Cartesian-i.i.d.. In the left panel, samples are drawn i.i.d, so each sample requires two new components to be drawn i.i.d. On the right side, the samples are Cartesian-i.i.d because the individual components are drawn i.i.d, and all combinations of them are considered.

Studying the generalization of algorithms that are trained on Cartesian-i.i.d data allows for many learning problems on combinatorial data to be bounded. Rademacher Chaos Complexity allows placing bounds on algorithms that use samples that combine components drawn from the same set (that is, $\mathcal{X}_1 = \mathcal{X}_2 = \cdots = \mathcal{X}_k$). Cartesian EMD further expands the data domain where learning can occur to include combinatorial data whose components are drawn from distinct sets. Some examples of these domains are given below:

- In the denoising domain, algorithms are trained to take noisy audio, video, or image files as input and output a cleaned file with the noise removed. Noisy files are often constructed by combining the corresponding clean file with artificially-generated or recorded noise. To analyze this learning problem using Rademacher complexity, the clean file and the noise file that comprise each sample must *together* be drawn i.i.d.; that means that no clean or noise file can be used in more than one training data point. By our analysis with the CEMD, we can draw samples Cartesian-i.i.d. and thus reuse our clean and noise data. We discuss this application in depth in the next section.

- Algorithms which estimate compatibility among tuples of data are easily analyzed with combinatorial data as input. That is, given a tuple of data points, the algorithm returns a real number estimating how compatible the two points are. For example, trying to predict the romantic potential of couples in a heterosexual or two-party dating context could have input data modeled as a pair of two individual data components sourced from separate distributions. To minimize the total number of people involved in the training of the algorithm, each person can be involved in multiple pairs that are all analyzed, and the generalization of the algorithm can be bounded with CEMD. (For a homosexual or single-party dating context, the pair are drawn from the same set, and Rademacher Chaos Complexity is a more appropriate measure of complexity.)

- In computer vision, training datasets can be amplified in size by applying randomly chosen transformations to the image, such as translations, reflections, and

added noise. All resulting images are added to the training set. Krizhevsky et al. [2012] used this to increase the amount of data in the pivotal AlexNet paper that applied neural networks to the task of image recognition. Given the vast quantity of data needed for accuracy in the visual domain, these techniques allow algorithms to operate on more data than would be feasible to obtain individually and label. This can be modeled as combinatorial data problem where input images are the combination of an original image and a transformation.

3.2 CEMD Definition

The Cartesian EMD is a generalization of the EMD by Cousins [2018] to the combinatorial domain, so that input data points can be Cartesian-i.i.d. rather than entirely i.i.d. We define it rigorously:

Definition 3.2. *Let $S, S' \subset \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$ be complete Cartesian-i.i.d. samples. Let $\mathcal{H}$ be a hypothesis class with $h : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathbb{R}$ for $h \in \mathcal{H}$. The* ***Cartesian Empirical Maximum Discrepancy*** *(CEMD) for S and S' over $\mathcal{H}$ is:*

$$\tilde{\mathfrak{C}}(\mathcal{H}, S, S') = \sup_{h \in \mathcal{H}} \left[\frac{1}{|S|} \sum_{z \in S} h(z) - \frac{1}{|S'|} \sum_{z' \in S'} h(z') \right]$$

Like the EMD and ERC, we can also define a non-empirical version of the complexity that obtains the samples from a distribution.

Definition 3.3. *Let $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{Z}^{+k}$ represent the number of each component sampled for two different samples from distribution $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_k$. Let $\mathcal{H}$ be a hypothesis class with $h : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathbb{R}$ for $h \in \mathcal{H}$. The* ***Cartesian Maximum Discrepancy*** *(CMD) for $\boldsymbol{m}$ and $\boldsymbol{m}'$ over $\mathcal{H}$ is:*

$$\begin{aligned} \mathfrak{C}_{\boldsymbol{m},\boldsymbol{m}'}(\mathcal{H}, \mathcal{D}) &= \mathop{\mathbb{E}}_{S \leftarrow \mathcal{D}^{\boldsymbol{m}}, S' \leftarrow \mathcal{D}^{\boldsymbol{m}'}} \left[\tilde{\mathfrak{C}}(\mathcal{H}, S, S') \right] \\ &= \mathop{\mathbb{E}}_{S \leftarrow \mathcal{D}^{\boldsymbol{m}}, S' \leftarrow \mathcal{D}^{\boldsymbol{m}'}} \left[\sup_{h \in \mathcal{H}} \left[\frac{1}{|S|} \sum_{z \in S} h(z) - \frac{1}{|S'|} \sum_{z' \in S'} h(z') \right] \right] \end{aligned}$$

3.3 Generalization Error Bounds

Like other methods of measuring hypothesis complexity, CEMD was defined with the purpose of obtaining a probabilistic generalization error bound.

Theorem 3.1. *Let $\mathcal{H}$ be a function class representing errors of hypotheses such that $h : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathbb{R}$ for $h \in \mathcal{H}$. For some vector $\boldsymbol{m}$ representing the number of samples*

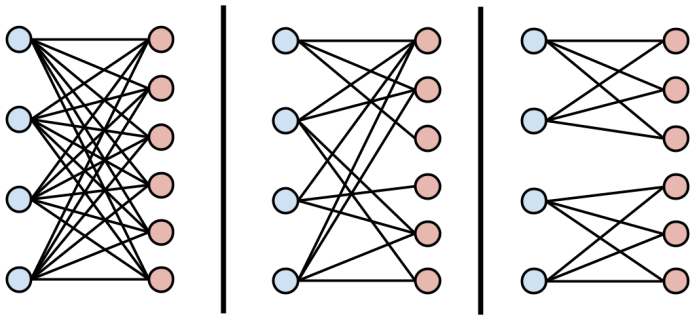

Figure 4: A comparison of Cartesian data points. From left to right, the graphs represent (1) a completely Cartesian sample, (2) a partial sample, and (3) a block sample with $c_1 = 2$ and $c_2 = 3$.

drawn from each k components, let $S \leftarrow \mathcal{D}^{\mathbf{2m}}$ be a complete Cartesian-i.i.d. sample. Let $S' \subset S$ contain only the k-tuples of components from the first k_i components sampled for each $1 \leq i \leq k$, and $S'' \subset S$ contain only those second k_i components sampled. The following holds with probability at least $1 - \delta$ for any $\delta \in (0, 1)$.

$$\sup_{h \in \mathcal{H}} \left[\underbrace{\mathop{\mathbb{E}}_{\mathcal{D}}[h(z)]}_{\textit{true error}} - \underbrace{\frac{1}{|S|} \sum_{z \in S} h(z)}_{\textit{training error}} \right] \leq \tilde{\mathfrak{C}}(\mathcal{H}, S', S'') + \frac{3}{2} \sqrt{\ln\left(\frac{1}{\delta}\right) \sum_{i=1}^{k} m_i^{-1}}$$

Note that $|S| = 2^k |S'| = 2^k |S''|$. While this bound is similar in to the one presented for the regular EMD, the requirement that we use 2^k times more combinations for training than we use for CEMD approximation reduces the tightness of our bound more substantially.

3.4 Partial and Block Sampling

The CEMD framework above governs samples drawn completely Cartesian-i.i.d. While that assumption applies in cases where the data contain values for all possible tuples of components, some circumstances call for combining some elements of each component with some of other components, to amplify the data without obtaining an uncontrollable amount. We refer to choosing samples that are Cartesian-i.i.d. but *not* completely-Cartesian-i.i.d. as *partial sampling*. *Block sampling* occurs when the samples all occur in certain clusters of fixed size. The differences between these types of samples are illustrated in Figure 4. We introduce the Generalized CEMD to handle these cases, with a very similar definition to the CEMD.

Definition 3.4. *Let $S, S' \subset \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$ be Cartesian-i.i.d. samples (that are not necessarily complete). Let $\mathcal{H}$ be a hypothesis class with $h : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathbb{R}$ for $h \in \mathcal{H}$. The* ***Generalized Cartesian Empirical Maximum Discrepancy*** *(GCEMD)Cousins [2018] for S and S' over $\mathcal{H}$ is:*

$$\tilde{\mathfrak{G}}(\mathcal{H}, S, S') = \sup_{h \in \mathcal{H}} \left[\frac{1}{|S|} \sum_{z \in S} h(z) - \frac{1}{|S'|} \sum_{z' \in S'} h(z') \right]$$

We can obtain a generalization bound for partial sampling based on this quantity in certain cases. We require that the two sets S' and S'' used to compute the GCEMD are *symmetrical*. For ease of notation, let $(j_1, \ldots, j_k) \in S$ if the tuple containing the j_ith element of component i for each component i is in S. For the two sets to be symmetrical, we insist that a tuple corresponding to indices $(j_1, \ldots, j_k) \in S'$ for $1 \leq j_i \leq m_i$ if and only if the tuple corresponding to $(j_1 + m_1, \ldots, j_k + m_k) \in S''$. In the bipartite graph analogy for $k = 2$, that means that S is composed of two completely separate bipartite graphs that are *isomorphic*, or have the exact same structure.

Theorem 3.2. *Let $\mathcal{H}$ be a function class representing errors of hypotheses such that $h : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathbb{R}$ for $h \in \mathcal{H}$. For some vector $\boldsymbol{m}$ representing the number of samples drawn from each k components, let $S \leftarrow \mathcal{D}^{\boldsymbol{2m}}$ be a complete Cartesian-i.i.d. sample. Let $S' \subset S$ contain only the k-tuples of components from the first m_i components sampled for each $1 \leq i \leq k$, and $S'' \subset S$ contain only those second m_i components sampled. The following holds with probability at least $1 - \delta$ for any $\delta \in (0, 1)$.*

$$\sup_{h \in \mathcal{H}} \left[\underbrace{\mathop{\mathbb{E}}_{\mathcal{D}} [h(z)]}_{\text{true error}} - \underbrace{\frac{1}{|S|} \sum_{z \in S} h(z)}_{\text{training error}} \right] \leq \tilde{\mathfrak{G}}(\mathcal{H}, S', S'') + \frac{3}{2} \sqrt{\ln\left(\frac{1}{\delta}\right) \sum_{i=1}^{k} \sum_{p=1}^{m_i} \left(\frac{|\{x \in S : x_i = p\}|}{|S|} \right)^2}$$

In the case of this generalized form when we use block sampling, this theorem gives a specialized result. A partial sampling is a *block sampling* when we divide Cartesian samples into n disjoint groups based on their components. That is, we can partition elements of each component i into sets of size some $c_i = \frac{m_i}{n}$. Then, we match a single set of each component to form a *block* and take all possible combinations of elements within those sets. Therefore, each block consists of $\prod_{i=1}^{k} c_i$ total combinatorial samples. Note that $|S| = n \prod_{i=1}^{k} c_i$. Thus:

$$|\{x \in S : x_i = p\}| = \frac{\prod_{\ell=1}^{k} c_\ell}{c_i} = \frac{|S|}{m_i}$$

We can therefore simplify the inequality from Theorem 3.2 for this case to obtain the identical bound as in Theorem 3.1:

$$\sup_{h \in \mathcal{H}} \left[\mathop{\mathbb{E}}_{\mathcal{D}} [h(z)] - \frac{1}{|S|} \sum_{z \in S} h(z) \right] \leq \tilde{\mathfrak{G}}(\mathcal{H}, S', S'') + \frac{3}{2} \sqrt{\ln\left(\frac{1}{\delta}\right) \sum_{i=1}^{k} m_i^{-1}}$$

4 Audio Denoising Experiments

To test the ability of our Cartesian EMD to impose tight generalization bounds on learning problems trained on combinatorial samples, we apply the CEMD to the problem of audio denoising. An audio denoising algorithm takes as input an audio file with added noise and attempts to remove the noise from the file. In a learning context, training data consists of a noisy audio file and a cleaned version of the file, which acts as a "label" that represents the correct output of the algorithm.

For Rademacher-based analysis, we must assume that all clean and noisy files used for training are drawn i.i.d. The noise used to corrupt each clean file is unlikely to be reused within a training sample. In a practical domain, this may not be feasible: training effective denoisers requires vast amounts of data, and reusing clean samples and added noise in different combinations may be necessary to obtain sufficient data. In this case, it makes sense to model it as a learning problem over combinatorial data, where one component is the clean audio and the other is the added noise. Therefore, creating bounds on learning audio denoising algorithms is a natural application of the CEMD.

4.1 Audio Denoising Domain

In this section, we introduce and define the audio denoising problem. Audio denoising has already been explored extensively within electrical engineering and computer science. While researchers previously approached audio denoising from a signal processing background, the recent successes of neural nets have led to a new wave of audio denoising algorithms that outperform older algorithms[Rethage et al., 2018]. Because numerous specialists have created audio denoising algorithms already, our audio denoising problem aims to choose the best of the preexisting algorithms for a given set of data rather than to create a new algorithm from scratch.

While audio files can be corrupted in numerous ways, we focus solely on the effects of additive noise in this thesis. Let $\mathcal{A}_C$, $\mathcal{A}_N$, and $\mathcal{A}_A$ represent the sets of clean audio samples, noisy audio samples, and audio noise respectively. Any audio samples can be thought of as a sequence of numbers in $[0, 1]$ representing the tonal pressures that are present at each time step. Audio files can be constructed by additively merging two files, which we denote as $z_1 + z_2$ for $z_1, z_2 \in \mathcal{A}_C \cup \mathcal{A}_N \cup \mathcal{A}_A$. Because noisy samples are constructed from clean samples and additive noise, for each $z_N \in \mathcal{A}_N$, there exists $z_C \in \mathcal{A}_C, z_A \in \mathcal{A}_A$ such that $z_N = z_C + z_A$. An audio denoising algorithm is some $\phi : \mathcal{A}_N \to \mathcal{A}_C$. It can be equivalently expressed in terms of the additive noise rather than the noisy input file as $\phi' : \mathcal{A}_C \times \mathcal{A}_A \to \mathcal{A}_C$ where $\phi'(z_C, z_A) = \phi(z_C + z_A)$.

In order to measure the success of a denoising algorithm on a noisy file, let $\mathcal{E} : \mathcal{A}_C \times \mathcal{A}_C \to [0, 1]$ measure the error between a file cleaned by the algorithm and

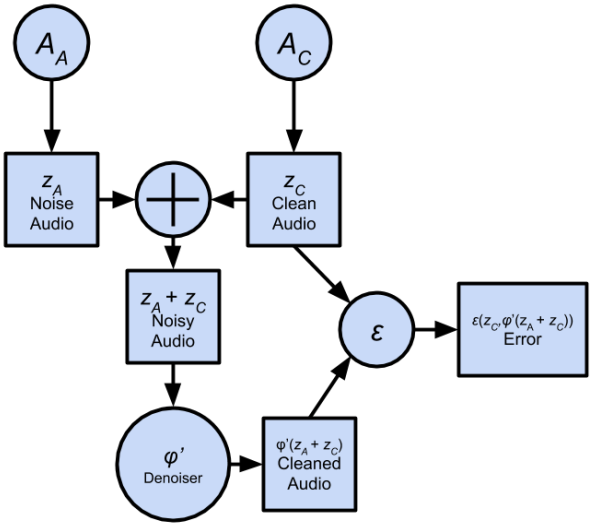

Figure 5: A visualization of the audio denoising process.

the true clean file we compare it to. To learn over hypothesis classes that can be input to Rademacher-like bounds, we learn from hypotheses $h : \mathcal{A}_C \times \mathcal{A}_A \to [0, 1]$ where $h(z_C, z_A) = \mathcal{E}(z_C, \phi'(z_C, z_A))$, which represents the difference between the cleaned version of the combined sample and the true clean version when denoising function ϕ is used. Figure 5 illustrates the audio denoising process in a more visual manner.

For a finite set $\mathcal{H} = \{h_1, \ldots, h_\ell\}$ representing the losses corresponding corresponding to ℓ denoising algorithms $\{\phi'_1, \ldots, \phi'_\ell\}$, we learn a weighted denoising model. We train the model with a training set S of pairs with $S \subseteq \mathcal{A}_C \times \mathcal{A}_A$. A simple model could be trained to select among the denoising algorithms the single algorithm we expect to perform best. However, there are several other methods to learn a better denoising function under the CEMD framework.

4.1.1 Learning a Weighted Denoising Model

One way to learn a denoising function given several functions to begin with is to learn an optimal combination of the solutions returned by several denoising algorithms. Our weighted algorithm returns the convex combination of the outputs of different denoising algorithms to optimally denoise a set of data. That is, we learn $h^* \in \text{conv}(\mathcal{H})$[5]

[5]$\text{conv}(\mathcal{H})$ represents the convex hull of a finite set $\mathcal{H} = \{h_1, \ldots, h_\ell\}$. That is:

$$\text{conv}(\mathcal{H}) = \{\sum_{i=1}^{\ell} a_i h_i : \forall i,\ a_i > 0 \text{ and } \sum_{i=1}^{\ell} a_i = 1\}$$

with corresponding $\phi^{*\prime}$ such that $\phi^{*\prime}(z_C, z_A) = \sum_{j=1}^{\ell} w_j^* \phi_j'(z_C, z_A)$ for $w^* \in [0,1]^\ell$ and $\sum_{j=1}^{\ell} w_i^* = 1$ that minimizes:

$$\sum_{(z_C, z_A) \in S} h^*(z_C, z_A) = \sum_{(z_C, z_A) \in S} \mathcal{E}(z_C, \phi^{*\prime}(z_C, z_A))$$

This is also equivalent to learning the vector w^* that minimizes that quantity.

Initially, we use a normalized mean squared error for $\mathcal{E}$. Because each audio file can be expressed in the time domain as a sequence of numbers in $[0, 1]$, such an error can be easily computed over the sequences. Moreover, minimizing the sum of mean squared errors means that we can find w^* using a convex quadratic program.

We can use this error of the output over the different algorithms as our hypothesis class for CEMD analysis. We discuss this in more detail in Section 4.2.

4.1.2 Perceptual Audio Models

While a normalized square error is mathematically convenient and sensible for comparing two sequences of arbitrary data, other measures of distance are better tailored to audio data. Because audio files are almost always intended for human listening, we can redefine success for denoising algorithms to be how clean an audio file sounds to a human listener rather than how close the audio vector is to the cleaned audio file. To measure that, we explore *perceptual audio models*, which attempt to measure how clearly humans hear different sounds.

Perceptual audio models quantify how much a human listener can notice changes to different audio frequencies. *Psychoacoustics* is a field that examines how individuals perceives sound, and perceptual audio models applies knowledge of pyschoacoustics to process audio files without compromising human perception of the sound. A central idea in perceptual models is *noise masking*, which occurs when the clean audio signals dominate the added noise from the algorithm in terms of human perception; that is, listeners only hear the certain strong frequencies that drown out other similar frequencies [Jayant et al., 1993]. Some perceptual models are *subjective*, meaning that they measure audio quality based on human ratings of sound quality, where listeners assess how much a modified signal resembles the reference signals. Others, like PEAQ (perceptual evaluation of audio quality) are *objective*, and are based more directly on psychoacoustic principles without relying on human-supplied data[Thiede et al., 2000].

Perceptual audio models are involved in designing compression algorithms that avoid making sacrifices to one's listening experience. Because MP3 is a lossy compression scheme, an audio file can be tightly compressed, but some data is permanently lost in the process. The creators of the MP3 scheme designed the encoding algorithm to remove as much audio data as possible without significantly compromising a user's listening experience [Library of Congress, 2017]. To do so, the MP3 algorithm determines which

signals are masked by other signals and safely removes the data needed to store those inaudible signals.

In future work, we plan on incorporating these perceptual techniques as a weighted mean squared error between frequency vectors. That way, we can more choose denoising functions that maximize audio quality for human observers.

4.2 Experimental Architecture

To assess the usability of our Cartesian EMD bounds, we compare the success of our algorithm when trained over two different types of samples. First, we train our weighted denoising algorithm over samples drawn i.i.d. and bound the generalization of the algorithm with Empirical Rademacher Complexity bounds. Then, we train the algorithm on samples drawn Cartesian-i.i.d. and bound generalization with CEMD bounds. We then compare the empirical errors and the theoretical bounds of the two by measuring how well each generalized and for how tight the bounds were.

When we compute the ERC and CEMD for generalization bounds, finding the supremum over the $\text{conv}(\mathcal{H})$ seems intractable because the number of convex combinations is are infinite. However, it can be shown that $\tilde{\mathfrak{R}}(\mathcal{H}, S) = \tilde{\mathfrak{R}}(\text{conv}(\mathcal{H}), S)$ and $\tilde{\mathfrak{C}}(\mathcal{H}, S, S') = \tilde{\mathfrak{C}}(\text{conv}(\mathcal{H}), S, S')$. Thus, it suffices to find $\tilde{\mathfrak{R}}(\mathcal{H}, S)$ and $\tilde{\mathfrak{C}}(\text{conv}(\mathcal{H}), S, S')$, whose suprema can be found by simply iterating over the finite number of functions in the function class[Shalev-Shwartz and Ben-David, 2014].

We coded the project in Java.[6] We acquired audio denoising algorithms primarily from code published online from well-regarded publications on audio denoising. Here are the algorithms we used:

- Rethage et al. [2018] created *A Wavenet for Speech Denoising*, which included a pre-trained neural network trained on speech data with added observed and artificial noise.
- *RNNoise* is a simple recurrent neural network framework produced by Mozilla. Its source code provides a light-weight pre-trained neural network implemented in Python. A publication is forthcoming.
- *NoNoise* is a noise-removal tool on Github that does not use neural networks and relies more on signal processing techniques.

In order to choose the weighted algorithm that best denoises the data, we use IBM's CPLEX to solve the quadratic program (QP) induced by the minimization of the normalized mean squared error of w^*.

[6]The code repository for this project, which includes code for audio denoising and codec selection, is at `https://github.com/chsanford/AudioEMD`.

5 Codec Selection Experiments

In addition to applying distribution-dependent uniform convergence bounds to the problem of audio denoising, we also use those bounds to tackle the similar problem of audio compression. To do so, we wish to determine which *codec* — a schema that converts a file to compressed format and back — best fulfills certain criteria, such as the amount of time needed to compress the file, how much smaller the compressed file is than the original, and the fidelity of the decompressed file's contents to those of the original file. To ensure statistical significance in the codec we select, we use Rademacher bounds.

In the *function selection* domain, a meta-algorithm aims to select a function to complete a task to maximize some objective value over a wide range of input data. Meta-algorithms in this domain can eliminate or prune proposed algorithms if the results so far lead to high confidence that the algorithm either fails the constraints or performs worse than another algorithm that meets the constraints. Being able to place tight bounds on the objective value of the candidate functions means that the performance of functions and their satisfaction of the constraints can be proven more quickly, which leads to a more efficient meta-algorithm. We see this as an appropriate domain to combine Rademacher bounds with sampling strategies to determine which candidate algorithm is best. In particular, we examine the codec selection domain, where functions encode and decode files to efficiently compress them without losing a significant amount of data.

5.1 Codec Selection Domain

The codec selection problem aims to choose an encoding scheme for a domain like audio, video, or images that meets certain criteria and maximizes an objective dependent on those criteria. For this project, we examined the audio domain and created a sampling procedure that determines which audio compression scheme performs best in terms of factors like the size of the compressed file and the similarity between the original file and the decompressed version. Unlike the previous experiment, we use audio data without added noise — the goal is simply to compress and decompress clean audio files.

Gupta and Roughgarden [2017] examined this kind of problem from a PAC-learning approach. They bounded the performance of different algorithms using bounds based on pseudo-dimension, which applies the concepts of VC-dimension to regression rather than classification problems. We built upon their model by replacing their pseudo-dimension bounds with Rademacher bounds in order to take the distribution of data into account to tighten the bounds.

Rather than simply finding a function that outperforms others with high confidence, we further expanded their model by also seeking functions that must meet certain constraints. For example, in the audio domain, we might seek a compression algorithm

that minimizes the amount of memory needed for the compression file while requiring that a compressed and decompressed file is sufficiently similar to the original file.

For our model, the encoding schemes are represented as a class of functions $\mathcal{H}$ with $h : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ is the set of input objects to encode and $\mathcal{Y}$ is a representation of the encoding along with useful meta-data about the transformation. We also have a set of criteria which describe the performance of the function $\mathcal{C}$ with $c : \mathcal{Y} \to [0, 1]$ for $c \in \mathcal{C}$ that assess the performance of a function in some aspect. In the audio domain, $\mathcal{H}$ contains compression schemes like MP3 and Ogg while $\mathcal{C}$ contains measurements like the ratio of the size of the original file to the size of the compressed file. We measure the success of an encoding with an objective $V : \mathcal{Y} \to \mathbb{R}$ such that $V(y)$ is a linear combination of $c(y)$ for $c \in \mathcal{C}$. We also seek functions that satisfy some set convex combination of linear inequality constraints in terms of the criteria: let $\mathcal{W} \subseteq \mathcal{X}$ be the region of the function's domain such that those constraints are satisfied. Thus, we seek $h^* \in \mathcal{H}$ such that with high probability for x drawn from $\mathcal{X}$ over some distribution:

$$h^*(x) = \arg\max_{h \in \mathcal{H}} \{V(h(x)) : x \in \mathcal{W}\}$$

We use empirical Rademacher complexity in the next section to discuss a progressive sampling algorithm that prunes functions in $\mathcal{H}$ that with high probability perform poorly by the objective or are not in the valid constraint region.

5.2 Progressive Sampling with Pruning

Our Progressive Sampling with Pruning (PSP) procedure samples inputs to the codec function uniformly at random in batches of increasing size. It is based on the progressive sampling method introduced by Riondato and Upfal. Riondato and Upfal [2015] Our technique differs in that we use the results of the progressive sampling to eliminate from consideration, or prune, functions whose results will be insufficient with high confidence. Based on the results of each batch, we empirically estimate the means of $c(h(x))$ for criterion c and hypothesis h over samples x. We bound the means with high probability with Rademacher complexity and prune functions which (1) with high confidence lie outside of $\mathcal{W}$ or (2) with high confidence has a smaller mean value of $V(h(x))$ than $V(h'(x))$ for some other f' that is in $\mathcal{W}$ with high confidence.

The algorithm is parameterized by δ and ϵ, whose meanings are analogous to those in PAC-Learning framework (Section 2.1). δ represents our level of certainty in each bound. That is, we require that each bound holds with probability $1 - \delta$. ϵ represents how bounded we expect our results to be prior to making a conclusion. That is, if the argument terminates, with probability $1 - \delta$, we can identify the best codec function, and that function's objective value varies by at most ϵ.

We repeat this process over batches which double in size after each iteration. Because each batch has more samples than the previous one, it can place tighter bounds than

the preceding ones can, thus allowing it to be more confident in its empirical means. We then disqualify more functions because we conclude with confidence that certain functions outperform other functions due to the shrinking confidence intervals. This means that fewer functions need to be run on each subsequent batch of inputs, which limits the number of unnecessary encoding steps. The algorithm terminates when there is exactly one remaining function that satisfies the constraints, when it is shown that no function satisfies the constraints, or when no more samples are available. Therefore, the algorithm finds the function with the smallest objective subject to the constraints with high confidence. We give pseudo-code for the algorithm in an attached figure:

Algorithm 1 PSP$(S, s_0, \mathcal{H}, \mathcal{C}, V, \mathcal{W}, \epsilon, \delta)$

Input: samples S, initial batch size s_0, hypothesis class $\mathcal{H}$, criterion set $\mathcal{C}$, objective $\mathcal{V}$, constraint space $\mathcal{W}$, confidence $\epsilon \in \{0, 1\}$, failure probability $\delta \in \{0, 1\}$

Output: optimal hypothesis $\hat{h} \in \mathcal{H}$, empirical criteria estimates$\hat{e}_{\hat{h},c}$, criteria confidence intervals $\hat{E}_{\hat{h},c}$, upper objective bound u

$\hat{E}_{h,c} := [0, 1]$ for all $h \in \mathcal{H}, c \in \mathcal{C}$

$n := \lfloor \log_2(\frac{|S|}{s_0} + 1) \rfloor$ [maximum number of iterations]

for $i \in \{0, \ldots, n-1\}$ **do**

  Let S_i be $2^i \cdot s_0$ unused samples from S

  for $c \in \mathcal{C}, h \in \mathcal{H}$ **do** [current batch]

    $\hat{e}_{h,c} := \frac{1}{|S_i|} \sum_{x \in S_i} c(h(x))$

    $\hat{E}_{h,c} := \hat{E}_{h,c} \cap [\hat{e}_{h,c} - 2\tilde{\mathfrak{R}}(c \circ \mathcal{H}, S_i) - 3\sqrt{\ln(2n|\mathcal{C}|/\delta)/2|S_i|}, \hat{e}_{h,c} + 2\tilde{\mathfrak{R}}(c \circ \mathcal{H}, S_i) + 3\sqrt{\ln(2n|\mathcal{C}|/\delta)/2|S_i|}]$

  Let $\hat{h} := \max\{h \in \mathcal{H} : \hat{E}_{h,\cdot} \subseteq \mathcal{W}\}$ [best hypothesis so far]

  for $h \in \mathcal{H}$ **do**

    if $\hat{E}_{h,\cdot} \cap \mathcal{W} = 0$ or $\max(V(\hat{E}_{h,\cdot})) \leq \min(V(\hat{E}_{\hat{h},\cdot}))$ **then**

      remove h from $\mathcal{H}$

  if $\mathcal{H} = \emptyset$ **then**

    error No valid h satisfies constraints $\mathcal{W}$

  else if $|\mathcal{H}| = 1$ or $\min(V(\hat{E}_{\hat{h},\cdot})) \geq \max h \in \mathcal{H}(V(\hat{E}_{h,\cdot}))$ **then**

    $u := \max_{h \in \mathcal{H}} V(\hat{E}_{h,\cdot})$) **return** $(\hat{h}, \hat{e}_{\hat{h},\cdot}, \hat{E}_{\hat{h},\cdot}, u)$

error Insufficient Sample size for $\delta - \epsilon$ guarantee

Based on the algorithm, we obtain theoretical guarantees about the usefulness of the results.

Theorem 5.1. *Suppose we run PSP$(S, s_0, \mathcal{H}, \mathcal{C}, V, \mathcal{W}, \epsilon, \delta)$ and obtain $(\hat{h}, \hat{e}_{\hat{h},\cdot}, \hat{E}_{\hat{h},\cdot}, u)$. Then, the following always holds:*

1. *The confidence rectangle of the criteria for $\hat{h}$ lies within the constraint space: $\hat{E}_{\hat{h},\cdot} \subseteq \mathcal{W}$.*

The following hold with probability $1 - \delta$*:*

2. *The true mean of each criteria for a given function lies within our confidence rectangle:*

$$\mathop{\mathbb{E}}_{x}\left[c(\hat{h}(x))\right] \in \hat{E}_{\hat{h},c} \ \forall c \in \mathcal{C}$$

3. *The objective of the true mean lies within our confidence interval for the objective:*

$$\mathop{\mathbb{E}}_{x}\left[V(\boldsymbol{c}(\hat{h}(x)))\right] \in V(\hat{E}_{\hat{h},\cdot})$$

   for $\boldsymbol{c}$ *as a vector of all criteria functions in* $\mathcal{C}$*.*

4. *The expected objective lies within a confidence is no less than* ϵ *worse than optimal and is no better than the upper bound:*

$$\mathop{\mathbb{E}}_{x}\left[V(\boldsymbol{c}(\hat{h}((x)))\right] \in [\mathop{\mathbb{E}}_{x}[V(\boldsymbol{c}(h^*((x)))] - \epsilon, u]$$

   for h^* *being the optimal hypothesis that satisfies the constraints, defined as:*

$$h^* = \arg\max_{h \in \mathcal{H}} \{\mathop{\mathbb{E}}_{x}[V(\boldsymbol{c}(h^*((x)))] : \mathop{\mathbb{E}}_{x}[c(h^*(x))] \in \hat{E}_{h^*,c} \ \forall c \in \mathcal{C}\}$$

5.3 Experimental Results

We coded the Progressive Sampling with Pruning algorithm in Java. Our codebase is sufficiently general to apply the PSP algorithm to a wide range of domains; one simply needs to implement our interfaces and abstract classes for Function, Criterion, and Sample to fit a desired domain.

We applied the algorithm to the codec selection domain. The function class $\mathcal{H}$ corresponds to compression algorithms. For this example, we choose between LAME MP3 encoders of different variable bit-rates — each one has a rating between V0 and V9 representing the how many bits are used to encode segments of audio [Hegemann et al., 2017]. V0 has the highest bit-rate, which means it features the highest quality sound yet also reduces the file size the least; V9 has the lowest bit-rate and thus has the lowest quality and the smallest output files.

We can also compare variable bit-rate (VBR) codec schema with constant bit-rate (CBR) and average bit-rate (ABR) schema. VBR schemes dynamically choose how many bits to compress, which tends to give the best results because more bits can be used to compress complex segments of audio than simple segments. CBR algorithms use the same amount of bits for each segment, which means that complex segments may lose key sounds, and simple segments may have too much redundancy; however, CBR

schema guarantees an exact compression ratio. ABR is a compromise between the two that aims for a certain compression ratio while allowing for some variance in bit-rate.

We considered several kinds of criteria for $\mathcal{C}$, which are meant to measure qualities of the compression algorithms that users would care about:

- *PEAQ Objective Difference* (c_1) measures the difference between two audio files as perceived by humans. This is an *objective* model because it is based on computational models of the ear rather than perception scores from individual tests. We discuss these models in Section 4.1.2. Because we ultimately care about whether humans can distinguish an audio file corrupted by compression from the original file, measurements that pay attention to human perception are especially important. We compute PEAQ using the GSTPEAQ codebase created by Holters and Zölzer [2015].

- *Root Mean Squared Error* (c_2) treats the two audio files as vectors of numbers in $[-1, 1]$ and computes a normalized L2-distance between the two. While this criterion is mathematically simple, it fails to account for the fact that differences between audio in certain frequencies may not be perceived by listeners.

- *Root Mean Squared Log Error* (c_3) takes the root mean squared error of the normalized logarithms of the elements of the audio vector. We logarithmically rescale the pulses of each pressure because we measure loudness for humans on a logarithmic scale: 60 decibels is 10 times louder than 40 decibels, which is in turn 10 times louder than 20 decibels. This criterion gives us an error metric that is slightly more linked to human perception of audio.

- *Compression Ratio* (c_4) represents the ratio of size of the compressed audio file to the size of the original audio file. Smaller values indicate that a compression algorithm is effective at reducing the size of a given audio file.

- *Compression Time* (c_5) is the time in seconds needed for the compression algorithm to compress the audio file. Because all criteria must output values in $[0, 1]$, we actually take the minimum of the compression time and 1 — this is a reasonable assumption because all compression schemes we have observed so far take much less time than one second.

- *Decompression Time* (c_6) is the same as Compression Time, except that it measures the amount of time needed to decompress the file back to WAV format.

From these criteria, we can construct an objective $\mathcal{V}$ that is a linear combination of these criteria. We decided that our objective should minimize the compression ratio and PEAQ difference with equal weights because accuracy of sound and amount of compression are key components for creating audio. That is, for a sample x and a

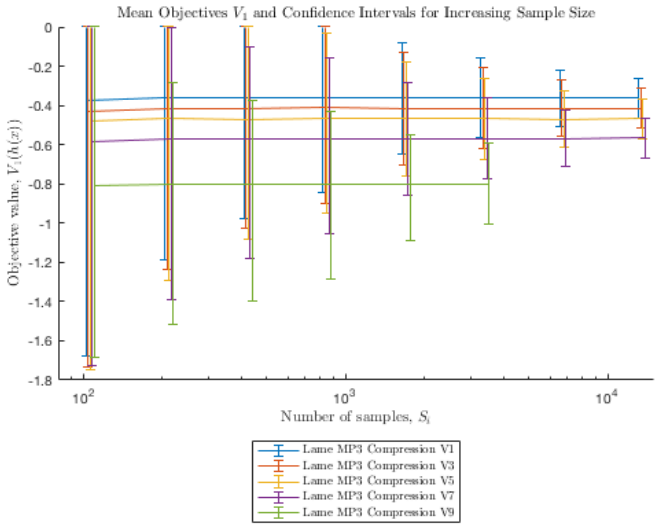

Figure 6: Bounds on the objective $V_1(x) = -c_1(h(x)) - c_4(h(x))$ for each step of the PSP algorithm for audiobooks.

function h, $V_1(h(x)) = -c_1(h(x)) - c_4(h(x))$.[7] However, this objective is just one way of quantifying the effectiveness of an audio compression algorithm. We also test two other variants: $V_2(h(x)) = -c_1(h(x))$ solely attempts to minimize PEAQ while $V_3(h(x)) = -c_4(h(x))$ only wants to minimize the compression ratio. Future users could modify this objective to be other combinations of criteria depending on what the listener values in a compression algorithm.

We also can implement constraints for our model with these criteria. For our application, we require that we each algorithm does not take too long to encode and decode samples. In this case, we want to only choose codec function $h \in \mathcal{H}$ if we are confident that compression and decompression each take no more than 0.5 seconds, or that $\hat{e}_{h,c_5} \leq 0.5$ and $\hat{e}_{h,c_6} \leq 0.5$. While these conditions are mostly trivial, they allowed us to test the constraint functionality, and we found that all of our compression algorithms satisfied this after several rounds.

For the tests, we let $\epsilon = 0.1$ and $\delta = 0.05$.

5.3.1 Tests on Audiobooks

To test the model, we ran it on ten-second clips from open source audio books on LibriVox[McGuire]. We collected over 25000 samples in that manner. When running the progressive sampling algorithm, we started by using 100 samples in the first round

[7] We flip the signs because our algorithm requires that we have an objective function to maximize.

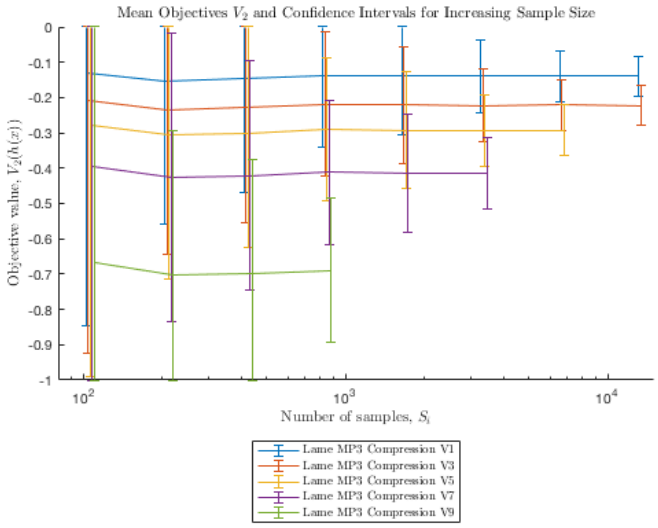

Figure 7: Bounds on the objective $V_2(x) = -c_1(h(x))$ for each step of the PSP algorithm for audiobooks.

and doubling that quantity until terminating after 12800 samples were used on the eighth round. Notably, our algorithm did not actually isolate the best encoding function — that would require more data. However, it succeeded in creating tight confidence intervals and in using those intervals to eliminate the worst function given our objective.

Figure 6 shows the empirical means and confidence intervals for the objective V_1 for each iteration and each compression scheme. While the means stay relatively stable, the bounds tighten with each iteration. Note that the V9 algorithm is eliminated after the seventh iteration because its confidence interval no longer intersects with the so-far optimal algorithm, V1. Given further iterations and more data, more algorithms could be eliminated as the bounds continue to tighten.

We can also visualize the bounds on our other proposed objectives. The difficulty of separating the performance of algorithms depends on the concentration of the objective values. In Figure 7, the empirical estimates of mean PEAQ divergence that comprise V_2 are very spread out, so the tightening bounds cease to overlap quickly. We can eliminate all but one suboptimal function this way. On the other hand, the estimates for V_3, which depends solely on compression ratio, in Figure 8 are more concentrated near zero and therefore fail to ever obtain tight enough intervals to separate functions.

5.3.2 Tests on Music

We also tested the algorithm on small segments pulled from music files. The music dataset contains the complete orchestral works of Debussy, conducted by Yan Pascal

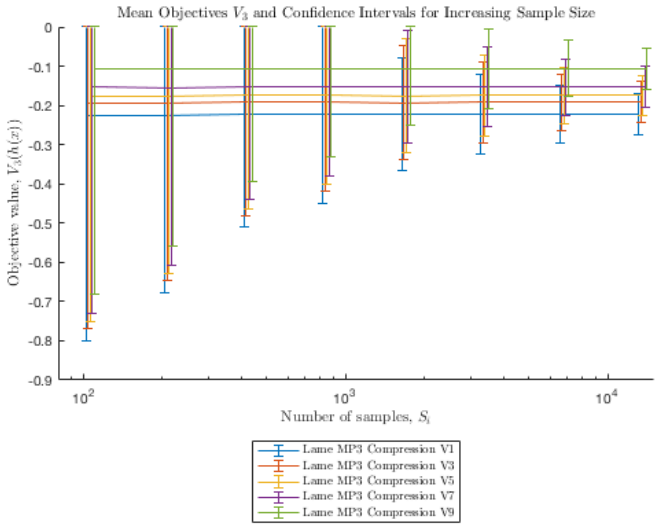

Figure 8: Bounds on the objective $V_3(x) = -c_4(h(x))$ for each step of the PSP algorithm for audiobooks.

Tortelier; the complete discography of hard rock band Led Zeppelin; and the 2000–2013 discography of progressive rock band Explosions in the Sky. We explored objectives over PEAQ divergence and compression ratios over a wider range of codecs; in addition to testing nine variants of VBR codecs, we also tested CBR codecs with fixed bit-rates of 320, 246, 128, and 64.

To visualize trends in these codecs, we compare PEAQ divergence and compression ratio in Figure 9. Each point represents those criteria values for a sample encoded with a given function. Note that we scale up the compression ratio for this experiment in order to increase the variance of the metric without compromising accuracy; because WAV files are encoded at a bit-rate of 750 and because the highest bit-rate encoded by an MP3 is 320, we scaled each compression ratio by $\frac{750}{320}$ to spread out the data more while keeping all ratios within the interval $[0, 1]$. We observe an inversely relationship with respect to compression ratio and PEAQ values as functions change. This makes sense because using more bits to encode a file leads to a smaller difference in sound between the original file and the decompressed file. Because they have fixed bit-rates, the CBR schemes have constant compression ratios, but vary widely in sound fidelity. The most accurate CBR schemes (256 and 320 bits) have zero perceived difference between the input and output files, but they have very high compression ratios.

In this more complicated case, we chose an objective that balances compression ratios and PEAQ divergences to choose a codec that effectively trades off between the two and does not simply maximize one of them. With the objective, $V_4(x) = -c_1(h(x)) - 2c_4(h(x))$, we can quickly eliminate codecs like the CBRs with bit-rates 320 and 256, as shown in Figure 10. The VBR encoding scheme V2 emerges as the scheme that best

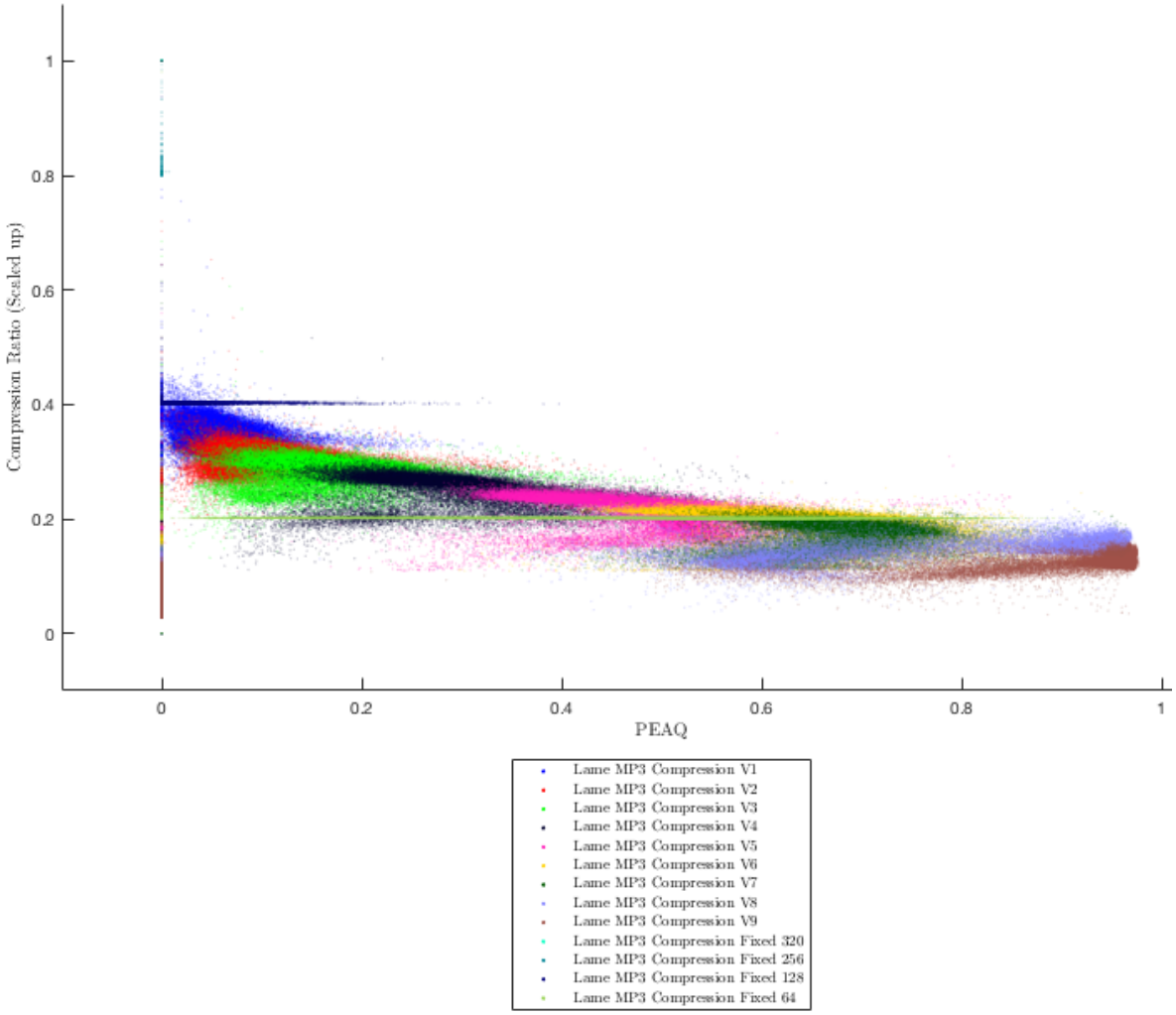

Figure 9: A scatter-plot of scaled compression ratios versus PEAQ divergences for music segments when encoded by a wide range of codecs.

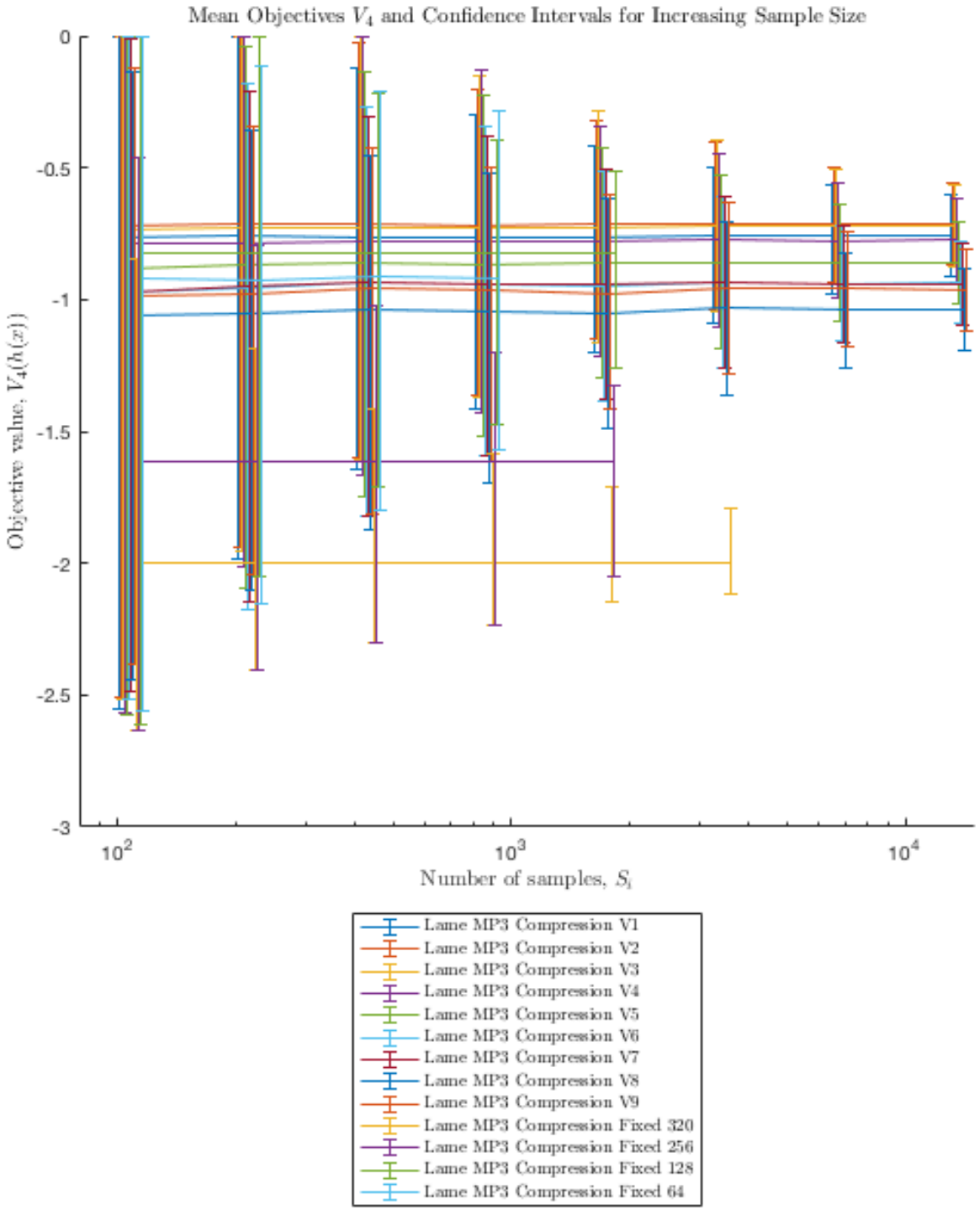

Figure 10: Bounds on the objective $V_4(x) = -c_1(h(x)) - 2c_4(h(x))$ for each step of the PSP algorithm for music.

trades off compression ratio and PEAQ, although it struggles to rule out other schemes because the trade off between the two criteria leads to similar objective values across a range of codecs.

For future experiments, we can use more data points to shrink the bounds further, to eliminate more compression algorithms, and to find the optimal encoding function with high certainty. We can additionally accelerate the rate of elimination by choosing more metrics like PEAQ that have significant differences in values for each compression function and incorporating them into the objective function. We also will test the audiobook data on a CBR codecs and compare those results to that of the music data on the balanced objective.

6 Conclusion and Open Questions

The Cartesian EMD framework we discuss allows rigorous generalization bounds to be applied to a wider range of machine learning problems. We relax the assumption that all samples must be chosen i.i.d, and create bounds that work for combinatorial data. Denoising problems are a domain that lends themselves well to this analysis. We also apply Rademacher-like bounds to create a sampling technique that works well for selecting between compression algorithms.

The following questions are potentially interesting next steps to explore for the CEMD:

- Which other applications would work nicely for CEMD bounds? How can we introduce rigorous learning theory to problems that currently learn from combinatorial data?

  One application we plan to explore is codec selection using *subjective* perceptual audio models based on scores assigned by human listeners. We can treat the human rating of audio divergence as a component of each data point, because people are inconsistent in their ability to distinguish sounds. We can use CEMD bounds on pairs of listeners and audio files to evaluate the best encoding function while accounting for variances in human perception.

- Can CEMD bounds be used for progressive sampling with pruning to provide bounds over algorithm selection problems that are trained over combinatorial data? In particular, can we apply our PSP algorithm to the audio denoising problem?

- The codec selection problem also seems to be a natural application for multi-armed bandits. It poses a slightly different question than the task of function selection. While our PSP algorithm encodes every file with every encoding scheme, the bandit approach would determine a single encoding function for each file. This may require more data, but it may also provide better bounds. How would an

algorithm like UCB1 compare to our PSP algorithm in determining which codec is optimal for the criteria we give?

- How can the CEMD be more easily predicted? Rademacher Complexity can be estimated using covering numbers, and this technique may be extensible to CEMD.

7 Acknowledgements

Without the help of my family, friends, collaborators, and mentors, this thesis would never have been possible. I am truly grateful for all the help I have been given in this process. Cyrus Cousins has been an incredible mentor and research partner. Our collaboration was unplanned; Cyrus and I met last November, and he started sharing papers on Rademacher Complexity with me. From there, we started work on this project that developed into my thesis. I appreciate Cyrus's guidance and direction throughout the duration of the project.

I'm grateful for Professor Upfal's mentorship. His group meetings have exposed me to a wide range of topics in learning theory. His advice on writing and presenting my thesis was insightful and improved the focus of my research. I am also thankful to him for financial support of the project and for being my first reader.

The project presented in this thesis is actually one of two I've worked on this year. I have concurrently researched with Carl Trimbach and Professor Littman on learning robustly across different distributions. I've enjoyed working with both of them, and their mentorship has given me a better understanding of reinforcement learning and of the overall research process.

Throughout my Brown education, I've been fortunate to have been advised and mentored by numerous brilliant and caring people. In particular, I would like to thank Professor Sandstede, Professor Klivans, and Professor Valiant for going out of their ways to help me find opportunities, discover my academic interests, and figure out my goals at and after Brown.

I am thankful for everyone who gave me feedback on the thesis and helped improve my writing clarity: Carl, Cyrus, Jacob Ruth, Kathy Jang, and Vince Kubala.

I am forever indebted to tea, chocolate, avocados, peanut butter, ShareLatex, and the CIT 3rd floor atrium for sustaining me and making these last few months of thesis work more bearable.

Kevin, thank you for your love and patience through this thesis process; I'm beyond grateful to have such a kind person by my side. Barbara, even when were on opposite coasts, talking to you never fails to energize me and make me smile. Mom, words cannot express how thankful I am for your thoughtfulness, compassion, and guidance.

8 Bibliography

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Peter L. Bartlett, Stáphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48(1):85–113, 2002.

Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Alexey Chervonenkis and Vladimir Vapnik. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Cyrus Cousins. On the statistics of combinatorial learning, 2018. URL `cs.brown.edu/people/grad/ccousins`. Online.

Victor De la Peña and Evarist Giné. *Decoupling: from dependence to independence*. Springer Science & Business Media, 1999.

Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

Robert Hegemann, Alexander Leidinger, and Rogrio Brito. LAME MP3 encoder, 2017. URL `http://lame.sourceforge.net/`.

Martin Holters and Udo Zölzer. GSTPEAQ – an open source implementation of the PEAQ algorithm. *Procedings of the International Conference on Digital Audio Effects*, 18, 2015.

Nikil Jayant, James Johnston, and Robert Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, 1993.

Jesse Johnson. General regression and over fitting, 2013. URL `https://shapeofdata.wordpress.com/2013/03/26/general-regression-and-over-fitting`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

Library of Congress. MP3 (MPEG layer III audio encoding), 2017. URL `https://www.loc.gov/preservation/digital/formats/fdd/fdd000012.shtml`.

Pascal Massart. Some applications of concentration inequalities to statistics. In *Annales-Faculte des Sciences Toulouse Mathematiques*, volume 9, pages 245–303. Université Paul Sabatier, 2000.

Hugh McGuire. LibriVox: Free public domain audiobooks. URL `https://librivox.org/`.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, second edition edition, 2017.

Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. *Proceedings of the 43rd IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

Matteo Riondato and Eli Upfal. Mining frequent itemsets through progressive sampling with Rademacher averages. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014, 2015.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Thilo Thiede, William C. Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John G. Beerends, Catherine Colomes, Michael Keyhl, Gerhard Stoll, Karlheinz Brandenburg, and Bernhard Feiten. PEAQ—the ITU standard for objective measurement of perceived audio quality. *Journal of the Audio Engineering Society*, 48(1):3–29, 2000.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984.

Index